ENABLING USER INDEPENDENCE AND CREATIVITY IN UBIQUITOUS COMPUTING ENVIRONMENTS: CONCEPTS, MODELS, TOOLS, USER INTERFACES

IRENE MAVROMMATI

ΕΝΝΟΙΕΣ, ΜΟΝΤΕΛΑ, ΕΡΓΑΛΕΙΑ, ΔΙΕΠΑΦΕΣ ΓΙΑ ΤΗΝ ΕΝΔΥΝΑΜΩΣΗ ΤΗΣ ΑΥΤΟΝΟΜΙΑΣ ΚΑΙ ΔΗΜΙΟΥΡΓΙΚΟΤΗΤΑΣ ΤΩΝ ΧΡΗΣΤΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΔΙΑΧΥΤΗΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΙΣΧΥΟΣ

ΕΙΡΗΝΗ ΜΑΥΡΟΜΜΑΤΗ

APRIL 2011

Πανεπιστήμιο Αιγαίου, Τμήμα Μηχανικών Σχεδίασης Προϊόντων και Συστημάτων
Επιβλέπων καθηγητής: Δαρζέντας Ιωάννης
Τριμελής Επιτροπή: Δαρζέντας Ιωάννης, Σαπίδης Νικόλαος, Σπύρου Θωμάς.

Δηλώνω πως είμαι συγγραφέας αυτής της διδακτορικής διατριβής και κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη διδακτορική διατριβή.

Ειρήνη Μαυρομμάτη

Acknowledgements

I would like to acknowledge the contribution of the people that encouraged me and had an impact on the research reported here. Without them this research would not have been possible.

I would primarily like to thank my supervising professor John Darzentas, as well as professors Thomas Spirou and Nickolas Sapidis for sharing their thoughts, criticisms and suggestions which have shaped my thought process, as well as their patience and encouragement. In particular, I would like to thank professor John Darzentas for his understanding and encouragement during the various circumstances I have encountered during the PhD process, and for inspiring me with his broad multidisciplinary perspective, and Nicholas Sapidis for his enthusiastic encouragement and positive comments.

I would also like to thank people who supported me and assisted me with their feedback, suggestions, criticism, and discussions: the members of the research group DAISy of Research Unit 3 at Computer Technology Institute, and in particular Achilles D. Kameas as well as the members of the e-Gadgets and ASTRA FET research projects, for sharing their thoughts and perspectives. I express my gratefulness to Panos Markopoulos for his assistance on evaluation and HCI issues as well as sharing HCI knowledge with me. I appreciate my partner George Birbilis for providing insightful inspiration, and my friend Guy Dugdale for his feedback and great assistance in editing the final text. Finally I would like to thank my family, and in particular my mother as well as George for their support in practical matters and my daughter Eriphille for enduring my unavailability.

I would like to devote this work to my family, the ones that are here and the ones that are missing.

Abstract

This thesis explored, through an interaction design process, aspects of enabling end user creativity in ubiquitous computing environments and has defined related conceptual and methodological tools to enable a vision of user empowerment and independence within pervasive computing environments. Concepts, models, tools and user interfaces are explored by use of a scenario based design approach.

Research presented in this thesis has made some headway in the effort to empower people to actively shape Ambient Computing environments. It has demonstrated the feasibility of letting end-users shape their ubicomp environments. Experience from system implementation case studies, as well as evaluation of expert and end-user trials, all suggest that an architectural model, where users act as composers of predefined components, is a worthwhile approach.

Evaluation results show that people understand the split in the dual nature of artifacts: their tangible and sensory characteristics and their connectable software counterparts. Conceptual models and alternative information visualizations are needed to support people in creating their own applications. Such visualization methods should combine different syntax styles that act complementarily to each other, thus allowing people (including non-computer experts) to use different ways to describe to the system what they want to achieve. More intuitive/natural ways to express user wishes should be provided in parallel with more formal structures that enable more detailed descriptions and advanced control.

A framework can act as an abridging tool, providing a conceptual basis and theoretical foundation, as common ground for the communication and cooperation between the disciplines involved in user-focused ubiquitous computing research. Theoretical and methodological constructs are presented here, towards the definition of a broad framework that can facilitate the design of ubicomp systems supporting end user development.

Table of Contents

Table of Figures

1

1. Περίληψη στα Ελληνικά

1.1. Θέμα της διατριβής

Στο κεφάλαιο αυτό περιγράφεται συνοπτικά στην ελληνική γλώσσα το θέμα της διατριβής καθώς και οι ερευνητικές υποθέσεις και στόχοι. Η διατριβή αφορά τα Συστήματα Διάχυτου Υπολογισμού (Ubiquitous Computing Systems - Ubicomp), και προωθεί την συμμετοχή στην διαμόρφωση και τον έλεγχο τους από τους τελικούς χρήστες. Διερευνώνται από την σκοπιά της διαδραστικής σχεδίασης μέθοδοι, μοντέλα και μηχανισμοί διεπαφής, έτσι ώστε οι τελικοί χρήστες να αποκτήσουν ένα μεγαλύτερο βαθμό προσβασιμότητας στα οικιακά περιβάλλοντα διάχυτου υπολογισμού.

Ο Marc Weiser, που εισήγαγε τον όρο 'διάχυτος υπολογισμός' (ubiquitous computing), αναφέρεται στο 'αόρατο' της υπολογιστικής ισχύος (Weiser, 1994), όπου η έμφαση πρέπει να δίνεται στην δραστηριότητα και όχι στο ίδιο το εργαλείο (υπολογιστή). Η θεώρηση του είναι πως *«ο υπολογισμός αποτελεί μία αόρατη τεχνολογική βάση την οποία δεν παρατηρούμε, όμως την χρησιμοποιούμε αβίαστα στην διάρκεια της ζωής μας»* (Weiser, 1994). Διαβλέπει *«μία τεχνολογία τόσο συνυφασμένη με την καθημερινή ζωή, έτσι ώστε δεν μπορούμε να την διακρίνουμε»* (Weiser, 1991). Η έρευνα που αφορά τον Διάχυτο Υπολογισμό εξακολουθεί κατά

βάση να στηρίζεται στο κυρίαρχο όραμα του Weiser (Chong et al, 2008), (Bell & Dourish, 2006), οι (Bell & Dourish, 2006) όμως επισημαίνουν πως αυτή η έρευνα πρέπει πλέον να επικεντρωθεί στο παρόν αντί να υιοθετεί ένα όραμα του παρελθόντος για τη μελλοντική (που τώρα πλέον είναι τωρινή) πραγματικότητα του διάχυτου υπολογισμού. Αν και η τεχνολογία είναι πλέον ευρέως διαδεδομένη και ήδη παρούσα (για παράδειγμα μέσω των ευρέως διαδεδομένων RFID tags) και οι ερευνητές μπορούν πλέον να αξιοποιήσουν καλύτερα αυτά που είναι σήμερα διαθέσιμα, είναι δύσκολο να φανταστούμε και να οραματιστούμε νέες καταστάσεις πέραν των συνηθειών μας και τείνουμε να προσκολλούμαστε σε ένα παρελθοντικό εξιδανικευμένο όραμα.

Ο διάχυτος υπολογισμός χρειάζεται να αναπτυχθεί με διαθεματικό τρόπο, σε ένα μεγάλο φάσμα από διαφορετικά περιβάλλοντα χρήσης του. Ο διάχυτος υπολογισμός πρέπει να είναι σε θέση να προσαρμοστεί σε διαφορετικές και μεταβαλλόμενες καταστάσεις χρήσης, καταστάσεις που θα μπορούσαν να είναι μη προβλέψιμες από τους σχεδιαστές και τους προγραμματιστές. Μια λύση είναι να δοθεί η δυνατότητα στους χρήστες ώστε να μπορούν να ρυθμίσουν, να προσαρμόσουν ή να κατασκευάσουν εφαρμογές διάχυτου υπολογισμού (Kameas and Mavrommati, 2001), (Newman, 2002), (Rodden and Benford, 2003). Η λύση αυτή έχει αρκετά πλεονεκτήματα όπως: (α) η βέλτιστη προσαρμογή των εφαρμογών διάχυτου υπολογισμού στις ανάγκες των χρηστών, (β) οι εφαρμογές αυτές να μπορούν να βελτιωθούν σταδιακά, από τους ίδιους τους χρήστες τους, (γ) οι χρήστες να ενδυναμωθούν ώστε να μπορούν να σχεδιάσουν τις δικές τους αλληλεπιδραστικές εμπειρίες για το δικό τους περιβάλλον (με δημιουργικό αντί για καταναλωτικό τρόπο). Τα πρώτα βήματα για την υλοποίηση αυτής της προσέγγισης είναι ο ορισμός ενός συνόλου εννοιών ως κοινή βάση τόσο για τους σχεδιαστές συστημάτων διάχυτου υπολογισμού όσο και για τους τελικούς χρήστες, και η δημιουργία εργαλείων και κατάλληλα δομημένων συστημάτων διάχυτου υπολογισμού που να τις εφαρμόζουν.

1.2. Υποθέσεις της διατριβής

Κύρια ερευνητική υπόθεση της παρούσας διατριβής είναι πως η παροχή δυνατότητας ανάπτυξης εφαρμογών από τους Τελικούς Χρήστες σε περιβάλλοντα Διάχυτου Υπολογισμού είναι μία απαραίτητη και έγκυρη προσέγγιση. Η προσέγγιση αυτή αξίζει να διερευνηθεί περαιτέρω ως ένα ξεχωριστό τμήμα της έρευνας πάνω στα Περιβάλλοντα Διάχυτου Υπολογισμού. Στην παρούσα διατριβή διερευνώνται σφαιρικά τα ευρύτερα ζητήματα που αφορούν στην ανάπτυξη από τους τελικούς χρήστες εφαρμογών διάχυτου υπολογισμού, προτείνεται ένα εννοιολογικό και τεχνολογικό μοντέλο για ανάπτυξη από τελικούς χρήστες (υποστηριζόμενο από τις κατάλληλες συσκευές επεξεργασίας και γραφικά περιβάλλοντα διεπαφής) και βεβαιώνεται η αρχική υπόθεση μέσω εφαρμογής του σε ερευνητικό πρωτότυπο και αξιολογήσεων του.

Η διατριβή έχει ως στόχο να προτείνει τρόπους για να αποκτήσουν οι τελικοί χρήστες ένα επίπεδο διαφάνειας, κατανόησης και έλεγχου της λειτουργίας των περιβαλλόντων διάχυτου υπολογισμού, στα οποία ζουν - όσο αποδοτικά αυτοματοποιημένα και ευφυή και να είναι τα περιβάλλοντα αυτά. Ο σχεδιασμός συστημάτων Διάχυτου Υπολογισμού θα πρέπει να αποτελεί τη βάση ώστε τα 'ψηφιακά επαυξημένα' αντικείμενα στο περιβάλλον (που συμμετέχουν στο σύστημα διάχυτου υπολογισμού) να είναι επαναχρησιμοποιούμενα, με δημιουργικό τρόπο από τους ανθρώπους, και όχι μόνο προσβάσιμα στους σχεδιαστές (Mavrommati & Kameas 2002), (Rodden & Benford 2003), (Rodden et al 2004).

Υποθέτουμε πως οι εφαρμογές διάχυτου υπολογισμού θα αποκτήσουν ευρύ πεδίο εφαρμογής στο οικιακό περιβάλλον, μόνο εφόσον οι άνθρωποι μπορούν να κατανοήσουν ένα βασικό σύνολο εννοιών που διέπουν την τεχνολογία που υπάρχει πίσω από το περιβάλλον που ζουν. Έτσι μπορούν να αναπτύξουν ένα αίσθημα εμπιστοσύνης, βλέποντας πως μπορούν εν δυνάμει να ελέγχουν τα συστήματα αυτά. Κατά συνέπεια, η προσέγγιση του αδιαφανούς «μαύρου κουτιού» που κρύβει την τεχνολογία μέσα του, όπου οι άνθρωποι δεν είναι σε θέση να παρατηρήσουν τη δομή του συστήματος, δεν αποτελεί στόχο της παρούσας έρευνας. Αντίθετα

προτείνεται μία προσέγγιση με 'διαφανή' διαστρωμάτωση, που αποκαλύπτει εν μέρει και επιλεκτικά τη δομή του συστήματος.

Στα πλαίσια της έρευνας αυτής προτείνεται μία ξεχωριστή συσκευή ελέγχου, η οποία μεσολαβεί μεταξύ χρηστών και περιβάλλοντος διάχυτου υπολογισμού, και με την όποια οι τελικοί χρήστες μπορούν να επηρεάσουν τις λειτουργίες του περιβάλλοντος αυτού. Προτείνεται ένα εύληπτο και επεκτάσιμο μοντέλο για το χώρο του Διάχυτου Υπολογισμού, το οποίο γεφυρώνει το νοητικό μοντέλο των χρηστών (που βοηθά στην κατανόηση τους συστήματος), με τις τεχνολογικές κατασκευαστικές έννοιες των περιβαλλόντων διάχυτου υπολογισμού.

Το μοντέλο αυτό επικυρώνεται και δοκιμάζεται μέσα από την υιοθέτηση του σε ένα πλαίσιο αρχιτεκτονικής υλικού-λογισμικού για περιβάλλοντα διάχυτου υπολογισμού. Το αρχιτεκτονικό αυτό πλαίσιο (Kameas, Mavrommati et al, 2003) περιλαμβάνει τη διαδικασία κατασκευής των επαυξημένων αντικειμένων, το περιβάλλον λειτουργίας του συστήματος και τη χρήση υποδομών της τεχνολογίας πληροφορίας και επικοινωνίας.

Σκοπός αυτής της έρευνας είναι να διερευνήσει κατά πόσο η ανάπτυξη από τους τελικούς χρήστες αποτελεί μια κατάλληλη προσέγγιση για εφαρμογές σε περιβάλλοντα διάχυτου υπολογισμού. Διερευνάται σε ποιο βαθμό μια τέτοια προσέγγιση είναι κατανοητή στους τελικούς χρήστες, καθώς και σε ποιο βαθμό αποτελεί χρήσιμο ή ακόμα και απαραίτητο συστατικό της τεχνολογίας διάχυτου υπολογισμού. Η εγκυρότητα της προσέγγισης αυτής αξιολογείται μέσω δοκιμών από τελικούς χρήστες. Οι ίδιες έννοιες που δίνονται στους τελικούς χρήστες μπορούν να παρέχονται και στους κατασκευαστές και τους επαγγελματίες σχεδιαστές εφαρμογών, σε ένα πιο λεπτομερές επίπεδο, καθώς επίσης μπορούν και να χρησιμοποιηθούν από ευφυείς μηχανισμούς αυτοματοποίησης (που τους προσαρμόζουν για τους σκοπούς λειτουργίας τους στην αδιαφανή προσέγγιση «black-box» που υιοθετούν). Δίνονται παραδείγματα από γραφικές διεπαφές χρήστη που βασίζονται (σε ελεύθερο βαθμό απόδοσης) στο μοντέλο αυτό, ώστε να

μπορέσει να εκτιμηθεί η ευρύτητα σε σχέση με το πεδίο εφαρμογής του μοντέλου. Παράλληλα προτείνονται λειτουργίες για την επίβλεψη και σύνταξη εφαρμογών, που αφορούν τις σχεδιαστικές προδιαγραφές των μηχανισμών για ξεχωριστές συσκευές ελέγχου. Προτείνεται επίσης ένα γενικότερο πλαίσιο, που περιλαμβάνει τις βασικές έννοιες και τα μεθοδολογικά εργαλεία, για τον σχεδιασμό συστημάτων διάχυτου υπολογισμού τα οποία να επιτρέπουν τη δημιουργία εφαρμογών από τους τελικούς χρήστες. Το πλαίσιο αυτό έχει στόχο να παρέχει ένα κοινό διαθεματικό πλαίσιο συνεργασίας, κατανόησης και ορολογίας μεταξύ των ειδικοτήτων που εμπλέκονται στον σχεδιασμό και την υλοποίηση συστημάτων διάχυτου υπολογισμού.

Οι επί μέρους στόχοι διαμορφώθηκαν ως εξής:

- Να διερευνηθούν τα επί μέρους ζητήματα που αφορούν την Επικοινωνία Ανθρώπου Υπολογιστή, τα οποία εμπλέκονται στην ανάπτυξη από τελικούς χρήστες εφαρμογών σε περιβάλλοντα διάχυτου υπολογιστή.

- Να δοθούν κατευθύνσεις για τις κατάλληλες έννοιες και εργαλεία τα οποία έχουν στόχο να εισάγουν τους τελικούς χρήστες στο Διάχυτο Υπολογισμό.

- Να δοθεί ένα κατάλληλο και εύληπτο νοητικό μοντέλο για την τεχνολογία διάχυτου υπολογισμού, το οποίο να μπορεί να λειτουργήσει ως γέφυρα μεταξύ της τεχνολογικής υποδομής και των αντιλήψεων των τελικών χρηστών.

- Να προταθούν διεπαφές για τους τελικούς χρήστες (όπως, για παράδειγμα, μέσω μιας εξωτερική συσκευή ελέγχου και σύνταξης εφαρμογών 'Editor'), ώστε να έχουν πρόσβαση μέσω αυτών στα περιβάλλοντα διάχυτου υπολογισμού.

- Να εφαρμοστεί το προτεινόμενο μοντέλο, στο πλαίσιο του σχεδιασμού συστήματος Υλικού-Λογισμικού, στα πλαίσια διαθεματικής ομαδικής ερευνητικής εργασίας. Επί μέρους στόχο αποτέλεσε η εισαγωγή μεθόδων

σχεδιασμού διάδρασης και η ενσωμάτωση της διαδικασίας σεναρίων στο πλαίσιο της διαδικασίας ανάπτυξης που εφαρμόζεται στην επιστήμη υπολογιστών.

- Να δημιουργηθούν απλές πραγματικές εφαρμογές σε περιβάλλον διάχυτου υπολογισμού, στο πλαίσιο όχι μόνο μιας διαθεματικής ερευνητικής προσέγγισης, αλλά και με τους τελικούς χρήστες.

- Να προταθούν συνδυαστικές και πολυτροπικές προσεγγίσεις κατάλληλες για την δραστηριοποίηση και επέμβαση του Τελικού Χρήστη σε περιβάλλοντα διάχυτου Υπολογισμού.

- Να καταγραφούν τα υπάρχοντα και τα αναδυόμενα παραδείγματα γραφικών διεπαφών που σχετίζονται με την δραστηριοποίηση των χρηστών και την επέμβαση τους στις εφαρμογές σε περιβάλλον διάχυτου υπολογιστή.

- Μέσω επαναληπτικών κύκλων σχεδίασης να πειραματιστούμε και να αναπτύξουμε ιδέες και παραδείγματα για εναλλακτικές γραφικές διεπαφές για τους τελικούς χρήστες, που μπορούν να έχουν εφαρμογή στο εν λόγω ερευνητικό πεδίο. Τα παραπάνω μας βοηθούν επίσης για την καλύτερη κατανόηση του πεδίου (μέσω της σχεδιαστικής διερεύνησης) και την διεύρυνση πιθανών εργαλείων και μεθόδων.

- Να εκτιμηθεί η εγκυρότητα του προτεινόμενου μοντέλου και η δυνατότητα εφαρμογής του, έτσι ώστε στην συνέχεια να μπορέσει να βελτιωθεί αυτό μέσω της επαναληπτικής/διαμορφωτικής διαδικασίας σχεδίασης.

- Να δοθεί μια ευρύτερη εικόνα των θεωρητικών και μεθοδολογικών προσεγγίσεων που αφορούν την σχεδίαση συστημάτων διάχυτου υπολογισμού τα οποία υποστηρίζουν τη δραστηριοποίηση και επέμβαση των τελικών χρηστών. Το πλαίσιο αυτό χρησιμεύει για να υποστηρίξει τα επόμενα στάδια

έρευνας στον χώρο αυτό, θέτοντας κοινές βάσεις διαθεματικής συνεργασίας στην έρευνα και ανάπτυξη τέτοιων συστημάτων.

1.3. Μεθοδολογία σχεδιαστικής έρευνας

Η παρούσα έρευνα αφορά το στάδιο της διαμόρφωσης ιδεών (concept phase) στη διαδικασία επαναληπτικής σχεδίασης (iterative design process). Στη φάση αυτή αναπτύσσεται η κατανόηση του πεδίου και διερευνούνται έννοιες, σχέδια και γενικές ιδέες που αφορούν τη σχεδίαση με στόχο την ενδυνάμωση των τελικών χρηστών σε περιβάλλοντα Διάχυτου Υπολογισμού.

Η διερεύνηση αυτή γίνεται από την οπτική γωνία της διαδραστικής σχεδίασης. Η διαδικασία διαδραστικής σχεδίασης, στη φάση της διαμόρφωσης της ιδέας, πραγματεύεται θέματα όπως λ.χ. προτεινόμενες λειτουργίες, εργαλεία, τρόπους αλληλεπίδρασης, παραδείγματα διεπαφών, νοητικά μοντέλα, γραφικές αναπαραστάσεις κτλ. Τα προτεινόμενα σχέδια και σενάρια λειτουργούν ως έναυσμα για περαιτέρω στοχασμό και κατανόηση των συστημάτων Διάχυτου Υπολογισμού που υποστηρίζουν την ανάπτυξη από τους τελικούς χρήστες, αλλά λειτουργούν επίσης και ως μέσα για την αξιολόγηση των προτεινόμενων προσεγγίσεων. Τα γραφικά περιβάλλοντα διεπαφής που παρουσιάζονται δεν τα αντιμετωπίζουμε ως τελικά αποτελέσματα της έρευνας, παρά ως ιδέες / σενάρια που χρησιμοποιούνται για την επικοινωνία, τον εμπλουτισμό των ιδεών και την εμβάθυνση στα θέματα που πραγματεύεται η έρευνα, τα οποία διευκολύνουν την ανατροφοδότηση της σχεδιαστικής σκέψης.

Όσον αφορά τη διαδικασία σχεδιασμού, ο Terry Winograd (στην εισαγωγή του (Carroll, 2002)) αναφέρει πως: *«ο Σχεδιασμός ενός καλού διαδραστικού λογισμικού δεν είναι ούτε επιστήμη, ούτε τέχνη. Δεν αποτελεί ακόμα τυπική διαδικασία σχεδιασμού μηχανικών, όμως πολύ περισσότερα εμπλέκονται σε αυτόν από την έμφυτη δεξιότητα και διαισθητική ικανότητα».*

Η ευέλικτη προσέγγιση είναι συνυφασμένη με τη διαδικασία σχεδιασμού (Cross, 2008). Ζητούμενο είναι μία ευέλικτη προσέγγιση, ανάμεσα στην σχεδιαστική-διαισθητική δεξιότητα και στην τυπική διαδικασία σχεδιασμού των μηχανικών. Ακολουθούμε λοιπόν το μεθοδολογικό πλαίσιο της Σχεδιαστικής Συλλογιστικής (Design Rationale) όπως περιγράφεται από τον John Carroll (Moran και Carroll, 1996), (Carroll 2003, pp.432) και ως μεθοδολογία του ακολουθούμε τη Σχεδίαση βασισμένη σε Σενάρια (Scenario Based Design) (Carroll 2000), η οποία είναι μία ευέλικτη μέθοδος, απευθυνόμενη ειδικά στη σχεδιαστική δραστηριότητα και το σχεδιασμό διάδρασης.

Η διαμορφωτική-επαναληπτική διαδικασία σχεδίασης ακολουθείται στο πρώτο αυτό στάδιο σχεδίασης, στο οποίο εντάσσεται η παρούσα έρευνα: αυτό της σχεδίασης των αρχικών εννοιών (concept phase). Η εξεύρεση απαιτήσεων γίνεται μέσω της συλλογιστικής σχεδιασμού και με την ανάπτυξη σεναρίων. Η επαναληπτική διαδικασία σχεδιασμού περιλαμβάνει κύκλους αξιολόγησης. Στην παρούσα έρευνα η αξιολόγηση στοχεύει στο να παρέχει ποιοτικές κατευθύνσεις στην σχεδίαση. Χρησιμοποιήθηκαν διάφορες μέθοδοι για την διεξαγωγή αξιολογήσεων, συμπεριλαμβανομένων των ερωτηματολογίων, δοκιμών σε πεδίο εφαρμογής, και μελέτες εμπειρογνωμόνων. Μεταξύ άλλων, έγινε επιτυχής χρήση του πλαισίου Γνωστικών Διαστάσεων (Cognitive Dimensions Framework), (Green and Petre. 1996), (Blackwell and Green 2003), από ομάδα εμπειρογνωμόνων σχετικών με το θέμα του Διάχυτου Υπολογισμού, επιβεβαιώνοντας έτσι πως το πλαίσιο αυτό μπορεί να χρησιμοποιηθεί ως μέθοδος αξιολόγησης για συστήματα Διάχυτου Υπολογισμού, τα οποία βρίσκονται στο στάδιο διατύπωσης των αρχικών εννοιών (Markopoulos et al, 2004) - δηλαδή πριν τα συστήματα υλοποιηθούν με συγκεκριμένο τρόπο εκτέλεσης, που θέτει εκτελεστικούς περιορισμούς όχι μόνο στην περαιτέρω ανάπτυξη τους, αλλά και στην αξιολόγηση της ουσιαστικής εννοιολογικής βάσης τους.

1.4. Ερευνητικά Αποτελέσματα

Στην διατριβή αυτή διερευνάται η σχεδιαστική δραστηριοποίηση των χρηστών στα περιβάλλοντα διάχυτου υπολογισμού. Τα αποτελέσματα της παρούσας έρευνας ισχυροποιούν την άποψη πως η πρόσβαση των χρηστών με στόχο τον σχεδιασμό εφαρμογών διάχυτου υπολογισμού είναι μία προσέγγιση έγκυρη, η οποία πρέπει να διερευνηθεί περισσότερο, και να υποστηριχθεί από κατάλληλα συστήματα και πλατφόρμες Διάχυτου Υπολογισμού. Προωθείται η αντίληψη πως ο σχεδιασμός του συστήματος οφείλει να συμπεριλάβει την προσβασιμότητα από τους τελικούς χρήστες. Στο πλαίσιο αυτό της αναγκαιότητας δραστηριοποίησης των χρηστών δίνεται μια επισκόπηση σχετικών εργαλείων και μεθόδων διάδρασης.

Προτείνεται και αξιολογείται ως βασικό εργαλείο για την ενδυνάμωση των χρηστών ένα νοητικό μοντέλο, που ταυτόχρονα αποτελεί και τεχνολογικό μοντέλο, εφαρμόζοντας στον χώρο των διάχυτων συστημάτων υπολογισμού το μοντέλο publish-subscribe.

Η έρευνα αυτή εισάγει στο σύστημα διάχυτου υπολογισμού χωριστές πληροφοριακές εφαρμογές για ποικιλία μέσων (συσκευών) που βοηθούν στη δημιουργία και διαχείριση εφαρμογών διάχυτου υπολογισμού από τους τελικούς χρήστες. Προτείνονται κατάλληλες λειτουργίες για τις εφαρμογές αυτές και γίνεται επισκόπηση στις γραφικές διεπαφές που χρησιμοποιούν παραστατικές και συντακτικές μεθόδους σχεδιασμού εφαρμογών.

Τέλος, η έρευνα αυτή προσδιορίζει σφαιρικά τις διαστάσεις ενός εννοιολογικού και μεθοδολογικού πλαισίου για το σχεδιασμό συστημάτων διάχυτου υπολογισμού, τα οποία υποστηρίζουν τη δραστηριοποίηση των χρηστών και την ενδυνάμωση της δημιουργικότητας τους. Είναι αναγκαίο να υπάρξει ένα τέτοιο πλαίσιο για την σχεδίαση συστημάτων Διάχυτου Υπολογισμού ώστε να χρησιμοποιηθούν ως κοινή βάση κατανόησης και επικοινωνίας μεταξύ των διαφορετικών ειδικοτήτων που εμπλέκονται, χωρίς να είναι απαιτούμενο να κατέχει ο κάθε εμπλεκόμενος σε βάθος τις υπόλοιπες ειδικότητες.

Ως παράπλευρο αποτέλεσμα της έρευνας, εφαρμόστηκε και αξιολογήθηκε το πλαίσιο με βάση τις γνωστικές διαστάσεις (Cognitive Dimensions Framework) (Green and Petre, 1996), για την αξιολόγηση των συστημάτων διάχυτου υπολογισμού. Επικυρώσαμε πως το πλαίσιο αυτό δίνει ικανοποιητικά αποτελέσματα και μπορεί να χρησιμοποιηθεί και για την αξιολόγηση των ερευνητικών συστημάτων διάχυτου υπολογισμού (και μάλιστα από το πρώιμο στάδιο της διατύπωσης των εννοιών των ερευνητικών προτάσεων).

1.5. Περιλήψεις κεφαλαίων

Κεφάλαιο 1: Περίληψη στα Ελληνικά

Το κεφάλαιο αυτό περιέχει την περίληψη της διατριβής στην ελληνική γλώσσα.

Κεφάλαιο 2: Θέμα της ερευνητικής διατριβής

Στο κεφάλαιο αυτό αναφέρεται η ιστορία των όρων και θεωρήσεων για τον Διάχυτο Υπολογισμό. Στην συνέχεια περιγράφονται οι ερευνητικές υποθέσεις, το πεδίο της έρευνας, και τα κύρια σημεία της.

Κεφάλαιο 3: Ερευνητική προσέγγιση

Στο κεφάλαιο αυτό περιγράφεται το μεθοδολογικό πλαίσιο με το οποίο έγινε η παρούσα έρευνα. Ακολουθήθηκε η διαδικασία διαδραστικής σχεδίασης και η σχεδιαστική συλλογιστική (Design Rationale), με βασική μέθοδο τη σχεδίαση βασισμένη σε σενάρια, όπως περιγράφεται από τον J. Carroll. Περιγράφονται επίσης οι μέθοδοι που εφαρμόστηκαν κατά την αξιολόγηση.

Κεφάλαιο 4: Από τα αντικείμενα προς τα επαυξημένα τεχνουργήματα

Στο κεφάλαιο αυτό περιγράφεται η συλλογιστική και η επιχειρηματολογία υπέρ της ενδυνάμωσης της δημιουργικότητας των τελικών χρηστών μέσα στα περιβάλλοντα Διάχυτου Υπολογισμού. Η δραστηριοποίηση της δημιουργικότητας των τελικών χρηστών δύναται να οδηγήσει σε αναδυόμενη λειτουργικότητα των συστημάτων διάχυτου υπολογισμού, όπως μπορεί να προκύπτει από τις εξειδικευμένες ανάγκες και επιθυμίες των ατόμων.

Κεφάλαιο 5: Διάχυτος υπολογισμός και προσεγγίσεις επαύξησης αντικειμένων

Εδώ περιγράφεται η σχετική έρευνα που έχει γίνει από την πλευρά της σχεδίασης συστημάτων διάχυτου υπολογισμού (σχεδίαση υλικού-λογισμικού). Περιγράφονται σχετικά ερευνητικά έργα και οι προσεγγίσεις τους, όσον αφορά επαυξημένα αντικείμενα και περιβάλλοντα: συμπεριλαμβάνουν αισθητήρες στα αντικείμενα και στο περιβάλλον καθώς και επικοινωνιακές και υπολογιστικές δυνατότητες, χρησιμοποιούν κατάλληλες μεταφορές ή μοντέλα χειρισμού των εφαρμογών από τους τελικούς χρήστες.

Κεφάλαιο 6: Ζητήματα επικοινωνίας ανθρώπου υπολογιστή στα περιβάλλοντα διάχυτου υπολογισμού

Το κεφάλαιο αυτό περιγράφει θέματα που αφορούν την Επικοινωνία Ανθρώπου-Υπολογιστή, που προκύπτουν από τις εξελίξεις στα περιβάλλοντα Διάχυτου Υπολογισμού. Θέματα που σχετίζονται με δραστηριοποίηση των τελικών χρηστών αναφέρονται από την σκοπιά της Επικοινωνίας Ανθρώπου Υπολογιστή.

Κεφάλαιο 7: Ανάπτυξη λογισμικού από τους τελικούς χρήστες: βασικές έννοιες

Στο κεφάλαιο αυτό παρουσιάζεται μέσα από βιβλιογραφική ερεύνα μία επισκόπηση των γενικών θεμάτων και βασικών αρχών που αφορούν την Ανάπτυξη Λογισμικού από Τελικούς Χρήστες. Παρουσιάζεται ο ορισμός της θεματικής αυτής περιοχής, τα

χαρακτηριστικά που έχουν οι χρήστες που αναπτύσσουν λογισμικό, οι έννοιες και οι προκλήσεις που αντιμετωπίζουν (μέσα από την χρήση οπτικών ή συντακτικών τρόπων προγραμματισμού).

Κεφάλαιο 8: Ανάπτυξη εφαρμογών από τελικούς χρήστες στο περιβάλλον διάχυτου υπολογισμού

Στο κεφάλαιο αυτό επικεντρώνουμε στα θέματα της Ανάπτυξης από Τελικούς Χρήστες που αφορούν συγκεκριμένα την ερευνητική περιοχή του Διάχυτου Υπολογισμού. Περιγράφονται οι δυνατότητες που δίνονται, οι μηχανισμοί που μπορούν να τους υποβοηθήσουν, καθώς και σχετικές έννοιες. Περιγράφονται διαφορετικές προσεγγίσεις που μπορούν να χρησιμοποιηθούν ως πιθανά παραδείγματα διάδρασης.

Κεφάλαιο 9: Ένα προτεινόμενο μοντέλο για τον συνδυασμό επαυξημένων αντικειμένων

Στο κεφάλαιο αυτό περιγράφεται ένα μοντέλο που μπορεί να χρησιμοποιηθεί για τη σύνθεση αντικειμένων σε εφαρμογές, που βασίζεται στο μοντέλο Publish-Subscribe, προσαρμοσμένο στα συστήματα Διάχυτου Υπολογισμού. Το μοντέλο ανάπτυξης λογισμικού επεκτείνεται ώστε να συμπεριλαμβάνει την έννοια της δυνατότητας δράσης (affordance, όπως περιγράφεται από τους (Gibson, 1977), (Gibson, 1979), (Norman, 1990), που αναφέρεται στις ιδιότητες-ίχνη προιδεασμού για τη χρήση του αντικειμένου). Το προτεινόμενο μοντέλο έχει διπλή λειτουργία, ως νοητικό μοντέλο για την ενίσχυση των χρηστών του συστήματος, αλλά και ως μοντέλο ανάπτυξης του συστήματος, γεφυρώνοντας με κοινό λεξιλόγιο και αναπαράσταση και τους δυο χώρους.

Κεφάλαιο 10: Εφαρμογή του μοντέλου - η περίπτωση του ερευνητικού έργου e-Gadgets

Μια μελέτη περίπτωσης της εφαρμογής του προτεινόμενου μοντέλου, παρουσιάζεται σε αυτό το κεφάλαιο. Η μελέτη περίπτωσης αφορά την εφαρμογή του μοντέλου στο πλαίσιο της δημιουργίας ενός τεχνολογικού πλαισίου, το οποίο υλοποιήθηκε στο ερευνητικό έργο e-Gadgets.

Κεφάλαιο 11: Αξιολόγηση της ανάπτυξης από τελικούς χρήστες εφαρμογών διάχυτου υπολογισμού

Σε αυτό το κεφάλαιο περιγράφονται δοκιμές από τελικούς χρήστες καθώς και αξιολογήσεις από ομάδες ειδικών. Τα αποτελέσματα των αξιολογήσεων περιγράφονται και αναφέρονται σημεία βελτίωσης. Επιβεβαιώνεται πως η προσέγγιση να υιοθετούν τα συστήματα Διάχυτου Υπολογισμού μεθόδους δραστηριοποίησης και ανάπτυξης από τελικούς χρήστες, είναι μία έγκυρη ερευνητική κατεύθυνση, που αξίζει να διερευνηθεί περισσότερο μελλοντικά. Διαπιστώνουμε πως το πλαίσιο γνωστικών διαστάσεων (Cognitive Dimensions framework) (Green and Petre, 1996) μπορεί να αποτελέσει την βάση αξιολόγησης για περιβάλλοντα Διάχυτου Υπολογισμού, δίνοντας ανατροφοδότηση από τα πρώτα ερευνητικά στάδια, πριν ακόμα αναπτυχθεί το πολύπλοκο σύστημα υλικού-λογισμικού.

Κεφάλαιο 12: Οι λειτουργίες της συσκευής ελέγχου και σύνταξης εφαρμογών

Στο κεφάλαιο αυτό δίνονται αρχικές προδιαγραφές για ένα μηχανισμό επίβλεψης, ελέγχου, και σύνταξης εφαρμογών διάχυτου υπολογισμού. Τέτοιου τύπου μηχανισμοί μπορούν να λειτουργήσουν σε ξεχωριστές φορητές συσκευές, τις συσκευές ελέγχου και σύνταξης εφαρμογών. Επίσης περιγράφεται πιθανή λειτουργική δομή, λειτουργίες και δυνατότητες τις συσκευής αυτής.

Κεφάλαιο 13: Γραφικά περιβάλλοντα που δίνουν την δυνατότητα ανάπτυξης εφαρμογών διάχυτου υπολογισμού στους τελικούς χρήστες

Σε αυτό το κεφάλαιο παρουσιάζεται μία σειρά από γραφικές διεπαφές, που μπορούν να χρησιμοποιηθούν από τους μηχανισμούς ή τις συσκευές ελέγχου και σύνταξης εφαρμογών που περιγράψαμε στο παραπάνω κεφάλαιο.

Κεφάλαιο 14: Προς ένα πλαίσιο για τον σχεδιασμό συστημάτων Διάχυτου Υπολογισμού, που υποστηρίζουν την δραστηριοποίηση των τελικών χρηστών

Με το κεφάλαιο αυτό ολοκληρώνεται η έρευνα, συγκεντρώνονται και οργανώνονται τα συμπεράσματα, και δομούνται σε ένα πλαίσιο οι έννοιες και σχέσεις που αφορούν τον σχεδιασμό συστημάτων Διάχυτου Υπολογισμού που υποστηρίζουν την δραστηριοποίηση των τελικών χρηστών. Η κατεύθυνση που προτείνεται είναι προς ένα ευρύ πλαίσιο, το οποίο γεφυρώνει την προσέγγιση του σχεδιασμού εμπειρίας χρήσης, του σχεδιασμού υλικού-λογισμικού, και των θεωριών μάθησης και γνωστικής ψυχολογίας. Η ευρύτερη θεωρητική και μεθοδολογική προσέγγιση παρουσιάζεται με κατηγοριοποίηση σε επιμέρους έννοιες και μεθόδους, οι οποίες συμβάλλουν στη σχεδίαση συστημάτων προσβάσιμων από τους τελικούς χρήστες, και εν τέλει στην ενδυνάμωση των χρηστών να δραστηριοποιηθούν στα περιβάλλοντα διάχυτου υπολογισμού.

Κεφάλαιο 15: Αποτελέσματα και Συμπεράσματα

Τα συνολικά αποτελέσματα και συμπεράσματα που προκύπτουν από την έρευνα αυτή παρουσιάζονται συνοπτικά σε αυτό το κεφάλαιο, καθώς και κατευθύνσεις για μελλοντική έρευνα στον χώρο των συστημάτων διάχυτου υπολογισμού που επιτρέπουν στους τελικούς χρήστες να δραστηριοποιηθούν και να αναπτύξουν εφαρμογές.

Αναφορές:

Στο τμήμα αυτό συγκεντρώνονται οι ερευνητικές δημοσιεύσεις που χρησιμοποιήθηκαν και αναφέρονται στην παρούσα εργασία.

Παραρτήματα:

Επιμέρους λεπτομέρειες της σχεδιαστικής έρευνας καθώς και των αξιολογήσεων, δίνονται με τη μορφή σειράς παραρτημάτων:

- **Παράρτημα 1:** Ανατροφοδότηση απο ειδικούς
- **Παράρτημα 2:** Αξιολόγηση μέσω του πλαισίου γνωστικών διαστάσεων
- **Παράρτημα 3:** Αξιολόγηση στα συνέδρια DC-Tales και BCS-HCI
- **Παράρτημα 4:** Δοκιμή στο i-Dorm και αξιολόγηση απο τελικούς χρήστες
- **Παράρτημα 5:** Ετεροαναφορές στην έρευνα που παρουσιάζεται
- **Παράρτημα 6:** Παράδειγμα σεναρίου δημιουργίας από τελικούς χρήστες εφαρμογών διάχυτου υπολογισμού.

1.6. Σύνοψη

Στο κεφάλαιο αυτό δόθηκε μία περίληψη των κυριοτέρων σημείων της διδακτορικής αυτής διατριβής στην ελληνική γλώσσα. Αναφέρθηκε το θέμα της διατριβής, οι ερευνητικές υποθέσεις, οι σκοποί και οι επί μέρους στόχοι της έρευνας, καθώς και τα κυριότερα ερευνητικά αποτελέσματα που προέκυψαν από την έρευνα αυτή. Αξίζει να σημειωθεί πως από την παρούσα έρευνα προέκυψαν ερευνητικές δημοσιεύσεις οι οποίες χρησιμοποιήθηκαν και αναφέρθηκαν σε περισσότερες από 40 ερευνητικές δημοσιεύσεις άλλων ερευνητών (ετεροαναφορές, όπως αναφέρονται στο παράρτημα 5).

2

2. Introduction to the subject of the thesis

2.1. Introduction

In this chapter the research assumptions, aims and objectives of the thesis are described. The assumption of seamless function of the Ambient Intelligence vision is questioned. The thesis complements this vision by arguing for user control, selective transparency and 'seamfulness[1]' into the working of the system. It attempts to provide methods, models and interface mechanisms for people to gain a greater degree of understanding of, and ease in handling the ubiquitous computing home.

---

[1] Seamfulness: 'exposing the seams', a guiding design principle that signifies comprehensive inter-connectedness and coherence. Here it is used to mean providing selective transparency into the interrelationships and workings of a system.

2.2. A history of terms, from Ubiquitous Computing to Ambient Intelligence

In 1991, Mark Weiser, a research scientist at the Xerox Palo Alto Research Center in California (PARC), published a paper in Scientific American introducing a new vision for the future of computing. The article was "The computer for the 21st Century", and it described a conception of the next generation of computing systems and of computing society (Weiser, 1991).

Third generation computing was seen as an integrated system of advanced computing devices and data communications, available anytime, anywhere and supported by intelligent interfaces (Weiser 1991). Central to this vision are pervasive networked computers, of different shapes and sizes, resident in various settings. To describe these future information systems, Weiser introduced the term Ubiquitous Computing, in his quote: *"Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives"*.

Two years later, in a follow-up article, Weiser stated that the ubiquitous computing environment would be one *"in which each person is continually interacting with hundreds of nearby wirelessly connected computers"* (Weiser 1993). He explicitly points out the 'invisibility' of computing (Weiser, 1994), since the focus should be that of the task and not the tool. He views computing as *"an invisible foundation that is quickly forgotten but always with us, and effortlessly used throughout our lives"* (Weiser 1994), and envisions ubiquitous computing technologies that will be *"weaving into the fabric of everyday life until they are indistinguishable from it"* (Weiser, 1991). To signify this, he introduces the term Calm Technology (Weiser and Brown, 1996), a technology that has the ability to move between the periphery of our attention and its focus. At the time Weiser wrote these articles, the internet

existed and was growing, but was by no means widely used. Nowadays, wired networks and broadband are an exponential part of the telecommunication infrastructure, and wireless LAN, Bluetooth, etc. are supported by many information appliances -with most prominent the mobile phones.

In the decade after Weiser's first article pioneered the future vision of ICT, several different 'flavors' of the idea were promoted, with slightly different orientations, signifying different national or corporate research agendas. Although addressing similar advances in ICT communications and infrastructure, a proliferation of terms were used, corresponding to different corporate or national foci. IBM, (representing the US perspective), introduced the term 'Pervasive Computing' and focused mainly on corporate computing systems, while in Europe, Philips, using the term 'Ambient Intelligence', concentrated on home computing and entertainment [the term has been introduced by Philips' Head of Research Emile Aarts (Aarts, 2001)]. Philips has introduced at the time a system that supported 'smart home' applications (Figure 1), (Figure 2), (Figure 3) and enabled personalization for home information and entertainment applications (Aarts and Marzano, 2003), (Baldus et al, 2000), (Philips Research website: Ambient Intelligence).

Figure 1: The 1999 Philips Project WWICE was an initial investigation into networked home devices and multimodal interaction. (Image from personal files – WWICE1 project video)

Figure 2: Ambient Intelligence: WWICE 2 is a project about the Connected Home that develops applications, user interaction concepts and system architectures allowing communication, content exchange and sharing experiences. (Philips Research website)

At the end of the 1990's, the European Union began promoting similar goals in its research agenda under the title 'Ambient Intelligence' and later 'Disappearing Computer', emphasizing "human-centered" developments in information-communication technologies. Aarts saw the convergence of the technologies of

ubiquitous computing, ubiquitous communication and user interface design (Aarts, 2001). In 2002 the European Union, with its funded IST-Future Emerging Technologies (FET) research agenda promoted the term 'Disappearing Computer' (see: Disappearing Computer initiative). This phrase emphasized the physical as well as the cognitive disappearance of computing, into artifacts used seamlessly in the human environment (be it work, home, education, public, mobile, or other). Sixteen IST research projects were funded by the European Union in the IST FET 5th framework program's (FP5) 'Disappearing Computer' umbrella structure (see: the Disappearing Computer website), while two more followed as part of the Future Emerging Technologies (FET) Open Funding Scheme of FP5, and the 'Disappearing Computer II' Proactive Initiative of the 6th framework (launched in 2004) that funded one additional Integrated Project (see: IST website: Projects launched by the Disappearing Computer Proactive Initiatives). Through these initiatives, the Disappearing Computer initiative gave an initial boost to ubiquitous computing research within Europe.

A point of departure of European IST research, and of the 'Disappearing Computer' (DC) initiative, was the ISTAG report (ISTAG, 2001). This report used four scenarios to communicate a view of future European research developments in Information and Communication Technologies, exploring both the social and technical implications of ambient intelligence, as well as issues of security and trust (SWAMI final report, 2006).

The term "ubiquitous network society" was used in Japan, to describe concrete action plans based on the aforementioned vision. The initiative 'u-Japan Strategy' has been subsequently introduced [(Murakami, 2003), through (SWAMI D1, 2006)]. The 'Realizing the Ubiquitous Network Society' advisory group addressed issues similar to those of the EU's ISTAG report and the Disappearing Computer research initiative program (SWAMI D1, 2006, p.p. 6-10). The final report of the 'Realizing the Ubiquitous Network Society' roundtable discussion was published in 2004 and

was publicized on the website of Japan's Ministry of Internal Affairs and Communications (MPHPT 2004).

Results from the advisory groups in Europe, as well as in Japan, were input to research and technology strategies for beyond 2006, in follow-up research framework programs for IST. In current research literature, a wide variation on the above names is used, with differing combinations of the words Intelligence, Distributed, Disappearing, Ambient, Ubiquitous and Pervasive, combined with the terms Environment and Computing. Some of the resulting compound terms employed are: Ambient Intelligence, Ambient Computing, Ubiquitous Computing, Ambient (Computing) Systems, Distributed Computing, Pervasive Computing, Ambient Intelligence, Intelligent Environments, Ambient Pervasive Computing, and Disappearing Computer.

It has to be noted that current Ubiquitous Computing research continues to revolve around Weiser's original vision. (Chong et al, 2008), (Bell & Dourish, 2006). As Bell and Dourish point out (Bell & Dourish, 2006), research should now focus on the Ubiquitous Computing of the present, rather than clinging to yesterday's vision of the then-tomorrow's ambient world. Ubiquitous technology is already present and visible, and researchers would do better to observe what is currently available, rather than sticking to a dominant, yet long past, idealized vision.

Research described here belongs to the area of Ubiquitous and Ambient Computing. It draws influences from the Philips perspective and by the Disappearing Computer EU framework where the author was actively involved (specifically the IST-FET project e-Gadgets). In this thesis we will use the term Ambient Computing, and Ubiquitous Computing, except where it may, from time to time, be interchanged with the other related terms mentioned above, where this serves better to clarify the perspective of the particular section.
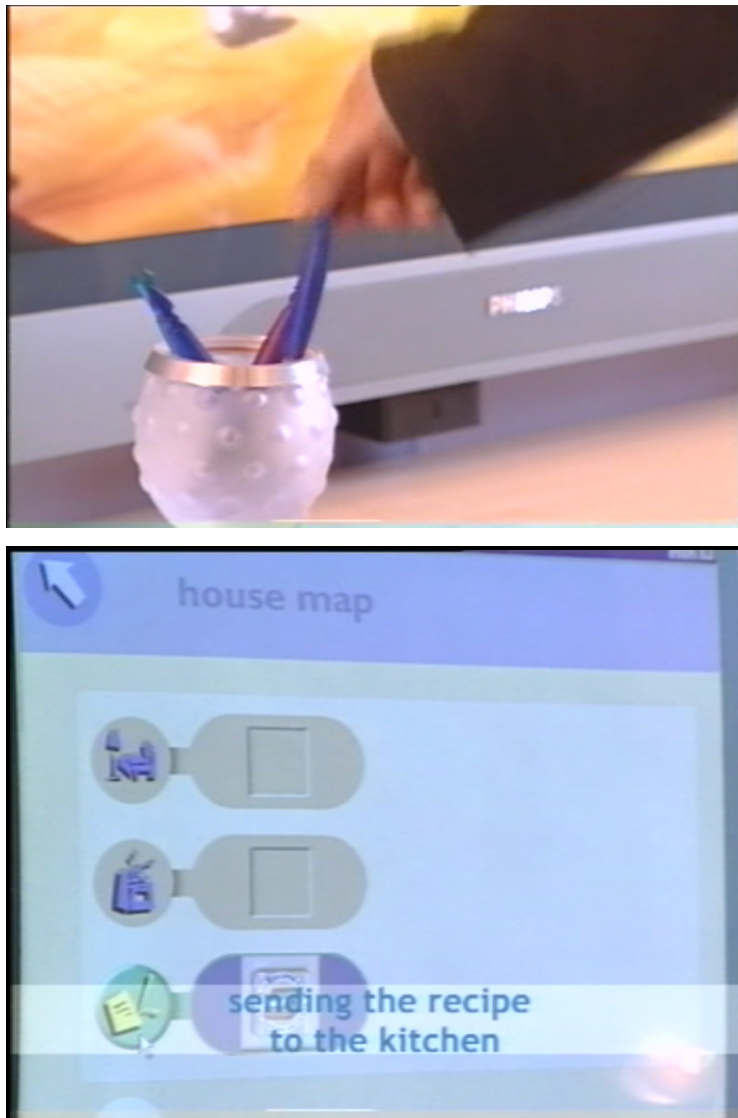
Figure 3: The Philips Research project WWICE is a coherent home network system for the convergence of entertainment, communication and information applications. Multimodal access was provided to various sources throughout the home (modalities included speech, gesture, RFID-tagged objects, and Graphical User Interfaces). (Images source from Philips Research website and personal files).

2.3. Introduction to the subject of the thesis, assumptions and key issues

Pervasive, ubiquitous computing promises that the environments we live in, where we engage in everyday activities, will increasingly consist of computationally augmented artifacts. These artifacts may be information appliances or just ordinary objects enhanced with computing and communication capabilities. Ubiquitous computing technology will need to be deployed and used in an immense range of different contexts, to fit seamlessly into the lifestyle of very different individuals, and to do so without requiring those individuals to attend to this technology instead of their own daily pursuits (Weiser, 1993) (Aarts, 2001), (Aarts and Marzano, 2003), (ISTAG report), (Weiser 1991), (Weiser, 1996).

Ubiquitous computing applications must be able to adapt to varying and changing situations and configurations, determined by the specifics of the environments in which they are participating, situations possibly unforeseeable to designers and developers. One potential solution is to enable users to configure, customize or even construct their ubiquitous computing applications (Kameas and Mavrommati, 2001), (Newman, 2002), (Rodden and Benford, 2003). This solution has several benefits: (a) applications are adapted in the best possible way to users' own requirements; (b) applications can be incrementally improved by their very users; (c) users are able to design their own environments and interactive experiences and shape their own environment in a proactive and creative way, rather than being mere consumers of technology.

The first step towards realizing this approach is the design of a set of concepts, common both to designers and users, and the provision of architectures and tools to implement them. Additionally researchers should consider common referents - such as unifying theoretical and methodological frameworks that view the system as a whole - that abridge the different perspectives involved in the design and development of ambient computing systems supporting End User Development, and

that can provide to all disciplines involved a broader understanding of the elements that are at play.

The research hypothesis in this thesis is that End User Development is a necessary and valid approach for Ambient Computing Environments, complementing Ambient Intelligence automation. It needs to be further explored, on its own merit, as a separate area of research. For this reason, this thesis explores the various issues involved in End User Development in the field of ambient computing and proposes a conceptual model for End Users that acts as both a user and technology model for the Development of Ambient Computing applications, supported by editing devices and Graphical User Interfaces. Using these, the research progresses to validate the aforementioned hypothesis.

Although influenced by Weiser's vision, it is not assumed in this thesis that the Ambient Intelligence Environment will ever turn out to be (Bell and Dourish, 2007) quite as seamless or relaxing as Marc Weiser hoped when comparing it to *"a walk in the woods…"* (Weiser, 1991). It is assumed that Ambient Computing will turn out to be an infrastructure as seamless as electricity or telephony, or even as networking is currently becoming (with interconnected devices such as mobile phones, computers, etc). These usually work seamlessly, supporting human activities unobtrusively, but when they do not, people can access and interrogate them to make the right settings, plug in the right cables, or run the appropriate applications. One needs also to have an idea as to how all the parts are connected when the system fails, so as to take the right remedial action. This thesis sets out to propose ways of gaining a level of understanding and control in the workings of the future ambient computing environment, however efficient or intelligently automated this may be, so that it can indeed be seamlessly integrated into a future form of our everyday activities. Our observation is that so called intelligence (within intelligent environments) is often reduced to automation; it is hereby argued to allow for adding human and societal intelligence as a part of Ambient Intelligent systems, in order to explore emergent uses and so as to potentially benefit from coupling the system's intelligence with the

human intelligence. The system should therefore be considered in its total form, that includes people, societies, mental or physical tools, computing systems, agents and intelligent mechanisms, as parts which evolve in a continuous interplay.

2.4. Scope of research

In a household, a chair is sometimes stepped up on, to reach a high shelf; clothes are hung from it in the bedroom; it can even become a barrier that prevents a child falling out of bed. A mug, when chipped or handle-broken, is used as a pencil holder or a toothbrush holder. Change in use of objects (repurposing) is a normal human practice. These usage changes derive from the object's 'affordances', the possible or potential or imaginable uses to which it might be put. Physical affordances stem from the particular shape, size or material of the object, or characteristics such as containment, grip, etc. Affordances of appliances refer to the possible services an appliance can provide: image, print, sound, light, etc. Information Technology-augmented artifacts should likewise allow for such changes in use. The design of Ambient Computing Systems should provide the basis for artifacts that are used and re-purposed creatively by ordinary people, and not only by designers (Mavrommati, 2002), (Rodden & Benford, 2003), (Rodden et al, 2004).

Such artifacts, because of their inherent connectivity, can support additional 'augmented' functions, more tailored to their owners' lifestyle, making user empowerment an important factor in their sustainable adoption. Returning to the example of the mug, a technology-augmented mug could be used by its owner for drinking, but also as part of a larger system of objects assigned to carry out health monitoring. Or, moved by a new owner to a different environment, the mug could be part of a different superset of objects, this time fulfilling an application that facilitates ordinary tasks: when the level of tea drops, it might assign to the kettle the task of switching itself on, to boil water for more tea. And when, after some time, the augmented mug changes use again to become a toothbrush-holder, it should be

able to evolve its digital function so as to be able to identify toothbrushes that need to be replaced. This vision of artifact sustainability emphasizes the fact that artifacts, as manufactured objects, have a different lifespan and use than do classic information appliances. The latter are normally disposed of altogether after a relatively short period, having no other use than the functions they initially offer. In contrast to these information appliances, sustainability of use should be a key aspect of artifacts as a natural consequence of their physical nature and usage.

Design researchers (among others, Dunn and Raby, Bill Gaver, of the Royal College of Arts, and Josephine Green of Philips Design) argue that ordinary people are the ones to add interpretation and meaning to designs, which need to be intentionally left open for this imagination and experimentation (Dunne and Raby, 2002). People are the interpreters of the open scenarios of technological objects (Gaver, 2002), and the designers' role is allowing these interpretations to occur by providing intentionally open-ended designs (Green, 2007). Philips has promoted 'open tools' as a design strategy (Andrews et al, 2002), in which design shifts away from delivering a finished product or experience, towards designing an 'unfinished' or 'open' solution that can be completed and evolved by users.
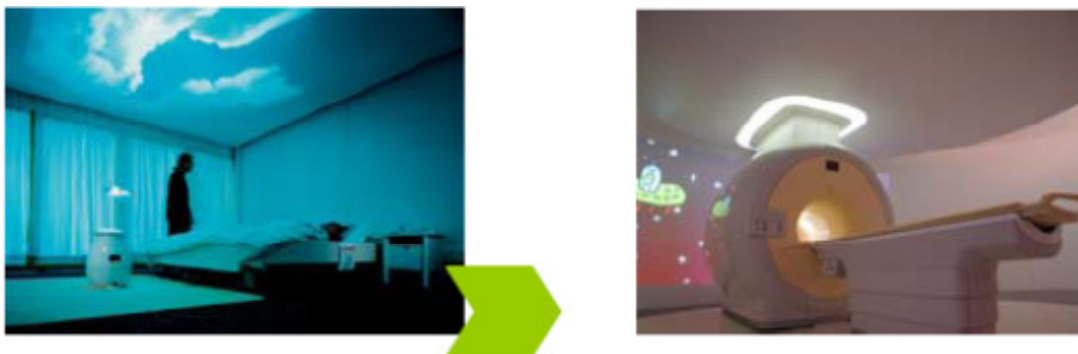
Figure 4: The Philips Nebula project was an 'open' tool, exploring the waking up experience through projecting customized images onto the ceiling. It was later customized for medical examination rooms for children's MRIs. [Image source: (Gardien, 2007)]

For example, Nebula, a Philips Design project presented in 2002 at the Consumer Electronics Show (CES2002), explores the sensations enabled by omnipresent

devices and future technologies which merge into one fluid augmented experience. Nebula is intended by Philips to be used as an 'Open Tool', supporting customized experiences (Gardien, 2007), (Kyffin and Gardien, 2009) by adapting the information streams presented. This contributes to the trend of co-designing by end users (Green, 2007), a concept that includes the idea of customization for different contexts of use and other markets (Figure 4).

A qualitative difference can be noted between, on the one hand, selecting elements within an experience predefined by a designer, or customizing the experience for the particular context and market (as the Nebula project does), and, on the other, 'Do-It-Yourself', allowing total definition of, and therefore control over the overall design and experience of the application. Although this research deals with the latter, it is assumed that designers of the future will retain their role of conceiving and creating meaningful ubiquitous experience applications. A comparison can be made with the development of personal computers and desk-top publishing, which did not make graphic designers obsolete. Rather, democratizing these processes and boosting the quantity of the visual communication material they produced actually increased the demand for better quality graphics from designers.

It is assumed that ubiquitous computing applications will have a widespread impact in the domestic environment only if people can understand a basic set of underlying technology concepts. This understanding is key to developing a feeling of trust, as is experiencing an ability to control such systems. As a consequence, the 'black-box' engineering approach, in which people are not able to observe the structure of the system, is not adopted in this research, but rather a semi-transparent approach which partially discloses the system structure. In this approach, a separate control device within the augmented environment, an Editor, is introduced and validated, making possible the visualization and total design of Ubiquitous computing applications by an ordinary user. A model, both scalable and easily comprehensible, is proposed (by adapting the publish-subscribe model to the context of ubiquitous artifacts) that can bridge the gap between technological constructs and the ordinary user's conceptual

models. This model is validated and tested via its adoption in a software architectural framework for Ambient Computing Home Environments (see Chapters 10 and 11).

This architectural framework, the Gadgetware Architectural Style (or GAS) as presented in (Kameas, Mavrommati, et al, 2003) (Kameas, Mavrommati, et al, 2005), includes the manufacturing process for augmented artifacts, as well as the ubicomp environment's Operating System, information communication infrastructure and other computer science related constructs. In evaluations and end-user trials the validity of a selectively 'transparent' approach into the workings of the ubiquitous computing system, based on the proposed model, has been assessed. The same concepts and constructs are provided to end-users and, at a more detailed level, to manufacturers and professional application developers. They may also be used by intelligent mechanisms that auto-adapt the environment's functions in a black-box approach. Graphical User Interface paradigms based on this model have been sketched to assess its scope and breadth, while mechanisms and functions for the Editor have been proposed which highlight what is required of it.

As a result of bibliographical research and concept development, a first organization of concepts and methodologies towards a broad Framework for the design of Ubiquitous Systems supporting End User Development (EUD) has been proposed. The Framework includes both theoretical and methodological constructs, attempting to address the issues that are involved in the development of ubiquitous computing systems that support end user development. It also aims to provide common ground to the multidisciplinary teams involved in ubicomp systems development, in order for them to understand the broader interplay of elements in an evolving system, and develop a common understanding so as to work together more efficiently. A key aspect is the consideration of co-evolution of people, artifacts and tools, and their social and organizational structures. Last but not least, End User Design is introduced as a separate set of methodologies in End User Development, to complement the dominant part of End User Programming.

Concepts described in this framework touch upon the three different interconnecting perspectives of theory, interaction, and system design. Key concepts in theory are echoed in design concepts, which, in turn, correspond to system design concepts and constructs. Social and cultural dynamics are addressed here as an inseparable part of Ubiquitous Systems, with Activity Theory playing a key role as a theoretical foundation.

2.5. Goals and Objectives

The goal of this research was to explore End User Development as an approach suitable for ubiquitous computing applications, and to validate to what extent it is an approach that is understandable by people, and, for that reason, necessary to ubiquitous computing research developments. The objectives were therefore the following:

- To investigate the various issues involved in end user development for ambient computing applications, from the perspective of Human Computer Interaction (HCI).
- To introduce end user development concepts in ubiquitous computing.
- To provide appropriate conceptual/technology models, that can act as a bridge between ordinary people's conceptions of ubiquitous computing and the actual technology infrastructure.
- To propose high level user-oriented constructs for End User Development for ubiquitous environments (such as an external editor device, its functionality, etc).
- To implement the proposed model in technological infrastructure (in Service Oriented Architectures (SOA) and artifacts). This was achieved in the context of multidisciplinary team work, in the course of which a sub-objective was to introduce interaction design methods and integrate scenario based development to computer science research and development (R&D) team practice.

- To test the potential for application creation by end users and domain experts.

- To report on existing User Interface paradigms emerging in end user development in ubiquitous computing environments; to iterate and validate the above via graphical user interface experiments; to propose combinatory multimodal approaches suitable for end user development.

- To assess the validity and scope of the proposed model: identify its weak points, the scope that it can be applied within, and the basis for iterations and improvement of the model.

- To provide an overview of constructs and related work, both theoretical and methodological, regarding the domain of ubiquitous systems design supporting End User Development. This effort towards a broad framework can further inform research in this domain.

2.6. Positioning and significance of the thesis

End user development can be seen as a controversial approach for Ambient Intelligent systems. Ambient Intelligence research often assumes agents that make the system opaque and a wireless distributed system infrastructure that works perfectly. Intelligence in this case often results in automations of presupposed functionality. In contrast, agents in this research are treated as a part of the system, but not the defining factor. This research was among the first published strands of work (Mavrommati and Kameas, 2002), (Mavrommati and Kameas, 2003), (Kameas et al 2003), in parallel with the work of Newman (Newman et al, 2002), (Edwards, Newman et al, 2002), to promote End User Programming in Ubiquitous Computing as a part of work within EU's Disappearing Computer framework program. End User Development in Ambient Intelligent environments remains a rather limited field to this date, the primary focus being on agents assisting with intelligent configuration. Small-scale research is facing inherent difficulties (multidisciplinarity, scope, robustness of systems and communication networks, etc.).

Weiser's vision (Weiser, 1993) was of a world populated with information devices and appliances (rather than augmented objects of everyday use), pre-supposing some sort of display adjusted to the devices and to be used for control and feedback. Nevertheless, these devices are information appliances rather than everyday objects which are RFID tagged or ICT enhanced. This has also been the perspective of many research projects within the Disappearing Computer initiative (see: Disappearing Computer initiative website). Research reported here rejects the adhering of a screen or LED lights to artifacts because of the limits this imposes on a scalable and inclusive approach. In the reported projects that focus on direct manipulation - for example, the Shiftables (Merrill et al, 2007) - programming is limited to certain functions pre-determined by application designers. In contrast, this thesis proposes the use of external devices, Editors, which can always access artifacts and services in an environment and in that way they can be used for end-user overview, access and control of all typologies of artifacts and their configurations (applications).

At the starting point of this research it was not conceivable by designers that they could leave the design of the application experience to end-users. Designers see themselves as orchestrators of applications, that can be open-ended only in certain predetermined aspects - such as changing the background of the desktop, or, analogous to that, at the Philips Nebula project (Gardien, 2007), (Green, 2007), (Kyffin and Gardien, 2009) where end-users change the image projected on the ceiling from their alarm clock, but do not conceptualize or create the entire application. This thesis, while keeping designers and application developers in their role of defining user-targeted and marketable applications, aims to promote the 'democratizing' of design to the people who actually use it, so as to be able to be creative for their own specific needs, and assumes that emergent function and use of ubiquitous systems will arise from people's use in this way. While this research has been ongoing, there has been a strong and growing trend to social media and open collaborative standards. There are now an increasing number of commercial applications and strategy reports promoting user-led innovation, mainly addressing the internet area - see (Leadbeater, 2008) and (Sharp and Salomon, 2008).

It is assumed that future public perception of interconnected artifacts will follow a trend similar to that which we can observe today in regard to the internet and social media. The many user-led developments in blogging, tweeting, and social networks have generated an almost universal awareness of and enthusiasm for the possibilities of accessing, participating in and innovating with electronic media. An absolute working assumption of this thesis is that this revolution in consciousness is the base from which will extend an exactly similar public grasp of the potential of interconnected artifacts.

2.7. Process and method

Work reported here pertains the concept phase of an Iterative Design process, regarding End User Development for Ubiquitous computing. The reported work is an investigation through the perspective of interaction design, of what such tools could be like, and what elements and functions they might have. The graphical user interface proposals supporting the End User Development (EUD) concepts act as prompts for further design and reflection on EUD system functionality and its comprehensibility by end-users. They are also an essential means of assessing the validity of EUD approaches for AMI environments. Graphical User Interfaces should not be seen here as end points of investigation, but more like scenario sketches, to be used for communication, the generation of ideas, and facilitating iterative design thinking.

The multitude of concerns leads to endless and sometimes confusing design details and choices (T. Winograd introduction in (Rosson and Carroll, 2002)). A flexible approach is inherent in the design process (Cross, 2008). For coping with this the optimal theoretical framework is Design Rationale, as described by Carroll (Moran and Carroll, 1996), (Carroll, 2003, p.432) and Scenario Based design methodology (Carroll, 2000), which is both flexible and specifically oriented to design activity. As Terry Winograd (introduction (Carroll, 2002)) has recognized regarding the

design process: *"Designing good interactive software is neither a science not an art. It is not yet a matter of routine engineering, yet there is far more involved than native skill and intuition".*

The iterative design process was followed at the concept phase: finding requirements through scenario-based design and design rationale, using textual as well as graphical and visual representations. The iterative design process includes evaluation cycles. Evaluation in the course of this research aimed to provide qualitative design directions. It was conducted with multiple methods (including questionnaires, field tests, and expert trials) which will be described in detail in further chapters. A notable development was the use of the Cognitive Dimensions framework (Blackwell and Green, 2003), (Green and Petre, 1996) by a group of subject experts. The Cognitive Dimensions evaluation framework was used successfully in the context of Ubiquitous Computing Systems design in its concept phase (see Appendix 2) and was thus validated as an evaluation approach that is usable in this area (Mavrommati et al, 2004), (Markopoulos et al, 2004).

2.8. Results and Research Contribution

This research has addressed the following:

The research was among the first to introduce End User Development for ubiquitous computing applications. The research validated End User Development as being a worthwhile approach for Ambient Environments. In addition, this research work promoted methods from the field of Human Computer Interaction and user centered design priorities within the context of computer science work in ubiquitous computing.

- This research has introduced to both designers and system engineers a new perspective for user led innovation in ubiquitous computing. This was done

by describing a rationale for End User Development (EUD), as a complementary approach to artificial intelligence, involving keeping people in a loop of creating their own environments and applications. The proposed design position statement was an influential starting point (Mavrommati and Kameas, 2002) for interdisciplinary research, as it argued for providing the means for people-led innovation within ubiquitous computing technological research and promoted emerging functionality as a result of ordinary people's creativity, alongside designer-led innovation.

- This research integrated design research methods, rationale, and scenario based design with ubiquitous systems computer science research practice, within the context of multidisciplinary teamwork (see: (eGadgets project website), (ASTRA project website)).

- An overview of approaches and issues relating to Human Computer Interaction for the field of End User Development has been presented.

- The End User Development approach to Ambient Computing environments was assessed for its validity, through user and expert trials. It proved to be a worthwhile approach, not alienating users in spite of the programming concerns introduced to them. It is therefore considered as a worthwhile approach for ambient environments that is complementary to artificial intelligence.

This research work proposed and evaluated a model supporting End User Development in ubiquitous computing.

- This research introduced End User Development concepts and constructs in the area of Ambient Computing Environments; an existing software model was adapted (the Publish-Subscribe model), by adding the concept of

affordances, so as to act as a conceptual model for people to reason about augmented artifacts.

- The proposed model was successfully integrated in a Service Oriented Architecture implementation and the creation of artifacts. The model was central to the proposal of a technological framework for Ubiquitous computing, within a multidisciplinary research context (in the e-Gadgets project case, part of the Disappearing Computer Framework program).

- End User Development, through its case study deployment in the e-Gadgets project case (via the instantiation of a wire-connections model), was validated as a useful approach for UbiComp environments and applications, allowing people to reason about and manipulate their environment. This approach is seen as complementary to intelligence (and not in conflict with it).

- Related graphical user interfaces and different interaction mechanisms for end user development.

The research assessed the Cognitive Dimensions as an evaluation framework that can be used in the context of validating ubiquitous computing constructs.

- As an additional outcome of the model's validation, the Cognitive Dimensions Framework was found to be a valid evaluation framework for the starting phase of conceptualization of Ubiquitous computing constructs.

The research introduced an overview of an Editor, its required functions, and investigated further possible end user notations and syntax (by outlining Graphical User Interface modalities).

- An interaction approach for an editing device and a set of specifications for it were proposed.

- Several Graphical User interfaces, supporting various graphically based structures and methods of interaction for Editing devices, were proposed. Sketching alternative / complementary proposals for Graphical User Interfaces was a means to expand on our concepts as well as to reason and expand on the proposed approach. An overview of AMI EUD Interfaces and alternative abstractions was provided.

This research has defined an initial framework for End User Development in ubiquitous computing applications.

- Following the investigation into several aspects that come into play in End User Development for Ubiquitous Computing applications, a classification emerged summing up the related theoretical and methodological constructs. Towards the end of this research path, an initial definition of a broad Framework for the design of End User Design in Ambient Computing supporting End User Development was outlined. In this context, End User Development is seen as consisting of two distinct but complementary parts: End User Design and End User Programming. The first, End User Design, is introduced as an important issue that needs to be addressed separately and on its own merits. The proposed broad framework combines cognitive theories and constructs from psychology, design and computer engineering.

2.9. Outline and contents of the thesis

The thesis consists of the following chapters.

Chapter 1: Summary in the Greek language

An outline of this thesis in the Greek language is provided in this chapter.

Chapter 2: Subject of the thesis

The first chapter gives an introduction to the thesis subject and an overview of terms used in the area of Ambient Computing. It describes the research hypothesis, the research assumptions, and the scope of research described in this thesis.

Chapter 3: Research approach

The methodological framework and approach used in this research is described in this chapter. Scenario based design process and evaluation methodologies are outlined.

Chapter 4: From Objects to Artifacts

The third chapter provides the rationale on the grounds of which end user empowerment needs to be promoted in the area of Ubiquitous Computing environments, and proposes that end-user creativity and emergence can arise from niche uses by different people.

Chapter 5: Ubiquitous computing and approaches to augmenting artifacts

Related research work on Ubiquitous Computing systems is described in this chapter.

Chapter 6: HCI issues for Ambient Computing Environments

Issues from the perspective of Human Computer Interaction that are emerging in the Ubiquitous Computing environment, relating to End User Development, are presented in this section

Chapter 7: End User Development in software: basic concepts

An overview of generic aspects of End User Programming, for software applications is outlined in this chapter. The profile of users, issues from Computer Systems Collaborative Work (CSCW) research, and software programming issues such as semantics, syntax and visual paradigms are described.

Chapter 8: End user development in ambient computing environments

Issues for End User Development focusing in the area of Ubiquitous Computing Environments are presented here. Different perspectives and approaches of related research projects are presented, as a possible toolbox of different interaction and technology paradigms.

Chapter 9: A proposed model for the recombination of artifacts

The 'Capabilities and Links' model, inspired by the Publish-Subscribe paradigm as applied to ubiquitous computing extended with the notion of affordances, is presented here. The model aims to bridge ubiquitous technology constructs with the human perception of artifacts and their use. In terms of End User Development the model was applied through a wire-connections programming paradigm.

Chapter 10: Application of the Capabilities and Links model: the e-Gadgets case

A case study for the application of the model in a technological framework for Ubiquitous computing, that promotes End User development, is presented here.

Chapter 11: Validation of end user development and the proposed model, through deployment in e-Gadgets

End user and expert evaluation trials are described in this chapter. The End User Development approach for Ambient Computing Environments is validated through these trials. The Cognitive Dimensions Framework is suggested as a method for evaluating Ambient Computing systems, providing valuable input for the early concept development phase in UbiComp research.

Chapter 12: The functions of the Editor

The Editor's possible editing functionality and its possible modalities and structure are presented here.

Chapter 13: Graphical User Interfaces: abstractions and syntaxes for End User Configuration in AmI

This chapter presents different graphical user interface approaches for the configuration of Ambient Intelligence environments, as well as existing variations on different syntax and semantics.

Chapter 14: Towards a framework for the design of Ubiquitous Systems supporting End-User Development

This chapter introduces a broad framework for the design of Ubiquitous Systems supporting end-user configuration, that bridges the perspectives of cognitive

theories, design and computer science. It is to be used for providing common ground to the different perspectives and disciplines involved in the design and development of ubiquitous systems. This chapter classifies the related theoretical and methodological tools and presents an emerging framework whereby End User Development is seen as a necessary affordance that ubiquitous systems research and development should seek to provide.

Chapter 15: Conclusions and Outcomes

Research outcomes and conclusions are summarized in this chapter, and future directions for work are outlined.

References:

The bibliographical references reported to in this thesis are described in this section.

Appendices:

Details of the evaluation sessions and their outcomes is described extensively in the following appendices as well as other related information on this thesis.

- **Appendix 1** - Expert review
- **Appendix 2** - Cognitive Dimensions framework
- **Appendix 3** - Evaluation at two conferences
- **Appendix 4** - the iDorm user test
- **Appendix 5** - Citations
- **Appendix 6** – Example EUD scenario in AmI

3

3. Research approach

"The basic argument behind scenario-based methods is that descriptions of people using technology are essential in discussing and analyzing how the technology is (or could be) used to reshape their activities. A secondary advantage is that scenario descriptions can be created before a system is built and its impacts felt". (Excerpt from 'Usability Engineering: Scenario-Based Development of Human Computer Interaction' by Mary Beth Rosson, John M. Carroll).

3.1. Introduction

The methodology used in the current thesis is described in this chapter. Systematic bibliographical study and search into the international science index is a fundamental part of research. In the case of this thesis, bibliographical research was broad, as it had to cover the different - and often multidisciplinary - areas that are considered as fundamental aspects of research relating to enhancing people's creativity and end user development in ubiquitous computing environments. Bibliographical research covered areas such as ubiquitous computing research, human computer interaction issues relating to ubiquitous computing, aspects of end user development - both for software applications as well as emerging research in

ubiquitous applications-, and last but not least, input from theoretical foundations of cognitive psychology. Bibliographical study lead to the understanding of issues that served as input to the problem's analysis phase –these issues are described in the different chapters of the thesis.

Following the interaction design process, after the extensive bibliographical study (which corresponds to the analysis and research definition phase), the concept phase followed. A number of proposals are typically suggested in the design process, as each proposal addresses different issues of the system, and alternative proposals help with reasoning the suitability and applicability of the proposed ideas with the problem issue addressed. The stages of the design process are not independent of each other, but iterative, as the issues addressed in the first phases may change according to the ideas that are being developed in subsequent phases. Innovation can result from the design process occurring from the re-formation of research issues and research questions, or from the development of alternatives and the selection and iteration and improvement of those that are more applicable to the problem at hand (Reeder, 2002).

A major issue with research about people in ubiquitous computing environments is that the environment cannot be replaced by a 'problem space'. A problem space includes only a selection of certain aspects of the world that have been coded. Interactive exploration includes richer aspects of the world (artifacts and world properties). Activity theory is an influence on this research in terms of understanding not only the design process, but also the actions of people in the ubiquitous computing environment. Cognition is not seen as only an internal mental performance, but as a result of the mind and the world interacting; pure mental performance in itself is not adequate (as justified in (Gedenryd, 1998) p215). Design is used in this thesis as a vehicle for envisioning, thinking about, instantiating and understanding a possible future environment (see Figure 5). As (Gedenryd, 1998) states: "*Design is not a purely intramental process closely tied to the fundamental mechanisms of intentionality and planning, but a similarly sophisticated cognitive*

technology; relying on subtle sophisticated co-evolved artifacts and working techniques".
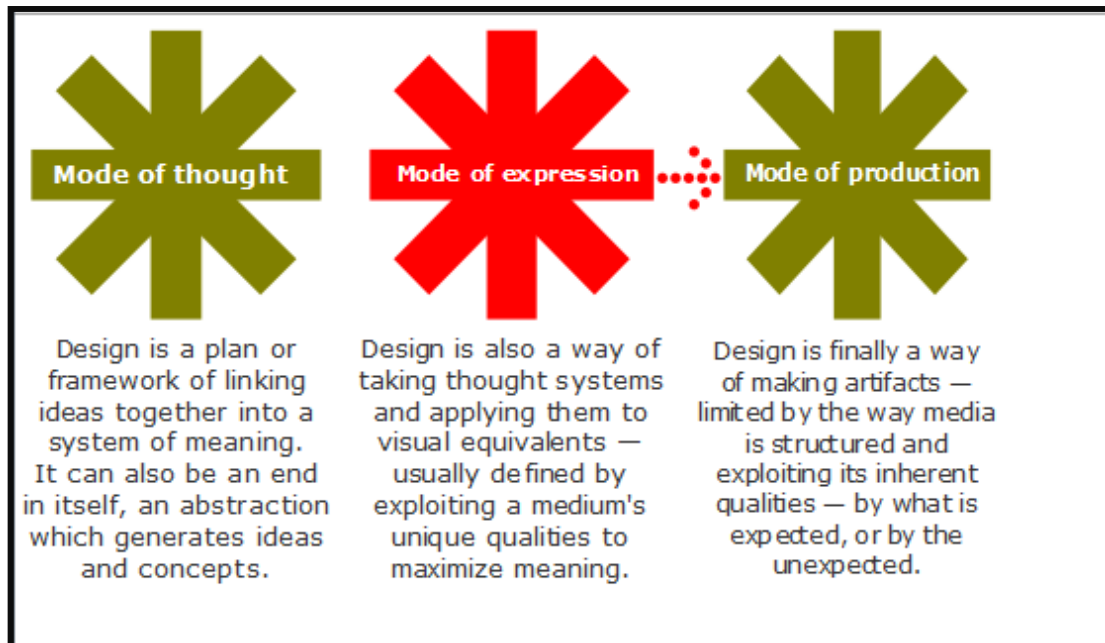
Figure 5: Design is seen as a mode of thought, while it has been traditionally associated with expression and production as well. Image source: http://en.wikipedia.org/wiki/File:Design_modes.svg

Scenarios and UI proposals reported here adopt the role of the means to the design process, a role that software design methodologies assign to requirement specifications. (Marc, 1995). Following the iterative design process, the work reported in this thesis is positioned in the initial stage of design analysis; proposals are seen as design sketches in order to explore the addressed research area. Graphical User Interface proposals are used as experiments, helping to get insight into the issues of end user creativity in ubiquitous computing environments. Interaction design is used as an analysis aid, aiming to develop ideas for End User Development in Ubiquitous Environments, which can open the path to more reliable concepts for the field, or act as a set of paradigms.

According to (Rosson and Carroll, 2002, pp15): *"The design of computing systems is part of an ongoing cycle in which new technology raises new opportunities for*

human activity: as people's tasks change in response to these opportunities, new needs for technology arise". They argue that descriptions of people using technology – in scenario based methods - play an important part in the discussion of how technology is formed and how people's activities could be reshaped through it. Scenario descriptions are made before a system is developed and deployed, giving us a better understanding of the system and its consequences.

Interface proposals are regarded as scenario sketches, which are an integral part of design activity. They are experiments that lead our thinking, rather than conclusive outputs (Gedenryd, 1998, pp129.). A multitude of ideas are sketched, - although they are at a high level of abstraction and their completion is not pursued - in order to determine the ones selected for further refinement or presentation. Draft ideas are enablers that help us recreate a future situation of use. Those few finally selected help understand aspects of the problem, as they offer starting points of many possible paths to address it.

3.1. Interaction Design Process

The aim of this research is to facilitate interactions between people and ubiquitous systems; in doing so, it tries to maintain a high level perspective, with the aim to provide directions for interaction within the ubiquitous environment rather than specific solutions. Thus it does not adhere to detailed specific instantiations or paradigms, but tries to leave intentionally open the specific design implementation details, and instead, focus on higher level directions and characteristics of the ubiquitous system, from an interaction design perspective.

"Interaction design is about behavior; it's the art of facilitating interactions between humans through and with products and services" (Saffer, 2007, pp4). Interaction design is considered as an applied art, in the sense that there are no set methodologies that are proven and that can hold true for all instances; it is

contextual and based on a hermeneutic approach, and it is defining 'on the fly' the related approaches and processes that can be used to gather data and draw conclusions (Rauterberg 2003) (Schon, 1983) (Saffer, 2007). Such approaches may be borrowed by interaction designers, depending on the context, from different disciplines and (such as, for example, art and design techniques, ethnography, cognitive psychology, computer science, human computer interaction).

According to Carroll (Carroll, 2000, pp42), design observes the world in action, in order to identify certain issues and then propose solutions to these issues. An interaction design project is not an isolated activity, since there are interrelations and consequences from the extended use of a design. People's tasks may change over the time that the design is used, even in unforeseen and unpredictable ways, which are sometimes inconsistent with the core functionality that is provided by that designed system. System design, according to (Saffer, 2007, pp30) can be considered as one approach to interaction design. A working design representation is tied directly to human needs and situations of use (Carroll, 2000).

There is a different set of problems that can be addressed by design than the ones that can be addressed via scientific methods. Design is considered a *prescriptive activity*: as Lawson (Lawson 1980, pp.90) stated, designers try to answer questions like: *"What could it be"* and *"What should it be"*. Designers try to define and create the future; they do not try to predict the future by forming an informed understanding of the present –as is the case with the *descriptive* approach of science.

Schon, in (Schon 1983) suggests that design is defined by the designer's own experience, that defines which process should be used and what part of knowledge is required for a given context. During the design process, the designer iteratively reflects on the problem, provides elements of solutions and evaluates them, then proceeds to another iterative cycle of re-consideration and action, that is based on the previous step (this is called by Schon *"reflection in action"*).

There are *no optimal design solutions*, but only selected design paths that satisfy to a certain degree the problem constraints, while compromising regarding other aspects. Lawson (Lawson 1980) has concluded that there is no absolute method of design that ensures that the designer will find the optimal design solution. Lawson claims that each design problem has its own unique characteristics that it is not possible to solve following the same unique methodological steps, but these steps should be defined per case. It becomes evident that design consists of vaguely defined phases, -the so called "design process" - that are cross influenced.

"The overall interaction design process does not exist" (Rakers George, 2001, pp32). Interactive System design is likely to involve not a single process, but several concurrent processes (Newmann and Lamming, 1995).

> In its basic form, the overall process is characterized by three broad generic phases:
> *thinking, designing, and realizing.*

These are *not necessarily subsequent to each other*, but orchestrated and inter-associated, with emphasis placed on getting the right information and finding a solution that appropriately fits (Rakers, 2001).

In providing a little more detail, the iterative process that is followed in Interaction design, has the following iterative (or sometimes even concurrent) steps:

- Analysis, (gathering data, materials, and inspiration about the problem area and sometimes organizing them in visual matrices and boards)
- Concept creation: brainstorms and workshops are in this phase, that generally correspond to the production of many different sketches/scenarios, done fairly quickly, highlighting different issues and addressing different aspects
- Concept development: taking elements from the above sketch 'inspirations' and detailing them further to design proposals

- Evaluation, that happens iteratively in between any of the process phases, providing input and insight for iteration or development in the next process steps

- Realization (sometimes followed by deployment and aftercare), that corresponds to helping the design to be implemented into production.

The above stages act in iterative cycles, without clear border lines between the phases, (especially more so in the first phases of the design process). In the software design life cycle, the iterative design process (based on the waterfall model, with iterative cycles added to it), is corresponds roughly to the above phases.

3.2. Design method

The Human-Computer Interaction (HCI) community has defined techniques for modeling the interaction between user and system, and has proposed user-oriented constructs for systems and software development (España et al, 2008). We will discuss issues from the HCI perspective (specifically from the orientation of Experience and Interaction Design), rather than from the Software Engineering perspective, -which has been defining methods to develop software and systems by specifying data, but not focusing on users and their experience-.

The two most influential paradigms are Positivism and Phenomenology, for scientific research and design respectively. Positivism for scientific research is a rational problem solving approach, whereby successful problem solving involves searching selectively within the possibilities of a problem space, and reducing it to manageable solutions, by logical consistency and deductive logic, and based on the underlying idea of an absolute truth. (Rauterberg, 2003). Humanities, on the other hand, use a hermeneutic approach, grounded on intuition, whereby truth is based on 'belief'; the most important basis for conclusions is a value system based on an individual knowledge base. A personal knowledge base (crafts and skills) is

exclusively accessible to the designer, sometimes without conscious reflection, often relying on intuition and experience, - or the so called *"reflective practice"* (Schon, 1983). In fact, experience / interaction design activity shares both paradigms, thus being able on one hand to predict and explain reality, but on the other, to become part of reality and change it (Rauterberg 2003).

There is no single way to practice design methods. There is an inherent flexibility in the design approach; it is a combined product of the right use of techniques and methods as much as of intuition and skill. (T. Winograd introduction in (Rosson and Carroll, 2002)), (Cross, 2008). Jones (1991) also recognized this by stating: "*Methodology should not be a fixed track to a fixed destination, but a conversation about everything that could be made to happen. The language of the conversation must bridge the logical gap between past and future, but in doing so it should not limit the variety of possible futures that are discussed nor should it force the choice of a future that is unfree.*"

In the context of this thesis, design methods are used with the aim of generating concepts for End User Development in Ambient Intelligence Environments, in order to provide an initial framework for the field. In interaction design, quick sketches, scenarios and rapid prototypes are used to help conceptualize the problem domain. The aim here is to address the analysis phase, via the interaction design process using scenario based design (including both visual sketches and text based descriptions).

In order to conceive the future ubiquitous environment, scenarios are created in verbal and iconic forms, as the most flexible, open and appropriate language for this discourse. According to John Carroll, in *Making Use* (Carroll, 2000): "*Scenarios of human-computer interaction help us to understand and to create computer systems and applications as artifacts of human activity, as things to learn from, as tools to use in one's work, as media for interacting with other people. Scenario-based design offers significant and unique leverage on some of the most characteristic challenges*

of design work: Scenarios evoke reflection in the content of design work, helping developers coordinate design action and reflection. Scenarios are at once concrete and flexible, helping developers manage the fluidity of design situations. Scenarios afford multiple views of an interaction, diverse kinds and amounts of detailing, helping developers manage the many consequences entailed by any given design move. Scenarios can also be abstracted and categorized, helping designers to recognize, capture, and reuse generalizations, and to address the challenge that technical knowledge often lags the needs of technical design. Finally, scenarios promote work-oriented communication among stakeholders, helping to make design activities more accessible to the great variety of expertise that can contribute to design, and addressing the challenge that external constraints, designers, and clients often distract attention from the needs and concerns of the people who will use the technology."

3.3. Can there be theory based design?

Herbert Simon – who was among the first to discuss scientific approaches in interdisciplinary design research - [(Simon, 1969) via (Carroll, 2006)] suggests that designers should not design with fixed goals [(Simon, 1969) pp162-167], because design activity is a means to identify further design goals – even goals that may be inconsistent with the original goals of the design activity. Each situation that design generates is a starting point for a new design activity, and with this process one gradually identifies different aspects of the problem space. Design activity is a problem solving activity, investigating and identifying aspects of the problem space. Design is therefore, as Gedenryd observes, at the core of human activity; it defines what humans *are*. The effort to classify design activity as being (or not) a scientific activity is therefore irrelevant, as that question is invalid (Gedenryd, 1998).

The design process often uses a hermeneutic approach; the designer is getting stimuli for inspiration, and starting the process somehow in the middle, assessing

and progressing the design as he/she goes along. Interaction and User-interface design is often considered a craft rather than an engineering discipline, as it is influenced by social as well as technical considerations, and has elements of fashion, acceptance, affect, playfulness (Blackwell & Green, 2003). On the other hand, Interaction design being part of the broader Human Computer Interaction area, it is a structured, although sometimes understated discipline. As such it uses techniques and methods that are used in design for thinking about a problem (i.e. quick prototyping and sketching is characteristic in the design process). These methods are not put together in a methodology by designers, (since designers are trained in practical problem solving rather than being fluent and analytic in verbal terms), but such a methodology is often articulated by system engineers and cognitive scientists, who, via interdisciplinary research, get an understanding of the design process and are better in formulating it in more structured argumentation (for example, see the work of Carroll or Gedenryd).

In Interaction Design, **Design Rationale** plays a central role in the "middle and around" process of design. Carroll and Rosson observed in their chapter *Design Rationale as Theory* in ((J. Carroll, 2003) pp.432.)) that *"the most troubling reaction to early theory work in HCI was to dismiss theory, science, and even research in general as not relevant enough to real HCI problems of system development). ... The touchstone design challenge for HCI in the 1980s was to understand and address difficulties experienced by new users... (learning, problem solving, and error management in workplace contexts). But the most articulate theory work of that period did not address learning, problem solving, or errors, and it had no means of describing the dynamics of thinking and acting in workplace contexts... The echoes of this dismissal of theory are evident still in HCI methods, many of which are grounded in raw experience and convention, not in theories or frameworks".* Carroll further observes that many issues discussed in the 1980's debates, about science and theory in HCI, are fundamental: *"What is the proper balance between rigor and relevance? A model of errorless performance can be more rigorous, but it is also less relevant to real situations. What are the boundary conditions on applicability*

for given theories?…What roles can models and theories play in invention,
development, and evaluation of new technology; <u>can there be theory-based design?</u>
Studies of technology development in other fields suggest that <u>science is more often</u>
<u>the beneficiary than the benefactor of technology</u>.” (Underlining is not in the
original; it signifies the writer's emphasis).

In this thesis, interaction and user interface design activity is used as a path for
reasoning, proposing and validating the approach for End User Development in
Ubicomp applications and further reflecting on the aspects of it, and the possible
functionality involved.

The further aspects of Design Rationale (and, in particular, Scenario Based
Development) as an methodological framework applicable to interaction design
investigation in AmI-EUD, in accordance with the arguments of John Carroll
(Carroll, 2003) (Carroll, 2000), follow in the next sections.

3.4.   On Task Analysis vs Design Rationale

Carroll (Carroll, 2002) argues that the choice of scenario-based design versus
traditional task-analytic methods reflect the differences in the underlying world
views and values of designers and researchers. He points out that task analysis (seen
as structured task decomposition) might be integrated as a part in the emerging
paradigm of scenario-based design.

Traditional task analysis assumes the goal of optimal performance. Carroll states
that overstressing this goal leads to the pursuit of task analysis to a point where
benefits are outweighed by risks. He argues (Carroll, 2002) in favor of ways for task
analyses to be made richer and sketchier, and they are seen from the perspective that
they are no more than tools for inquiry. Carroll argues that designers need
representations of work activity in order to engage with the design problem

intimately, to visualize people and their activities. Designers need media and sketches to generate and explore design proposals, to discuss and refine them (Gedenryd, 1998). <u>Design representations need to go beyond 'correct and complete' description.</u> (Carroll, 2002).

In design activities, designers have to prioritize and choose. Thus the different so called 'design methods' are different to methods such as task analysis because they pursue different goals. Scenario based design (as articulated in Carroll's book "Making Use") contrasts with traditional task analysis, because the latter assumes that there is a correct and complete symbolic description of user tasks, and attempts to capture it (Carroll, 2000), (Carroll, 2002). In traditional task analysis tasks are seen as hierarchical structures of operations that are systematically decomposed to a very detailed level. The representation of work activity is the main objective, while informing the design so as to optimize performance is a side benefit that is not always achieved.

Scenario-based design on the other hand has a different goal; it tries to facilitate the emergence of new designs from richer descriptions, participatory methods, and observation of people's habits and practices. The emphasis is on making the issues visible so that meaningful design work can be achieved.

Carroll and Rosson argue for the adoption of Design Rationale as a theoretical foundation, and scenario based design as a subsequent methodological framework, because they are stemming directly from the design discipline, but are also providing a broader scope that can encompass not only design for interaction, but also more analytical engineering methods (such as traditional task analysis), (Carroll, 2002).

Carroll points out that we should question implicit assumptions of absolute criteria when discussing design. The identification of opportunities in a design project – be they exploration, lucidity, continuity, or respect for nuances- are complex cognitive

objectives that cannot be diminished to the optimization of tasks output. Such design objectives are formed by design interventions; therefore they have to be considered in the iterative context of design prototyping and refinement (design, deployment, adoption, redesign). Scenario Based design is seen as an appropriate methodology, since, *"scenario-based design does not merely provide task analysis input to design, it is design."* (Carroll, 2002)

HCI methods widely incorporate scenario-based design as a standard method. Scenarios are a pervasive design representation (Carroll, 2002) because:

"(1) Scenarios are concrete in the sense that they are experienced as low-fidelity simulations of real activity. (2) They are also flexible in the sense that they are easily created, elaborated, and even discarded. And (3) Scenarios keep design discussion focused on user activities, more specifically, (4) they keep design discussion focused on the level of task organization that actors experience in their basic tasks. ... (5) This makes it easier for all stakeholders in a design, including end-users, to participate fully, and (6) creates a focal, use-oriented design representation that can be reused throughout the system development process for constructing prototypes and mockups, requirements analysis, use cases and software object models, user interface metaphors, design rationales, usability specifications, formative and summative evaluation test tasks, task oriented training, help and other documentation, etc."

Human–computer interaction design gives high priority to innovation, and this is an increasing trend. The search for innovation involves iterative prototyping from the start, formative evaluations, rapid development, software tools, interdisciplinary teams and participatory methods. These are supported by attributes of scenario based design rather than by traditional task analysis. Moreover the scenarios employed in software design and development are not as rich as those used for requirements analysis. Again, in Carroll's words (Carroll, 2002): "*Scenario based design is more than a matter of task analysis. It involves understanding what people already do,*

how they do it, how they experience it, how they learn it, how they pass it along to others. It involves envisioning possibilities that have not yet been dreamt of—new ways to do familiar things, entirely new things to do. .... It involves walkthroughs, design rationale, many varieties of formative and summative usability evaluation, and installation and maintenance".

Carroll's book *Making Use* addresses the use of scenarios for envisioning future systems, identifying and analyzing requirements, developing usability rationale, and designing usability evaluations. It does not present a single, comprehensive scenario based method, but discusses aspects of scenario based approaches. The collaborative volume edited by Rosson and Carroll's on *Scenario based methods* can be used as a related framework for HCI.

3.5. Design Rationale as theoretical foundation

Design rationale is the process of documentation and analysis of specific designs that are created: (Carroll, 2003, chapter 15). The overall design, its specific features, technological selections, and hypothesized intended use of these features, are described. Hypothetical user interaction scenarios are created and observed, regarding their advantages, tradeoffs, and shortcomings. The taxonomy of features, tradeoffs, and technology solutions, are described in Design rationale, but it also provides assumptions for human behavior and explanations of that behaviour. In the process of interface and interaction design, many of these decisions may happen internally, in the mind of the designer (those that relate to the minor interaction elements, such as boxes, and scrollbars, etc.), while the major decisions (regarding interaction style, technology platform, etc.) are often articulated in reports.

Carroll (Carroll, 2003, chapter 15) mentions that design rationale includes thinking about design specifics and features and their relationship with user actions and needs, and the environment and context of use. Often, the design rationale is

informal, based in the designer's intuitive theories of human behavior; sometimes it may have a more scientific grounding in anthropology, sociology, psychology. Carroll in his book (Carroll, 2003) considers design rationale as an effective HCI theoretical framework, as it helps get an insight into new methods of effective interaction, it can guide the development of tools and environments, and it can propose new measuring methods for interaction. According to him design rationale addresses three things: content, applicability and scope. According to him, design rationale addresses three things, roughly corresponding to the approach's scientific foundations: context, based in ecological science; applicability in action science, and scope in synthetic science. As the system design is being developed, the design rationale is evaluated and refined (action science). When it is generalized it contributes to theory building (ecological science). If design rationale is generalized and justified in established scientific theoretical foundations, it serves as an integrated framework for understanding and exploring further the problem area (synthetic science).

As an approach to theory in HCI, design rationale is in the middle, between approaches that apply theories from outside the HCI domain and approaches that dismiss theory as irrelevant and restrictive (and therefore value heuristic usability engineering). Design rationale focuses on describing the domain of application in terms of design tradeoffs, but it uses theory as necessary to justify those (Carroll, 2003).

Applying Design Rationale: Scenario Based Design method

In this thesis, text Scenarios are used along with UIs and diagrams for investigating models or possible features, the image or prototype being used in a similar way that a verbal interaction scenario is used for enabling discussion, in the Scenario Based design method.

ANALYSIS

Analysis of
stakeholders,
field studies

Problem scenarios

Claims about
current practice

DESIGN

Activity scenarios

Information
Design Scenarios

Interaction Design
Scenarios

Metaphors,
information
technology, HCI
theory, guidelines

Iterative Analysis
of usability
claims & redesign

PROTOTYPE & EVALUATION

Summative
evaluation

Usability specifications

Formative
evaluation

**Figure 6: Overview of the Scenario-Based Framework (Image source: Image from
http://ldt.stanford.edu/~gimiller/Scenario-Based/scenarioIndex2.htm)**

Different disciplines within HCI approach Scenario Based Development differently, focusing on different goals of the process. The scenario types they use vary in scope. Go and Carroll (Go and Carroll, 2004) identify several communities that work with the Scenario Based Approach (Figure 7):

1- Strategic Planning: using abstracted artifacts and future hypothetical situations, involving to some degree current technology and trends in the assumptions.

2- Human computer interaction. It adopts work oriented approaches, focusing on the user and the broader context of user tasks. These approaches typically involve 'day in the life' scenarios of use, involving user tasks, and their degree of tangibility is much lower than software engineering approaches.

3- Software Engineering: scenarios here are seen from the systems viewpoint, aiming to identify requirements, or to specify a model of the system. Scenarios

used in the analysis and design of software Systems are based on real world artifacts, they are 'smaller in scope' scenarios, typically keystroke-command level scenarios. Use case models, consisting of actors and use cases, are using scenarios for the design and development of software systems.

Scenarios can range between several categories: (Go and Carroll 2004, pp.45-55), such as the following:

- Problem Scenarios tell a story of current practice, because it emphasizes on describing activities in the problem domain
- Activity scenarios describe concrete ideas about new functionality, new ways of thinking about users, their needs and how to meet them.
- Information and interaction design scenarios specify task objects and actions that help users understand and act on the proposed System. The goal is to specify functions and mechanisms for manipulating the task information and activities.

A scenario is a description that includes (Go and Carroll, 2004) 1) background information on a situation and assumptions about the environment of use 2) actors (people, or other) and 3) their goals, 4) sequences of actions or events. Scenarios are expressed in various media and forms. They can be text narratives, graphics, images, storyboards, video mockups, or scripted prototypes. In addition, they may be in formal, semi-formal, or informal notation. (Ronnald et al, 1996) identify four dimensions: the form view, the contents view, the purpose view, and the lifecycle view.

Carroll points out that scenarios are not specifications, nor should they be considered as specifications. They are fragmented, open ended and focus on selected instances, they are informal and sketchy; these characteristics contrast with the formal, specific, and abstract technology descriptions that characterize System Requirements.

In this research we use scenarios from the Interaction Design perspective (as described at 2 above), using narrative as well as form view (Ronnald et al, 1996) in an informal manner. The aim is to get insight into the aspects involved in the ubiquitous system, and the functional and non-functional requirements and broader context of use that EUD for AMI may entail, and to propose and evaluate possible models for such systems. In the iterative development process, that is typically followed in interaction design, our research work is specifically located in in the 'concept' and 'early analysis' phases, using an iterative cycle of concepts, designs, prototypes and evaluation, not to propose a conclusive system, but specifically to provide via experimentation the insights and paradigms needed as input to research that further proceeds with the design and development of such EUD for AMI systems.

COMMUNITY	UNCERTAIN FACTOR	GOAL OF SCENARIO-BASED APPROACH	DEVELOPMENT PROCESS	VIEW POINT	BACKGROUND GOAL OF COMMUNITY
Strategic Planning	Environment	List "what-if" questions and their answers	Iterative	Organization Technological changes Economics Social, political regulations Consumer attitudes	Plan a course of actions
Human-Computer Interaction	Use of system	Envision user requirements of (future) system use	Iterative Prototyping	Human Usability Cognition Emotion	Describe use of (future) systems Design usable computer system
Requirements Engineering	System requirements Functionality	Acquire user requirements and specify them	Waterfall Spiral	System architecture Development process	Specify systems Provide a good transition to the next development phase
Object-Oriented Analysis/ Design	Objects Data structures Class hierarchy	Identify objects, data structures and model class hierarchy	Iterative Incremental	System Object	Design a model of world

COMMUNITY	SCENARIO USAGE
Strategic Planning	Envisioning uncertain future environment Providing communication tool Organizational learning Sharing a mental model among stakeholders
Human-Computer Interaction	Analyzing user tasks Envisioning future work Mock up and prototyping Evaluating the constructed system Deriving learning materials Developing design rationale
Requirements Engineering	Eliciting user requirements Deriving specifications Analyzing the current system usage Describing the current system usage Constructing test cases
Object-Oriented Analysis/ Design	Modeling objects, data structures and class hierarchy Analyzing problem domain Providing a model of real-world objects

Figure 7: The communities using scenarios (above) and the factors that can be used to categorize scenario design, and typical scenario usage in design (below) (Source: Go Kentraro, Carroll John, Interactions, November-December 2004, pp45-55).

Strategic Planning

Vision scenarios: "year in the life" examples

Human Computer Interaction

Outline of use scenarios – "Day in the life" focus

Experimentation, Understanding of design problem space and possible solutions Developing design rationale, experimenting on ideas, mock ups and prototypes, providing input to system requirement.

Requirements Engineering

Requirements elicitation, analyzing system and its elements, description of system usage, specifications, prototypical development cases and test cases

Software Engineering

Detailed commands and system tasks

Figure 8: Scenario forms vary in breadth of focus and detail, from the broader scenarios used by strategists, to more narrow scenarios of software engineering.
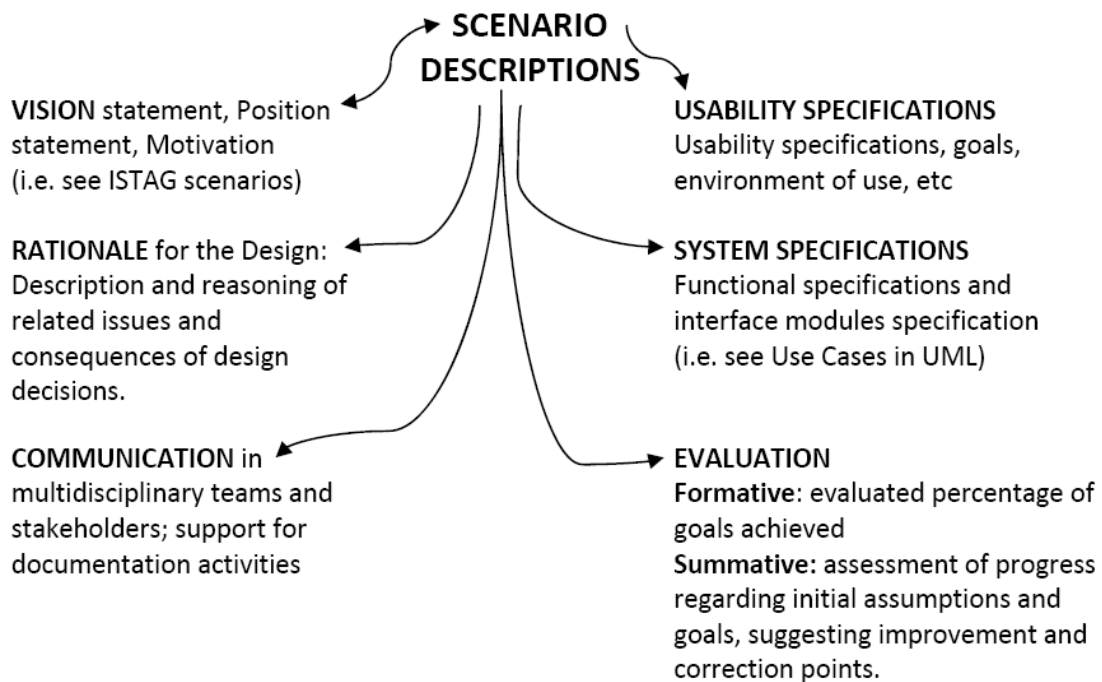
Figure 9: Scenarios provide a common language for design

There is an evolutionary aspect in design as it may introduce new concepts or extend established ones. Design is a reaction to a perceived problem description. A design is a response to certain selected concerns (that are prioritized, often informally) and it often acts as a compromise solution: some elements of the problem situation, after the design artifact is introduced (design intervention), are made easier, others more difficult, and still others obsolete and unnecessary. At the same time carrying out certain tasks creates opportunities for devising new artifacts; the new artifacts that are introduced eventually alter the original tasks and create new needs and further design occurrences. This is the task-artifact cycle.

Imagining system or problem
space, introducing new
artifacts as solutions to the
assumptions/issues

Assessment of limitations
and scope; Pros and Cons of
the solution

Rationale analyzing how new
proposed features enable
new situations;
consequences of the
decitions, pros and cons.

Specific Scenarios of Use for
detailed Task Analysis
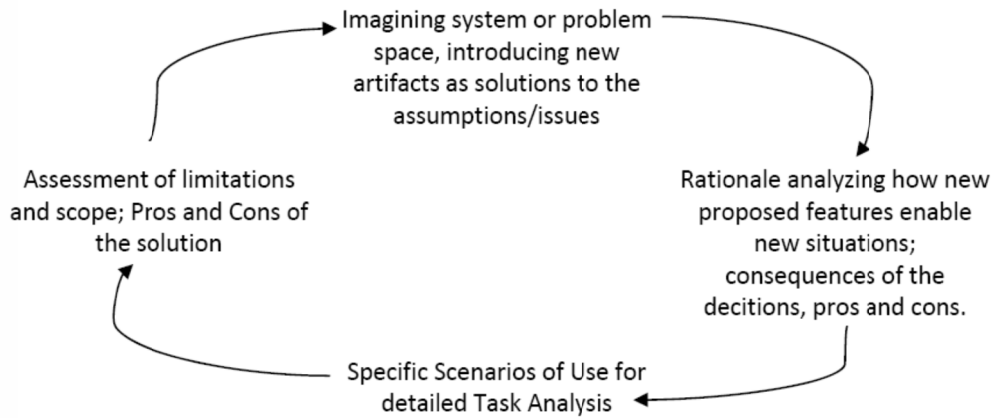
Figure 10: According to Carroll [(Carroll, 2003) page 435]: In the task-artifact cycle, human ideas and activities raise technology requirements and new technology subsequently raises new opportunities for human action. The interrelated flow is emphasized in the task-artifact cycle above.

3.6. Evaluation approach

User Centered Design as an approach involves end users to the development of the system, by iterative cycles of evaluation that feed back into design. Interaction Design and HCI follow User Centered Design methods, and the Iterative design process. Four essential activities are identified (ISO13407) in user-centered design projects:

a) Requirements gathering (understanding and specifying the context of use)
b) Requirements specification (specifying the user, the environment of use and organizational requirements).
c) Design (producing designs, UIs and prototypes).
d) Evaluation (carrying out assessment of the proposed designs by end users or subject experts).

The evaluation techniques that are used, sometimes in combination, for requirements gathering and concept validation in this research are:

- Focus Group Evaluation, because it provides qualitative data, has low implementation cost, and requires a low sample size of subjects.
- Questionnaires in Focus Group evaluations were based on the Cognitive Dimensions framework (which will be described in further sections)
- Short usability trials by experts (in the course of conferences) followed by short questionnaires.
- Usability testing in small end-user groups, followed by interviews. This method is used for evaluation of concepts, the drawback being that it has a higher implementation cost

Focus groups are very often used as an input to design concept phases. They generally produce non-statistical data and are a good means of getting information about a domain. An experienced (preferably external and neutral) moderator is needed for the process to be effective, as well as a carefully selected small team of experts. The process involves encouraging an invited group of people that have knowledge on the certain area that is being evaluated to share their thoughts, feelings, attitudes and ideas on a defined topic.

The Cognitive Dimensions (CD) framework was used in this research, (a non-analytical method used at the early concept phase, aimed at focus groups) as it provides a set of discussion tools to be used by designers and evaluators of designs, for qualitative feedback. CD enables the discussion by providing a shared set of terms. The aim of its use is to improve the quality of discussion, as experts make sophisticated comments and suggestions about systems.

Usability testing can be used as an input to design or at then more completed stages of prototypes, or at the end of a project. It is a prominent method to find out what are the most likely conceptual / usability problems. In usability testing sessions data is

collected in order to evaluate a system. People are carefully selected (as representatives of the targeted profile of end users, and so that they are unbiased, and neutral to the system under evaluation) and are invited to attend a session in which they are asked to perform a planned series of tasks. A moderator takes note of any difficulties they encounter. Users are often asked to follow the think-aloud protocol which asks them to verbalize what they're doing and why they're doing it. The evaluator can note their difficulties based on their discussion and also how long it takes them to complete the given tasks, which is a good measure of efficiency. Two specialists' are normally required per session - one to moderate, one to note problems. Some form of design has to be available to test: this can range from paper to a partly functional prototype. Non-statistical or statistical data can be generated from that method. In our case we used this method tor gain qualitative feedback, gathering non-statistical data based on a particular design and using the think aloud protocol. The evaluation was conducted by an external moderator to ensure neutrality and was aided by the author.

Interviews are usually employed early in the design process in order to gain a more detailed understanding of a domain/area of activity or specific requirements. They usually involve the interviewer speaking directly to the participants (to only one or two participants usually). A participant's unique point of view can be explored in detail. Any misunderstandings or clarifications can be identified. The output of an interview is of a qualitative nature and reports of the interviews are then carefully analyzed by experienced practitioners, in order to produce valid results. The experience and skill of the interviewer and analyst is of utmost importance on applying this method successfully.

Questionnaires are a means of asking users for their responses to a pre-defined set of questions and are a good way of generating insight or data. Questionnaires are usually employed when a design team looks for a larger sample size than can be realistically achieved through direct contact. Questionnaires allow statistical analysis of results, which can increase a study's credibility through its scientific appearance.

This makes it all the more important that the questionnaire is well-designed and it is of the utmost importance that it asks non-biased questions.

In all the evaluation methods it is best to have evaluators who are neutral to the proposed design or concepts. In our case the difficulty was that they needed to be sufficiently informed about the system in order to make useful reports, but not deeply involved in the design itself.

The exact application of these methods will be further described in subsequent chapters.

A drawback of evaluation methods is that they can only give generic responses - some of which may seem profound or trivial- and they do not directly suggest new features or preferred taxonomies of features or tasks. Evaluation is reactive: it assesses a system that is proposed and manifested in a given form, but it is not a design activity. The form of presentation and the usability may themselves get in the way of assessing the key concepts and features of a conception of a future system. Nonetheless, evaluation can give valuable general insight into such systems as those we are investigating, thus guiding the design towards an iterative redesign process resulting in a new or different proposal which includes (or omits) extra features and additional system specifications.

Participatory design can be a useful approach when selected participants are able to conceptualize a technological future. It was not selected as a method because of the outlook of our research which is this: that people in such environments will evolve with technology to have very different skills and requirements in 10-15 years than the current generation does. This is already the case with the digital emancipation of the web generation: the use of the internet and social/awareness media, as well as a more general technology generation gap to which we can already bear witness. For this reason a focus group was a better match to the profile of our research project.

4

4. From objects to artifacts

4.1. Introduction

Objects are enhanced with new digital properties and information-communication capabilities, gradually becoming computationally augmented artifacts. In this chapter an approach towards defining artifacts is introduced, and their potential influence in people's lifestyle is addressed.

Although ubiquitous computing is not a new notion (Weiser, 1993), in the last few years there has been growing interest in the technological as well as human aspects of research in the area of the Ubiquitous Computing - in Europe most notably through two European funded FET initiatives both named: 'Disappearing Computer' (Disappearing Computer Initiative website). As a consequence of this type of research, miniaturized, 'disappearing' computers are being embedded into various kinds of artifacts; thus artifacts become enhanced with new digital properties and information-communication capabilities.

The perspective assumed in this research pertains ordinary objects of everyday (non-computing) usage, (such as furniture, utility or decorative objects), that traditionally had no computing capabilities build into them and are thus not regarded as typical information appliances. The view presented here on disappearing computer artifacts stretches to involve interfaces that differ from information appliances with centralized processing power (such as mobile phones, Personal Digital Assistants (PDAs) etc). Rather, in these other applications, processing power is distributed: the function the user aims for when using the artifact is no longer achieved by a single processing application but by a collection of tangible objects. These enhanced objects may vary in shape, materials and capabilities, but they are able to communicate with each other through an invisible network, and share the processing capabilities they may individually have. Collective functionality emerges within a group of such artifacts that can work synergistically together in an environment, through these invisible links. Such artifacts are of a dual nature: a tangible and a software part, interconnected. In addition to processing, they may also be enhanced with sensing or actuating capabilities of their own.

We hereafter describe an approach towards defining augmented artifacts (hereon mentioned as 'artifacts'), and examine to what extent their advance can affect people's existing lifestyle.

4.2. Adding Information Technology into objects

The main idea behind the 'ubiquitous computing' or 'ambient computing' or 'disappearing computer' concept is that the computer 'disappears' as an object (Weiser, 1998) and computing services are made available to users throughout their physical environment (Markopoulos, 2001), be it the office, the home, the street, etc. More and more applications are designed, realized, and presented in research context which involve dispersed objects that work together to create a unified experience. But are there *any* proposed technologies that support designers and people, which enable them to create applications for ubiquitous computing, without always have to start from scratch? Are these technologies flexible enough to allow design with sustainability of augmented artifacts regarding their use over time?

The view of disappearing computer artifacts stretches to involve objects that may be varied in shape, materials, capabilities, but are able to communicate with each other through an invisible network, and share the processing capabilities they may individually have. The purpose of use is no longer achieved by a single processing appliance, but by a collection of tangible objects. Collective functionality emerges within a group of such artifacts that can work together through invisible links. Such artifacts are augmented with processing and communication abilities and can also have sensing or actuating capabilities, along with their physical, tangible characteristics.

Disposable objects (such as shopping bags, milk bottles), items that we surround ourselves with (such as furniture, decorative items, lights, carpets, pots), or even architectural elements in buildings or public spaces (floors, walls, rooms, buildings, squares), can be enhanced with processing capabilities, communication modules and sensors. This may eventually become a part of the manufacturing procedures. Processing artifacts will be capable of interlinking with other artifacts in various possible associations. Several questions arise, assuming that such a world is possible. First of all, do we need such artifacts? Then, how do they present themselves? What is the role of people? How can we build such environments

without overwhelming people? Research should put a great deal of consideration into the development of technologies appropriate to enabling people rather than superseding them.

4.3. People shuffle objects' usage

People buy objects and make their surroundings from them, arranging them and rearranging them as fits their needs. Human environments are 'object-scapes', spaces that are constituted from collections of objects, positioned in ways that facilitate or express the life of their owners[2]. (Rodden, 2004), (Alexander 1964), (Alexander 1977).

Artifacts are a subset of information appliances, where the term 'information appliance' is used in the broader sense of the word (Sharpe, 2003). Artifacts are ordinary objects that are commonly used for routine tasks (objects such as tables, chairs, cups, shelves, lights, carpets, etc.), and which in the future can be enhanced with communication, processing and possibly sensing abilities. People buy furniture, as well as all sorts of other things, but then define their own home environment and often rearrange the objects over time or change their use. We surround ourselves with objects, for aesthetic, functional, playful or emotional reasons. We adapt and change the use of these objects over time. We may rearrange our surrounding objects at times, when we move into a different environment, or the family's needs

---

[2] Christopher Alexander (Alexander 1964) discusses a design process where designed items are shaped gradually and continually over time, so as to fit the evolving and changing context of use; people participate in this 'unselfconscious design process' fluidly recognizing a failure in their settings and taking corrective action to make the form of their surrounding fit better to their changing situations. Continuous adaptation is seen as *piecemeal building* (Alexander 1977).

are changing (i.e. when a baby is born). Reusability -adopting a new function-, is a key issue for the sustainability of objects. We should equally be able to sustain and reuse in different context the augmented objects of the ubiquitous environment too.

A tangible object can have functions, depending on what it can 'afford' - from its tangible shape and capabilities. Although it is mostly used for one purpose at a time, over time it may change its' use. A cup with a broken handle, for example, is then transformed into a toothbrush or pencil holder. People are flexible and inventive when it comes to using objects to fill their needs. Purposes of use may change over time, providing the object's physical properties are not violated, and allow for these new purposes. The arrangement of objects, their location and clustering with other objects, make for different purposes of use at any one time. The shape of common objects has been adapted over an extended period of social history precisely to enable them to meet different purposes in changing contexts of and based on the objects' affordances.

Let us consider the example of a cup. Putting aside attempts to draw conclusions beyond the object's physical characteristics, a cup can be categorized as a container object, and can be used to contain things, whether this is liquid, toothbrushes, sea-pebbles, pencils, flowers, or alternatively someone can think of different uses for it such as to trap a spider with it, or to use it as a paperweight. Depending on the properties of the material it is made of, the capabilities the exact shape offers, as well as other objects/substances it interrelates with, this container object may be used to drink from, to pour with, to carry liquid with or to keep things in. The cup is initially mostly located in the kitchen but can also be mobile within the home, and it is linked in the context and process of use with the water boiler or coffee machine. Later on, when its use becomes that of a pencil holder, it becomes more static, on a desk, and contains a certain set of pencils and pens. Similarly, a chair can be used as a coat hanger, a seat, a ladder, or a bedside table.

A table for example, is a surface raised to an appropriate height, which in some cultures is used to put things on, which can then be handled while sitting in front of the table. There are more specialized uses of a table, e.g. a table that is used to facilitate study is a 'desk', a table to eat on is a 'dining table'; a shorter and smaller version may be a 'coffee table', while there may also be co-existence of the many purposes of use in one instantiation of a 'table'.

It is in human nature to convivialize an object and use it in other than predetermined ways, as long as the physical properties of it allow for that. Although things should be designed to serve a purpose and serve it well they should also be allowed to deviate from it when this is required. People may also change objects intentionally, 'pimping', 'hacking' or 'cannibalizing' them to achieve the 'improper' repurposing of objects. Because of the creativity of people and their effort towards the reuse and sustainability of some objects deviations of use happen unconsidered. If future objects had not only physical properties, but digital capabilities, (such as sensors, processing, communication, etc), the supporting technologies should allow for this normal human behavior to still be able to happen.

Figure 11: Familiar artifacts are enhanced with sensing, processing and communication capabilities.

The notion of the appliance as defined by Bill Sharpe (Sharpe, 2003) is 'a device of specialized and widespread use; a device that does one specific thing to information

of a certain type'. Based on this notion, an *'artifact'*, as defined and considered in this research, can be considered as a subset of appliances. Artifacts originate from an evolution of common everyday objects, and therefore are in widespread use. Artifacts' material shapes coupled with additional digital capabilities enable them to be used collectively in order to achieve a superset of functions. Therefore artifacts are capable of doing a *range* of 'specific things to information of certain type'. Artifacts express their capabilities to other objects through their digital aspects; artifacts can synergize with each other and share their (processing, sensing, actuating or other) capabilities within a communal pool. The observations regarding the use of 'normal' (un-enhanced) objects, mentioned in previous paragraphs, also apply to the use of Artifacts; in fact variations of use can be taken to a much greater extend because of the Artifact's inherent connectivity and collectivity in behavior.

4.4. Towards open, flexible, collaborative systems

The use of material objects does not remain static over time. Neither does the use of more solid structures, such as buildings and architectural constructions (Rodden, 2004). Human environments are 'object-scapes', spaces that are constituted from collections of objects, positioned in ways that facilitate or express the life of their owners.

People buy objects and make their surroundings from them, arranging them as fits their needs. They buy furniture, decorative, play or utility artifacts from shops, but then they take them and make up with them their own home setting as they want; they rearrange the objects at times, according to necessity or aesthetics (e.g. if they move into a different situation, or when the family's needs are changing, e.g. when there is a new family member). As Tom Rodden has pointed out (Rodden, 2004), people do that even in the construction of their houses. Houses are perceived as non-changing, solid architectures; yet, over time, new plumbing is installed, walls are demolished and new ones are built, kitchens are redone, ironwork is changed, the

whole look and feel and functionality of the building may change. All things and tools people use and maintain over time, they also adapt and change. And we should remain able to continue doing this deviation and alteration in our forthcoming, ICT augmented environments.

Observing these natural changes in uses of normal objects it becomes evident that we should allow for artifacts to be used creatively by people, let alone by designers. This could prove crucial for the successful adoption of artifacts: because of the inherent connectivity of artifacts additional 'connected' functions can be supported that are tailored better to their owner's use (Markopoulos, 2001). Returning to the example of the augmented cup; it could be used by someone for drinking, as a primary function, but also as a part of a greater system of objects that is assigned to do health monitoring of the owners. Alternatively, for someone else it could be part of a different grouping of objects (figure 12) that facilitates ordinary tasks; e.g. when the level of tea drops, it may be so assigned for the kettle to switch on to boil water for more tea. When, over time, the augmented cup purpose is deviated to that of a pencil-holder that in turn should be able to identify missing pens, or faulty ones, and play a different role in the everyday life of the household.
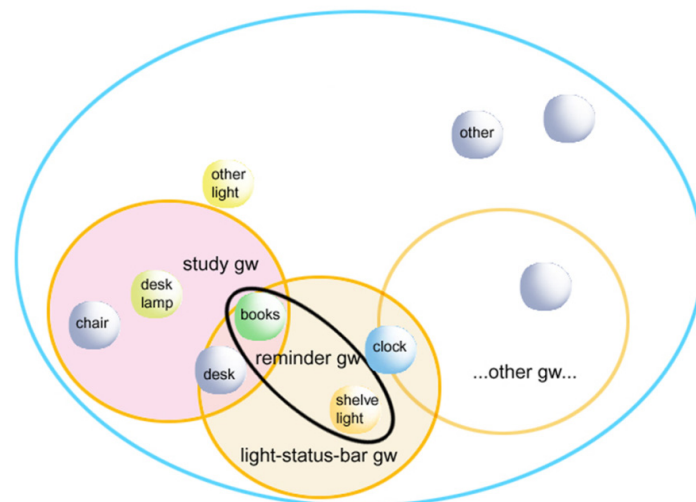
Figure 12: An artifact can simultaneously participate in different application clusters; each of those functional clusters can serve a different user or a different goal

Several design researchers (among others, Dunn and Raby, Bill Gaver, Philips Design) have argued that people should be the ones to add interpretation and meaning to designs that are intentionally left open for imagination and experimentation (Gaver, 2002). People are the interpreters of open scenarios of technological objects (Dunne and Raby, 2002), while designers can allow for these interpretations to occur by providing intentionally open-ended designs (Green, 2007, pp46-48). From a constructivist engineering perspective, adding to the above approach, artifacts can be treated as the building blocks that enable and inspire unique individual applications to be created. Provided that flexible infrastructures are supported the use of these artifacts, at the same time, on the one hand, niche applications can be created by and shared between people, while on the other, experience designers could come up with other types of application, targeting certain user groups and needs. But before we can get to this point, we need the appropriate concepts and the enabling tools that end users and designers can use, and these in turn need to be based on open approaches from the level of the technological infrastructure.

4.5. Usage and interface issues

Each object in our everyday world has been designed with certain tasks in mind - and often its design has been developed and adapted over the years to best suit for these tasks. The ways that we can use an ordinary object (implied by the 'object's affordances') are a direct consequence of the anticipated uses that object designers 'shape into' the object's physical characteristics. The objects have been designed to be suitable for certain tasks, but it is also their physical properties that constrain the tasks people use them for or 'afford' their different uses. As everyday objects are "enhanced" with computing and communication capability, the user will need to learn the new ways in which they can be used (which can be indicated by designing new affordances) and the collective tasks they can participate in. This ubiquitous computing paradigm introduces several challenges for human-computer interaction.

Firstly, users will have to update their task models, as they will no longer be interacting with an ordinary but with a computationally enabled object. Secondly, people will have to form new conceptual models about the everyday objects they use, and thus they may change their habits.

As computing becomes embedded in everyday objects, so the Human Computer Interface is now engaging with the physical world as well as its digital representations. In such a world, the 'direct manipulation paradigm' will have to include metaphors describing interaction with tangible objects. Unlike some parts of Weiser's vision (Weiser, 1993), it may not be appropriate to the nature of many artifacts to have screens added to them. Such an interface approach applies to the more specific category of information appliances, and although convenient for interaction it does not fit the nature of all objects and augmented environments (therefore new HCI issues occur regarding the loci and nature of feedback). The design of the object's form and physical properties affects the interaction with it. In fact the design of objects, which constitutes their interface, may have to be reconsidered so that their new capabilities can be promoted to the user (indicated by appropriate elements for the nature of each object). In this broad picture, information appliances as we know them are only a subset of these objects. Information appliances are often screen dominated, but this is not so for artifacts. Most artifacts cannot have the on-screen (or audio) feedback that is defining interaction with many information appliances. This will be a challenge for designers. It will require a broader approach, whereby the tangible interface of the object not only provides for an optimal user-experience, but is also assuming the role of the interface to a larger set of interconnected causes and effects, - while not all of the effects that are caused are visible to the end users.

4.6. Meta-issues of use

People can, then, act upon their environments, be they physical or enhanced ubiquitous computing environments, by setting goals, forming plans and perceiving results. At the cognitive level, the disappearance of the computer forces people to form new mental models about those tasks which involve objects and environments (that now may use hidden ICT capabilities). On the other hand, if the appearance and function of everyday objects and environments change (or new objects appear into our everyday life), then people will have to adapt or form new models of tasks involving these objects (Kameas & Mavrommati, 2001).

Most objects around us have been designed for specific tasks; but this specificity constrains the ways in which we might use them for. In general, everyday objects can be used in different ways, providing that their physical properties are not violated. As everyday objects are 'enhanced' with sensing, computing and communication capabilities, in order to become artifacts, people have to learn the possible new ways that they can be used (that may have to be indicated by appropriate new affordances) and the tasks these objects might participate in. People may initially have to use objects in more complex ways, as they may end up interacting at the same time with individual objects and with their configured collections.

Living with and using artifacts, may not seem easy at first, and may require certain new skills to be developed (including abstractions and models to reason about them). Nevertheless, it may be the case, as it is for example with writing, that once the skill - however complex it may be - is acquired, over time, it comes to feel natural, easy and transparent in use (Sharpe, 2003). The case of writing is highly relevant: it is a complex skill, it involves understanding the abstraction of sounds into written symbols, recognizing these symbols, but also reproducing them, without consciously thinking about the process of doing so. The process of people growing into the skill-set needed for using artifacts and their associations can be viewed in the same light as acquiring the mental and kinetic skills for writing, and its relation

to oral expression. In contrast, people are acculturated to recognize and to use the tangible part of the artifact as is, because this knowledge is embedded in their socio-cultural history.

The introduction of artifacts can be expected to affect people's everyday lives in several ways. People will have to change established habits, learn new skills, and form new mental models for the objects and spaces that surround them. Task models will need to be updated, as people will start interacting with artifacts (that are capable of participating in many more new and complex tasks) rather than the accustomed ones. The conceptual models people have of objects and of computing will have to evolve in order for them to utilize the new possibilities offered by the computationally enhanced Artifacts.

There are several issues to be investigated at this level, which can be considered as the 'syntactic' level:

- Which is the set of people's actions that artifacts should respond to? How will these be expressed in objects' capabilities?

- How will people perceive artifacts? What patterns of usage can emerge from adopting and adapting artefacts usage?

- How can augmented artifacts be designed so as to be introduced to people's lifestyles and not to contradict their existing behavior patterns and conceptual models? Is there a general architecture upon which artifacts could be based?

Possible advantages and benefits

Why should all the trouble be taken, of adopting artifacts, coping with augmented environments and acquiring new skills? To do it, people have to see clear benefits from it. At the early stages where the ordinary user is just beginning to understand how to use and what to do with ubiquitous computing technologies, several

applications used as research validation scenarios can be criticized as being uninteresting, of no apparent value or importance, or understating the potential of ambient technologies. In contrast, some factors that can be motivating to using artifacts are the following:

- New tasks and new enhanced services become possible.

- Faster services, savings in effort and time in doing complex ordinary tasks, can be achieved. Everyday life tasks take a lot of effort; this can be seamlessly facilitated.

- Particular and special needs of targeted groups can be better met. (For example, fitting the individual needs of young children, the elderly, disabled people, or other groups where the applications have clear and strong benefits).

- Unpredicted niche applications can be made possible, by empowering end-users to act according to their needs and wishes.

People can initially ignore the difference between objects and Artifacts. They may begin using artifacts according to their existing habits, gradually growing into full use of the added digital capabilities. People may choose to use artifacts after a time of initial apprenticeship. In this ways, the intrusion of the artifact in existing task models will be perceived as small.

We must assume that, as it is often the case with fundamental technology advances (Norman, 1999), once we get more accustomed to the new medium, people will come up with applications and uses that cannot be presently anticipated. Later generations of applications will be nothing close to what we can currently capable of imagining.

4.7. Conclusions

Although tangible artifacts should be designed to serve a purpose and serve it well, they should also be allowed to deviate or evolve away from this purpose when that is required. People often put an effort into continuing to use objects even after they become antique or have deteriorated (such sustainability is often achieved by alteration of use). People can be very creative with the way they use things. So far this has happened in a rather natural way, as it is in human nature to use things in other than predetermined ways, as long as the physical properties of objects can afford this. If future objects are to have not only physical properties, but ICT capabilities too, their designs should still allow for this normal human behavior.

Artifacts have the potential to gain their own place in our future everyday life. Designers need to aim for a smooth transition in order for people to adopt artifacts and embrace their benefits. To be effective in this transition we need to develop or adapt visions of how (even the more mundane of) tangible objects may evolve in the future, and what role people may have in this landscape. Such visions will support people in creating their own ambient experience applications they want, or changing of the pre-constructed applications they are given.

Research has to consider approaches for augmenting artifacts that are scalable. Approaches are needed that cover in their conceptual framework a wide range of objects of different kinds (from a tag to a desk to a house, from service carriers such as heating, TV, light, audio, to tangible furniture, flowerpots, clothes, carpets, boxes, etc). We need to consider and develop referents between designers, people and artifacts, giving bridging technological solutions that allow for reuse and sustainability.

5

5. Ubiquitous computing and approaches to augmenting artifacts

5.1. Introduction

One way to explain the method employed in Artifacts' creation is that artifacts are created by giving to objects the ability to be associated with each other (having invisible links to each other, and affect each other's behavior). This, in turn, translates to different methods for embedding the hardware, software and design aspects that are added to objects, so as to include, make visible, manage and utilize these links. This not only involves solutions for embedding processors, sensors and communication modules to objects, but also suggests appropriate mental models and relevant tangible-interaction languages, on how these objects can be used, and well as redesigning existing objects to indicate their new affordances.

Companies such as Philips (Aarts and Marzano, 2001) promote an ambitious vision for the private domain with In-Home networks and intelligence, while at the same time simpler scenarios using more established technologies such as RFID (radio frequency identification) tags are already being implemented for the home, mobile or public environments. Available early in the time-frame of transition to pervasive environments, these RFID tags can be attached to objects; they contain radio-enabled microchips which can be read out wirelessly. Their growing use covers areas such as supply chain management, and their low cost and widespread application today indicates that the ambient intelligence environment, despite the technological challenges still present, is not so far into the future.

The various approaches to augmenting artifacts in UbiComp environments are based on different underlying system architectures, ranging from distributed computing and peer to peer systems to centralized systems. Recombinant systems publish-subscribe methods, direct manipulation and graphical programming methods are different elements that come into play. Associations between artifacts are achieved either by following a given tangible interaction language (ie see Shifteo, Accord, etc), or by use of third intermediate devices that act as editors (see: Speaskeasy, e-Gadgets). The appropriate appearance, nature and loci of feedback interaction are interaction issues that most approaches have to address. Various attempts to address the loci of feedback interaction are noted, either by adding an LCD display onto artifacts, or by using nearby devices (that can provide audio or visual feedback), or using separate dedicated devices within the system (such as palmtop computers as editors).

This chapter presents software and hardware engineering approaches from projects that deal with the inter-association of artifacts. Platforms such as SMART-ITs, SIFTABLES, ACCORD, NEBULA, CAMP, e-Gadgets, the TERESA environment, Plants, ASTRA, SpeaksEasy are outlined, while other related system engineering approaches (such as TinyOS, Phidgets, Oxygen, etc.) are briefly presented.

5.2. Approaches to creating and associating artifacts

SMART-ITs:

In the context of the disappearing computer initiative, the project 'Smart-Its' (Holmquist et al, 2001) aims at developing small devices, which, when attached to objects, enable their association based on the concept of 'context proximity'. The collective functionality of such a system is mainly composed of the computational abilities of the Smart-Its, these work as added tags to the physical shelf of the participating objects. In this approach action, such as shaking together at the same time two personal devices, groups the devices together to a certain functional cluster (SMART-ITS project website).

SIFTABLES:

David Merrill and Pattie Maes have introduced sensor network user interfaces through the "Siftables" platform (Merrill, 2007), and a tangible interaction language for it. Here, wireless sensor network technology and methodology is applied in the area of tangible user interfaces. Applications, such as photo 'sorting' or 'scrabble' word game, are handled through the Siftables distributed tangible user interface (a kind of generic square compact tiling). Shiftables themselves (figure 13) are a collection of small compact tiles (36x36mm) with accelerometer, a color LCD screen, infrared transceivers, and RF radio, as well as a rechargeable battery. An interaction language for this platform is introduced, with a library of manipulations analogous to the point and click or drag and drop, but related to sensor network UIs, such as shake, group, pile, etc. Visual feedback appears in the LCD screen while audio feedback can be heard through a nearby computer. The devices are uniform tiles; therefore it is more a Peer to Peer (P2P) generic platform for games and specialized applications rather than for generic objects of everyday use. More information about this platform can be found in (SIFTABLES project website).

Figure 13: The Siftables platform of small ubiquitous devices manipulated by gestures

What the Smart-Its and Siftables approach have in common is that they attempt to introduce conventions and *use a tangible interaction language, while no external editor or Graphical User Interface is used.* Rather, they introduce a vocabulary of gestures and tangible manipulations for associating artifacts, within the context of given (and not user-configured) ubiquitous applications. Applications are to be used, but not configured, by end users.

NEBULA:

Nebula (Kyffin and Gardien 2009) is a Philips Design project that explores experiences enabled by omnipresent devices such as wall-embedded digital windows, wearable electronics, and flexible displays. These devices are intended to merge in order to create one fluid augmented experience (Figure 4, Figure 14).

Nebula (Gardien 2007) supports different customized experiences, making it possible to be used by very different markets and use contexts (ie home experience, hospital etc.). Customization is achieved by adapting the information streams that are projected. Allowing the customization of the content stream brings Nebula a step towards co-designing by end users (Green, 2007).

Figure 14: An image from the Nebula project; the Nebula Alarm Clock projects end user defined content to the ceiling, as a wake up alarm. (Image from Philips Design Website).

ACCORD:

The project ACCORD focuses on a tangible UI approach. Its focus was in developing a Tangible Toolbox (based on the metaphor of a tangible puzzle) that would enable people to easily embed functionality into existing artifacts around the home (Rodden et al, 2007). ACCCORD has a compositional approach to Home Environments, based on a component model using the notion of a shadow digital space that acts as virtual representation of the physical environment (Rodden et al, 2004). The project has developed a tablet editor and a graphical user interface (GUI) representation of a jigsaw puzzle for it (Figure 15). The ACCORD component model was evolved with collaborative design methods, in partnership with end users. The Jigsaw pieces that were used as artifact's respresentations were readily understood by users, who could subsequently reason about simple interconnections of devices and make assemblies of them. The ACCORD toolkit is publicly available at the (ACCORD project website).

Figure 15: Left: the Paper Puzzle Editor using paper based identification technology. Right: the tablet editor and the editor screen. Each component is represented as a physical puzzle piece; a service is created by connecting pieces in a left-to-right order. (Source: ACCORD project website).

TERESA

TERESA (Mori et al 2004), (Paterno et al, 2008) is an environment for authoring pervasive multimodal user interfaces. It is composed of a set of XML-based languages (DSL), transformations among such languages, and an authoring tool. It provides designers with the possibility of designing interfaces for a wide set of platforms, which support various modalities. TERESA provides application examples for a number of platforms.

CAMP

CAMP (Truong et al, 2004) is a system that enables end-user programming for smart home environments based on a magnetic poetry metaphor – a simple interface for creating applications. CAMP aims to enable users to create applications taking as a starting point the users' goals and tasks – rather than the developers' perspective of devices and their interactions. (Truong et al, 2004) presents a study making use of automated capture and playback of home activities which examines how users

conceptualize applications. The study reveals a breadth of home applications that people desire.

THE GADGETWARE ARCHITECTURAL STYLE (GAS):

The Gadgetware Architectural Style approach (GAS) views the process where people configure and use complex collections of interacting artifacts as having much in common with the process where system builders design software systems out of components (e-Gadgets project website). The domestic environment in this approach has a multitude of peer to peer interacting artifacts, from furniture and appliances to environmental sensors (ie temperature, humidity, light, or sensors in the building features), which people dynamically associate, aided by the Gadgetware Architectural Style (Kameas et al, 2003), (Mavrommati et al, 2002), (Kameas et al, 2005)   (Figure 16, Figure17). A style is in a way a dictionary of elements, together with the syntax which indicated how these elements can be put together.
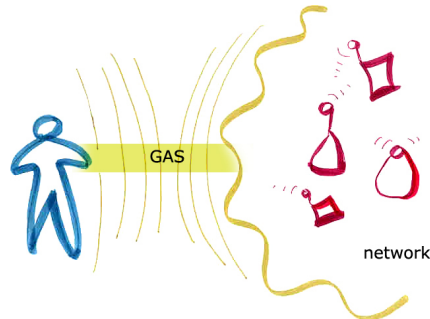
Figure 16: The Gadgetware Architectural Style provides a way for people to manipulate Ubiquitous Computing Applications

The Gadgetware Architectural Style proposes a Style for Peer to Peer Ubiquitous computing applications, and as such it proposes the generic concepts and syntax from which individual instantiations can be generated.
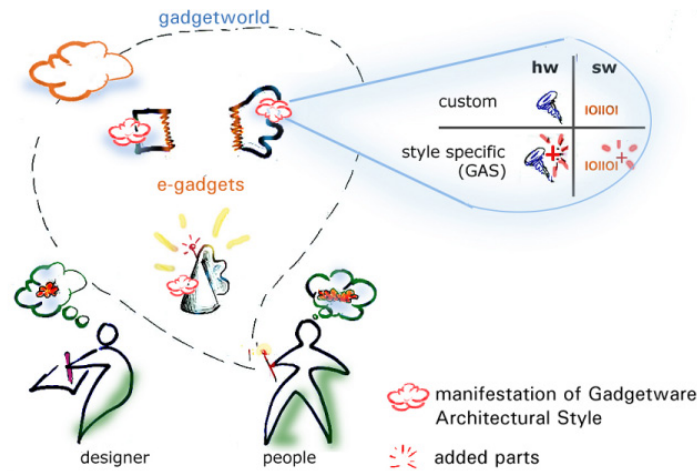
Figure 17: The Gadgetware Architectural Style (GAS) is a common referent between artifact manufacturers, designers, and users

In the GAS style approach, a vocabulary is developed, so that people can understand the nature of artifacts and manipulate them as a result (Figure 16, 17). This approach can scale both 'upwards' (towards the assembly of more complex objects, ie from objects to rooms, up to buildings, cities and so on) and 'downwards' (towards the decomposition of eGadgets into smaller parts, i.e. towards the concept of 'smart dust'). Different objects are treated similarly within GAS. These objects are GAS enabled artifacts, thatmay range from the computationally powerful ones (having their own processing and communication), to the very minimal (that can be considered as tagged artifacts (ie with RFID), which may borrow processing and storage capabilities from servers or other proximate artifacts, for example). The artifacts and the GAS Operating System Architecture are explained in Figure 18 and 19. An editor is used by end users and designers alike, to create and manipulate the GAS enabled artifacts and configure their collective behavior. More on this approach can be found in chapter 10.
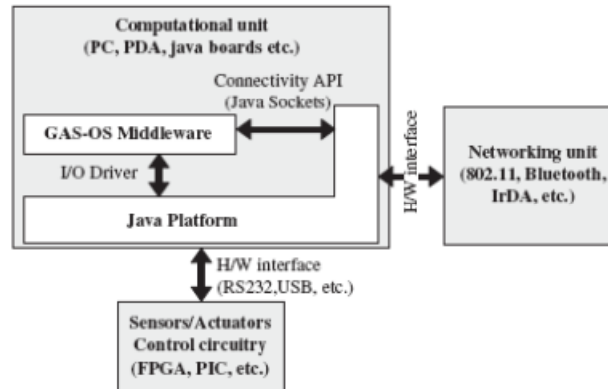
Figure 18: Artifact high level architecture. Source: (Drossos et al, 2007)
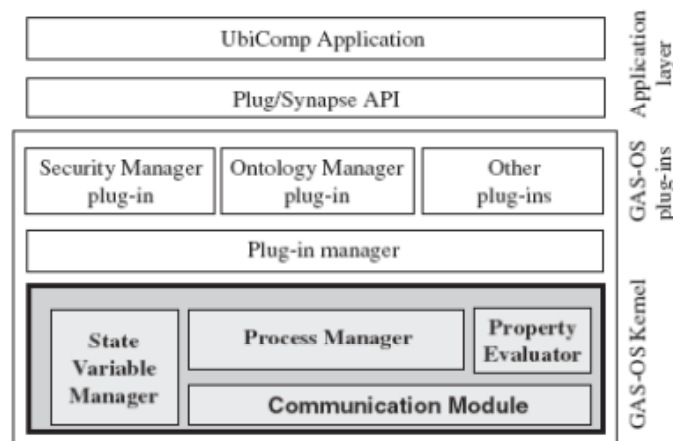
Figure 19: The GAS OS modular architecture according to (Drossos et al, 2007)

PLANTS:

Project PLANTS (PLANTS project website) introduces three-way communication between plants, people and artifacts, following the Gadgetware Architectural Style approach. PLANTS project starts from the basis of the e-Gadgets project (envisioning an environment consisting of people and artifacts and the interactions between them), and expands this into mixed plant-artifact environments (bio-

gadgetworlds), (Figure 20), aiming to optimize the efficiency and productivity of plant growth. An array of sensors placed among crops detects biological plant signals and uses them as the basis for precision applications of water, pesticides or fertilizers.
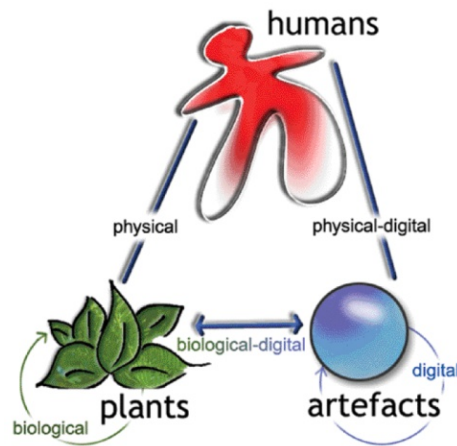
Figure 20: PLANTS explores mixed environments, in which humans, plants and artifacts are computationally enabled and can act together, in the context of precision agriculture.

By delivering resources when and where they are needed, the PLANTS system seeks to minimize their wastage and the environmental and human health damage their overuse can cause. PLANTS software, ePlantOS, enables plants to communicate with other components of the PLANTS system by converting their botanical signals (detected by sensors) to digital signals, and then into actions from the actuator-artifacts (such as watering). The ePlantOS software system acts as a 'parallel universe' in which each physical entity of the crop set-up has a second 'digital self'. A plant thus becomes an 'ePlant' by having an added technological layer. Pumps, lights etc. are similarly represented as 'eGadgets'. ePlantOS is a middleware software layer that manages the network interactions of the system by linking up all components into a virtual mixed society of plants, people and gadgets called a 'bioGadgetWorld'. Decisions (ie whether to irrigate) are based on the 'ontology', a directory of rules and definitions about plant parameters and

characteristics, the core of which is designed to fit the limited memory capacity of ePlants and eGadgets. A higher-level ontology elaborates on the core ontology. The users (crop managers) can build their own bioGadgetWorlds of interacting elements (using Bio-Gadgetworld Editor) and are able to edit the underlying ontology rules through the Supervisor Logic and Data Acquisition Tool (SLADA), which is also intended to be used by crop managers.

ASTRA:

The ASTRA project (ASTRA website) explored the concept of pervasive awareness, ie where awareness information is automatically generated as a result of personal/home devices exchanging information about the user and the situation of use semi-autonomously. The project therefore investigated systems and concepts that support such pervasive awareness applications. To this end ASTRA defined a framework for supporting the conception and the design of Pervasive Awareness systems, specifically those that are intended to support social relationships. These consist of theories and technological solutions (service oriented architecture, tools and applications) that support communities to create, adapt and appropriate Pervasive Awareness applications.

ASTRA envisions pervasive systems that can help fulfill fundamental social needs: knowledge about each other's emotional status, health, location, and availability, but also the exchange of messages can contribute to feelings of connectedness. The feeling of belonging (labeled by ASTRA as 'social connectedness', the term signifying feelings such as being in touch with others or the level of closeness and social contact in one's social life) is seen as fundamental to human well-being.

The ASTRA project has created an experimental infrastructure: a service oriented architecture and tools for enabling communities of end users to create their own awareness systems and services. ASTRA aimed to support the process of innovation driven by end-user communities themselves acting as designers of their own awareness services, coupled with the ubiquitous computing applications as a means

of defining the awareness service, or displaying it. Astra couples awareness applications with ubiquitous input and output on the user's environment. An inherent part of ASTRA end user tools is the definition of ubiquitous computing applications that signify or portray awareness information as well as the fact that these awareness applications are shared within a social group.

ASTRA produced a computer mediated awareness system, exploring pervasive awareness and making possible the configuration of awareness applications by end users (Mavrommati I., Calemis J., 2008), (Figure 21). ASTRA uses a Service Oriented Architecture (SOA) for service delivery in a dynamic environment. Graphical User Interfaces aimed at the end users enable the configuration of "awareness connections" and the definition of pervasive applications, as well as the overall observation and control of running active configurations.
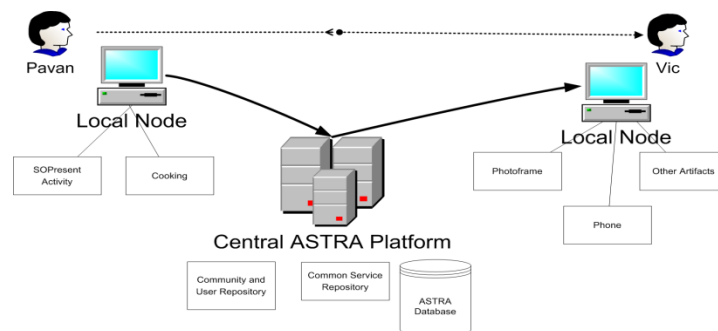
Figure 21: ASTRA: Awareness communication is transmitted between two ubiquitous environments

Different variations for end user configuration of a ubiquitous application have to be noted:, On one hand, projects such as Nebula focus on allowing the customization of the experience (by selecting different content assets as elements within a predefined experience) while approaches such as GAS, ACCORD and ASTRA focus on allowing for the end user design of the total ubiquitous application and experience.

SPEAKSEASY.

Speakseasy is an example of 'recombinant computing' - a set of common interaction patterns that leverage mobile code to allow rich interactions among computational entities with only limited a priori knowledge of one another **(**Edwards et al 2002), (Newman et al 2002)**.** Speakseasy is designed to support ad hoc, end user configurations of hardware and software, and provides patterns for data exchange, user control, and contextual awareness.

Speakseasy is used to support ad-hoc spontaneous peer-to-peer collaboration. Peer-to-peer systems have deficiencies (such as inflexibility in terms of discovery protocols, network usage, and data transports) that diminish their ability to support this domain. The Speakeasy framework addresses these issues and supports these P2P types of applications; a demonstration application, called 'Casca', supporting ad hoc peer-to-peer collaboration by taking advantages of the mechanisms provided by Speakeasy, was created as a case study.

Distributed computing software technologies

Various software technologies are being employed in the context of distributed computing research projects. Several of these technologies are considered as paving the way to the future pervasive computing environments (most prominent examples are, for example Cloud computing, or the Internet of Things). Such developments are briefly outlined here:

- **RPC**: Remote Procedure Call - legacy mechanism for implementing distributed systems.
- **RMI, Java RMI, DCOM** (using ORPC wire protocol), CORBA: legacy Object-Oriented (OO) distributed programming mostly using a client-server model. When using callback channels (connection from server back to client) these systems tend to have problems with firewalls and the range of open ports which limits their applicability.

- **JINI:** Java specific - distributes/leases computational capability to JVM (Java Virtual Machines) nodes on the network. Designed with hardware-based JVMs in mind which didn't gain enough traction in the market.

- **JavaSpaces:** an early cloud computing pioneer, offers publish/subscribe methods in a resource cloud. This approach is Java specific which lowered its usage/adoption.

- **Web Services based on SOAP** (Simple Object Access Protocol) XML based protocol for invoking services and getting back results. SOAP is an official W3C standard, advanced but a bit verbose and hard to implement all related protocols for security, transactions etc. defined by WS-I (WebServices Interoperability group). For device interactions REST (Representational State Transfer), also XML-based but simpler, stateless and less verbose, has become a defacto protocol standard (e.g. used by Microsoft Robotics Studio).

- **Semantic Web** (dubbed 'the Internet of Things' (IoT)) is an effort backed by W3C consortium which define World Wide Web standards and is spearheaded by its director Tim Berners-Lee. It is adding semantic metadata in standardized forms (eg RDF - http://www.w3.org/RDF) to services, coupled with structured, Web-based ontologies (eg in OWL or SKOS) which enable richer integration and interoperability of data among descriptive communities.

- **Grid Computing** is a combination of computer resources in loosely coupled, heterogeneous, and geographically dispersed grids, coordinating them to reach a common computational goal.

- **Cloud Computing** is a trend in Internet-based computing approaches, whereby shared virtualized computational and storage resources, software, and information are provided to computers and other devices on demand, as with the electricity grid. Such developments are seen as paving the way to future ubicomp applications.

Other related approaches to distributed computing, pertain to the development of platforms, such as the following:

OXYGEN: A more generic approach has been undertaken by the project "Oxygen" which enables human-centered computing by providing special computational devices, handheld devices, dynamic networks and other supporting technologies. (Oxygen project website)

GAIA (Roman et. al., 2002) provides an infrastructure to spontaneously connect devices offering or using registered services. Gaia-OS requires a specific system software infrastructure using CORBA objects, while mobile devices cannot operate autonomously without the infrastructure;

BASE/PCOM (Becker, 2004), (Weis, 2006) use component-oriented middleware, supporting heterogeneity and abstraction of services, nevertheless the application programming interface requires specific programming capabilities by end users. A tree structure is used here for manipulating the components.

TinyOS is an event driven operating system, designed to provide support for deeply embedded systems (i.e. sensor networks), which require intensive operations while they are hampered by hardware resources that remain minimal.

PHIDGETS: Phidgets supports the easy development of physical interfaces through physical widgets (Phidgets, 2001).

A multitude of commercial solutions exist (eg CORBA, JINI) but are rather heavyweight for most applications and too complex to be used by ordinary users.

5.3. Conclusions

This chapter has presented related work pertaining to different approaches to augmenting artifacts in UbiComp environments, allowing for different behaviors, in recombinant or configurable ubiquitous systems. Two different aspects of end user

development of a ubiquitous application have to be noted: on one hand, projects such as Nebula, or Siftables, focus on allowing the customization of a given contextual experience (by selecting different content assets as elements within the experience of a predefined umbrella application). On the other hand, approaches such as GAS, ACCORD, and ASTRA focus on allowing for the total experience design of the ubiquitous application by end users.

Among the variety of concepts in the different underlying system architectures, we note recombinant approaches in peer to peer distributed computing systems as a software and hardware system design approach being most relevant to Ubicomp applications accessible by end users. The most prominent themes in related work in this area is treating artifacts as recombinant computing modules, using for their communication publish-subscribe methods, allowing in some cases for direct manipulation or in others using graphical programming methods. Artifact associations are achieved via given tangible interaction languages, or by use of Editors as programming overview devices. The nature and loci of feedback are commonly mentioned as interaction issues that need to be addressed for the context of each research domain.

6

6. HCI issues for ambient computing environments

6.1. Introduction

In an Ambient Intelligence Environment the application and the interface merge into one entity; the augmented artifacts and spaces become access points to applications but they also carry the application's functions. The nature of interaction within the AmI environment differs to the familiar paradigms of Human Computer Interaction. We argue for the emergence of a new generation of User Interface Design Tools that facilitate users to see into and manipulate the behavior of the augmented artifacts

and spaces, and through these manipulate the applications of the Ambient Intelligence Environment. The development of such tools has to face the technological issues posed by ubiquitous computing, but also the challenges arising from shifts in human computer interaction within AmI environments. Some of these issues (relating to the shifts in Human Computer Interaction within AmI environments), which can affect the design of these tools, are discussed in this section.

6.2. Concerns about the Ambient Intelligence World

The main focus of ubicomp research is on developing the technology that enables Ambient Intelligence Environments (ie sensors technology, miniaturization, processes, networking, middleware, energy provision), only subsequently addressing the HCI issues within these environments. There are insufficient existing HCI practices, theories and models regarding HCI for AMI, while User Centered Design is at its very early stages in this field of research.

AmI's theoretical vision tends to present people as passive consumers happily accepting their increasing dependance on AmI systems. AmI sees people increasingly and comfortably relying on AmI for a number of activities (for reminders, surveillance, health monitoring, entertainment, home automation etc) (Crutzen, 2006). Yet, it remains unclear how such systems are maintained. This vision, according to Crutzen, does not seem realistic because Ubiquitous Computing systems cannot be absolutely problem free.

It is not clear whether new technologies will eventually provide a means to escape gracefully from being always connected (Swami report, 2006), (Markopoulos et al, 2004). Moreover it is questionable how people will accept living and evolving within AmI technological environments and how this 'nurture' will impact human nature (Swami report, 2006). Social issues relate to how people's individual as well

as collective behavior will evolve by living in AmI environments, where it is clear they will develop certain expectations, assertions, and habits (Mavrommati et al, 2003d)

A network of interconnected devices has as a consequence that the quantity of information increases beyond the capabilities of human perception, resulting in information overload. Increased connectivity also leads to blurring of the physical borders between people and spaces. Several studies attempt to respond to the need to understand the broader implications of AmI - such as Digital Territories Study, (Daskala and Maghiros, 2006), (Daskala and Maghiros, 2007), or Safeguards in the World of Ambient Intelligence report (SWAMI report, 2006) - but are still at too early stages to be taken into practice.

AmI system design has to face serious challenges regarding the system's usage, concerning how users can control the AmI system (for example with appropriate means, or aided by intelligent agent interfaces), how they can predict what the complex networked artifacts will do, and how the whole AmI system in turn will function appropriately and unobtrusively, providing for qualitative experience and safeguarding its user's privacy. HCI issues such as avoiding stress and confusion in order to achieve actions, avoiding errors, and facilitating recovery from errors, are important to be addressed.

6.3. Issues Introduced in HCI

New elements that are introduced by the nature of living and interacting within an ambient intelligent environment lead to new HCI paradigms. Some of the issues that impact not only AmI HCI research practice (Mavrommati and Darzentas, 2006) but also End User Development are described in the following sections. These issues belong into the following generic categories:

- Change in Human Computer Interaction models

- A shift in the nature of interaction

- Organizational concerns of users

- Interaction channels

- The scope of intelligence

- Visibility of actions, reversibility of actions, error tolerance

6.4. Change in HCI models

A good case for study in the context of Ubiquitous computing environments is Norman's (Norman, 1990) model of interaction which is widely used in HCI: the so called execution-evaluation cycle. The execution-evaluation cycle splits the interaction into a sequence of sub-actions, each of them being a result of a specific user intention. Initially the user forms a goal, and then forms a sequence of intentions, followed by specific actions. The user then proceeds to execute these actions, and perceives the state of the system after these actions are performed (change state, communicated to the user via appropriate feedback). The user then interprets the new state of the system and finally evaluates the outcome, comparing it to his/her initial goal (to what extent the goal is achieved). Belloti (Bellotti, 2002) attempts to re-think Norman's '7 stage' interaction model, focusing more on interactions that are most appropriate for AmI environments (assuming the interactions are not GUI based). Belloti suggests the following five interaction challenges for AmI researchers, and exposes a number of design challenges that result, which designers should address:

1. *Address* - how to direct communication to a system. (Disambiguate signal from noise; disambiguate intended target system; define how one can avoid addressing the system accidentally).

2. *Attention* – ensuring that the system is attending (Embody appropriate feedback so that users are aware of system's attention; feedback should be of such nature and loci so that it is directed to user's attention).

3. *Action* – defining what can be done with the system [Identify and select a possible interaction object. Identify and select one action and bind it to the object(s) and avoid unwanted selection. Handle complex operations (such as multiple objects and actions, more abstract functions)].

4. *Alignment* – monitoring systems response. (Make the system state perceivable at any given moment; direct timely and appropriate feedback; provide feedback on the system's response to users).

5. *Accident* – avoiding errors and misunderstandings, or being able to recover from them. (Control or cancel system action that is in progress; system intervention in case of error; disambiguate what can be undone in appropriate time).

Some pitfalls that are applicable for End User Development in AmI which are identified in the Belloti (Bellotti, 2002) model include: unintended actions, leading to undesirable results; failure to execute an action; limited operations available; wasted input effort in a non-attending system; inability to detect mistakes; difficulty in evaluating a new state; and inability to detect mistakes and to recover the previous state. The Bellotti framework is not only useful for the design of intelligent automations in AmI systems, but is also applicable for understanding HCI issues in the complex, interlinked Ambient Intelligence environment.

6.5. A shift in the nature of interaction

In Ubiquitous Computing environments people do not act on the world, but act *with* the world. Thinking, according to Distributed Cognition theory (Lave, 1988), is not

just within the head, but in the external relationships with things in the world and with other people. People, who are in constant dialog with the physical environment, they use the information and location of artifacts in order to guide their actions (Dix et al, 2004). Incidental interaction that happens within an AmI environment thus impacts the system design and has relevance for End User Development.

Interaction in Ambient Intelligence Spaces can range from explicit to implicit interaction. The user may be unaware where the interaction is taking place (ie via gestures, sensors, movement detectors, secret cameras) (Dix et al, 2004). Unintended user actions in the environment may result in unintended control and manipulation of the application of the direct environment of the user, or of associated environments in different locations. The implicit nature of this interaction (achieved in AmI by both sensing and physical action) assumes a seamless human computer relationship where there may often be no conscious interaction. On the other hand, since no system is completely error free, issues of the appropriate level of visibility and transparency of the system are raised. The possibility of facilitating a degree of transparency (providing upon request some visibility into the workings of the ubiquitous system) is an element that could be considered for End User Tools. (Dey, 2004), (Mavrommati et al, 2004). Schmidt argues for an AmI interaction model in which users can always choose between implicit and explicit interfacing: *'The human actor should know ... why the system has reacted as it reacted'* (Schmidt, 2005).

We are witnessing a radical shift in the field of HCI: the basic models of interaction that have proved universal across technologies are questionable for AmI (Dix et al, 2004), (Bellotti, 2002), (Scholtz and Consolvo, 2004). The main drifts from traditional HCI, as described in (Dix et al, 2004), are:

- The focus becomes that of activities rather than tasks.
- Emphasis is given to the design of continuously available interaction
- There are no starting or ending points for interaction in AmI environments

- Interruptions and multiple actions can happen within the environment, that are not (or are loosely) connected to achieving certain goals. Such interruptions and multiple actions can be mistaken as options for the system.
- There are many different perspectives in operation within the AmI environment and therefore the reuse of information for different functions should be a consideration. This points at the need of, for example, associative models of information.

6.6. Organizational concerns of users

In an AmI environment there is more than one inhabitant and therefore more than one user per application. The input in AmI is often distributed, while the user may not always be aware that his or her action is in fact an interaction within the UbiComp environment.

AmI systems may be accessed by single users, but also by users who operate in larger groups (Bellotti, 2002). There can potentially be more than one user of an AmI application, working on it simultaneously, from the same or remote locations. The same may also hold as a requirement for Tools aimed at the creation and editing of AmI applications. It may be in the interest of more than one user to co-edit an application that involves them, synchronously or asynchronously, but doing this collaboratively and from different locations than the application environment.

6.7. Different interaction channels

In AmI the direct engagement is with the world itself (Dix et al, 2004). In the real world the input is physical, yet in the AmI world physical action gets converted to digital information too, and has consequences as an action of direct manipulation. Here the physical world becomes the interface of the AmI system, but is more than

120

just interface, as it cannot be separated (mentally or in fact) from the rest of physical reality.

In AmI the nature of input and output devices is shifted and is very different that in traditional HCI. Interaction becomes multimodal and ubiquitous: many appliances and artifacts within an environment can be used and in many different operations, as well as sensing capabilities of the environment. The same applies to the nature of the input of the application, that can be similarly diverse; speech, gesture, tangible interfaces, biometrics, are a few of the interaction methods in play. Complex command languages could be replaced by certain direct manipulation actions upon the objects, which make use of multimodal interface combinations to interact with the system. Among the issues affecting end user development, as identified by (Bellotti, 2002), are:

- how to identify and select a possible interaction object,
- how to select one action and bind it to the object while avoiding unintended selection,
- how to handle more abstract functions, and
- how to embody appropriate feedback and direct it to users' attention

6.8. The role of intelligence

Actors coming into play in an AmI environment can be human, or agent software. Intelligent agents can be used in 'programming by demonstration' techniques, thus facilitating a set of actions for end user development within the environment. Proactive agent behavior holds the promise of seamless interaction within the AmI environment. On the other hand, agent intervention may result in unexpected behavior of the AmI system that may surprise the user – and such surprise must be avoided.

Visibility of the workings of the agent and the intelligent application and its rationale should be available upon request, as well as an overwrite function for the applications or the agents within them (the overall off switch). Appropriate feedback so that users can be aware of having the systems attention has to be considered.

6.9. Visibility, Reversibility of Actions, Error tolerance

Feedback should be provided for actions upon the physical environment that involve the AmI application. Syntactic correctness of sequences of actions has to be constantly and continuously checked in order to appropriately inform users, so as to avoid errors before they are made. In the CollaborationBus example (Gross & Marquant, 2007) feasibility checks automatically deactivate inadequate operations. To be able to undo actions (reversibility of actions) is also very important in the case of error: a requirement stemming from this is the ability to recover previous state. Visibility, reversibility, and syntactic correctness of actions can also be dependent on issues of context awareness and compatibility of platforms. In an AmI environment many interoperating platforms may come into play, while the locus and nature of both feedback and of stored actions is distributed. Some actions are upon the (local or remote but linked) environment and cannot be undone (eg a sound that has been played, or a heat that has been generated, cannot be 'undone'). Therefore, error prevention, tolerance and reversibility all pose interesting and serious system design challenges (Bellotti, 2002) (Crutzen, 2006), (Mankoff et al, 2000), that are only beginning to be addressed in current research.

6.10. Conclusions

This chapter has outlined issues regarding Human Computer Interaction with Ubiquitous Computing environments. We note the work of Bellotti (Bellotti, 2002) as an important outline of the HCI issues that can serve as design challenges for the building of UbiComp systems and applications.

In ubiquitous environments there is a shift in the nature of interaction, with one or more users interacting within an environment rather than a single computing device with defined input and output. Ubiquitous computing environment will eventually be used by groups of users, simultaneously, from the same or remote locations. Interaction channels become multimodal and ubiquitous, since users now interact with the world itself. Identification of interaction objects, avoiding unintended actions or selections, handling more abstract functions, and the nature and location of feedback are emergent Human Computer Interaction issues. Since ubiquitous computing systems are facilitated by intelligent mechanisms there is the danger of unexpected system behavior, and no visibility on the intelligent decisions that are being made. Serious HCI challenges are those of visibility, error prevention, syntactic correctness of actions, error tolerance and reversibility of actions in ubiquitous computing environments.

Last but not least, in order to understand the nature of interaction in ubiquitous computing environments, we need to divert from the models that are established in classic HCI theory, evolve and adapt them (as (Bellotti 2002) did), or adopt new interaction models that are more appropriate and applicable to these new, more complex computing environments. Such new models are better founded in Activity Theory, Distributed Cognition Theory, and Situated Cognition Theory, as reported in (Dix et al, 2004).

7

7. End User Development in software: basic concepts

7.1. Introduction

End User Programming (EUP) - or the broader term End User Development (EUD) - describes the situation where users, who are not primarily software developers, configure or program their devices or services in order to meet their own needs. This chapter briefly describes the field of End User Programming, seen here in its more general form, for software applications of various kinds, a subset of which is the domain of ubiquitous computing applications. In the following section what is End User Programming will be outlined and the profile of end users acting as developers will be described. The main challenges (semantic and syntactic) of programming for end users will be explained as well as some programming paradigms.

A concise summary of the field, albeit with a primary focus on Awareness Systems, is provided by P. Markopoulos in ASTRA deliverable D4 (pp.5-14) (available from

ASTRA project website). Reference book for the field is "End User Development" edited by Lieberman, Paterno, and Wulf, (Springer, 2006).

7.2. How is End User Programming defined?

End User Development is defined as *"the broader set of activities, techniques, methods, and tools that allow people (users of software systems), who are acting as non-professional software developers, at some point to create, modify or extend a software artifact"* (Lieberman et al, 2006) (EUD network website) and (Costabile, 2002). The purpose of this activity is to get the devices to do what people wish them to do.

This is not the standard model of software development, where professional programmers or developers create these applications on behalf of business clients, who then package the ready-to-use applications and retail them to their end users. It is nevertheless a model that is becoming increasingly popular, from the creation of a simple webpage (that can be exported by a word editor for example, without any knowledge of programming skills) to blogging, tweeting, configuration of private web pages in social networking sites, to the configuring of wikis. To some extent, programming is been made possible to an increasing number of ordinary people. Application configuration is available to everybody, and the resulting applications are more tailored to everyone's own needs.

End User Development is a term gaining popularity in recent years, preferred to the term End User Programming as it signifies better the broader scope of activities. What were previously the concerns of software developers now are handled by consumers who are non-software-developers. End User Programming issues range from the design of the application, to creating the code for it, installing it, testing and debugging it, updating and optimizing it. Examples that aim to make programming

easier, such as the filling of forms (often web-based), or using graphical notations, are exemplary manifestations of End User Development.

7.3. From adaptation to new functionality

The End User Development definition is broad, as it includes all forms of adaptation and personalization of a software application. Different levels of manipulation of applications are inherent in this definition. Nevertheless, the core of End User Development is considered (according to the EUD-net report) to be i) change of application/interface behaviors, and ii) creation of new functional behaviors. Associated activities include installation, trouble shooting, debugging, and updating.

End User Customization, meaning the configuration of settings (by selecting options) in order to customize an application, is considered as one end of the spectrum of End User Development. End User Customization includes, for example, actions such as setting a background image in one's mobile phone or desktop for example, or setting up style settings on a word processor.

Between these two poles of end user customization and end user programming, there is a range of activities from setting up a mobile phone or a set top box, to one's email, to an Excel project sheet, up to developing one's own webpage; such examples are more typical representations of end user programming.

Within the context of End User Development, in the specific domain of Ubiquitous computing applications, the term '*End User Design*' is introduced in this thesis as complementary to '*End User Programming'*. The term primarily signifies the descriptive design process of conceptualization rather than the more structured process of programming. It relates to describing scenarios (in text based or visual form) and experience details, with the user acting from the perspective of an

experience designer rather than the software design process and problem solving implied by the mind-frame of a system engineer.

So, by use of the term '*End User Design*' we refer to the easier and more frequently used parts of End User Development (such as customization), the more abstract descriptions (such as scenarios or narrations) created by end users to describe applications, and some extra manipulation functions upon artifacts. At the other end of the spectrum is the development of an application in a programming language by an enthusiastic amateur programmer. End User Development tasks and approaches in this research are thus seen as being twofold, *End User Programming* and *End User Design*. The latter will be described in more detail in further chapters (see chapter 9, chapter 14).

7.4. The profile of End Users – Developers

End users acting as developers can range from enthusiastic amateur programmers to professionals who are not computer scientists or programmers (who may act in their professional capacity, as teachers, designers, doctors, etc.), or consumers – not acting in their capacity as professionals, but as individuals. (Lieberman et al, 2006, pp7).

With the intergration of computing technology into artifacts of everyday personal use and the penetration of internet technologies and Semantic Web, the wider public is becoming increasingly accustomed to computing and fluent in the use of computing technology. More and more people engage with activities that were considered to be programming a few years ago. The range of End User Development skills are gradually shifting, from the simple configuration of forms to more complex configurations.

End User Development activities are defined by moving boundaries and so are end-users, whose constantly shifting skills and profiles cannot be readily defined. It is not assumed that programming is a regular activity for people engaging in end user development. End users acting as developers may not have any formal training in programming concepts or methods and may not think in programming terms for problem solving activities. Environments supporting End User Development should therefore aim to minimize the memory strain required to remember concepts and structures. End User Development should strive to make it easy for its users to get accustomed to the programming environment and principles involved, with an easy and low threshold introduction.

7.5. Visions and contradictions

An application that can be considered an end user development application is in the shifting zone of skill development; it pushes the boundaries of what programming tasks are given to end users, which were previously in the domain of professional software developers. End user programming follows the development of computing, starting from the single computer access point, and gradually progressing to the ubiquitous computing environment augmented by sensors and computer networks.

Traditional interaction design principles suggest making invisible to the end users the internal workings of the system. The so called "black box" approach is about providing only the required input and output, in the right proportion, without unnecessary cluttering or confusing user interaction with the details of the internal workings of the system. End user programming, on the other hand, suggests that some developer tasks should be handed to end users; the motivation being that the details of the wishes of each user cannot be known to system developers in advance, nor all the context of use. The danger in this is that too many functions and extra features are packed into systems interfaces, resulting in complexity and clutter that

can be overwhelming for end users, even where they do wish to engage in redesign of the system.

Managing complexity is a key issue in End user development. Allowing cascading views of the internal working of the system (perhaps supported by multiple views) can help towards decreasing complexity. The vastly increased functionality made available to end users can be handled by gradually revealing to the users the functions, when and as they need them while they become more accustomed to using the system. According to Nardi (Nardi, 1993), end user programming is an attitude to attempt to simplify things, allow reuse and automation through certain mechanisms within a system.

The domain of Ambient Intelligence (AmI) (Aarts and Marzano, 2001) that has emerged over the last decade is an area where end user programming might be valid; the vast number of applications and services as well as context (social or technological) cannot be known in advance. The solution of giving to users the possibility to adapt the functionality of ubiquitous computing environments is emerging as a possible approach to bridge the gap between developers of the ubiquitous environment and it's inhabitants.

The end user development paradigm in the domain of Ambient Intelligent environments contrasts to the AmI vision advocating cognitive disappearance of the computer which was presented by Weiser (1991) and the subsequent AMI visions endorsed by EU IST programmes (DC, ISTAG report). The case here is that end users are actively engaging with AmI technology, and shaping it, in ways that cannot be foreseen a priori by developers (Mavrommati et al, 2004). The two paradigms need to co-exist, focusing on managing different levels of complexity, according to the activities supported.

7.6. Challenges: semantics, syntax, visual paradigms

End user development requires the creation of *domain specific abstractions at a semantic level* (Nardi 1993 p.27-42). Nardi argues that people are capable of learning and using formal languages and systems when these are relevant to their tasks (eg music scores, gaming notations, etc.). Presentation end user development concepts to users can be done in many various ways at a syntactic and lexical level, imposing many design challenges (i.e. design choices can range from visual manipulation to textual syntax).

Nardi (Nardi 1993) argues that task specific languages have to provide primitives that are high level (that is, they are not composed from many low level statements), task specific (allowing application of domain knowledge, and users execute directly the tasks they want), familiar and accessible (so that learning non-task-specific concepts is reduced), and have simple and useful controls.

In the case of EUD for Ubicomp, semantics can be provided by the use of the Connectivities-Links model (Plug-Synapse) - which will be described in further chapters. This has to be linked with platforms that support it, exchanging information between artifacts and user-control-related devices. At a syntactic level, interfaces need to be developed for EUD in Ubiquitous computing environments, to support users in the visualization, customization, and creation of new applications. Interfaces can be graphical, natural, programming by example, agent based, or mixed, but should avoid the difficulties that the conventional programming languages pose.

An overview of *generic guidelines for designing a programming language* aimed at end users is presented by (Reppening and Ioannidou, 2006). These guidelines are less applicable to the context of use of Ubiquitous computing nevertheless they provide useful considerations.

Apart from guidelines, there are several visual paradigms that can be used in end user development interfaces. In *form based systems*, for example, a rich set of functionalities is offered through a preset of pull down choices, checkboxes, and text fields in a form. End users can easily specify the parameters of the system, without having to understand or use programming language syntax. Forms are characteristic in many website configurations, (eg e-government, social networks, etc.). Although the range of applications that can be constructed with forms can be limited, they are easy to learn and can be used effectively even by novice users. Moreover they provide a guided structure that the users can easily comprehend, move through and make corrections to. For web services in particular they are a credible and standardized solution, which most users are accustomed to.

Visual language challenges of End User Programming

The suitability of a visual language depends on the mapping of the representations to the concepts related to the domain. It is often claimed that visual languages are more easy to use for programming. For example (Fernando et al, 2006) used a 3D graphical interface for the manipulation of audio connectivity between users of a virtual environment. Nevertheless, more abstract concepts can be better represented with textual representations (Green and Petre, 1992).

The appropriateness of different interfaces, textual or visual or combinations, needs to be explored for the domain of ubiquitous computing, in order to achieve a balanced representation of concepts of that specific domain. Mixed or swapping views on an interface can be a visually useful for handling multiple representations.

Programming language and syntax challenges for non-professionals:

Programming by end users poses several syntactic and lexical challenges, such as the following:

- Booleans: Difficulty in the use of Boolean expressions (AND, OR, NOT) as has been reported by (Greene et al, 1990). This partly owes to the fact that

these expressions, when spoken, are colloquially used to mean the opposite to their programming meaning. In particular the use of NOT is often unclear to non-programmers (Markopoulos in (ASTRA D4, 2009a)).

- Using formal Syntax is difficult for end users. As a guideline, minimizing syntax, and try to make syntax errors hard or impossible, is suggested by (Reppening and Ioannidou, 2006).

- Control structures (i.e. conditionals, loops, etc.) are difficult for end users acting as programmers to understand and use, and they require certain background knowledge and training. One suggestion is to closer match control structures with mental models of iteration by the users (Markopoulos in (ASTRA D4, 2009a).

- Object oriented programming, object's attributes, relations and abstractions, are difficult to comprehend and be used by end users who are non-programmers (Detienne, 1990a), (Detienne, 1990b).

It has to be noted that, with the introduction of programming skills in the school education curriculum, in the future we may witness greater affinity of younger users in particular with some of the syntax of programming languages. End user development skills are constantly evolving, with end user development being a shifting area that cannot be defined as a confined skill zone. In the specific case of End User Development for Ubiquitous computing, that is itself a developing field, the future profile of end user programmers has to be taken into consideration. It is assumed that Ubiquitous computing will become mainstream technology in a decade or more. The skills of the majority of end users will be different than it is now, as it benefiting hugely from their affinity with computing both outside and as part of their formal schooling.

Case based programming

Problem solving is seen as a process whereby a new, unseen problem is matched against the accumulated knowledge in the cases captured in past experiences. In Case based reasoning (Kaneko and Onisawa, 2005) users have little knowledge of

software programming skills, while they can instead interact through linguistic expressions with the system. Programming in this case involved converting the ideas as expressed by users into machine language. Gu (Gu2005) has experimented on Case Based Reasoning based on conversations, in a system that assists users by having a visual dataflow programming environment for image processing, in order to retrieve certain modules of image processing functionality.

Case Based Reasoning can be considered a midway solution between programming by example and other direct manipulation paradigms of end user programming. A pre-constructed case can match the problem, thus providing an initial solution (program). An additional benefit is that the cases can be shared between users in a repository or expert system. The program can then be further adapted to users, by use of agents or by being deliberately manipulated through user interaction.

7.7. Conclusions

The domain of ubiquitous computing, a relatively new field, is not sufficiently investigated with regards to appropriate end user development paradigms. This area is considered a promising ground for end user development (ASTRA project website). In this domain, two contradictory approaches, that of the 'black box approach', prominent in the vision of Ambient intelligence (the computer working as an enabler to the background while disappearing from the foreground of people's attention) and that of End User Development, (where users can understand and manipulate the manipulate their environment and shape unforeseen applications in AmI), need to co-exist. Managing the levels of complexity, and gradually revealing the system's functions, are key elements to bridge this gap.

The principles of Human Computer Interaction, arguing for a good conceptual model and appropriate feedback are valid in the case of all interfaces, including EUD. The strength of the interfaces themselves, is only loosely coupled with the quality of their manifestation; the strength of the Graphical User Interface, as Norman points out (Norman, 2010), has more to do with ease of remembering

actions, what actions are possible and how to invoke them. Visibility is a generic principle of Human Computer interaction, which needs to be taken in consideration in EUD for Ubiquitous computing environments.

Application development in the case of Ubiquitous computing needs to explore the use of a small set of primitives, the suitability of different, possibly combined forms of programming, and possibly combine different notations. Form based systems can be introduced as an easy way for starting with programming of Ubiquitous Computing applications, having obvious benefits (since they provide an interaction style most are already familiar with). Forms also have significant benefits regarding transporting implementation between different platforms.

Programming-by-example techniques are a field that can be further explored, yet they are not a satisfactory solution to the issues that non-programmers are challenged with. Such techniques do not cover the spectrum of applications possible within a configurable ubiquitous computing environment, while user problems may occur from low visibility of what the system does.

Last, but not least, Ubiquitous computing research will take time to mature. In the meanwhile the profile of end user developers is shifting; people's skills, knowledge and affinity with technology and with end user development are evolving. The range and depth of skills typically available to end users acting as developers at the time Ubiquitous systems reach the market cannot be fully foreseen at the moment.

8

8. End User Development in Ambient Computing Environments

8.1. Introduction

Continuing from the previous chapter, where end user development for software was outlined, this chapter gives a summary of concepts, paradigms and mechanisms that are applicable for End User Development in the specific context of the Ubiquitous Computing Environment. Automatic interface generation, programming by example, and abstractions and metaphors, are the elements that are seen as methods applicable for Ubiquitous Computing.

8.2. Rationale: why EUD for AMI applications

In an Ambient Intelligence Environment the application and the interface tend to merge into one entity, as it is the augmented artifacts/spaces that are the access points to applications, but may also carry the application functions. As ubiquitous computing develops, prototyping tools for ubiquitous computing applications will be in demand, for developers, but also for end users. Such tools will initially be aimed to application designers so that they can participate in the development of applications that currently require a high-level of technical expertise (Li & Landay, 2005), (Mavrommati et al, 2004). End User Tools can also be appropriated to facilitate users reasoning, as well as manipulating the behavior of the AmI environment, so that they can supervise and eventually create or modify AmI applications to fit their own idiosyncratic wishes and needs. Emergent functionality in AmI can be the result of niche implementations, previously unforeseen, created by end users themselves (Drossos, 2007), (Mavrommati, 2004), (Mavrommati and Darzentas 2007). This new generation of prototyping tools can help shape the future of ubiquitous computing and eventually accelerate the development of next generation ubiquitous applications.

End User tools aimed at inhabitants of AmI environments stem from the perspective that it does not seem possible in ubiquitous computing environments to cater for all the potential needs of all categories of users (Mavrommati et al, 2004). (Rodden and Benford, 2003), (Mugellini 2007) suggest that it seems much more reasonable to enable people to cater for some of these needs themselves, and empower them - via provision of appropriate tools - to create ubiquitous applications that fit their own idiosyncratic needs. End User tools can also act as a selectively-transparent observation window into the AmI system, helping users to reason about the workings of the AmI environment. The development of such tools has to face not only the technological issues that Ubiquitous computing poses but also the

challenges that will occur due to the shifts in Human Computer Interaction inherent in the specific nature of interaction within AmI environments.

8.3. Ambient Intelligence Vision and End Users

The highlight of the Ami vision is that computers will be everywhere, in objects of various sizes, from keys to cars to buildings. These computers will be invisibly integrated into everyday life and will be supporting people in their diverse activities. The main components of this vision are (SWAMI, 2007):

- **Reliable robust hardware** in varied sizes and with long lasting power supplies (possibly self-managing or energy harvesting).

- **Wireless and wired communications** between computers, with various collaborating networks.

- **Intuitive interfaces** around the environment that are accessed by everybody (eg multimodal interfaces, various sensors, biometrics).

- **Embedded intelligence** unobtrusively reasoning about people's actions so as to provide them with services when needed or assist in controlling interfaces. From the system's perspective, intelligent automation can manage communications and maintenance (eg self-repairing).

Stemming directly from the Ambient Intelligence vision, the idea of seamless interoperation and homogeneity is the basis of an idealistic, clean and orderly UbiComp world. This turns out to be a false assumption (Bell and Dourish, 2007) as currently the UbiComp world turns out to be a pretty messy one, even in laboratory situations.

The idealistic UbiComp vision implies less direct and less conscious user input than the current systems. As Crutzen states (Crutzen 2006): *"Physical invisibility or perceptual invisibility mean that one cannot sense the AmI devices anymore; one cannot sense their presence nor sense their full (inter-)action, but only that part of interaction output that was intended to change the environment of the individual user"*.

The position proposed to resolve the contradiction between perceptual invisibility and the need to assist inhabitants in dealing with the messiness of the UbiComp world, is that the technology should reveal the system in order to motivate users to relate the possibilities of the technology to their actual needs, dreams and wishes (Crutzen 2006). As (Petersen, 2004) states *"...domestic technologies should be remarkable rather than unremarkable"*.

8.4. Approaches for accessing Ubiquity

There are two approaches regarding the visibility of AMI systems to end users. They can be seen not necessarily as opposite but rather as complementary:

1. People should not care about what's going on inside computers. Interaction within AmI should be seamless. This is generally based on the assumption that the AmI systems will be robust enough and error free, and intelligent agents will be based on the appropriate data and make appropriate judgments to appropriate actions.

2. People should be given a degree of transparency into the workings of the system. Transparency could be varied, according to the user and the context of use. (Markopoulos et al, 2004) A recombinant constructivist approach is often proposed, aimed at end users (Newman, 2002), (Rodden, 2004), (Mavrommati et al, 2004), (Merrill, 2007). These recombinant models are promoted with the claim that emerging niche applications can be achieved as

a result of people's inherent creativity and understanding of the application's building clocks. From being able to see and handle applications comes the building of trust and adoption of AmI systems. One can reason and self-assist with system failures, servicing, or safeguarding privacy.

It is obvious from the above that the human factor is a crucial element in the construction of an 'ambient intelligence' world and needs to be taken into account early in the research and development process of AmI systems. The success of ambient intelligence will depend on how individuals perceive AmI environments, how secure the AmI world is made, and how their individual rights (including privacy) are protected. If people get to trust the system and the intelligent decisions made in the background they should be willing to adopt AmI and appropriate it - via suitable interfaces. Tools for End Users providing transparency and reasoning into the workings of AmI may well provide one of the means towards that goal.

8.5. Applications and Interface Paradigms of EUD approaches in AmI

A number of existing infrastructures – such as Jini, UPnP, and others- address the configuration of Ubiquitous environments and applications. Nevertheless they are addressed to the developer rather than the ordinary person living in a ubiquitous environment. Development Environments for experts include the iQL programming model (Cohen 2002), a non-procedural language for specifying the behavior of components in pervasive environments, and Papier-Mache (Klemmens, 2004) that provides tools for programming tangible user interfaces.

Since 2002 there has been a growing number of efforts towards the creation of Tools facilitating End User Development of Ubiquitous applications (Newman, 2002), (Rodden, 2003), (Mavrommati et al, 2004), (Drossos et al 2007), (Gross et al 2007). The example of CollaborationBus (Gross et al, 2007) uses the concepts of a Pipeline

and collaboration sharing. Memodules and Accord projects (Mugellini, 2007). (Rodden et al, 2004), (Rodden et al, 2007), (Accord project website) propose a visual editor based on the puzzle metaphor. Many of these approaches (Mavrommati 2004), (Rodden and Benford 2003), (Newman 2002), create an intermediate component model allowing for the recombination of AmI elements, but also interfacing to the user with appropriate constructs. The Phidgets toolkit (Greenberg, 2001) facilitates the development of physical user interfaces by providing a range of sensor and actuator elements. iCAP (Sohn & Dey, 2003) allows users to rapidly prototype Ubiquitous Computing environments. ACAPpella (Dey, Hamid et al, CHI2004) supports context aware programming by example. An interface for creating sensor-based environments and configuring tables is provided by eBlocks (Cotterell, 2005). Similarly, e-Gadgets (Mavrommati et al, 2004) provides an editing tool for creating device associations in a home environment – the difference to most other approaches is that it is aimed specifically at the end users. The jigsaw (Rodden, 2007) is an editor for getting control over the technological home environment through assembling pieces of a jigsaw puzzle. Some systems that are based on mobile devices to control configurations are systems for PDAs and TabletPCs (Humble, 2003) while other approaches (Mavrommati et al, 2004), (Drossos et al, 2007a), (Drossos et al, 2007b) try to be device independent by separating the interface layer from the function mechanisms.

8.6. Broad perspective on AmI development tools

We can consider as overall broad categorization of tools that facilitate the development of AmI applications the following:

- Mental models (and interfaces supporting them)
- Ontologies
- Application/Software mechanisms

Mental Models

People are an intrinsic part of a Disappearing Computer environment as it is their actions and behavior as well as their needs that define the environment. The human element can be catalytic for the whole system: one of the things that can create 'emerging' and previously unforeseen functionality is the inherent human creativity and the human capability for problem solving and expression. That, however, relies on people's adoption of ubiquitous computing, and in turn the technology's understandability and openness for adaptation. Mental models can be considered End User tools, since they facilitate end users gainging an understanding of the workings of the AmI system, so that they can reason about the AmI applications. Such models need to be suitable to act both as high level technology models as well as people's conceptual models.

One such example is the adoption and appropriate adaptation of component models that allows for the recombination of functions (Drossos et al, 2007a), (Drossos et al, 2007b), (Newman et al, 2002). To enable the recombination of elements into new functions, the basic concepts and elements of a component model need to designed in a way that they are capable of being easily communicated to people, so that there is a degree of transparency into the – otherwise invisible - workings of a ubiquitous environment. This can be done by an appropriately designed conceptual model – that embodies the basic technology concepts which allow for inter-associations of artifacts (Mavrommati et al, 2004). An example of a high level programming model that provides a conceptual abstraction allowing end users to describe Ubiquitous scenarios is described in (Drossos et al, 2007b).

In fact, such model acts as a high level interface for the user within a ubiquitous computing environment; it acts as a communication layer which people can understand, and by having access to it they can manipulate the (otherwise) 'disappearing computers' within their environment. The creation of such models as interfaces, for the broader interaction with ubiquitous computing environments, often proceeds in parallel with the creation of middleware, that acts as a bridge

between core technology layers (such as protocols, communication etc), devices, and people.

To support such mental models, the glimpse into the ubiquitous system is often decoded in metaphors. Metaphors stem from already existing (non-ubiquitous) widely recognizable paradigms that imply interconnectivity. Examples that imply interconnectivity can be appropriate familiar terms - like the verbal term 'Plugs' used in (Drossos et al, 2007), or familiar images, as for example a 'Puzzle' (Mugellini, 2007), (Rodden, 2003), (Humble, 2003) - which can be used in the context of Ubiquitous Computing.

Ontologies

An ontology can provide a common basis for communication and collaboration between heterogeneous artifacts and AmI environments. The ontology describes the basic conceptual terms, the semantics of these terms, and defines the relationships among them. It is therefore fundamental for the creation of ubiquitous applications, and can be considered a tool, in the broad sense of the term; see: (Christopoulou and Garofalakis, 2010), (Christopoulou et al, 2005), (Goumopoulos and Kameas, 2009).

Application/Software mechanisms

What is generally understood by the term 'Editing tools' is application mechanisms that support the establishment and management of applications. With a range of external devices, the 'Editors', people can supervise available sources (artifacts, services, etc) and create associations between them, thus making AmI applications. These mechanisms' core structure can be independent of particular modalities (Mavrommati, 2004) so that various point-application editors can be implemented with a variety of multimodal interfaces, and in a variety of devices.

8.7. Mechanisms and resources for EUD in AMI

There are three basic questions that we can ask regarding the software mechanisms for End User Tools for creating AmI applications:

- What resources are available as from the environment to the Tools?
- What should the Tools do upon the environment?
- Which elements that may come into play can be considered for these Tools?

Available as resources from the environment to the Tools are (Olsen, 2005):
- Multiple sensor approach
- Ambiguity of input
- Diversity and distribution of computing platforms
- Diversity of contexts of use
- Diversity of users
- Amount of Data
- Interactions available in the environment

The handling of these resources can lead to a number of challenges, the most prominent of which is *context awareness*. Such resources can also provide the input needed for the concept of *computing ahead* the next likely editing stages, so that guidance in editing actions is provided by the software tools.

Context-aware applications are one of the most important forms of next generation interactive systems. As ubiquitous computing develops, prototyping tools for context-aware applications will be in demand. Such tools can also help produce more usable context-aware applications in an efficient way (Yang Li, 2005). Context awareness therefore has to be used by editing tools, as it has to be specified or configured, so that it can then be used as an element of the AmI applications that are being created.

Decision making is a related aspect according to which the developer or advanced user may want to dynamically define or change the rules determining an individual artifact or even an application's behavior, in a high level manner. Dynamically defining the parameters for an application is another aspect that can be defined or altered using tools aimed at developers. As (Drossos, 2007b) reports, a tool providing a GUI for creating or changing rules can also provide the advantage that rules can be dynamically altered in a high level manner without disturbing the operation of the rest of the system.

Assuming substantially more computing power in the application context of distributed, multimodal sensing and recognition techniques we might want to consider constantly *"computing ahead"* – modeling the user's actions and pre-computing the results along perhaps the five most likely next inputs - as Scott Hudson states in (Hudson, 2005). This can be used to provide new kinds of feedback as well as shortcuts for the user, but can also enable proactive pre-fetching from other media. According to Hudson, related to the concept of "computing ahead" is the notion that we should move from the idea of a single state of a system or object of interest to maintaining multiple alternative states simultaneously. This will be useful both in interesting new interaction techniques which allow 'what if' explorations to happen naturally (SWAMI report) and as basic support for dealing with ambiguity and asynchrony. Several challenges may arise from this proposed approach that can lead to sophisticated models of probability - which in turn could be based on machine learning approaches for adaptation to users and tasks, and models of ambiguity of inputs - so that it is made easier to deal with (Mankhoff, 2000).

Collaborative sharing of applications can provide the supporting means for community participation, for adopting, encouraging and supporting the creation of AmI applications. In the CollaborationBus example (Gross et al, 2007), three types of sharing of compositions are possible: *event sharing* allows users to either share events from their own sensors or processed event data from their filters; *actuator*

sharing allows users to share the control of a personal actuator with other users, so that other users can send commands and control the system behavior; *pipeline sharing* allows sharing of complete compositions with others. Efforts towards sharing of pervasive awareness applications by communities of end users are also pursued in the ASTRA project, see (Astra D4, 2009b).

Clearly, not all people will be inclined or able to create or even configure an AmI application. Enthusiasts and motivated individuals should take the lead in addressing niche needs 'bottom up'. More diversity and business opportunities can rise in the long term from such a shift in perspective.

8.8. Various Concepts for Tools Interfaces

Elements that can be considered for the interfaces of AmI editing tools are:

- Automatic interface generation
- Programming by example techniques
- High level abstractions
- Metaphors

Several concepts are described in the report of the Future of User Interface Design Tools workshop (Olson, 2005) at CHI2005, which, although more generally aimed, can provide food for thought in the area of End User Development for AmI.

8.9. Automatic interface generation

Automatic interface generation requires specific information about the user and the current situation to be incorporated into the design of the user interface. A user interface could be displayed on whatever device the user has available. Another

possibility is that the interface could use familiar elements that the user has encountered recently, and personalize according to the user's profile. Model-based systems attempt to formally describe the tasks, data, and users that an application will have, and then use these formal models to guide the generation of the user interface (Nichols, 2005).

Systems can use this input to automatically design the user interface, or to design assistance to people. When a user requests an interface to control an appliance, the user's device downloads a functional model from the appliance and uses that model to automatically generate an interface. Although there have been successful developments in limited domains (i.e. remote controls), it is noted that model-based user interface tools have not become common (Nichols, 2005). Nevertheless, assuming a manageable scope of foreseen interfaces for tools, we should note that those model-based techniques may hold potential for the interface instantiations of AmI tools.

8.10. Programming by example

The 'programming by example' techniques and intelligent agents can help users with routine complex tasks. 'Programming by example' is a demonstration approach that requires an initial time for extended observation of relevant sensor values. In the latter learning phase, the users specify relevant sensor events so that appropriate artificial intelligence algorithms can detect patterns in the observed sensor values and automatically execute desired actuators (Dey, 2004). 'Programming by example' tools hide most specific details of the underlying mechanisms from the users but this reduces the barrier for nontechnical users to configure ubiquitous computing environments. Camera-based technology can be considered as providing visible and accessible sensors within the AmI environment. Camera-based technology has been introduced to the home due to recent commercial developments - such as Microsoft's Kinect sensor, a depth camera combined with spatial audio

input to do voice/face recognition and motion capture, originally developed for Microsoft's X-Box360 game platform (see KINECT website). Although at the moment the use of such technology in the home pertains to gaming purposes (having the limitation of a restricted viewing angle), we can expect that such developments will make it easier to observe gestures and actions upon an AmI environment, get people more accustomed to camera sensors, and make programming by example more prominent. On the other hand there are issues regarding reasoning about the behavior of the intelligent environment with users requesting more visibility into which decisions were made and why.

Programming by example techniques cannot remedy all cases of application configuration. There are situations where there cannot be a task example performed - because the application splits between different locations, different time periods or in situations that cannot be replicated (for example, where the application pertains to many people present, to a certain combination of outside humidity and temperature, or a specific point in time, where, in fact, the application creator does not actually need to require the availability of the above in order to configure an application). Although programming by example provides, at a syntactic level, ease of use for end user programming of AmI applications in specific cases, it cannot be generalized as an interaction form in broader aimed end user tools, as more abstract high level semantic ways of description of applications are needed. Nevertheless, it can prove a useful complement to the tools' functions as one of the alternative syntactic methods available.

The IST-FET project e-Gadgets explored the use of agents to programme the behavior of an ambient intelligent environment (the 'i-dorm') and combined this approach to an end user programming approach using Graphical User interfaces. While there is a lot of appeal in the programing by demonstration approach, the same problem was identified (Markopoulos et al 2003) as had been noted by Myers, Ko and Burnett (Myers et al, 2006): *users need a way to observe what the agent has learned and reason about the agent's actions.*

People need to control the learning of the agent, as well as to be notified when the system learns. This implies that a combination of two contradictive approaches may be applicable: allowing for seamless intelligent configurations (in a 'black-box' approach), while when aided by intelligent configurations being able to observe this and allow for a level of transparency into the agent's learning and reasoning, as well as intervening and revealing the more detailed workings of the system on demand.

8.11. High level abstractions

In prototyping an AmI application, designers and end users need to explore the large number of ubiquitous inputs and specify the contexts of use. A design tool can facilitate this task by providing *high level abstractions*. In Topiary (Yang et al, 2005) a map abstraction was used to represent spatial relations of entities and thus allowed designers to capture location contexts of interest by demonstrating scenarios. In e-Gadgets on the other hand, a high level conceptual model was used to explain the workings of the system to the users and enable them to make connections. In this approach more complex artifact behavior can emerge from interactions among more elementary artifacts. This approach, can scale both 'upwards' (towards the assembly of more complex objects, i.e. from objects to rooms, up to buildings, cities and so on) and 'downwards' (towards the decomposition of given gadgets into smaller parts) (Mavrommati, 2004).

A high-level mechanism to abstract context, that also allows the rapid construction of ambient computing applications, is presented by (Jacquet et al, 2005); this is complemented by a clear conceptual model for AmI. *A well-defined vocabulary is used in this model that tries to map the physical and virtual world to elementary component objects which can be interconnected* in order to create AmI applications. The criticism is (Drossos et al, 2007b) that this architecture seems to limit the representation of the real world, in all its richness and complexity, to sets of sensors; that in turn restricts the model's scope, as well as the autonomy of the components.

8.12. Metaphors

Metaphors are a commonly used way to facilitate the use of a tools based on paradigms that are known and familiar to the user through his or her own (real world) experience. Various forms of connectible puzzles are metaphors often used in editor's user interfaces. A *fridge magnet* metaphor is used in (Truong et al, 2004) while a *browser* approach is used in the Speakeasy system (Newman, 2002) - where components are connected using a visual editor based on file-system browsers. Specify the application's behavior by assembling metaphorical pieces of a *jigsaw puzzle* is often used, (Humble, 2003) (Rodden, 2004), (Mugellini, 2007) as it is immediately intuitive, offering a recognizable analogy for connecting services. Sensors as well as devices are represented by puzzle piece-shaped icons that the user 'snaps' together to build an application. Nevertheless, this metaphor can restrict the potential for development and richness of programming expression (Rodden 2004 p.74), (Drossos, 2007). The interactions are simplified to sequential execution of actions and reactions which limits the potential to express many of the user's ideas, while the non-existence of emergent properties and the absence of rule based logic, can result in very simplistic application behavior. Other representations are needed to give users more control over their application than the metaphor of jigsaw pieces allows.

The *Pipeline metaphor* is used by CollaborationBus (Gross,2007), using nested and parallel pipelines allowing logical (AND, OR, NOT) conditions; in the pipeline metaphor, sensors are the sources of any event in the pipeline, filters represent single conditions, and actuators (software or hardware) are at the end side of the pipeline.

8.13. Conclusions and challenges

Editing tools are required to manage the ubiquity and understand the logic of the AmI environments. Several of these tools are addressed more to the developer or the advanced user rather than an everyday end-user. The purpose of these tools is to configure certain aspects of the system's behavior, and implement certain applications within AmI environments.

Such tools need to allow for utilizing context awareness, but also provide adaptive interfaces and propose alternative syntactic methods to cater for a variety of user profiles. The consideration of modalities (augmented reality, gesture, or speech interfaces for example), poses a number of challenges to the design of editing tools. Robustness and adaptation to changes, like different environments and infrastructure, being able to dynamically define the parameters of an application, providing many perspectives for accessing services, are but a few of the challenges posed.

Further challenges involve developing techniques for providing support for debugging and testing the applications build, providing feed forward, as well as feedback, and allowing for transparency into the workings of the system, as well as the design and development of the system and its tools so as to prevent errors, minimize them, tolerate them, and recover from them – all within the ubiquitous computing infrastructure.

Tools aimed at end users need to be researched on their own merit, as they constitute a very important part of the Ambient Intelligence vision. The research community needs to work on defining a roadmap towards appropriate, efficient and effective mechanisms for the AmI editing tools by end users as well as application designers and developers.

9

9. A proposed model for the recombination of artifacts

9.1. Introduction

A model is presented in this chapter that augments physical object of everyday use into Ubicomp artifacts and that includes in its conception people as a part of the system. The proposed model bridges software architecture constructs and people's conceptual models of ubiquitous computing artifacts, by introducing the concept of affordances to the Publish-Subscribe model.

A 'black-box' engineering approach - whereby people are not able to observe the structure of the system- is not adopted in this research, but rather a transparent approach which partially discloses the structure of the system. A model that can act as a conceptual aid for people to understand the ubiquitous applications, and how to manipulate them, is an important step towards transparency. In later end-user trials the validity of a transparent approach for Ubiquitous computing engineering is assessed. The same concepts and constructs are provided at a more detailed level to the manufacturers and builders of the applications, and may also be used by intelligent mechanisms to adapt them in a black-box type approach. The approach also proposes Editor mechanisms towards the (re)configuration of Ubiquitous Environments and applications.

9.2. Background summary

As was explained in previous chapters, people are an intrinsic part of a Disappearing Computer environment, as it is their actions and behavior, their nature and their needs, that define any living environment. The human element can be catalytic for the whole system: one of things that can create 'emerging' and previously unforeseen functionality is the inherent human creativity and the human capability for problem solving and expression. This relies on people's adoption of ubiquitous computing technology, and in turn the technology's understandability and openness for adaptation. An interaction model for recombinant ubiquitous computing is

proposed, whereby people can get a conception of the workings of the system, and augmented everyday objects can be manipulated by them and combined into applications.

A number of information appliances and 'smart' products from the consumer electronics and white goods industries are gradually being introduced into consumers' homes. Mundane objects of everyday use that are enhanced with computing and communication capabilities are being developed experimentally while some are appearing in the market already. Some examples of such processing and communication enabled products are the Internet fridge, internet microwave oven, UPnP hi-fi sets and DVD players, sensor enabled toys such as Furby, Nabaztag and other pet or robotic toys, tagged coffee mugs, digital picture frames, sensor/internet enabled furniture, to mention but some. It seems that our future environments will consist of an increasing number of objects, furniture and appliances that will be enhanced with Information Communication Capabilities, which will be able to work synergistically with each other (Aarts, Marzano, 2003). The future home is expected to become populated by distinct devices that are interconnected via an invisible web of network based services (Aarts, Harwig, etc, 2001), (Harwig and Aarts, 2002). The research challenges relating to creating this future home environment are not only technological (how do the objects and appliances share common standards, how they communicate robustly with each other). Equally important are the challenges that relate to acceptance of this new metaphor by people and the resulting human behavior (how do people develop an affinity with these environments; how will the mental models of their environments be re-shaped; what are the skills they will develop to cope with behavioral dependencies between objects that are not always visible) (Mavrommati, 2003b).

Several research projects are currently under way aiming to develop the necessary hardware and software modules that will enable devices to communicate and collaborate (Disappearing Computer site), (Humble et al 20003), (Newman et al, 2002); thus we can safely assume that future in-home devices will be functionally

autonomous but able to synergize with each other (Figure 22), regardless of the differences in their shape and functionality (Mavrommati, 2003b).
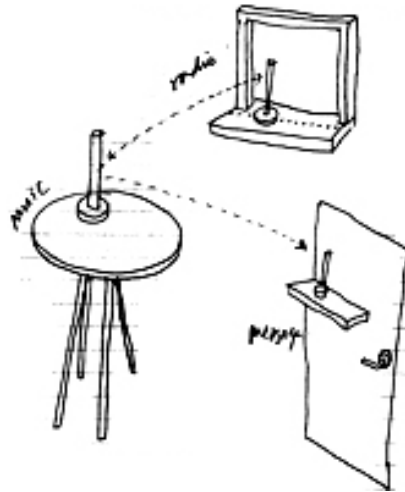
Figure 22: Different artifacts can communicate and synergize with each other

9.3. Assumptions

In order to achieve the aim to facilitating human innovation in a ubiquitous computing environment the definition and adoption of appropriate models is necessary. Such models need to be suitable to act both as high level technology models as well people's conceptual models for the ubicomp system.

One example is the adoption and appropriate adaptation of component models, which allows for the recombination of functions. To enable the recombination of elements into new functions, the basic concepts and elements of the component model needs to be designed in such a way that they are capable to being easily communicated to people, so that there is a degree of transparency into the (otherwise invisible) workings of a ubiquitous environment. This can be done by an appropriately designed model, which carries along the basic technology concepts that allow for inter-associations of artifacts, as well as supporting mechanisms and terms to carry out this manipulation.

In fact, such a model acts as a high level interface for the user within a ubiquitous computing environment; it acts as a communication layer, which people can understand, and by having access to it they can manipulate the 'disappearing computers' within their environment. The creation of such interfaces, for broader interaction with ubiquitous computing environments, goes hand in hand with the creation of middleware, which acts as a bridge between core technology layers (such as protocols, communication etc), devices and people. To develop such middleware, human centered notions as well as technology requirements have to be equally considered from the very start of defining the system's concepts. (Mavrommati and Kameas, 2002).

To support such models, metaphors may be used that stem from already existing (non-ubiquitous) widely recognizable paradigms which imply interconnectivity. Examples implying interconnectivity can be the appropriate terms used, such as the verbal term 'Plugs' (Kameas et al, 2003) (Kameas and Mavrommati, 2005), (Mavrommati et al, 2004) or familiar visual metaphors - such as a 'Puzzle' (Rodden et al, 2004), (Accord Project website) - which are used in the context of Ubiquitous Computing. Interconnectivity is not the only thing in question; what is to be interconnected is equally important. The nature of services, capabilities, sensor data, state of the service or application have to be specified, -with people's understanding in mind, into the technological constructs, in order be able to clearly communicate to people. In this research case, focus was on *physical objects that have computing and communicating capabilities not readily perceptible in an ordinary living environment*, and specific consideration was given to the specifics of the nature and perceived usage of these artifacts. The selection of physical everyday objects, in turn, adds some new elements to existing ubiquitous computing research (as it is the 'information appliances' that are more readily associated with Ubiquitous computing since the concept's introduction by Mark Weiser, and later, the 'Invisible Computer' by Don Norman). People use objects (such as surfaces, tables, floors, lights, cups) in other ways than they handle services or information appliances (such as TV, printer, computers, stereo sets). Moreover the interface characteristics of

ordinary artifacts are different to those of information appliances; the variety in their nature of use, shapes and sizes, their affordances (implied by their design characteristics), their potential for different uses (stemming from their physical shapes, and history) and their interfacing capabilities (often objects are not suited/do not have visual feedback or screen capabilities to assist with interfacing with them, as most information appliances have) (Mavrommati and Kameas 2003b). Adopting a different focus, on augmented ordinary physical objects, adds to existing Ubicomp research which has a predominant focus on information appliances.

It has to be noted at this point that a human environment populated by augmented objects poses HCI with several open questions: issues of acceptability, visibility, reversibility of actions, syntactic correction of actions, intelligence, trust, safety, to mention some, and last but not least the definition of appropriate HCI theoretical frameworks for understanding and exploring Ambient Intelligent Environments (such as (Bellotti 2002), (Scholtz and Consolvo (2004)). These issues have to be noted as important elements for different research paths in the area of augmenting human environments and objects.

9.4. Methods proposed

Ubiquitous computing systems are characterized by distributed resources (in the form of physical artifacts). Recombination of artifacts' services / capabilities is considered an appropriate approach that suits the nature of ubiquitous computing environments **(**Edwards et al 2002), (Newman et al 2002), (Kameas et al, 2003**).** One method of achieving this is the definition of an architectural style (middleware) that supports the underlying technology as well as human centered notions (Kameas et al, 2003), (Kameas and Mavrommati, 2005), (Drossos, Mavrommati, Kameas, 2007b) in order to combine the capabilities of artifacts.

An architectural style can be characterized as a conceptual and technological framework for describing and manipulating ubiquitous computing applications (Figure 16). An Architectural style typically consists of: a vocabulary and layers of semantic associations between terms, sets of configuration rules, and a technical infrastructure to support it. Moreover, a style can use terms and concepts that serve as a common referent between Ubiquitous computing systems, designers, and people (Figure 17). In order for the architectural style to be enabling to people, it needs to be supported by tools that are part of the system. Syntactic and interpretation mechanisms, editing mechanisms, methodologies for artifact creation all have to be intertwined and integrated into the middleware, as well as into the adopted model that enables human understanding of the workings of the system.

An Editor (Figure 25) is also added to the picture of the Ubiquitous Environment, as an external super-control device that is needed to review and associate Artifacts and their collections (Mavrommati et al, 2004), (Mavrommati and Kameas, 2003). The Editor can be a software program that is run by an information appliance (for example, a mobile phone, tablet, or a laptop/PC can be used as editor, with the implication that the information appliance running the operating system is equally, by doing so, an artifact within the system).

9.5. Models, Abstractions, Affordances

Augmented artifacts can be associated by people using the straightforward model of **Connectable Capabilities** (Plugs) and **Links** (Synapses) occurring between them (from now on referred to as the Plug-Synapse model). Capability-Plugs are software classes that make visible the artifacts' capabilities to people and to other artifacts. They can correspond to artifacts affordances relating to the embedded sensors the artifact may have (eg containment, surface, pressure, temperature etc) or to services they may provide (eg light, sound, etc). An artifact has physical properties, as a result of its tangible self, while it offers services, resulting from its digital self. These connectable 'capabilities' (both physical/sensory and digital service) are

expressed as 'Capability-Plugs' (otherwise called 'Plugs'). Then, a Link (otherwise called a 'Synapse') is an association between two compatible Plugs (Figure 23). This is an invisible link (wireless or wired) explicitly created to achieve a particular working of the two capabilities together. Synapses are formed using an external overview device, the Editor (Figure 25), making end user programming possible. The editor is used to review artifacts and their collections and associate them together (Mavrommati et al, 2004). By using the proposed abstractions of connectable Capability-Plugs and Links-Synapses, one can combine the augmented objects and build new types of applications (functional groups of artifacts' associations) by connecting their connectable capabilities in synaptic associations, and then describing the properties of these associations.

The term 'affordance', refers to the opportunities for action provided by a particular object or environment (Gibson, 1966), (Norman, 1990). The affordances stemming from the physical characteristics of the object and its perceived use are the connectivity elements (called 'Capabilities', or 'Plugs' in the model); they act as an in-between layer between sensors and connectable usage that people can relate to and use.
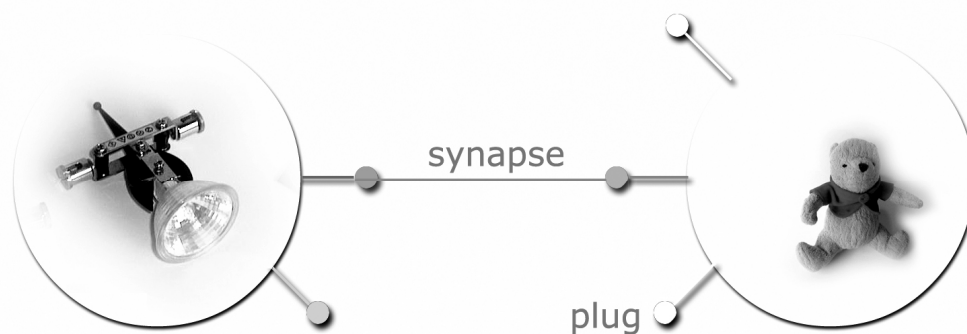
Figure 23: The Plug-Synapse model: The artifacts' capabilities (Plugs) can be inter-associated with invisible links (Synapses) to form ubiquitous computing applications

In the proposed model, physical objects can be adapted to be able to act as components of an augmented Ubiquitous computing environment. Their physical properties and characteristics can translate into connectivity possibilities (Mavrommati et al, 2004). Introduction of the idea of *affordances* (stemming from objects' physical disposition, sensors and perceived use) enriched the Plug-Synapse construct. Such affordances have the potential that they can be used via an ontology, which provides a second level of semantic interpretation of the physical characteristics. Thus the system abstractions of Capability (i.e. 'Plugs') and the Ontology relate to the affordances that stem from the physical characteristics and perceived use of the object.

According to D. Norman (Norman, 1990), affordances *"refer to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used"*. Up to now, the ways that an object could be used and the tasks it could participate in have usually been determined by these physical affordances, that are, in turn, determined by its shape, material, and physical properties. By giving to the physical objects the possibility to act as components of an augmented environment, and to interconnect (via synaptic links) to form ubiquitous applications, the objects acquire in fact new affordances, that are given to them via their digital self. The ways that we can use an ordinary object are a direct consequence of the anticipated uses that object designers 'embed' into the object's physical properties. This association is in fact bi-directional: the objects have been designed to be suitable for certain tasks, but it is also their physical properties that constrain the tasks people use them for (and therefore define their use in ubicomp applications too).

Due to their 'digital self', artifacts can now publicize their abilities in the digital space. These include properties (what the object is), capabilities (what the object knows how to do) and services (what the object can offer to others). At the same time they acquire extra capabilities which, during the formation of ubicomp applications, can be combined with capabilities of other augmented objects or

adapted to the context of operation. Thus, augmented objects have two new affordances to their users:

Composability: Artifacts can be used as building blocks of larger and more complex systems. Composeability is perceived by users through the presentation - via the object's digital self, presented in Editors - of the object's connectable capabilities, in this way giving users the possibility to achieving connections and composing applications of two or more objects. In implementation terms this is achieved via a communication unit that artifacts have, which in turn requires universal descriptions of tasks and services.

Flexibility: artifacts that have embedded digital storage - or have access to it - can change the digital services they offer. The tangible object can thus be partially disassociated from the artifact's digital services (for example, an object with an RFID tag can use processing that is located elsewhere). Flexibility is the result of this disassociation of the software and the digital self of objects.
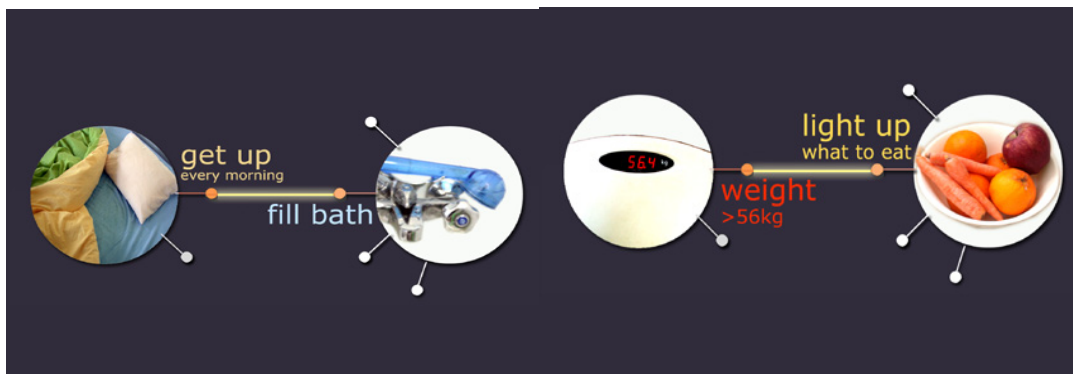
Figure 24: Graphical examples of two applications (that are sets of functional links between augmented artifacts).

These new characteristics of augmented objects need to be somehow indicated to people. A way to do this is via a special purpose software mechanism that can run in several interface modalities and devices. Such mechanism/tool can be generally

referred to as the 'Editor' (Figure 25). In addition, according to manufacturers or designer's choice this affordance of connectivity can also be indicated by such elements of the object's physical design as the material shape, auditory, or haptic, or else by those visual design elements for control and feedback that present the object's digital interface. Such elements depend on the exact nature of the object and result from its industrial/interaction design (this involves the specific design briefing and is dependent on the design constraints and the designer's expertise).

9.6. Concepts, constructs and application

A high level recombinant model (the Capabilities and Links model) was adopted and adapted (based on the Publish-Subscribe model), and provided an enabling conceptual abstraction of the Ubicomp system to the Users (Figure 17). It has to be noted that the e-Gadgets project drew initial inspiration from the e-Slate component platform for educational applications (e-Slate website), a platform that applied component architecture to specific educational applications called "microworlds" (Birbilis et al, 2000), (Roschelle et al, 1999); e-Gadgets has adapted concepts of component architectures, manipulation and microworlds, for the ubiquitous computing environment, whereby the components are the tangible artifacts, and they are selectively composed into ubiquitous applications ('Gadgetworlds'). A middleware software layer, (the Gadgetware Architectural Style, or GAS (Kameas et al, 2003), (Drossos, Mavrommati, Kameas, 2007b), (Drossos et al, 2007) - managing software resources, implementing the connectable capabilities of objects, supporting service discovery and management of associations - has enabled artifacts to become components of Ubiquitous Computing systems. A supporting ontology provided a common basis for collaboration between heterogeneous artifacts (Christopoulou and Kameas, 2005), (Christopoulou et al, 2005), (Goumopoulos and Kameas, 2009). About a dozen, varied ordinary objects were augmented with computing and sensing capabilities, for demonstration purposes in order to test them with end users. The objects were varied in size and affordances; they included furniture, room sensors

(such as for temperature and humidity), a pressure sensitive carpet, a bed, a desk, chair, a few books, lamps, music player, a cube lamp (changing colours when flipping side) and a clock. Editing tools, in the form of an application mechanism that supports the establishment and management of applications based on this model, with a core structure that is independent of particular modalities, were implemented. Editors were implemented with various interfaces (two different interface versions were built and tested: one running on a PDA and one for the PC). With the 'Editors', people can supervise artifacts and create/edit associations between them, thus creating/altering ubicomp applications (Mavrommati and Kameas, 2002), (Mavrommati and Kameas, 2003), (Mavrommati et al, 2004). Examples of interfaces for working editor prototypes were used to facilitate the end user evaluation sessions, in the course of the e-Gadgets project (see chapter 11).

9.7. The Editor role and functionality

People can supervise the functional capabilities of devices in the home via other specialized devices, the Editors. GAS Editors allow users to do this with an integrated graphical user interface; potentially, Editors may be linked to multimodal interfaces integrated in the environment, such as speech or gesture input systems. With Editors it is possible to supervise and interface with the various augmented artifacts. In this way one can supervise the inter-connectable capabilities of an appliance (the Capability-Plugs) or be given the possibility to enrich them. Finally, people can act upon these capabilities by associating them together into matching pairs that serve specific functions (the Links/Synapses); they can break existing associations, pause them, or add parameters in the association to influence the details of the function (Figure 3). The synaptic associations thus created or edited are not stored locally, (i.e. in an Editor), but in the e-Gadgets project's case are stored in the distributed appliances and objects within the home environment. Thus, in the case of failure or object movement beyond network range, much of the application functionality can be restored.

Editing capabilities can be used by designers to create Ubiquitous Computing applications without having to start from scratch, as they can reuse existing component objects. Editors may equally be used by end users to personalize ubiquitous applications, or for using their own creativity in creating novel associations for niche or innovative functions. The core editor functionality can also be accessed directly by intelligent agents that construct and adapt applications by monitoring user behavior. Potential uses of the editor can include the finding and customizing collaboratively created applications from communal application pools (as was explored in the case of the ASTRA project, (see chapters 5 and 13)).
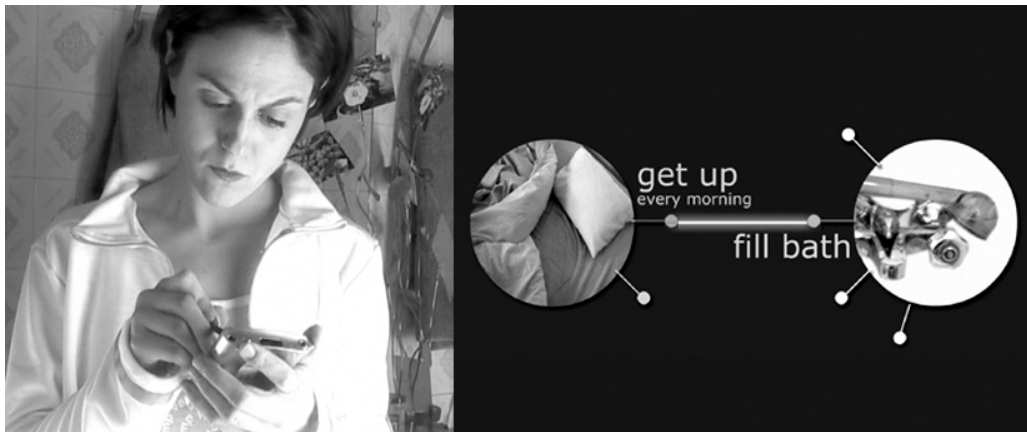
Figure 25: People can create certain associations between artifacts. The Editor is a overview/ control-device used to (re)design applications within a ubiquitous environment, using artifacts as a starting point.

For classifying capabilities (plugs), we initially speculate on patterns of usage of these artifacts by people, and address the issue of how people perceive artifacts. In this way we indirectly address the issue, which set of peoples' actions should artifacts perceive. Capability-Plugs can be classified into higher and lower level plugs. Lower level plugs represent a single capability (ie a certain kind of sensor), while a higher-level plug is manifested to the user as a single capability-plug that represents a higher-level function (and is in fact an organized collection of more basic plugs). Capability-plugs can be hierarchical, and they can also move from being a lower level plugs (with a *quantitative* capability) to higher level plug (with *qualitative*, more abstract goals related to human conditions).

9.8.   Affordances as Connectivity Plugs

A valid and common engineering approach would be to model the ubicomp system and its subsystems (ubicomp sub-applications) as virtual artifacts, where (sub) system parameters and trigger conditions (input events) are exposed as input plugs, and output state, together with outcome events (resulting functionality) are exposed as output plugs.

While software engineers and programmers would be at ease with such a conceptualization of a ubicomp application (potentially complex enough to be comprised of several, even nested, sub-applications) the average end user will have difficulty to reason about and constructively manipulate a non-tangible entity. Moreover this type of representation would naturally cause an asymmetry between the number of objects perceived by the user in their surrounding environment, and the representations displayed in the context of an editor's interface, since the editor would have to display the system and sub-system virtual artifacts as well.

A higher level, but more user friendly approach to system towards system conceptualization by end users, is suggested, where plugs of sub-systems respective virtual artifacts are hosted as dynamic plugs at tangible augmented artifacts of the system. Such plugs are dubbed as 'dynamic' since they stay attached to the respective tangible objects only in the context of their participation in the specific (sub-) system.

This approach maintains symmetry between the tangible end user environment and the virtual ubicomp system context, enabling more fluent direct manipulation in editor interfaces and allowing for natural interaction paradigms such as programming by example.

The proposed dynamic plugs correspond to affordances of the augmented artifacts in the context of the specific ubicomp application (sub-system / virtual artifact).

Benefits of virtual artifact plugs:

These affordances (that can be considered as 'hidden affordances[3]' (Gaver, 1991) , more on which is explained in chapter 14),signify perceived potential usage of the artifact in a given situation or environment (context of use). By exposing such affordances – as designers hint design objects - on reusable artifacts, the end user applies end user design in practice, allowing for more comprehensive repurposing of the given artifacts by people interacting in this environment.

This approach also enables the case where an artifact itself carries with it one or more persisted applications that are portable. Such applications have as only member this artifact. These applications can extend permanently the respective artifact by exposing more affordances on it irrespective of the environment of use. Persistence of the respective logic can be done on the artifact itself, provided enough internal data storage, or be hosted at a cloud based repository with the artifact storing a reference ID to those applications in the cloud.

When such artifacts are detached from the context of the ubicomp application they stop hosting these dynamic plugs and thus they no longer present their respective affordances, since they can only be perceived in the specific ubicomp environment.

---

[3] Hidden affordances are technology affordances not always directly and immediately perceived (Gaver, 1991).

The aforementioned concepts can be summarized as follows:

1. System as artifact
2. Hosting system virtual artifact plugs onto real world augmented artifacts
3. Affordance is specific subsystem context and dynamic plugs.

To explain the above concept, the following scenario is considered: In an augmented home, a sofa is alternatively used as a bed for guests. Aided by a camera-based recognition system that watches how the sofa is being used, a ubicomp application is defined that exposes the affordance of a bed as virtual plugs on the sofa. If the sofa is moved to another location, nota accessible by the vision system, the respective dynamic plugs are automatically removed from the sofa-artifact. So, when the sofa is being used as a bed, other ubicomp applications can be set up to trigger appropriate actions (like automatic dimming of the lights, or adjusting the room temperature).

9.9. Issues

Self-explanatory terminology: The Capabilities-Links model was initially referred to as the 'Plug-Synapse' model, the reasons for this being partly historical, since initial inspiration was drawn from the e-Slate software architectural platform for educational purposes (e-Slate website), and then transported to the context of ambient intelligent environments in the course of the e-Gadgets research project. Nevertheless the terms Plug and Synapse, applied to augmented tangible objects, were judged by experts in the course of validation and appraisals (see chapter 1 and related appendices 1 to 4) as too unclear, and much to be too technical verbal terms. As a preferred, more suggestive term, instead of 'Plug' for the general public (in the English language) the terms 'Connectivity', 'Capability' or 'Connectable Capability' have been proposed. Additionally, the more straightforward words 'Link' or 'Association' are preferred to the term 'Synapse'. These suggestions were

the result of brainstorming on the terminology during an expert workshop. The model is subsequently better referred to as the Capabilities-Links model. Nevertheless, the terms Plugs and Synapses gained the inertia of habitual use (in e-Gadgets project research and publications), and so proved difficult to replace. For this reason the two pairs of words have been linked to form the new combined terms Capability-Plugs and Synaptic Links to express the model. These not only avoid confusion by a complete change of terminology, but are also considered better and more self-explanatory for addressing end users with, than the initial terms.

People's expression: In articulating an application, a person will express his or her own goals and intentions, rather than how the information is organized or presented. For example:

'I would like to have music in the house when I come in'.

In contrast, the Capabilities-Links model requires the human goal to be decomposed into available artifacts and their capabilities, then synthesized to form a collective artifact behavior:

'Player>start playing music>when>ID Anna> steps on doormat'.

However adept human beings are at using systems and notations, the above line does not come instinctively as human thought. Nor it is easy to grasp and configure applications on paper using this model. However, using multiple representations for editing, including natural language for expressing intentions, and then defining the specifics by help of an expert system, could ease this gap between natural expression of intentions and the application configuration. The model still remains valid as a way to explain and express the internal workings of the system, and being able to reason between multiple representations.

Visualizations and representations: the Capabilities-Links model can act as a background model for people to understand the technology, but may or may not be

used as a direct visualization. The 'Bubbles' concept (Daskala and Maghiros, 2007) or Tag Cloud visualizations, may prove to be suitable visualization alternatives. The Capabilities-Links model can be at their conceptual basis but manipulation can be done via different visual approaches.

9.10. Conclusions

In this chapter a conceptual model has been presented that can be used to understand as well as manipulate ubiquitous computing applications. This conceptual model maps directly on the 'publish-subscribe' model: augmented artifacts are seen as components of the ubiquitous environment. Artifacts (in a component based approach) can associate with each other, via invisible Links that people create between them, and specifically between their digitally expressed connectable capabilities (or Capability Plugs). The digitally expressed capabilities of artifacts are treated as new artifact affordances. The Capabilities-Links model provides an easy to explain, understand and retain conceptual schema that can be directly used for the creation of applications by special purpose devices, called Editors.

10

10. Application of the Capabilities and Links model: the e-Gadgets case

Parts of the content of this chapter were published in the following:

Kameas, A., Mavrommati, I. (2005). Extrovert Gadgets. Configuring the e-Gadgets, Communication of the ACM (CACM), vol. 48, no. 3, p.69.

Drossos, N., Mavrommati, I., Kameas, A. (2007). Towards ubiquitous computing applications composed from functionally autonomous hybrid artifacts. Disappearing Computer book (eds: Streitz N. Kameas A. Mavrommati I.), Springer Verlag, LNCS.

10.1. Introduction

The model proposed in previous chapters was applied during the 'extrovert-Gadgets' (e-Gadgets) project. The case for its application is described in this chapter. At the time of preparing this text, several research projects are taking place in the broader area of ubiquitous computing systems. Most of them focus on specific, mostly technological parts of the architecture. In the e-Gadgets project the effort was to provide a vertically integrated system that would offer a complete technological solution while taking into account the user perspective.

10.2. An infrastructure which supports communication

Artifacts are considered components of the in-home environment, which can be freely associated in several different ways, thus collectively achieving different functions within the home (eGadgets project), (Newman, 2002) (Mavrommati and Kameas, 2002). The functionality of collections of objects can serve different purposes: pleasure, ludic use, or home automation and task facilitation.

The ability to 'configure' and 'reconfigure' (Newman, 2002) those in-home devices stands as a driving concept behind this vision as it allows for end user creativity to emerge in a ubiquitous environment, where people can create their own niche applications or adapt their ubiquitous surroundings (Mavrommati and Kameas 2002), (Mavrommati et al, 2003d) (Rodden et al 2003). In order to realize this possibility, the concepts need to be developed that will serve as a common referent among people, designers of applications and developers of technology. Based on these concepts, one has to:

a) Implement a universally applicable model of the communication between components of the system (that is, services, distinct objects and in-home

appliances). Such a model needs to provide a common technology-independent upper layer (providing the end-user functionality) and a lower layer that can accommodate existing standards, system architectures (such as UPnP, Jini, etc) and communication protocols (such as Wi-Fi, Bluetooth etc).

b) Build mechanisms with which people can be enabled to act upon their environment, and manipulate the information appliances, augmented artifacts, and services within it. Such control mechanisms can be associated with existing appliances or those that are especially built.

c) Reinforce the willingness and the ability of people to use the in-home environment by developing appropriate interfaces that provide them with a basic level of understanding of that environment and the applications running. Such an environment will consist of tangible or intangible components invisibly interconnected via local or remote network links. Thus the end user tools must reveal the internal structure of the environment and applications, present their current state and content, and explain their functionality and enable (or help) people to manage the associations between components.

As was explained in the previous chapter, in the proposed mode augmented artifacts can be associated by people using the straightforward model of connectable Capabilities (Plugs) and Synaptic Links (the so-called 'Plugs and Synapses' model). An artifact has physical properties as a result of its tangible self, while it offers services as a result of its digital self. Plugs are software classes expressing the connectable capabilities of these artifacts. A synaptic link (Synapse) is an association between two compatible Plugs (Figure 26). This link is explicitly created by people to achieve a particular synergetic function from the two artifacts, often a cause-effect type of relationship. Synapses are formed using an external overview device, the Editor, (a software program that is run by an information appliance).
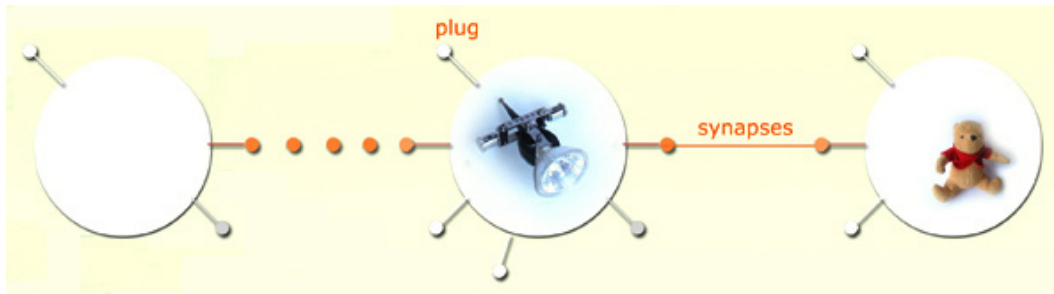
Figure 26: A vocabulary acts as a common referent between people, objects and their collections: the artifacts' capabilities (Plugs) can be associated together via invisible links (Synapses) in many possible ways. Thus, the adopted style provides an infrastructure for open applications. An application is formed by a collection of objects functioning together in this way to serve one specific purpose.

This recombinant approach has been adopted and exploited further in the extrovert-Gadgets (e-Gadgets), an EU-IST Future Emerging Technologies (FET) research project of the Disappearing Computer Initiative, wherein artifacts have been the building blocks that enable applications to be created by people. In 'extrovert Gadgets' (or e-Gadgets), the term 'extrovert' in the project's name signifies the negotiation and communication among artifacts. In the context of this project the function of an application (the project uses the term 'Gadgetworld', corresponding to the term 'microworld' used in component software) requires, at the technology level, intense message exchange between the associated artifacts (called 'eGadgets'). P2P, client-server or other network protocols will deal with this traffic, ensuring it remains outside people's awareness.

The e-Gadgets project produced several example artifacts (eGadgets) that can be considered pioneers of augmented objects. The result of linking artifacts together via invisible links is an application configuration (the project calls it a Gadgetworld) (Figure 24, 25, 26). It is a distinguishable, specific configuration of associated artifacts formed purposefully by a designer, a user, or even an intelligent agent.

People need to be aware of the results of the application will produce once it operates. It is assumed that people will have to learn to create applications, even debug them, with the help of an overview device, the Editor. Transparency of use and robustness are key issues. According to the AmI vision, most important is the

achievement of 'seamless integration' of artifacts. Artifacts are heterogeneous appliances created by different manufacturers, and compatibility of these produced objects is better insured by their integration within a single computing architectural style. As a result of this integration, the artifact becomes aware of its environment.
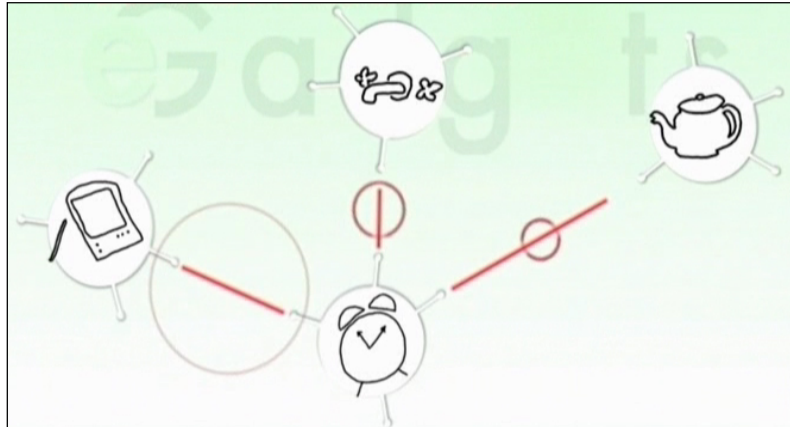
Figure 27: Negotiations and data exchange happening between artifacts

The solution proposed by the e-Gadgets project (Kameas et al, 2003) has been to enable communication rather than merely exchanging messages. Communication aims to achieve a shared understanding (Habermas, 1984); it happens so as to make known an intention or a desire, so that in turn others can respond to the suggestion. In the e-Gadgets project approach, a Synaptic Link is formed as a result of negotiation among eGadgets. Negotiations and subsequent data exchange are based on ontologies possessed by the e-Gadget artifacts; the only intrinsic feature of an artifact is the ability to engage in structured interaction in this way (Christopoulou and Kameas, 2003).

The e-Gadgets project has used the abstractions of the Capabilities and Links model (Plug and Synapse model). It defined the Gadgetware Architectural Style (GAS), which includes a set of concepts (including the above model) and application rules, the enabling middleware, a methodology and a set of tools that enable people to compose distributed applications from the services offered by artifacts and devices. The Gadgetware Architectural Style (GAS) applies the concepts of component-

based software engineering to ubiquitous computing environments, as is extensively reported in (Drossos et al, 2007a) (Drossos et al, 2007b)

Using the GAS technological framework a control device - as editor - was realized, via which a user can associate together a number of artifacts in several different ways. The Editor, by applying the Plug-Synapse model and operating GAS-Operating System, allows visualization and control in the augmented home environment. The interoperability between objects is assumed as given, as it can be realized by existing/proposed infrastructures.

10.3. Enabling artifacts to become components in the home

One assumption underlying GAS system architecture is that appliances and objects in the home are manufactured to be autonomous and functionally self-contained. Moreover, they can locally manage their resources (processor, memory, sensors/actuators etc). GAS provides a compatibility framework, specifically a layered middleware that enables them to share definitions of services, exchange data, interpret incoming messages correctly and act accordingly. Thus GAS enables in-home artifacts and appliances to be used as components of a non-a priori defined system, and at the same time enables users to play an active role in understanding and defining the functionality of their ubiquitous environment. GAS provides a middleware that can directly interface with hardware components and also serves as a layer on top of existing network protocols or distributed system architectures, enhancing them with GAS-specific functionality.

The objects in the augmented in-home environment, in line with the proposed model, can range from simple to complex ones, while sizes can vary extremely. By associating these artifacts together into collaborating clusters, people can shape their own environment. This approach supports the development of open systems but can

also be used to explain the system to people; the latter is necessary in order for them to be able to manipulate such environments.

At the conceptual level, GAS includes the Capabilities and Links (Plug-Synapse) model, which regards each object in an in-home environment as presenting a set of connectable affordances and offering or requesting a set of services; these abilities, that can be inter-associated, are visualized via the software construction of Capability-Plugs. People can associate compatible Capabilities (Plugs) (thus creating Synaptic Links) and establish a composition of the respective services / functions (Figure 24, 25).

Figure 28: A domestic object can become augmented with computation and communication capabilities

10.4. Creating Artifacts

For research purposes a number of domestic objects and furniture were augmented with computation and communication capabilities and become e-Gadgets (Figure 28, 29, 31). This "GASification process" is a stepwise methodology that ensures that all the necessary hardware and software modules are installed in the object and initialized correctly.

According to the e-Gadgets approach, in order to make an everyday object into an e-Gadget, firstly one has to attach to it a set of sensors and actuators. For example, in order for the e-Table to be able to sense weight, luminosity, temperature and proximity, it has to be equipped with pressure pads, luminosity sensors and an infrared sensor. Pressure pads are mounted underneath its top surface, covering all of it. Luminosity and temperature sensors are evenly distributed on the surface. Four infrared sensors are placed on the legs of the table and another on its top (figure 29).
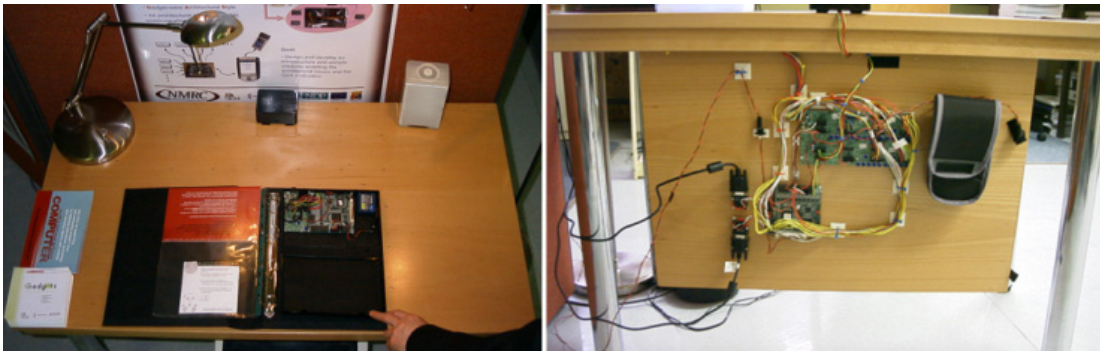
Figure 29: The top surface of the e-Table and the supporting circuitry underneath

A great deal of circuitry that connects these sensors with power sources (replaceable batteries in this case) and with driving hardware modules. The latter are required in order to collect and filter sensor readings into the GAS-OS middleware, which currently runs on a Personal Digital Assistant (PDA), an iPaq, attached to the object (to preserve autonomy of objects, an iPaq per artifact was used, providing the required processing and communication hardware).

Once the hardware is installed, one has to install GAS-OS in the artifact's iPaq and configure it to interface with the hardware. This is achieved via a special software module, the Gadget-OS, which interfaces with an FPGA that drives the sensors. When these objects have reached beyond prototype stage, all necessary hardware (including sensors, processor, wireless module, battery, boards and circuitry) will be embedded into them during their manufacture; all that will be to download GAS-OS in them and configure it for the specific object.

The latter step includes the definition of the artifact's Identity and its Capabilities (Plugs) in a way that will be understandable to other artifacts. This is done by creating XML-based descriptions of plugs using universally agreed core ontology (a vocabulary of basic terms) (Christopoulou and Kameas, 2005). The uniqueness of the ID is achieved with a process similar to the one used for MAC addresses (Ringas et al, 2002). The above are used by the GAS-OS modules in order to manage Plugs and Synapses, translate sensor readings into messages, translate incoming messages into service requests etc. GAS-OS runs in Java, but relies only on features available in the Java Personal Edition, compatible with the J2ME Personal Profile. This allows the deployment of a wide range of devices from mobile phones and PDAs to specialized Java processors. The proliferation of end-systems capable of executing it makes Java a suitable underlying layer providing a uniform abstraction for the middleware.
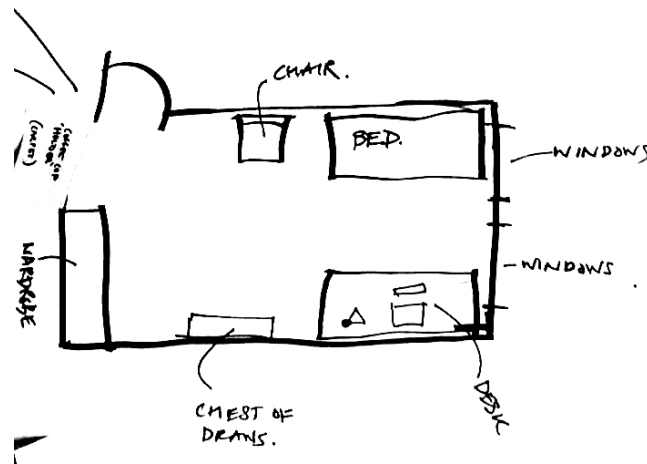
Figure 30: The intelligent dormitory in the University of Essex was used in the e-Gadgets deployment, so that agents had access to GAS-OS.

Figure 31: Example of prototype augmented artifacts: the augmented chair, table, book, lamp, and cube-light.

10.5. Editing functionality

The Editor is designed to be compatible with the artifacts in the home. It is a piece of software that can run on top of an existing information appliance (such as a PDA or PC), while the top software layer utilizes the particular interface resources of the appliance at which it is residing. It's core is thus independent from the device it runs on, allowing for a the implementation of many different interaction modalities. Within the context of the e-Gadgets project two implementations have been created using on-screen interfaces, on a handheld computer and a laptop computer, as this enabled the testing of the Editing concepts in a portable form. The Editor's high-level architecture is shown in (Figure 37).

The main role of the Editor is to make visible the available artifacts in one's ubiquitous environment, and their existing functional dependencies and to allow the user to create and edit synaptic associations between appliances and thus create, edit and debug ubiquitous applications. An association can be established between two capabilities, when these are available via the digital self of a device. The user has to indicate which two capability Plugs to associate and link them with each other;

therefore a cause-effect relationship is forged between the objects. The Editor uses the services of GAS-OS to support editing actions.

The Editor identifies the Information Communication Devices in the home environment. It also sees the capabilities offered by each device that can be inter-connectable. Such capabilities have a direct relationship with the actuating / sensing capabilities of the objects and the functions that are intended by the object's manufacturers. Nevertheless, some of these capabilities (especially more complex ones) may not be obvious to people, apart from via the editor.

In addition the Editor identifies the available application configurations, in the form of current working groups of associated appliances in the home, and displays them for supervision. The links between the compatible capabilities of appliances are visualized and can be manipulated through the Editor. In the test bed implementation this is done by a graphical user interface using an association matrix, as shown in Figure 36. Associations between certain capabilities of appliances/objects can be formed thus creating configuration sets for a certain purpose.

Figure 32: Two implemented versions of the Editor (on PC and PDA).

The identification and selection of capabilities (Plugs) via the editor is a task that depends on the user expertise. A novice user might not be interested in or understand more than just the description of the capability; (s)he can then base

his/her selection on a natural language description that is proposed in the design of the Editor. A more advanced/experienced user may prefer to see more of the technical details in order to make his/her selection of the capability.

Once such a set of Synaptic associations is established, the part of the operating system that runs on each participating device ensures that it works. The associations that are activated will operate as long as the user wants them to (until (s)he deactivates or deletes them) unless there's technical inability to maintain its functionality (i.e. one participating object is out of range, non-responsive, e.t.c.) .

10.6. The implemented Editor interfaces

Two versions of the Editor have been created, in order to test the primary editing functions (Figure 33). One with richer functionality was implemented for a personal portable computer, having with 'professional' designers/developers in mind. The second and simpler one runs on an iPAQ handheld computer and is intended for the non-trained end-user. Twelve sample devices have been modified in order to be compatible so that can be seen and associated together in a number of ways by the Editor (Figure 29,31). In this implementation the Graphical User Interface (GUI) is handled at the Interface Layer and all the functions required of the GUI (like discovery of devices, activation of functional association sets etc.) are mapped to the actual Operating System functions in the Function Layer via the Intermediate Layer.

The PC visualization of the Editor consists of several screens: One that lists the devices in the vicinity (the Listing Pane). From the listing pane the user can drag and drop selected devices into the Editing pane. Upon selection of two devices, an association matrix that shows both devices capabilities opens up. Tagging a square on this matrix (Figure 36) associates the two capabilities together (with preset properties). If the user wants to change the specific details of this association, they can do so by twiddling with the properties ("mappings") of each capability in the

association. The newly created application then needs to be named and activated. When this is possible the user can test the association set by using the physical devices, (in order to ensure that the configuration is working as originally intended); alternatively (s)he can use the editor and re-establish the association mappings (properties) or delete the total set of connections.

The PC interface of the Editor (Figure 32, 36) allows several ways of working– the two most prominent are the 'clockwise' operation   and the counterclockwise operation. In the clockwise operation, the user creates a new configuration starting from the selection of the devices; then (s)he identifies which capabilities (s)he wants to use and links them in an association and finally specifies the parameters of this association (its properties). The counterclockwise operation is suggested so that the formation of the Functional set can be based on a description of the functionality the user intents. (i.e. Searching for already established Sets (based on similar target functions) and using a natural language description interface in order to figure out the devices needed and the synapses required to achieve the requested function).

The layers that are illustrated in the grid-form design of the Editor allow the Editing task to start from any level: one can search for the capabilities required first and then deduce the devices, search for devices based on their class, and perform operations on sets of the devices.

In the working prototype of the Editor the primary functions that enable the creation of associations were quite robustly implemented (discovery and visualization of artifacts and their capabilities; creating associations; setting the properties of these associations; naming and activating a functional set of associations; activating, deactivating, deleting each association or each set), while the auxiliary natural dialogue and facilitating operations part have not been implemented. The working prototype using the screen and Graphical User interface on a PC/laptop, gives the primary functions using similar panes as the ones suggested in the design phase. It

provides in a rather extended way the editing items for supervision and manipulation.

The working prototype of the Editor on a PDA contains the same primary functions but in a more condensed way, that guides the user through a stepwise process for creating a Gadgetworld . This is partly dictated by the small display size. Due to its stepwise approach and the limit imposed in the amount of presented information and options at any one time it is easier to use. Consequently the PDA based implementation was the one used for evaluation purposes (Figure 32 and 33).

10.7. Conclusions

The e-Gadgets project has applied the conceptual model of Capabilities and Links (initially referred to as Plug-Synapse Model), proposing an architectural style (GAS) and building a system (middleware, tools and artifacts). Ubiquitous computing artifacts that follow the proposed architectural style can be reused for several purposes, in order to build a variety of Ubiquitous computing applications. The value of this approach is that, via the e-Gadgets tools, artifacts are treated as reusable components.

The proposed model is easily comprehensible; therefore, by the appropriate use of tools, the e-Gadgets technology can be usable by designers of Ubiquitous computing systems, but also by untrained end-users. Subsequently this approach opens possibilities for emergent uses of ubiquitous artifacts whereby the emergence occurs from people's own use. Potentially it can enable the acceptability of Ubicomp technology into people's environments, as well as enabling the making of emerging niche applications.

Communication (hence the term 'extrovert' that stands for the 'extrovert Gadgets' or e-Gadgets) is promoted rather than mere message exchange. In the e-Gadgets

approach, a Synapse is formed as a result of negotiation among artifacts, which, in turn, are based on e-Gadgets ontologies.

The deployment of the conceptual model proposed in chapter 9, in the case of the e-Gadgets project, as part of the system architecture, enabled the assessment of the model and validation of End User Development within AMI. The evaluation sessions, based on the deployment of the e-Gadgets project, will be described in detail in the following chapter.

11

11. Validation of end user development and the proposed model, through deployment in e-Gadgets

11.1. Introduction

In this chapter, we describe how End User Development in the area of Ubiquitous Computing applications has been validated. The proposed model was assessed as an abstraction that is understandable and usable, especially by young adults. The four

different evaluation sessions, their process and outcome are described in detail in the Appendixes 1 to 4.

The proposed conceptual-technological model for end user development of Ubicomp environments was evaluated through several user and expert trials. An expert review workshop and an analysis based on the Cognitive Dimensions framework were conducted in order to assess the concepts in the preliminary phases of the prototype implementation. Feedback was also collected using a hands-on Demonstration (using three artifacts and an editor), and shown in two conferences (British HCI and DC-Tales), where feedback questionnaires were received. Finally, a short user evaluation was conducted in a specially constructed student dormitory at the University of Essex (i-Dorm), using several sensing and actuating components and artifcts; that room and its components had been equipped to support the proposed model and were controlled through GAS-OS middleware (Kameas et al, 2003), (Drossos 2007b). An environment of more than a dozen augmented objects (of various sizes and types, such as a carpet, lamp, desk, chair, books, bed, music player, cube, etc) and two implemented instantiations of the Editor (on a PDA and on a laptop) were used in this evaluation, which aimed to monitor how potential users grasp the concepts and whether they can create or modify their own applications, using the concepts of Capabilities and Links ('Plug- Synapse') and editor within a GAS enabled environment.

11.2. An example scenario deployed for evaluation

A fictional scenario that was used to guide implementation of a demonstrator during the e-Gadgets project, and then for deployment in the i-dorm, is described in this section.

This example scenario addresses a day in the life of Patricia, a 27-year old single woman, who lives in a small apartment near the city center and studies Spanish

literature at the city's University. A few days ago she passed by this store, where she saw an advertisement about 'extrovert Gadgets'. Pat decided to enter. Half an hour later she had given herself a very unusual present: several pieces of furniture and other devices that would turn her apartment into a ubicomp one! The next day, she was anxiously waiting for the delivery of an e-Desk (it could sense light intensity, temperature, the weight on it), an e-Chair (it could tell whether someone was sitting on it), a couple of e-Lamps (she would be able to remotely turn them on and off), some e-Book tags (they could be attached to a book, tell whether a book is open or closed and determine the amount of light that falls on the book) and an e-Carpet (you just had to step on it). Pat had asked the store employee to pre-configure some of the e-Gadgets, so that she could create a smart studying corner in her living room. Her idea was simple (she felt a little silly when she spoke to the employee about it): when she sat on the chair and drew it near the desk, then opened a book on the desk, then the study lamp would be switched on automatically. If she closed the book or stood up, then the light would go off. She hadn't thought of any use of the carpet but she liked the colors.
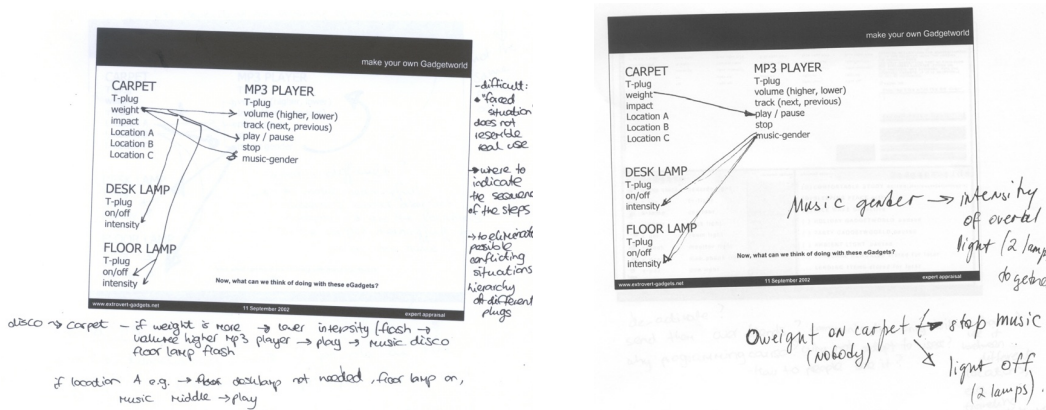
Figure 33: Annotated diagrams that were used during the expert evaluation

The scenario and deployment were used in some of the evaluation sessions. For research purposes a number of domestic objects and furniture mentioned in this scenario have been augmented with computation and communication capabilities, and turned into artifacts (eGadgets), running GAS-OS middleware that uses the

proposed Capabilities-Links model, within the scope of the extrovert-Gadgets (e-Gadgets) research project (chapter 10).

Figure 34 : Use of the PDA based Editor by test subjects.

The behavior requested by Pat requires the following set of Artifacts: e-Desk, e-Chair, e-Lamp, e-Book. The collective function of this can be described as:

When the particular CHAIR is NEAR the DESK AND ANY BOOK is ON the DESK, AND SOMEONE is sitting on the CHAIR AND The BOOK is OPEN THEN TURN the LAMP ON.

In order to achieve the collective functionality required by Pat, the employee in the store had to create a set of Synapses among e-Gadgets' Plugs (see Figures: 35, 24, 25). This type of functionality and component structure is created, inspected and modified through the Editor. For example, Pat can subsequently define the intensity of the e-Lamp when it's being automatically switched on; thus the light won't blind her. Or, if an intelligent agent is used, it could adjust each time the light intensity based on the overall amount of light in the room, as it is recorded by luminosity sensors distributed on objects in the room.
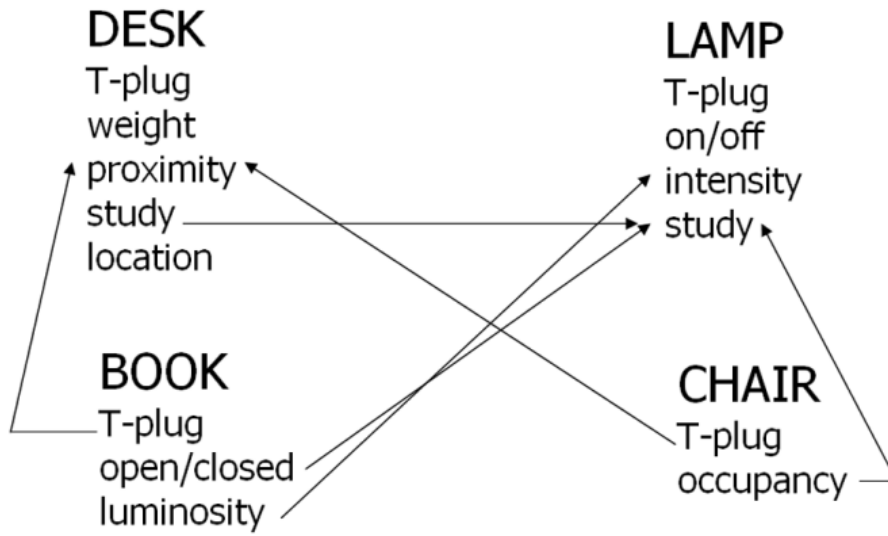
Figure 35: Schematic representation of the connections between appliances, in the above scenario
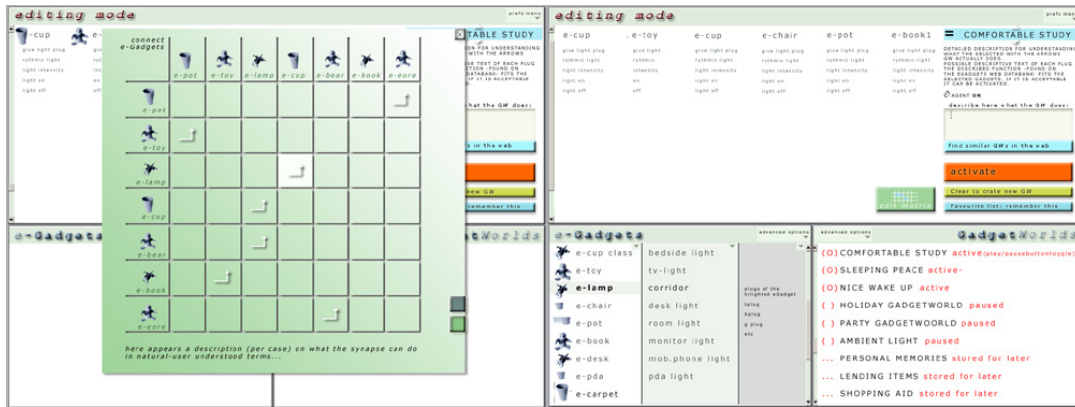
Figure 36: Draft design of a Graphical User Interface proposed for the Editor (for PC)

11.3. Concept evaluation

A central research question is how acceptable it is for end users to manipulate the various devices in the home environment. This is linked to the extent to which users comprehend the concepts underlying the Editor, and their willingness to use it. A concept evaluation was conducted with the Editor that revolved around two axes:

a. Comprehensibility of the underlying concepts and

b. Willingness to use such technology.

The Editor was evaluated as one part of a broader system and concepts that were proposed by the e-Gadgets project. The Editor evaluation did not focus on the specific design details, i.e., the look and feel, of the graphical user interface. Rather it aimed to assess the role that 'editing' can play in the future home environment and how it the extent to which it is perceived to satisfy user needs fulfill user needs.

Initially an expert appraisal was carried out for evaluating the proposed interaction concepts and technology, with respect to the end user requirements. The evaluation was conducted in phases. First an expert review was conducted in the form of a workshop. Subsequently, the cognitive dimensions framework (Green and Petre, 1996), was applied to assess how well the e-Gadgets concepts support end-users to compose and personalize their own ubiquitous computing environments. This first round of evaluation, a first version of the Editor was used, together with paper-mock ups. The nature of the evaluation was formative, i.e., it aimed to suggest directions for the next steps of the project, which would ensure that user needs are taken into account. After a working prototype of the Editor was developed (using a Graphical User Interface in a PDA), the concepts were tested through a hands on demonstrator at two events, and a wealth of questionnaires were collected. Last, but not least, a series of short user test took place in an actual ubiquitous environment (i-dorm) supported by GAS technology.

In total, the following evaluation sessions were conducted, that will be described in the following sections:

a) An expert appraisal of the concepts,

b) An assessment of concepts using the Cognitive Dimensions framework

c) Short user tests and questionnaires by conference delegates

d) Short usability tests (7 subjects) in a ubiquitous computing environment

These are reported in their full detail in the corresponding appendixes 1 to 4.

11.4. Expert appraisal

The end-user that would engage in editing applications in ones in-home environment is considered to be a 'technophile' but not a programmer; three experts in User System Interaction that matched this profile performed a set of evaluation activities during a workshop evaluation session (the workshop activities and outcomes are described extensively in (Mavrommati et al 2003d). The basic concepts were introduced in a short introductory session, in order to familiarize experts. Four scenarios were discussed that highlighted different usage/interaction design issues. A small discussion in a focus group format followed each scenario. A problem solving exercise was set to gauge the extent to which these experts could construct an in-Home application and further to reflect on what they consider as problems of doing this. A design draft (paper mock up) of the graphical interface proposal for the Editor and some video-prototypes presenting different multi-modalities were demonstrated and expert opinions were solicited. Finally an open-ended discussion elicited global level feedback for the e-Gadgets project.

In the evaluation session a broader set of issues was addressed, relating to a model of a component based approach for the Ubiquitous computing home. A subset of the

evaluation tasks concerned the Editor. Some of the most recurring themes of the discussion relating specifically to the Editor were:

- Ubiquitous computing technologies embedded in physical objects add hidden behavior and complexity to them. Problems may arise if this behavior is not observable and predictable for the user.
- Intelligence causes problems of operability and of unpredictability for users. It must be used with caution and this should be reflected in the demonstrations built.
- Constructing and modifying applications in the Ubiquitous home, is a problem solving activity performed by end-users. As such, it has an algorithmic nature and thus good programming support should be offered.

11.5. Cognitive Dimensions evaluation

Following the last observation, the expert conducting the evaluation has conducted a further assessment using the Cognitive Dimensions framework (Green and Petre, 1996), a broad technique for the evaluation of visual notations or interactive devices. It helps expose trade-offs that are made in the design of such notations with respect to the ability of humans to translate their intentions to sequences of actions (usually implemented as programs) and to manage and comprehend the programs they compose. Broadly, the Capabilites and Links model and the Editor facilitate the creation of in-home applications that are composed in non-textual manner. Thus, this theoretically founded technique can be used to provide insight into selecting between alternative choices with respect to providing tools for applications construction. Some of the most interesting points resulting from the evaluation with the cognitive dimensions framework were the following:

- There will always be an initial gap between the users' intentions and the resulting functionality of a user composed application. Users will have to

bridge this gap based on the experience they develop after a trial-and-error process. An Editor can shorten this initial gap, by ***allowing several different ways of expressing the user's goals.***

- Since an object can be part of several in-home applications at the same time, the effect it has on each is not easy to understand from the physical appearance. <u>Developments of the Editor will need a way to illustrate to the user how the specification of the parts influences the dynamic behavior of the whole application</u> (similar to debuggers in Object Oriented environments).

- Editors should aim to bridge the gap between architectural descriptions of an application and the user's own conceptualizations, which might be rule-based, task oriented, etc. (improving on the Closeness of Mapping dimension).

- To edit in-home applications the Editor requires only a few conventions need to be learnt by the end user (appropriately low terseness).

- The Editor should make observable logical dependencies between seemingly unrelated physical objects (hidden dependencies). There are side effects in constructing applications. A state change in one component may have non-visible implications on the function of another. In the conceptual diagrams used during the discussion (Figures: 33, 35), dependencies were directly visible. However, in the graphical user interface mock up shown in the evaluation, connections and their rules were not shown. Some way of visualizing and inspecting such connections would be a useful addition.

- An object can belong to several applications, the effect it has on each is not easy to understand from the physical appearance (Role Expressiveness). (A way to show this is to represent it via the Editor).

- The abstraction level is appropriate for the target user audience (Abstraction Gradient Dimension).

11.6. Surveys at two conferences

In the second stage, the working Editor prototype that was developed (using a Graphical User Interface of a PDA and one on a PC) was presented through two hands-on demonstrations at two events. One session was during the "TALES of the Disappearing Computer" event in May 2003 (where 10 completed syrvey questionnaires were received). The second session was at the British annual conference on Human Computer Interaction (where 29 completed questionnaires were received).

The demonstration featured an Editor running on a PDA that supported discovery and use of another three appropriately converted devices in the room: a Mathmos tumbler light, an MP3 player and a pressure sensitive floor mat. Conference delegates were invited to make their own associations by connecting the components available using the editor. After a short introduction, delegates were able to compose applications in order to control the music played by positioning themselves on the floor mat or by flipping the Mathmos tumbler on its' side (as this lamp resembled a luminous glass brick).

A wealth of comments was collected in the questionnaires, the most notable of which, are.

- 13 delegates found that this technology will not be used because it is too complex;
- 11 noted that is very easy to create and modify applications
- 4 that it is very easy to learn to do it.
- 5 delegates noted that it would not be easy for users to appreciate the benefits of this technology.

11.7. Short user tests at the i-dorm

The e-Gadgets project evaluation at the iDorm of the University of Essex, took place, in 2004, in the specially created augmented environment of a student dormitory that was made to be GAS enabled. The student room was furnished with several augmented artifacts (ie, desk, book, audio equipment, clock, desk lamp, chair, etc) as well as compatible room equipment (such as room blinds, temperature, room light, etc), that were accessible for configuration to seven participants in total (via two editor interfaces) and was also be able to be controlled by agent software (that could observe the users actions and use the developed architecture (GAS) to create applications via the establishment of synaptic links and tweaking their parameters.

The focus of this evaluation was to provide an account of the problems that users encounter with these concepts and their current state of realization was attempted, as well as conducting an overall 'non acceptance' test of the concept by checking whether potential users are not willing or able to understand, make and edit Gadgetworlds.

The iDorm evaluation tried to provide insight to the e-Gadgets concepts and technology, by answering, among other, wheather participants could understand the basic concepts (given only a brief introduction), if they would be able to create applications by associating together artifacts' capabilities, if they could predict the behavior of an application from seeing what it consists of, and wheather they could be able to slightly modify ubicomp applications.

The study was a combination of conventional short tests, which involved 6 participants (split into three pairs in order to be able to use the 'think aloud' protocol) and one single test that took place overnight (with one participant, who had spent more time and slept overnight in the i-dorm). The short tests aimed to gauge how potential users grasp the concepts and the overnight test aimed to get a relatively longer term and more realistic test of e-Gadgets when it is used in anger.

One participant stayed overnight after the tests. In the evening he was invited to play around with the gadgetworld, not as a programmer or an experimenter which tries to reach boundary conditions, but trying to get it to a state reasonable to live with (even if that was only for one night). On the day after in the morning his was asked to state his opinion and was invited to make changes to the Gadgetworld once more.

All evaluation participants that had a hands-on experience using this technology familiarized with the concepts very quickly, within only a few minutes (5-7 min) of explanation. The editor was used successfully (it was intended as a functional tool, with a preliminary interface, that aimed to test the research concepts, providing an appropriate level of robustness). The majority of subjects succeeded in creating simple applications for themselves (using 2-3 objects with 2-3 connections), using the editor provided.

Overall, the iDorm test, seems to provide conclusive evidence towards the viability of the concepts and at least rest the fears of putting too complex tasks on the shoulders of the end-user. Although the positive findings of the iDorm tests cannot be generalized for the wider public, due to the limited set of subjects, still, these test results do show the clear potential of the concepts presented. An interesting and detailed account of the i-dorm uset test and overnight stay test can be found in appendix 4.

11.8. Summary of outcomes

Through the various evaluation trials (reported in detail in appendices 1 to 4) it is indicated that End User Development has validity as an approach for approaching Ubiquitous Computing Environments.

The proposed Capabilities-Links (Plug-Synapse) conceptual-technological model was assessed as an easily comprehended abstraction that is usable especially by younger people. This model is bridging technological and conceptual notions, and

can underlie the design and development of middleware software architectures. When used as a part of a more generic framework for recombinant ubiquitous computing, it can act as the foundation for the creation of Editing tools and their interfaces.

In the first evaluations skepticism was noted among HCI experts regarding the ability of end-users to grasp the concepts we proposed. The short user-tests seem to rest the fears of an impossible to use complexity. This can hold true especially for younger users growing up surrounded by technology.

The feedback that relates specifically to the Editing tools can be summarized as follows:

- The application behavior should not surprise the user, i.e. automation or adaptation actions should be visible and predictable (or at least justifiable).
- End-users acting as Ubiquitous application developers should be supported with at least as good tools as programmers have at their disposal, e.g., debuggers, object browsers, help, etc.
- Multiple means to define user intentions should be supported by the graphical interface of the Editor, as the users tasks tend to be comprehended and expressed in a variety of ways.
- The acceptability of the Gadgetworld concepts, depend on the quality of the actual tools and their design.

11.9. Conclusions

This research has set out to provide a level of transparency into the Ubiquitous environment, by giving end-users the ability to construct or modify ubiquitous computing environments. To do this, concepts of component based software (i.e. microworlds) were adapted and applied in order to treat physical objects as

components of Ubiquitous Computing environments. Though this process, the application of End User Development into the area of Ubiquitous Computing applications has been validated, and the proposed conceptual-technological model was assessed as an abstraction that is understandable and usable, especially by the younger and more technology adept.

A novelty of the model and Editing approach in the home environment is that artifacts can be treated as reusable "'components'. The component architecture is made directly visible and accessible via the Editor. This enables end-users to act as application developers. This end user programming approach may be especially suitable for ubiquitous computing applications, as has been indicated in the evaluation trials reported in this chapter. The possibility to reuse devices for several purposes - not all accounted for during their design- opens possibilities for emergent uses of ubiquitous devices, whereby the emergence results from actual use.

An important outcome from the evaluations (Mavrommati et al, 2004), relating to follow up work on Graphical User Interface visualizations, is that *multiple means to define user intentions* should be supported by the Editor, as people conceptualize their intentions in a variety of ways, which are not necessarily structural abstractions of the system. End-users programming their environment should be supported with similar tools as programmers, e.g., debuggers, object browsers, help, etc.

A lot depends on the Editing tools in terms of interface and interaction but also extensive auxiliary functionality to aid with the editing tasks, which should be researched in their own merit.

12

12. The functions of the Editor

12.1. Introduction

As explained in the previous chapters, an approach which partially discloses the structure of a system is adopted in this research and supported by a conceptual-technological model that assists people to understand the ubiquitous applications, and how to manipulate them. The approach also assumes Editor Mechanisms towards the (re)configuration of Ubiquitous Environments and applications, and graphical User Interfaces that provide different syntax possibilities for End User Development. The functionality, architecture and modules, which can be considered for Editors from the perspective of user experience design, are outlined in this chapter.

12.2. The Editor role

As has been explained in the previous chapters, [eople can supervise the functional capabilities of devices in the home via special software applications running on

specialized devices, the 'Editors'. Editors provide the possibility for multiple representations of configuration activities in the form of Graphical User Interfaces (GUI). Such Editor GUIs can provide for different visualizations and in parallel be linked to multimodal interfaces - such as speech or gesture input systems. With Editors it is possible to supervise and interface with the various in-home appliances and also have an overview of applications of the ubiquitous environment. People can supervise the inter-connectable capabilities of an augmented artifact; they can act-upon the augmented artifact's capabilities by associating them together into application clusters serving specific functions, they can edit or delete existing associations, pause them, or add parameters in the associations to influence the details of it's function. The associative links created or edited may be stored locally (in this case, for example, in an Editor accessing the functions of a Central Processing Unit), while in the case of a distributed system (such as the e-Gadgets system) they are stored in the distributed appliances and objects within the home environment, with the advantage of this approach being that in the case of failure or object movement beyond network range, much of the application functionality can be restored. A set of associations between artifacts is a ubiquitous computing application. Such applications can be collaboratively shared between users, by collaborative mechanisms, or adapted to different user environments assisted by appropriate ontologies.

The editing capabilities can be used by designers to create ubiquitous computing applications, without having to start from scratch, as they may reuse existing component objects. They may be used by end users to personalize ubicomp applications or being creative in novel associations for niche or innovative functions (figure 17). The core editor functionality can also be accessed directly by intelligent agents that construct and adapt applications by monitoring user behavior, (as, for example, was the case in the e-Gadgets project).

12.3. Editor key functions

The main role of the Editor is to provide an interface to the user for handling artifacts and ubicomp applications and assisting him/her to create and edit synaptic associations between artifacts. An association can be established between two capabilities, when these are available via the digital part of a device. The user has to indicate which artifact capabilities to associate and link them with each other; therefore a cause-effect relationship between the objects is achieved. The Editor uses the services of an operating system (OS) to support the editing functions – for example the GAS-OS or the ASTRA OS in the cases of the e-Gadgets and ASTRA respective project cases.

In summary the Editor goals are:

1) To indicate/make visible the ubiquity in one's home environment: artifacts and their augmented affordances as well as their functional dependencies: the existing pervasive applications
2) To form new associations between devices, in order to achieve certain functions or to insert new artifacts in an application, editing their specific rules, etc.
3) To assist with sharing applications, servicing, debugging, etc

12.4. Editor high level architecture

The Editor's high-level architecture (see figure 37), according to (Mavrommati and Kameas, 2003) can have three separate interacting layers:

1. The Operating System Layer, which offers to the Editor its communication capabilities and knowledge of other connectable appliances interfaces,
2. The Editor Manager Layer, which provides abstraction of the interface layer as well as compatibility with the function layer. It contains all the structures and functionalities needed by the Editor

3. The User Interface Layer that is responsible to provide any interactions with the end-use as well as any other operations the Editor provides.
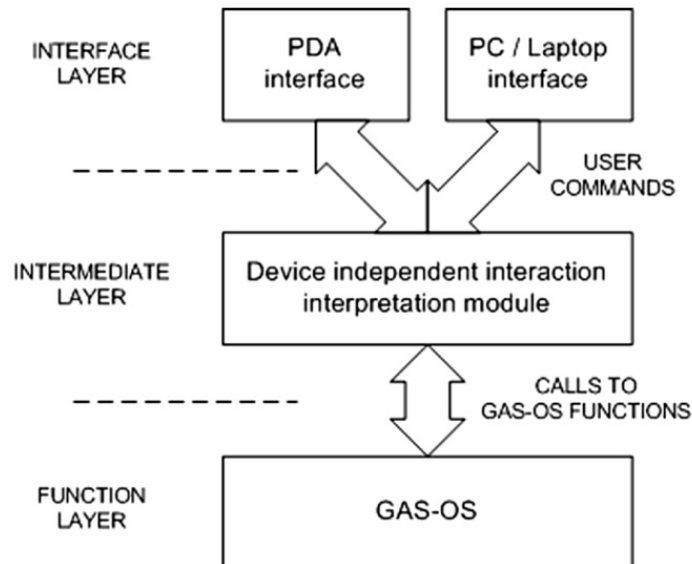
Figure 37: Schematic representation of the Editor layers, in the case of GAS-OS, for the e-Gadgets research project.

The Editor can identify the augmented artifacts in the vicinity of the home. It also sees the capabilities offered by each device that can be inter-connectable. Such capabilities have a direct relationship with the actuating / sensing capabilities of the objects and the functions that are intended by the appliance manufacturers, as explained in chapter 9. Nevertheless, some of these capabilities (especially more complex ones) may not be obvious to people, apart from via the editor.

In addition the Editor identifies the available application configurations, in the form of current working groups of associated appliances in the home, and displays them for supervision. The links between the compatible capabilities of appliances are visualized and can be manipulated through the Editor. Associations between certain capabilities of appliances/objects can be formed thus creating configuration sets for a certain purpose (figures 24 and 34).

The identification and selection of connectable artifact capabilities via the editor is a task that depends on the user expertise. A novice user might not be interested in or understand more than just the description of the capability; he or she can then base his/her selection on a natural language description that is proposed in the design of the Editor. A more advanced/experienced user may prefer to see more of the in-depth details in order to make his/her selection of the capability.

Once such a set of associative Links is established, the part of the operating system that runs on each participating device ensures that it works. The associations that are activated will operate as long as the user wants them to (until he or she deactivates or deletes them) unless there's technical inability to maintain its functionality (ieone participating object is out of range, non-responsive etc).

12.5. Tasks that can be supported by the Editor

Editor can have modules in order to support the following main tasks:

- Overview of applications and their state
- Creating new ubiquitous applications or editing old ones
- Sharing ubiquitous applications and assisting with collaborative development
- Assisting in augmenting artifacts

In the following sectiona the corresponding sub-tasks of those are outlined:

Overview of applications and their state

1) Provide an overview of the existing ubiquitous application configurations in one's environment
   a) show an overview list of all ubiquitous applications
   b) Indicate which applications are active, running, paused, or only created and stored -but not activated.

c) Show the stakeholders of each application (when applicable)

d) When possible, visualize hidden dependencies of each application

2) provide overview and control over the state of the ubiquitous environment

a) provide an overall ON / OFF SWITCH for all applications in the environment

b) provide on/off switch per application

It has to be noted that the overall OFF switch has emerged from evaluation test sessions as a pressing requirement, that is yet missing from ubiquitous environments and applications functions (see chapter 11 and Appendix 4), and so, it is addressed here on its own merit.

Figure 38: An example visualization of the Idle/Observation Screen. The overall OFF switch for each application that is currently running can be seen[4]. The applications scroll left, in a loop.

---

4 The Idle/Observation screen: a proposed example showing an scrolling overview sequence of ubiquitous applications, as well as providing access to the Editor, a universal OFF switch, and OFF controls per application. This control and overview screen is suggested to be visible in the home, through an interactive digital picture frame, or other screen in the house. In the ASTRA project case, the parties sharing their awareness information via pervasive applications in their environments were also viewed (via their profile photos), alongside with the application's name.

Creating new ubiquitous applications or editing existing ones

a) Create new applications by establishing new application configurations, associating links and rules of links, between certain connectivities of artifacts.

b) create more complex applications from the ones created in previous time and show them as a list

c) edit the parameters between connectivity links (i.e. edit the parameter values, introduce timers/diary parameters, etc), in existing applications or those that are being created.

d) delete certain applications

e) delete certain links from existing applications

f) enrich/adapt applications by creation of new associations/ (adding more connectivity links) in existing applications

g) negotiate with other stakeholders of applications, or adapt dependency aspects (i.e. privacy settings, etc), when applicable

Sharing ubiquitous applications and assisting with collaborative development

a) Collaboration: Users can form communities, and share application configurations, knowledge, tips, or even software that defines the artifacts connectivities.

b) People can find (ie by keyword search) applications that may fit their needs, and then adopt them for use in their environment. Ubiquitous applications should be made to adapt to their environment semi-automatically.

c) Developer communities can publicize and share different interaction modalities for the editor.

Assisting in augmenting artifacts

a) Create the code for transforming the associative capabilities of an artifact

b) Port new software constructions (connectivities) into artifacts.

c) Share the code of artifacts, via a collective repository; find, within a collaborative shared repository, suitable code for artifacts that can directly be adopted or adapted.

12.6. The overview / control screen

Applications in ubiquitous computing environment have to face the problem of visibility. It is not clear to inhabitants of an augmented environment that there are ubiquitous applications running in the space, nor can they easily know which applications are currently active. People have to also distinguish between these two states of applications: the ones that are active in the background (not currently colleting sensor data), and the ones that are currently being used (running). Moreover, in the context of the Editor, they have to be able to see an overview of the status of applications, either being draft (saved applications that are in progress of development) , or created (finalized but not activated), the ones that are activated (but can be paused), and the ones that are currently running.

An overview screen is considered a very useful addition to a ubiquitous system, that shows applications that are active (that is, in the background but not functioning at the time) or being used (applications that collect data and appropriates responses, that is currently interacting within the environment and with users).

An overview screen (figure 39) has been proposed in the context of the ASTRA project (that enables pervasive awareness applications) showing not only the currently running applications but also the user groups that share awareness information via each pervasive application. In the ASTRA context of collecting and presenting awareness information using the ubiquitous home, this control view is an idle screen that can become interactive when touched. It is intended to be a form of interactive small frame (like a picture frame), that is always visible in a prominent location within one's home environment and running in idle mode showing in a loop

the active applications. Via this overview screen, direct control of switching OFF or ON the applications is provided (pausing the currently used or running ones), as a response to the feedback reported from users (see appendix 4). In addition, via this screen, one can directly access more detailed functions of the editor , through a graphical user interface.
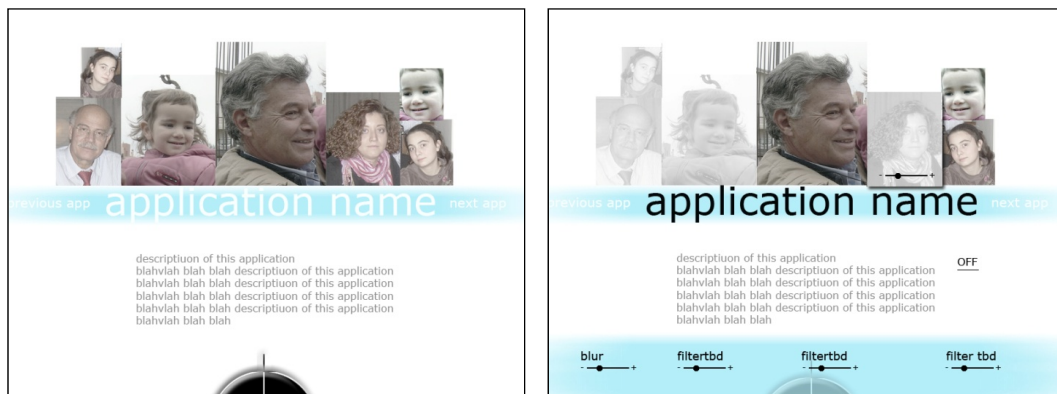
Figure 39: Visualization of the Idle/Observation Screen. At the lower part the OFF switches for each individual application can be seen.

12.7. The role of communities in end user development: the ASTRA project case

EU IST FET ASTRA (2005-2009) explored the role of communities in End User Development, for Pervasive Awareness Systems. ASTRA (Astra project website) aimed at a supporting infrastructure that allows users not only to use services but also to develop new applications that will support and enhance their social relationships. Services can be developed by the community that is going to use them, but might also be provided by external parties, including both other communities and individuals.

It has to be noted that sharing applications means that someone that has developed and used an application is making it available to others as well - this not necessarily

implies that they use the application together. Moreover sharing can be supported by the total application rules, but also of subsets of rules from an application. For example sharing sets of rules that define the triggering of applications, or application related vocabulary used by a community, can be possible paths that promote the adoption of configurations. An extensive report, which describes in detail the different community mechanisms as they are briefly reported in this section, can be found in (Astra D4, 2009b), authored by M. Divitini.

End user development for ubiquitous computing applications consists of two parts: a) finding the appropriate application and b) appropriating it for one's own environment.

Using a repository for finding applications

Considering EUD and communities, systems have to enable functionality for the sharing of applications between users, and of other application relevant communication and information. This can be made possible by agents that actively help with the process of finding the applications that match certain criteria/requirements, and by providing means of communication between users, in order for them to help each other.

- A shared repository of ubiquitous computing applications. This should include context explanations, which could be collaboratively built (ASTRA D4, 2009b). In the context of pervasive awareness applications in particular, this could assist users to understand the behavior of the applications (i.e. why the lamp is turning green, what exactly is signified, that is then forgotten?)
- For application designers there is a specialized need for shares spaces for collaborative software development from a software development perspective (such as sourceforge.net) as well as support for selling ubicomp applications.

An overview of repository functionality (ASTRA D4, 2009b) identified for sharing of applications in ASTRA is the following (figure 40):

- Sharing of applications
- Tagging of applications
- Browsing for applications
- Searching for applications
- Recommendation of applications
- Annotation/Rating of applications

The rules that trigger an application may be intertwined with contextual information from the environment, that only makes sense when considering the specific context of the environment (i.e. the specific artifacts present). Appropriation between different environments is necessary (i.e. automatic, via intelligent systems, or user-configured). A process of abstraction and revision can protect users (for reasons of non-disclosing private data and information). A user may be, for example, willing to share an application but not the specific rules that are associated within his or her particular instantiation.
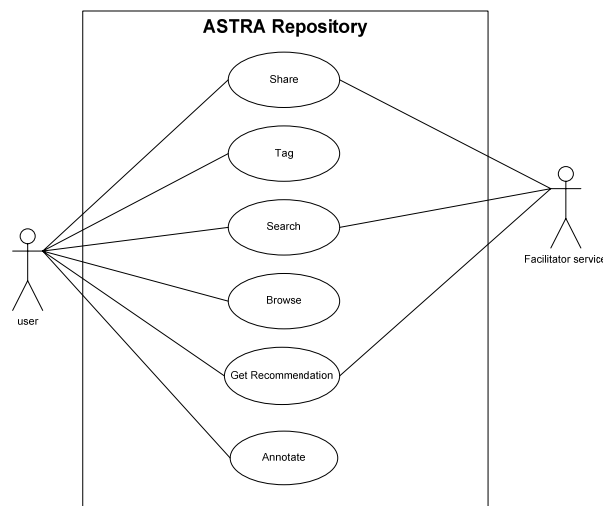
Figure 40: Use cases as they are identified for the ASTRA repository of Shared applications. Source: (ASTRA D4, 2009b)

Tagging provides a way of describing items by collaboratively applying tags to them. Tags can then be used for different categorization of items. In shared repositories tagging functionality is a primary part of the system.

Searching is a common way to seek applications from a repository that can be shared. Search types can include search by keywords, by similarity of environments/artifacts, or by certain criteria (i.e. intrinsic properties of the application, aspects that are important to the user).

Finding applications to appropriate into one's environment can also be done by *browsing*: browsing by tags that are most related, or by applications that are used by people that have similar profiles. *Recommendation* can also assist users; recommendation can occur when the system can observe the users behavior and can suggest a list of potentially interesting applications to end users. This can be assisted by an agent that compares the applications the user has active, with a list of popular applications in the EUD community (ASTRA D4, 2009b). Recommendations can also occur from individuals, directed to their common interest groups.

Appropriating shared applications to one's own environment

When deciding to use an application from a shared repository, one must appropriate it. By appropriation, a user has to tailor an application to his specific environment, artifacts within it, and his or her specific wishes and needs. In this process some artifacts, links or rules, may be omitted or replaced with others in the new environment. Help of agent systems or expert people, through the community EUD support tools, may be needed to appropriate the applications (figure 41).
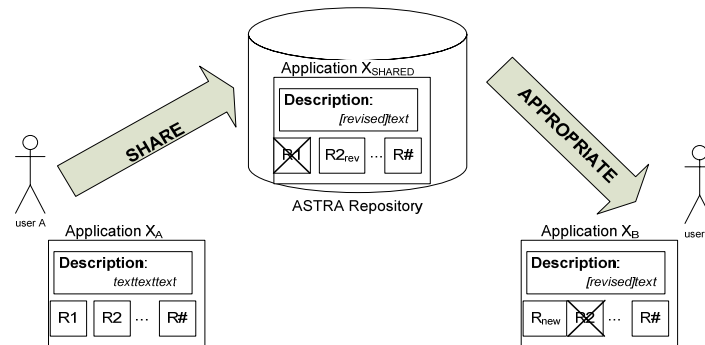
Figure 41: Sharing and appropriation of applications, in the case of ASTRA. (Source: ASTRA D4, 2009).

12.8. Interaction Diagrams for editor: the ASTRA case

From an experience design perspective, a sequence of interaction diagrams are needed in order to sketch out the specifics of the information flow and the GUI controls. The interaction annotated diagrams take into consideration the interface elements and use web forms for proposing indicative screen layouts. Subsequent web implementation for ASTRA project has been based on the proposed interaction structure, with iterations that were considered necessary by the development team in order to reach a working version of the implementation.

In the case of the ASTRA pervasive awareness systems, separate groups of functional elements were defined (that are linked to each other), that loosely correspond to the different system modules. There are three main clusters of user actions, proposed in the UI (figures 42, 56). The first part handles Awareness Connections between users and groups (that is, defining to whom to make one's state known to, or whom to accept aware notifications from). The second part is about creating the Pervasive Applications, this is where the user can associate their awareness state to a pervasive awareness application and configure it (this involves the configuration of the ubicomp application that that will define the user's state to be communicated, or the configuration of how to visualize the states of connected

others in the user's specific augmented environment; as such it is the part involving the main functions of end user development for ubicomp applications). The third part allows the management of Users and Communities, which is about registering users and allocating user groups, in a manner similar to current awareness systems. An additional idle mode is suggested (figures 38, 39, 42), to allow not only observation of currently active ubiquitous awareness application but also provides the user with direct GUI controls for tweaking specifics (i.e. rules, filters) each awareness application.
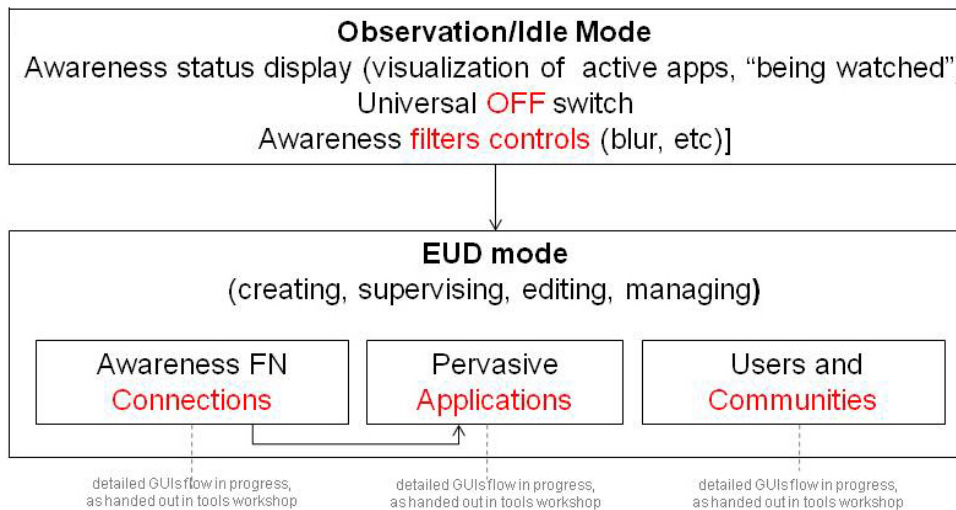
Figure 42: An overview of the separate GUI parts, as proposed for the ASTRA Editing Tools interface; the swap between an observational idle mode and an editing mode is noted.

12.9. Conclusions

In this chapter, main functions that are considered as prominent in Editors were discussed. The following were identified as important issues:

- An overview control observation screen is introduced; it allows observation of the current running applications, as well providing the overall OFF switch, or the specific off/on switches per application. Tweaking of properties can be

possible here but only through simple gauges, while one can use this idle mode as an access screen to the 'mainstream' editor GUI which contains the detailed functions of the End User Design and Configuration part.

- There is a need to visually identify in the editor interface which applications are created, which are activated, and which are currently running.

- The interface layer needs to be separated from the intermediate device interaction layer and the function layer of the system (Figure 37). This is suggested for the better management of the interface in order to provide different instantiations for different devices, interfaces and modalities.

- The addition of modules for community-sharing are proposed as an important element, assisting in the finding, sharing and porting ubicomp applications (that can then be adapted to a certain environment). Appropriation of applications for different ubiquitous environments is a key issue that remains to be addressed by ubicomp research.

- Programming of new ubiquitous applications, by associating artifacts and creating specific rules, is the most difficult issue for end users in their function as programmers. Their success in this relies on appropriate visual syntaxes and Graphical User Interfaces that need to be developed through iterative design experimentation, in parallel with developments on ubicomp system architectures.

13

13. Graphical User Interfaces: Abstractions and syntaxes, for End User Configuration in AmI

13.1. Introduction

Different syntaxes and visualizations that enable end users to design and to program their ubiquitous computing environment are outlined in this chapter. Cooperating alternative views of the perspectives of design and programming in order to create applications, are presented. Information visualization approaches for End User Programming appear with many different syntax and notations. Several scenario sketches, in the form of visual explorations used as stimuli for further considerations, are presented in order to explore possibilities regarding the interface.

Some different approaches for End User Development, ranging from the most prominent syntaxes to the less explored, are the following:

- **Tree structure** of execution flow (Figures 57, 58), with choices in each step: this is a typical syntax for programming visualization, addressing End User Programming from a software developer's perspective. Nevertheless since this visualization is becoming more common in operating systems, casual users are not utterly unfamiliar with it.

- **Pipeline:** this is a syntax visualization that assumes the view of a flow, whereby components are positioned and connected to each other. This view is familiar to application designers, but is not difficult for end users to grasp either (examples can be seen in figures: 44, 45, 46, 54). The specific terminology and complexity in structures that it allows is a defining factor regarding how proficient with programming end users need to be to manipulate them.

- **Puzzle:** this is a metaphor commonly used in component based systems. It primarily relates to end user design aspects, as it provides an existing model for users to be able to readily use; yet it is exactly there where the problems begin: it has only one to one associations with other components, and it can promote a rather simplistic view that prevents from reasoning on more complex applications. Yet it holds potential when taken as inspiration but used in a more abstract form (abstracted on a broader model, rather than a directly applied metaphor that is visually implied), and can be generalized in the form of a pipeline or a model schema. In ACCORD the Puzzle metaphor (see Figure 15, Chapter 5) is found as an easy to use interface by naïve users (Rodden et al, 2007), (ACCORD project website). Siftables (Merrill, 2007), (SIFTABLES project website) seem to resort to the same puzzle metaphor principle in their proposed tangible proximity based interaction language,

whereby puzzles take shape as they are put next to each other, rather than being predefined and associated (Fig 13, Chapter 5).

- **Direct association with lines-wiring:** this approach involves creating lines as links between components, in order to define an application (figure 52, 53). This can cause problems to end users, since, however easy it may seem at first, for two or three artifacts, it is rapidly becoming visually confusing and the sequence of action, or the timeline of actions, becomes increasingly more unclear as more artifact-components are introduced, resulting in a 'spaghetti-like' view (see figure 52).

- **Wizard step by step dialogue approach:** This is a form of an easy-to-use dialogue (with step-by-step questions and answers), but on the downside it takes a long time to complete, and provides no overview of what application functionality is created - that is especially important when one wants to be able to re-visit it and change it. This can be used by non experienced users; nevertheless it can prove very cumbersome and time consuming, and it is very hard to remember the previous choices and steps to get an overview of what has been done, unless the visualization offers a more complex alternative in parallel with the wizard.

- **The Text-based scenario:** This proposal is the most challenging, in terms of inference involved in it's implementation, but it also holds more promise for end users acting as designers. All the previously mentioned approaches have as a downside that they expect users to rationalize on the applications on the basis of artifacts and how these are inter-associated with each other. Although such views are valid to get an overview of how the connections of the system are made, people do not usually think in these terms. They tend to think of their own goals they want to accomplish, and not the medium that is in between. For example: Jenny would express an application in the following way: "When I wake up, I want to have fresh coffee and music, and

have ample daylight". Decomposing this into reasoning about artifacts and their connections is not as natural an expression, as it would result in something like: "When the bed-sensors weight value is 0, then coffee maker Turn ON, Music player Turn ON". It should also be noted that the duration that this (or any other similar) application will run will be endless, causing a malfunction not obvious at first, that would need some sort of system assistance, ie via a dialogue, at the configuration stage. The text based scenario when alternated with one of the application-structure programming views that were mentioned above (such as the 'pipeline', see fig. 54), can act as a bridge between expressing and understanding/manipulating. It acts as an End User Design tool, but one that can give input to visualization of programming interfaces. This approach is the most suitable to promote the method of scenario based development for End User Design.

- **Visualization of a Mental Model** can be used as the basis for an interface. For example the Capabilities and Links model (see chapter 9) can be visualized in interlinked devices, borrowing elements from the Jigsaw-puzzle, as well as the Line-wiring associations, that are described above. It provides clarity to the end users at first use, but it is not suitable for complex applications involving many artifacts and many connections. Nevertheless it can perhaps be scaled up in visualizations using the "tag-clouds"[5] (Lohmann et al, 2009) –(a clustered layout of tags whereby distance between tags follows certain semantic relatedness criteria, and related tags are positioned in close proximity, with the most contextually appropriate ones appearing

---

[5] A Tag Cloud is a visual depiction shown as a weighted list in visual design. Tags are usually single words while the importance of a tag is shown with size or location. The tags are usually hyperlinks that lead to a collection of items that are associated with a tag.

larger in size), and "Level of Detail"[6] type of visualization approaches. Such approaches require more extensive visual design experimentation and may need to be accompanied by an alternative view of a programming syntax (such as the pipeline or tree diagram visual approaches).

- **Form-based system:** Most graphical user interfaces for end user development of ubiquitous systems involve the use of forms. They typically use mixed interfaces combining pull-down menus, checkboxes, drag and drop selection mechanisms, and can even provide alternative views to swap between different syntax. Their advantage is that users are already accustomed to them from web configurations, and such implementations are easily ported in a number of devices.

- **Advanced interfaces:**, Alternative more elaborate models can also be explored, with a starting point from the mental model. The bubble metaphor, for example [see JRC's Digital Territories report (Daskala & Maghiros, 2007)] can hold potential to be visually explored, combined with visualizations borrowing elements from Level-of-Detail and Tag Clouds visualizations, which were mentioned above.

- **Programming by example techniques, or tangible interaction languages:** This approach needs to be coupled with a visual representation (ie. Pipeline,

---

[6] Level of Detail is a visualization of context presentation that combines smooth zooming (i.e. by use of pyramidal tiff), and GIS technology (georeference info, see for example the GoogleEarth and GoogleSky application). An example implementation of Level of Detail visualization is Microsoft's Deep Zoom for image viewing applications. Users can pan around and zoom into a large, high resolution image or a large collection of images, through high quality transitions. Image regions are downloaded as the user zooms into them, while animations hide any jerkiness in the transition. Items can be rearranged by using keyword tags where the user zooms towards one single item depiction / name and a description/view of that item appears in a separate text pane (for a DeepZoom example see the hard-rock café memorabilia website, http://memorabilia.hardrock.com/).

Tree-diagram), so when checking back on the application that is created, one can see what is achieved, can reason about the configuration and manipulate it's specifics parameters. Programming by example techniques can be applicable for a subset of applications, whereby the artifacts are such that can be manipulated. However they cannot be used in all application cases, that can contain, for example, more abstract information, (consider that can be described as follows: "when it is Christmas day, and the outside temperature is less than -5, and there are more than 3 people in the house, then play festive music related to snow"). Not all these conditions can be replicated at programming time.
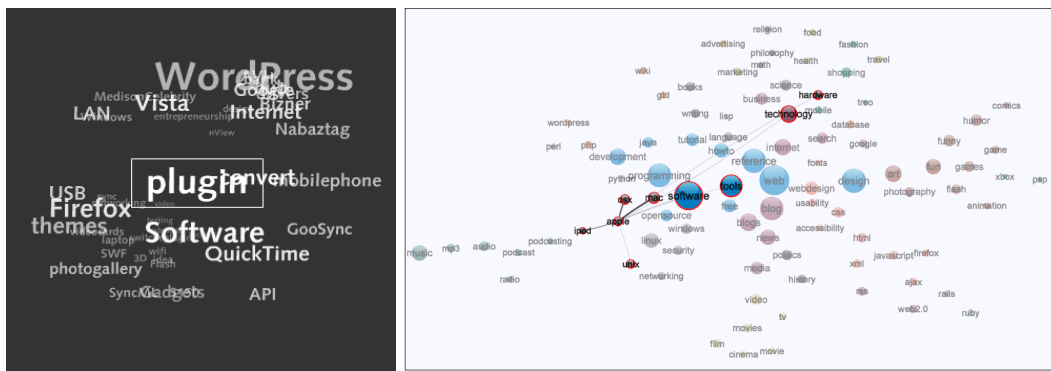
Figure 43: Examples of Tag Cloud Visualisations (source:
http://en.wikipedia.org/wiki/Tag_cloud)

In the following sections, visual experiments for graphical user interfaces are presented, using several of the above notations and syntaxes.

13.2. Pipeline style: interface examples and experiments

The pipeline view uses a linear organization of elements, in sequential flow visualization. It can be easily understood by end users and developers alike, depending on the technical language and level of detail that is being used.

Visual scenarios were created with the pipeline style in order to explore the design space considering this visualization as a suitable one for End User Development. It is also a visualization that can apply to the Capabilities and Links model (referred to initially as the Plug Synapse model) that has been proposed in previous chapters (see chapter 9), therefore bridging people's understanding on the working of the system, with the manipulation of the components of applications.

Pipeline examples: the case of the CollaborationBus project

 A Pipeline view for editing is used in the the CollaborationBus example (see Figures 44, 45) (Gross and Marquardt, 2007). In the Pipeline model, abstractions of Sensors, Filters, and Actuations, are selected and sequentially placed after each other in a row. Many elements can be placed in parallel with each other (see Fig.44), so that AND functions are achieved. It is noted that, In the CollaborationBus example (Gross and Marquardt, 2007), the most popular function among users was the sharing mechanism. Ready-to-use pipeline compositions, found in the shared repository, can then be used as a template to modify parameters or build new configurations. A drag and drop mechanism can be a further development on this interface.

We note that in the CollaborationBus example the components are used in stacks, with filters added between sensors and actuators. Multiple pipeline rows have the advantage that they can indicate many connections. Yet, in the specific implementation, the resulting extended pipeline with the terminology used by the CollaborationBus project, is considered as extended, complex too technical for the average end user.
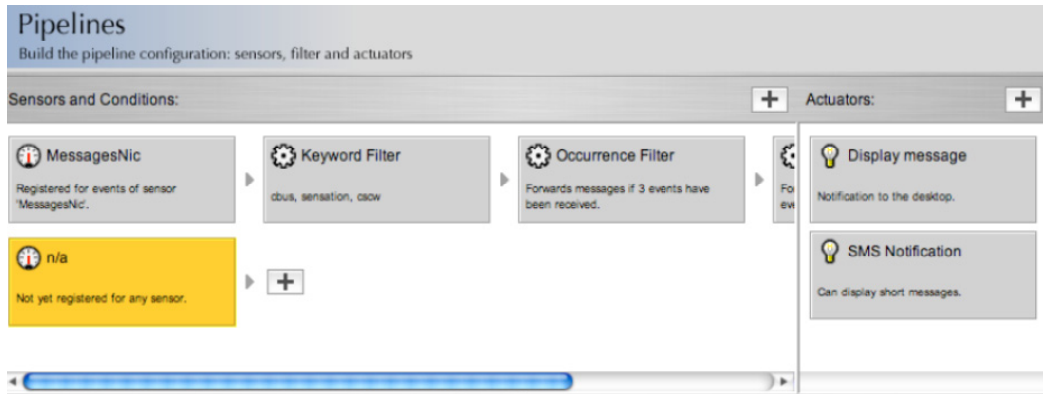
Figure 44: Detail of the CollaborationBus Pipeline editor. Source: (Gross and Marquardt, 2007)
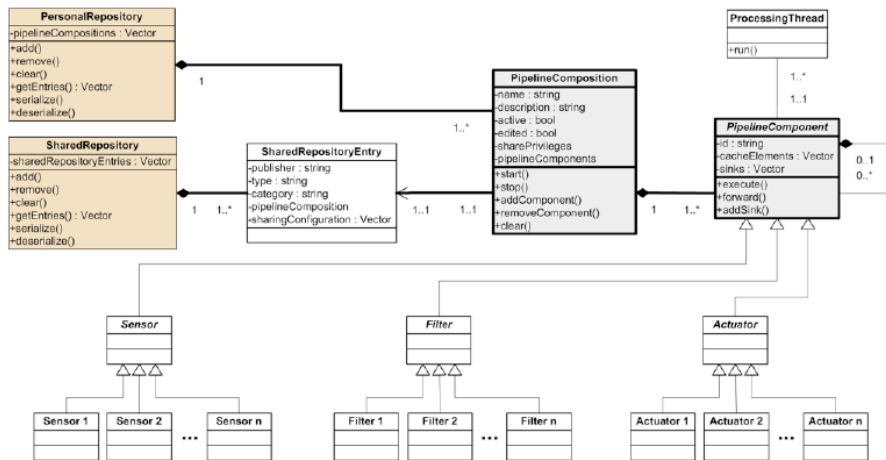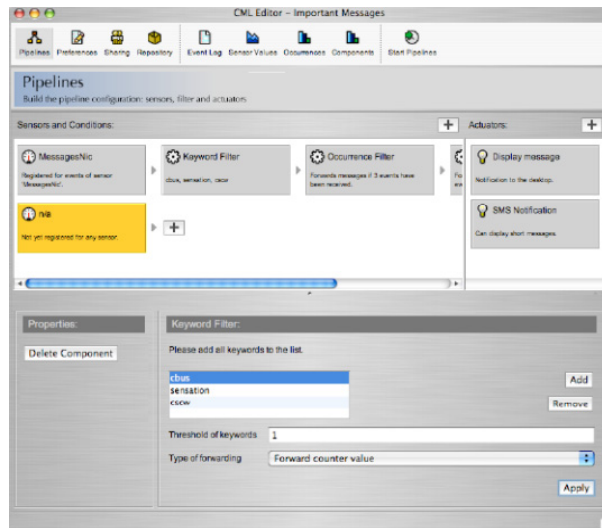
Figure 45: The Pipeline Editor in the CollaborationBus example. and its repository and pipeline UML class diagram. Source: (Gross & Marquardt, 2007)

Pipeline examples: experimentation based on pervasive awareness application scenaria

Alternative Graphical User interfaces for the creation of Pervasive Awareness applications, as well as different models and visualizations for Rule Editing, aimed at different target audiences were explored in GUI design scenarios (Fig. 46, 47, 48). Pervasive awareness applications are applications that enable friends or family to be aware of what the other's situation is, via ubiquitous computing applications in their environments. Pervasive Awareness applications therefore can involve a part whereby the ubiquitous computing application, at each node, needs to be configured by people that wish to share awareness information with others.

The interface proposed in (Fig. 46, 47, 48) is used for configuring a ubiquitous computing application for awareness communication, and is aimed at young teenagers, using the characteristic comic-strip style of the artist Keith Haring (Fokidou et al, 2008). This pipeline view ismore simplified to the one presented in the CollaborationBus example. In this case all the elements are put in one line (and not in stacks), with the specific properties being selected from pull-down menus and connected by Boolean expressions.

The rule editing proposed here is based on a pipeline model: Artifacts, applications and operators are presented in the form of floating entities on a scrolling window; they can be drag-and-dropped onto the pipeline on the lower part of the screen, where they are arranged into rules for pervasive awareness applications. The specifics of the associations between the selected objects in this pipeline visual experiment are selected via pull-down menus (AND/OR). The particular combination is then given a name (in this case, of an awareness application that signifies awareness information to be shared with friends).
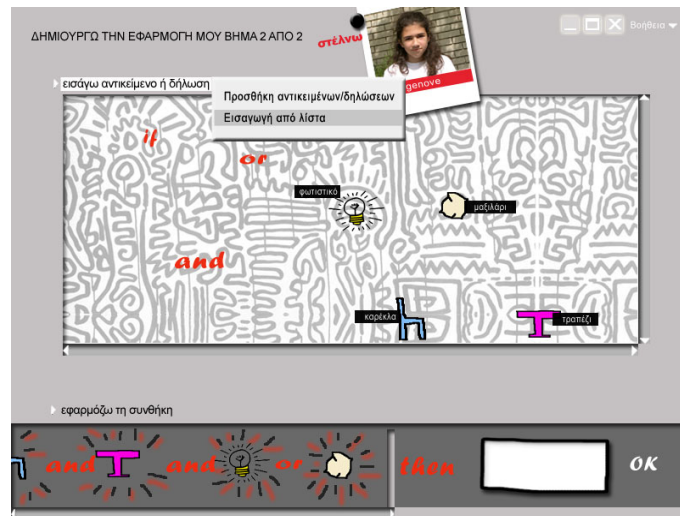
Figure 46: The Pipeline Editor visual scenario, aimed at teenagers. The artifacts are selected from the above window, and could be alternatively presented based on Tag-Clouds, or Deep-Zoom visualisations. Source: (Fokidou, Romoudi, and Mavrommati, 2008)

The applications created can be shared with selected friends; upon receiving it they can associate this awareness 'message' to certain actions of the artifacts around them, thus developing the receiving pervasive awareness application part, in a similar way. Direct messaging of friends while creating or sharing an application, is proposed as a part of this End User Tool scenario.

The pipeline concept is easy to grasp; this example can be expanded by including other application parts in the pipeline itself (by their name), and treat them in the same way as a part of the total configuration (and not only as output result).

Figure 47: An interface using the pipeline model, aimed at young teenagers, using for artifact selection the characteristic comic-strip style of the artist Keith Haring. Source: (Fokidou, Romoudi, and Mavrommati, 2008)

In the artifact selection pane, visualizations can use combinatory approaches, including, for example, tag-clouds (Fig.46), so that related artifacts appear more prominent, especially when combined with dynamic zoom to select artifacts from. It

can also be combined with a search function, and 'selection by example' functions (by this selecting the artifact in the actual environment is suggested, or even the artifact function with programming by example techniques). After selection of the artifact or its property, the pipeline view provides a programming syntax for the configuration of easy applications. The breaking up of applications into smaller (true or false) elements adds benefits for checking and reasoning on the sub parts of more complex applications. Yet, it remains an issue with the pipeline model, the use of programming language and syntax that can be challenging for non-professionals; especially the usage of Booleans is not always well understood.

Figure 48: Proposed functionality in this GUI scenario includes a chat-space, that enables synchronous discussion between teenagers, to facilitate collaborative End User Development

13.3. Graphical Interface Experiments for End User Programming: the e-Gadgets case

In the case of the e-Gadgets project (e-Gadgets website), implementation experiments for End User Programming included a graphical user interface using an association matrix (figure 53). Certain capabilities of appliances/objects could be associated thus creating groups of configurations for a certain purpose-application.

Selection by proximity (see figure 49) is a possibility for selecting artifacts, or manipulating them via 'programming by example' techniques (ie via a camera based system). Nevertheless a list view is also needed, because some artifacts may be too small, or not possible to select by proximity (i.e. a temperature sensor, a heating system, a certain combination of values, etc).

Figure 49 : Handheld device (storyboard): The editor functions in the GUI of a handheld device (PDA), which is a specialized extrovert gadget.

Figure 50: Handheld device: application creation (storyboard). The GUI corresponds directly to the connectivities – links (plug-synapse) model, using a simplified form of 'line wiring' connections, but suitable for 'low-level' applications.

Figure 51: Handheld device (storyboard): deletion of a connection between artifacts, and therefore deletion of an application that has only one connection.

Capability-plugs, indicating the affordances of artifacts (see chapter 9), are used in the visualization of figure 50, 51; even though they are based on sensor readings they are more complex constructs than plain sensors. These connectable abilities are created and allocated (ie by artifact manufacturers as factory settings) to each artifact as set of capability-plugs, that is predefined in their software (as in the e-Gadgets project assumption), or can be evolved or adapted by users who would be capable or willing to create and port such software constructs into artifacts, or could be instantiated using ontology based definitions.

Advantages and disadvantages of the interface

A clear advantage of the graphical user interface representation shown in figure 50 is that, following the visualization of the plug-synapse mental model for the underlying technology, it is very easy for novice users to grasp the concepts and use the interface (as can be seen the evaluation results reported in chapter 11 and appendices 1 – 4). People in the evaluation sessions were able to make simple directional associations between two artifacts, and thus created simple ubiquitous applications, getting initialized to the idea of programming their environment.

When two artifacts are associated, a link between them is created, with standard, automated specific settings on the configuration. Simple commands (IF …THEN) or

Booleans, used in order to modify the preset-values can be added via menus on top of the association links.
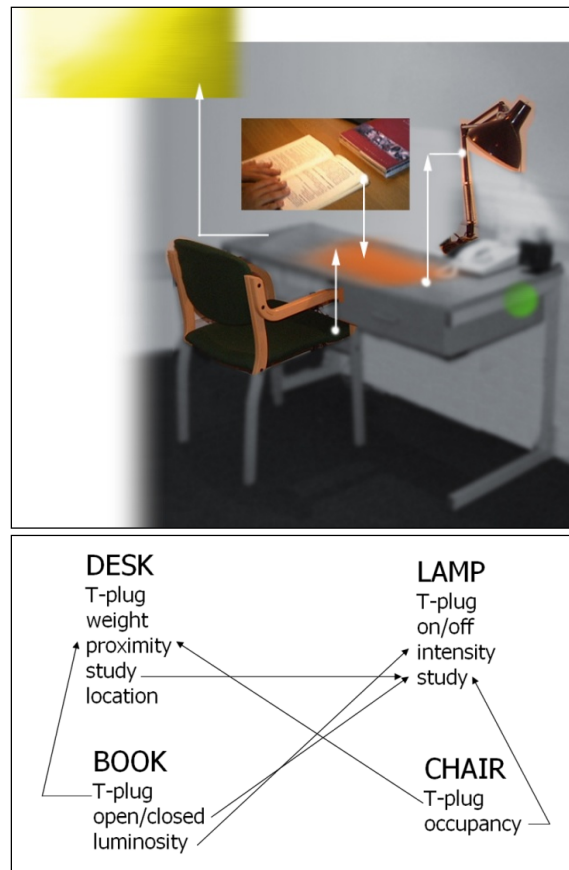
Figure 52: Top: the application here is described visually: "when the book is on the desk, and the specific chair is near the desk, then turn the light on" (T-plug refers to the Identity of the artifact). Bottom: the resulting 'spaghetti'-like visual effect of many links, based on the direct model visualization.

The disadvantage of this visualization approach is that it enables rather simplistic configurations. When more than one association is required between artifacts, the visualization becomes that of link wiring, that is more complex and visually messy (figure 52), and it is difficult to reason on the cause and effect relationships. (To solve this, a matrix visualization for associations (figure 53) was proposed, that is described in the following section)

A more important issue is the fact that starting from the level of artifact selection (and actuator / sensors), for establishing an application, is how most people think of ubiquitous applications. People tend to express their wishes in terms of functions they want. i.e. "I want to always have enough light when I am reading", rather than "WHEN the book is ON the desk, AND the luminocity of the space is low, THEN turn the closest available light source ON). Moreover there is an issue with people reasoning how to cease the running of the application., i.e. once the light is ON in the above example, one has to set a second application in order to Deactivate the first one, and there is still the <u>risk of getting in a spurious activation/deactivation cycle when specifying trigger conditions without enough thought.</u> Also, apart from the user of artifacts, <u>use of generic conditionals and timers need to be added,</u> that are also elements that have to be represented as artifacts, although they are more abstract and generic in nature.

13.4. Other Interface experiments for End User Programming

A grid visualization was also a visual experiment conducted (Mavrommati and Kameas, 2003a) as a graphical user interface for associating artifacts (see below). Although this visualization was providing an overview, it is also not a fully scalable approach: the matrix has a limitation due to the certain amount of lines – even if scrollable it would have limited visible area; however it could support zooming. Users can click on the boxes of the meeting points of artifacts rows, in order to initiate links between them. The grid, and the need for directionality of links, makes it necessary to list artifacts twice (horizontally and vertically), which can confuse users and adds unnecessary repetition in the available screen space. This grid overview can be useful to the more adept application designers, since a larger amount of links and thus more complex applications can be created. Nevertheless for novice users it is complex to understand the established application, and therefore is not considered as valid an approach as the more direct associative one using links

wiring, (that is mentioned in the previous section). To indicate directionality of links, arrows are shown in the intersection cells.

One advantage of the grid (figure 53) is that all connections can be displayed (while the pipeline model only displays as many rows as connections, at the grid-view all possibilities are shown). An additional benefit is that it allows easy view of the events from a device to itself (which is possible in devices that are systems comprised of subparts, i.e. a stereo system or a video recorder, or an in-house security system).
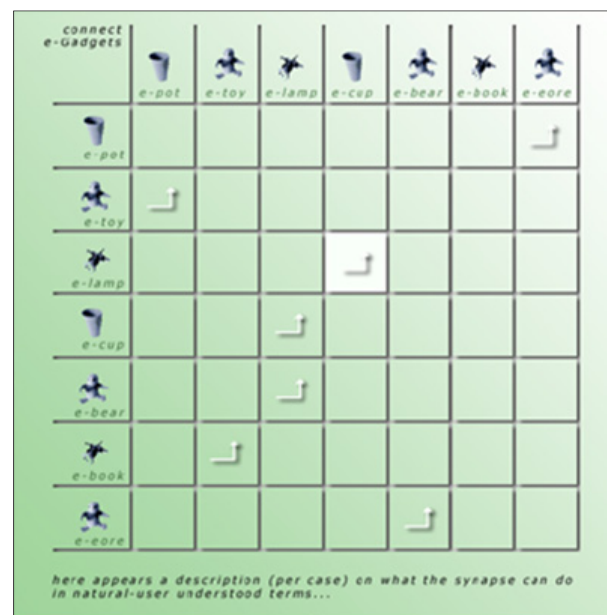
Figure 53: The grid view for establishing links between artifacts in the e-Gadgets project case.

Steps for users to associate artifacts and create Ubicomp applications:

In the above grid interface, users need to proceed by a limited number of actions:

1. Think of application, in terms of artifacts and the relationship between them
2. View existing applications, or View existing artifacts
3. Select artifacts from their environment (or their properties)

4. Select and associate artifact capabilities, into pairs, establishing links

5. Add the parameters of the links ('rule editing' of the application).

6. After creating all the applications, give a name to the application that is created and a description so that one can remember it by.

Several problems are expected to occur with testing the applications that has been created. One of them is that not all conditions may be present at the time of creation, (for example, there may be daylight and the application may require low lighting conditions, or it may be summertime, and the application pertains winter temperatures, or it may be regarding a party situation, while only one person is present in the space at programming time). Subsequent problems occur with debugging the application, since usage conditions at debugging time need to be replicated. Emulators and debugging facilitation mechanisms are needed to solve this problem. Collaborative development, where one may ask for help from a team of more advanced programmers, may also provide ways to ease such difficulties.

Another issue is for a user being able to remember over an extended time period, what the application does, judging by its name. Description and keywords are therefore considered a useful addition to Editors.

13.5. Form based editors: the ASTRA interface scenarios

The design and scenarios for the ASTRA End User Tools (EUTs) (see ASTRA Deliverable D4a) are described here as a case study for exploring visualization and functional possibilities of End User Development Tools. The ASTRA tools concept use a combination of the Pipeline model and forms that can be alternated in use with a natural dialogue interface, whereby end users can design application scenarios (that they can then observe or alternate to the Pipeline view to understand the interconnections better).

Goals and approach

The scenarios of the ASTRA set of tools aimed to exploit the underlying ASTRA Service Oriented Architecture (SOA), and give users the possibility to take advantage of predefined services, as well as support the easy creation of applications by communities and the interaction among these applications. The aim was to develop a prototype End User Tools Suite that would show user management, rule editing, and application management in a SOA based implementation, mapped to the ASTRA connectivity theory and model (Calemis and Mavrommati, 2007), (Mavrommati and Calemis, 2010). Flexibility in terms of representation was another objective: being able to run in many devices, but also to support alternative interfaces (in order to test for example alternative models for rule editing –supported by appropriate information visualization).

Approach

The ASTRA End User Development Tool (EUT) proposes the use of a web interface and is based on the ASTRA SOA via the respective Application Programming Interface (API) (Mavrommati and Calemis, 2010). This API can be used as a basis to integrate future interface developments. The ASTRA EUT aim to serve as a basis for integration of other developments, such as alternative interfaces, adding an ASTRA pervasive application part to other third party awareness applications and mechanisms for enabling community development. So it aimes to serve as the basis for enabling community driven breakthrough in end user development for pervasive awareness applications.

Initial experiments with rule editing implementation, in the ASTRA project case, progressed with the definition of Scenarios (and subsequent Use Cases based on the SOA and mapped on the ASTRA Formal Model), through annotated interaction diagrams, to the design of GUI interfaces supporting the interaction and finally an implemented version of EUT that was interfacing with the SOA. Still, what is presented in this section is the concepts from the design phases, and not the ones that

were finally developed; this is so as not to limit the ideas into a subset occurring from the constraints of deployment.

Considerations regarding the Interface

Programming by example was ruled our as a selection for the running version of EUT, due to the problems and the complexity that it could introduce. While in some cases (the straightforward cases, where there are fixed sensors, no ambiguity, no implied inferences, no other programs or intrusions) programming by example can help, these are also cases that are considered easier to program manually. In the most complex cases where it would make sense to program by example, then problems may occur - due to ambiguity of input, ambiguity of intention, what is implicit and what not. To achieve programming by example in these most complex cases, usage of historical data and observation over time may be needed. It is noted again here that not all ubiquitous computing application scenaria can be realized by programming by example.

More complex forms or interaction paradigms can allow more possibilities for creating applications, but are harder to use for end users (unless the target group becomes very specific and therefore more limited in number). For the target group of elderly for example (that is a key target group to pervasive awareness projects such as ASTRA) and their families, simplicity in the form of interaction is a key element, so a step by step wizard can be a suitable interface. There is an obvious trade-off with simplicity: it limits the scope of possibilities in EUD. The trade-off is that by simple, easy to use mechanisms, there are fewer possibilities for creating more complex applications.

Some of the possible UIs that have been ruled out, for the ASTRA project case, after initial consideration are:

TUIs: Tangible User Interfaces (radio dials, knobs, etc) can often be very target-group specific. Moreover they may provide limitations regarding the possibility to

generalize their functions; they need to have predefined functions, and can be more appropriate to be used for certain applications as their tangible controls. This is feasible by associating TUIs (i.e. sliders, buttons, objects) with certain application functions via the Editor, for manipulating certain aspects of the application, rather than using them as generic editing mechanisms. Some initial experimentation with tangible interfaces and video prototypes were conducted at the start of both the ASTRA and e-Gadgets projects but were not taken as viable solutions on their own. Designing and developing a tangible interface would take a prohibitive amount of time compared to a form based/GUI interface and it would divert effort from the exploration of the appropriate conceptual model for supporting End User Programming. Furthermore tangible interfaces make distribution and scope for deployment logistically more complex - especially if the aim is to support the involvement of communities of users. Distribution and support would be costlier, less scalable and overall more difficult by tangible UIs than what is currently possible via the internet. Tangible interfaces are considered as a possible path addressing the specific target groups that one could explore in the future, but is a rather limited one in terms of broader scope.

Wizards were initially considered either in spoken or text based system, aimed especially to the target group of the elderly. Wizards have the risk of increasing the interaction time and end up tiresome, because of offering the many programming choices in a step by step way. The steps involved here can be simplified by allowing selection of choices from an application list, labeling it as required, then followed by a (semi) automatic appropriation.

Although Forms and Wizards may seem at first glance an approach not as rich for Ubicomp environments as tangible UIs or direct manipulation, yet, they constitute approaches more portable to different devices; in addition we consider them as being more suitable for the average user with some experience in filling of forms or navigating in simple interactive applications. Moreover forms can be complemented

by widgets and query dialogues, as alternative controls, making them more flexible and broad.

Regarding GUIs for more detailed development of applications, alternative views of rule editing are considered, such as using the visualization of the pipeline, connection node diagrams (using tree-structure visualization), etc. Two alternative views (within the same GUI) for rule editing have been proposed in the design phase of ASTRA (figure 54). A connection mode, using a combination-view of a pipeline-tree diagram and an alternative free natural text dialogue, was proposed (but could not be implemented in the scope of the project). The rule editing is supported by visual feedback at the lower part of the screen that combines text and tree structure, and can be manipulated at the nodes.
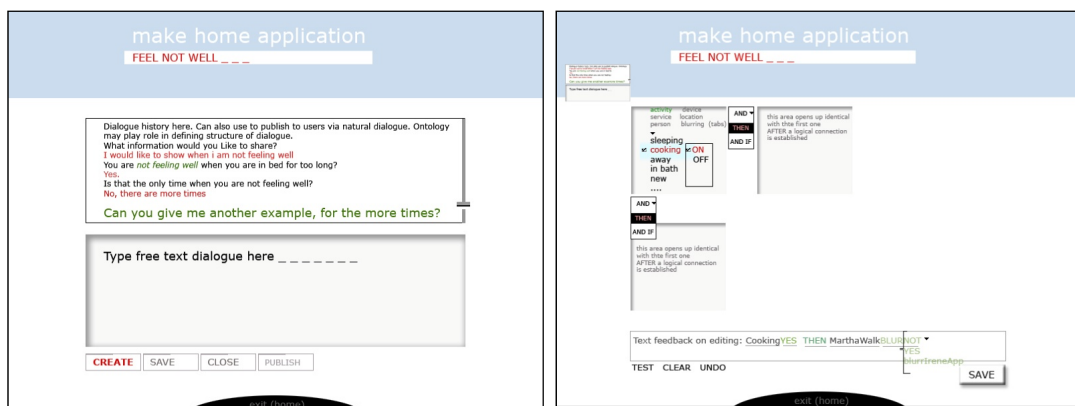
Figure 54: Alternative views for rule editing: Text and Pipeline view. In the pipeline view a number of preset awareness applications can be selected, as well as applications from the community repository that can be modified. Other parameters can be added, configured as part of the ubiquitous application.

In the ASTRA tools, the availability of a list of predefined applications was proposed, where the novice user will only have to select from the ASTRA awareness applications menu (from pool of ASTRA applications created by the community) and also select the person(s) they want to share awareness information with. Appropriation can then be done automatically in their own environment, informing

them on the predetermined specifics of the devices used for capture/actuations. There are follow up issues raised on being aware of which awareness applications are accepted and which devices participate in them (i.e. remembering), that need to be addressed in supporting elements of End User Tools (a proposed design solution for this is the 'Idle mode', (described in chapter 12). Awareness states (ie sleeping, cooking, location, etc.) are introduced as a way to provide ready-made components for awareness applications, which can be inferred from separate sub-systems within the home.
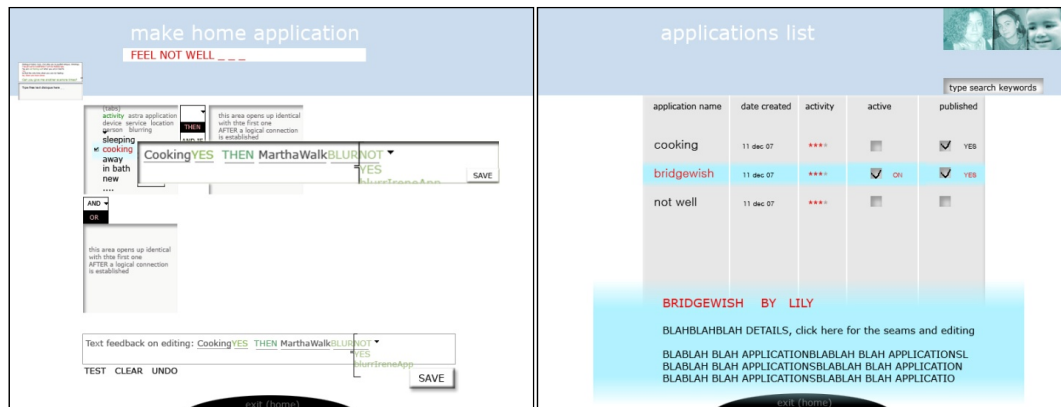
Figure 55: ASTRA sample screens (drafts): Rule Editing, and Application List
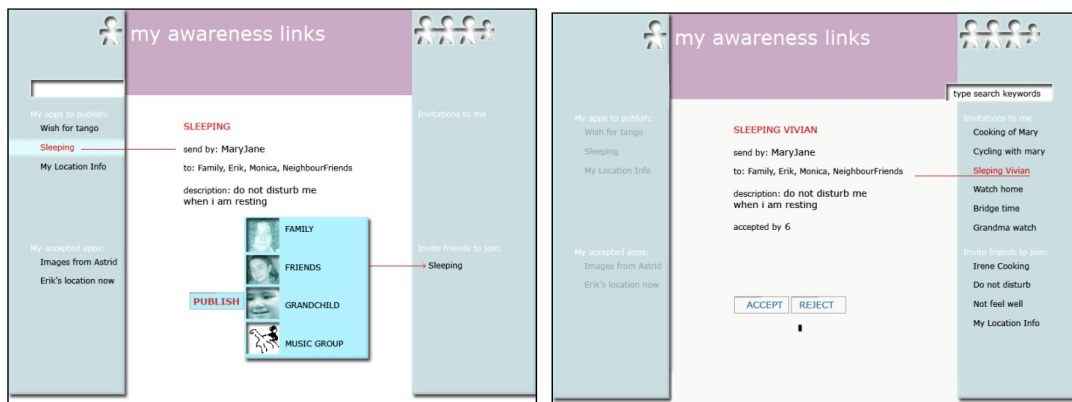
Figure 56: Awareness connections Screens, displaying the shared receiving (left) and sending (right) side of the application. As well as allowing for publishing and subscribing awareness applications to communities, the screen also provides a quick overview of all ASTRA awareness applications shared. A Search function that directs to the community repository of applications is also available on top.

13.6. General programming issues:

Some programming issues (that have been observed in the course of the design work reported in previous sections) can occur when the status is 'momentary' versus being 'event based' (having duration). For example Anne wants to know when John is back from school; she has set the pervasive awareness system to let her know that he is back, when someone sits on the sofa - which has embedded sensors (i.e. pressure sensors on the seat). So it gives the signal that John is back from school when he sits on the sofa, but without taking into account when that activity should end in time; i.e. the next day is he still back from school (from the previous day's state), or is he not in yet? The system knows that John is back in the house from school, by checking if he sits on the sofa. Once this is set to TRUE then he is back, without the possibility to reverse this state. Nevertheless, once he has sat on the sofa, he is perceived as being in the house and this status is not checked again; thus the following day, the system perceives him as still being in the house, although he is no longer sitting on the sofa.

To solve this problem, models are required, that take into consideration the length of time, status updates and transitions. The status should not be event-based but based on the transitions that occur. This may be hard for end users to figure when the applications they designed exhibit erratic behavior (other than the expected behavior). They need to program into their applications the time when to restart the application. In this case, the end user programming model should not be based on the definition of Booleans (event based models), but on state transitions. Events should be handled by system designers as input/output transitions, involving an entering state and a closing state, rather than a momentary event occurrence.

Figure 57: Extract from a service that employs the expression editor to let the user express to somebody that she is in a good mood on weekends with nice weather (Image is credited to G.Metaxas, Amelie system, Source: ASTRA D4, 2009).

Figure 58: Example of two heterogeneous ranges combined in a common group based on their common aspect (Source: G. Metaxas, ASTRA D4, 2009).

13.7. Conclusions

In this chapter interface characteristics for Editors were presented, such are the following:

- The use of web forms or xml constructs is suggested, providing an umbrella structure that most are adept with, and also providing the possibility of cascading detail.

- The need to have multiple representations for End User Programming is stressed: natural language dialogue, in parallel with other modalities. Interfaces based on the pipeline view or the Capabilities and Links (Plug-Synapse) model can be effective, but also step by step wizards are needed for less computationally adept audience.

- Attention to the language used: it has to be easily understandable, and be based on simplified concepts of the model of the system, so that the users know what they are expected to do. The Capabilities and Links (Plug-Synapse) model is seen as a possible model where system explanation can be based, although it does not need to be followed in a direct model representation within the programming interface (nevertheless for easy-to-understand and use systems, this coupling helps).

14

14. Towards a framework for the design of Ubiquitous Systems supporting end-user development

"Generalization is a verbal act of thought and reflects reality in quite another way than sensation and perception reflect it". Lev Vygotsky

14.1. Introduction

This chapter outlines theoretical and methodological constructs, towards the definition of a Framework for the design of Ubiquitous Systems supporting end-user development. The framework is based on a multi-disciplinary perspective, attempting to bridge design and system constructs from a User Experience design perspective, in order to provide common understanding to Ubicomp Development teams.

End User Development in ubiquitous computing environments is an area still in the early stages of research; design frameworks targeting End User Development in Ubiquitous Computing systems do not exist. A first structure of concepts and methodologies, towards what can be a framework that addresses the design of Ubiquitous computer systems that support End User Development, are reported in this chapter. The proposal is broad and consists of theories and methodologies for the design and engineering of Ubicomp systems that support End User Development; elements of the theories and methodologies will be outlined in the following sections.

The Broad Framework for the design of Ubicomp Systems supporting End User Development came as a result of a design rationale process on several sub-issues of Ubiquitous Computing and End User Development (Mavrommati and Darzentas, 2011). Design Rationale and Scenario Based Design were considered a fit way to explore the area of ubiquitous computing and end user development because *"it links creative intuition and grounded analysis, and it's constructive enthusiasm is best suited for ambiguities of technology"* (Carroll, 2000, pp315). Ambiguity and the open ended nature of scenarios support both creative design and analytical thinking, with scenarios helping to form a more concrete hypothetical view in the technological future.

Sub issues addressed include Human Computer Interaction issues, Ubicomp System Architectures, End User Development and its users; End User Development Tools, functionality and interfaces. Previous chapters examined those sub-issues, and by responding to them and questioning solutions, as provided insight that was brought into the proposal for a broader framework, as well arguments supporting the necessity of End User Development in Ubicomp systems (Figure 59). An example scenario of users developing Ubicomp Applications can be found in appendix 6.

What is defining this approach is the attempt to form a broad multidisciplinary framework. It is broad in that it tries to view the system in its totality, determining

all its component parts, from the theoretical foundation, to user experience, to system design. The broader system (and therefore the framework that explains it) is considered to be more than the sum of its parts. There are mutual implications in the relationship of Users, Social structures, and Ubiquitous - End User Development systems as they function together in an iterative evolutionary cycle. A key aspect is the potential for emergent behavior, evolution and growth of people, artifacts and tools, social and organizational structures. Concepts described in this framework are interacting and interconnecting among the three different perspectives that are proposed: theory, interaction, and system design. The parts are interconnecting while key concepts in theory are echoed in design concepts, which, in turn, correspond to system design concepts and constructs. The social and cultural dynamics are addressed here as an inseparable part of ubiquitous systems (with Activity theory playing a key role as a theoretical foundation).
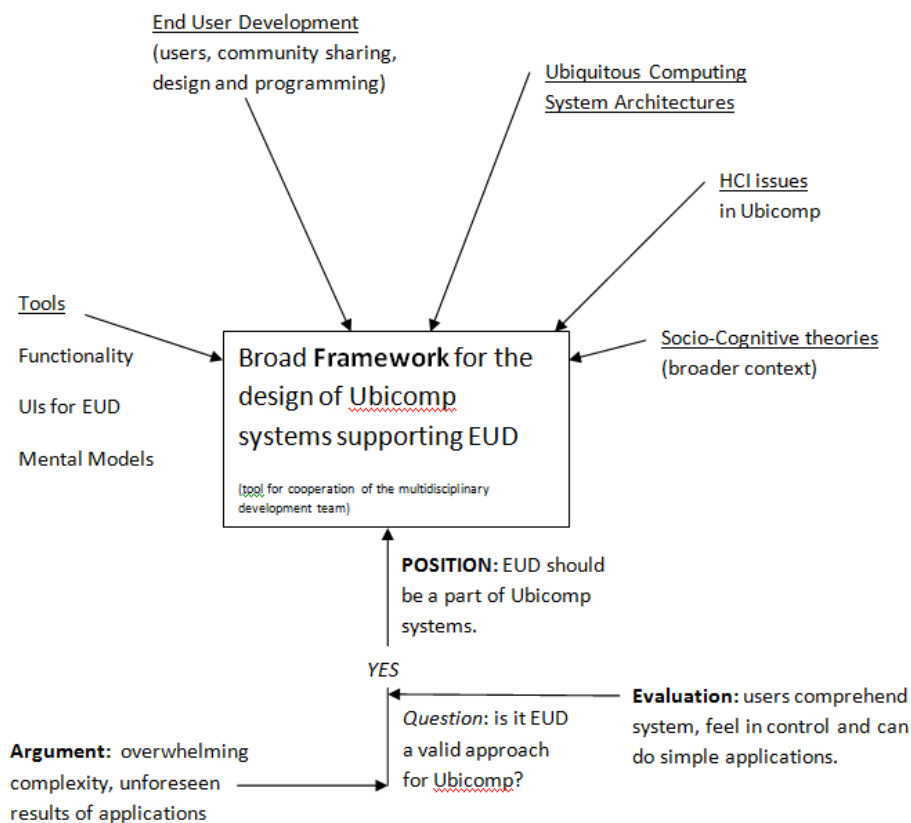
Figure 59 : Chapters of the thesis provided insight for a broader Framework for the design of Ubiquitous Computing systems that support End User Development.

14.2. Related work regarding Ubicomp EUD frameworks

At the moment of writing this chapter, in scientific literature no other frameworks are mentioned that specifically concern end user development within the context of Ambient Computing Environments. It has to be noted that there are significant advances in the more general area of End User Development, that are summed up in the collective volume: "End User Development" (Springer, 2006), edited by Lieberman, Paterno and Wulf (Lieberman, Patterno, Wulf, 2006). The work of Bonnie Nardi (Nardi, 1993): "A small matter of programming" (MIT press, 1993) has also to be noted as not only influential but still very up to date with current concerns of End User Development.

In Chapter 18 of the collective volume on End User Development (Lieberman, Patterno, Wulf, 2006), a semiotic overview for end user development is attempted by de Sousa and Barbosa (De Souza and Barbosa, 2006.). This work provides an overview of grammatical structures and syntaxes, and different manipulation structures based on semiotics.

In Repenning and Ioannidou (Reppening and Ioannidou, 2006) design guidelines are discussed for End User Development. They list thirteen generic guidelines, extrapolated from their experience with AgentSheets simulation authoring environment (they address errors, syntactic guidelines, incremental development, among other suggestions). These guidelines focus on virtual worlds and object programming guidelines, and are not concerned with the challenges of EUD in ubiquitous computing environments.

The Meta-design is a notable framework for the future of the overall field of End User Development (Fischer and Giacardi, 2006), (Fischer et al, 2004) outlines the overall field of End User Development, in many application areas (i.e interactive art, social creativity, open source, to mention a few areas covered). The authors note that they believe that End User Development *requires a change in mindsets and*

cultures: people that want to be active contributors and designers, and not just costumers. (Liebernan, Patterno, Wulf, 2006, pp454).

(Dey et al 2001), for a phD thesis, have proposed a Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, that is concerned with Ubiquitous Computing Environments, although it is not addressing the user as a designer and developer. This framework describes abstractions that form a conceptual framework for supporting context-aware applications.

In summary of future perspectives for EUD, Klann, Paterno and Wulf note (Klann et al, 2006, pp479) that the architectural challenges for EUD enabled systems becomes apparent in the vision of Ubiquitous computing, whereby an array of distributed interconnected devices is supposed to provide a consistent and context sensitive service to end users. The context of use is the combination of users' profile, (interests, tasks, and background knowledge), the environment, and the available augmented devices. They note that "*while adaptivity can carry a long way, user driven adaptability remains crucial so that users can finetune the system to their goals and work practices. These* adaptation activities also enhance the users' competence and support their understanding of the system". In attempting a roadmap of research, (Klann et al, 2006) note that representational forms and interfaces for end user development environments need to be researched on their own merit, particularly regarding creating and evaluating domain specific and graphical (2D and 3D) formats. Collaborative aspects of EUD are also mentioned as a key element.

The adopted supporting technological framework and infrastructure for Ubiquitous recombinant systems that can support end user configuration, within the proposed framework, is based on the constructs described in detail in (Drossos et al, 2007) (Goumopoulos et al, 2009). This is a Framework for the Software and Hardware Systems Design, referred to as the Gadgetware Architectural Style, -used for the design and development of ubiquitous recombinant systems-, and it is is subsequent

work stemming from the e-Gadgets[7] project, (e-Gadgets project website), (Mavrommati and Kameas, 2003b), (Kameas and Mavrommati, 2005), (Drossos, Mavrommati, Kameas, 2007). This methodological framework addresses the hardware and software system design for recombinant computing systems that can also support end user configuration; this framework focuses on the hardware and software system, without assuming the perspective of user experience design, or addressing the broader theoretical foundations.

Current EUD frameworks are either generic umbrella frameworks of all the application domains of EUD, or concentrate in virtual and software environments. End User development needs to have a focus in the application domain (Klann et al, 2006); ubiquitous computing is such a domain whereby research needs to provide approaches with practical relevance for this specific domain. A first structure for treatment of underlying theories, as well as experience design and system design concepts and methodologies, towards a framework for the total design of Ubiquitous computer systems that support End User Development is presented in Figure 60.

---

7 The e-Gadgets project has drawn inspiration from the e-Slate component platform for educational applications (e-Slate website), applying concepts from microworld component architecture (Birbilis et al, 2000) to the world of tangible artifacts.
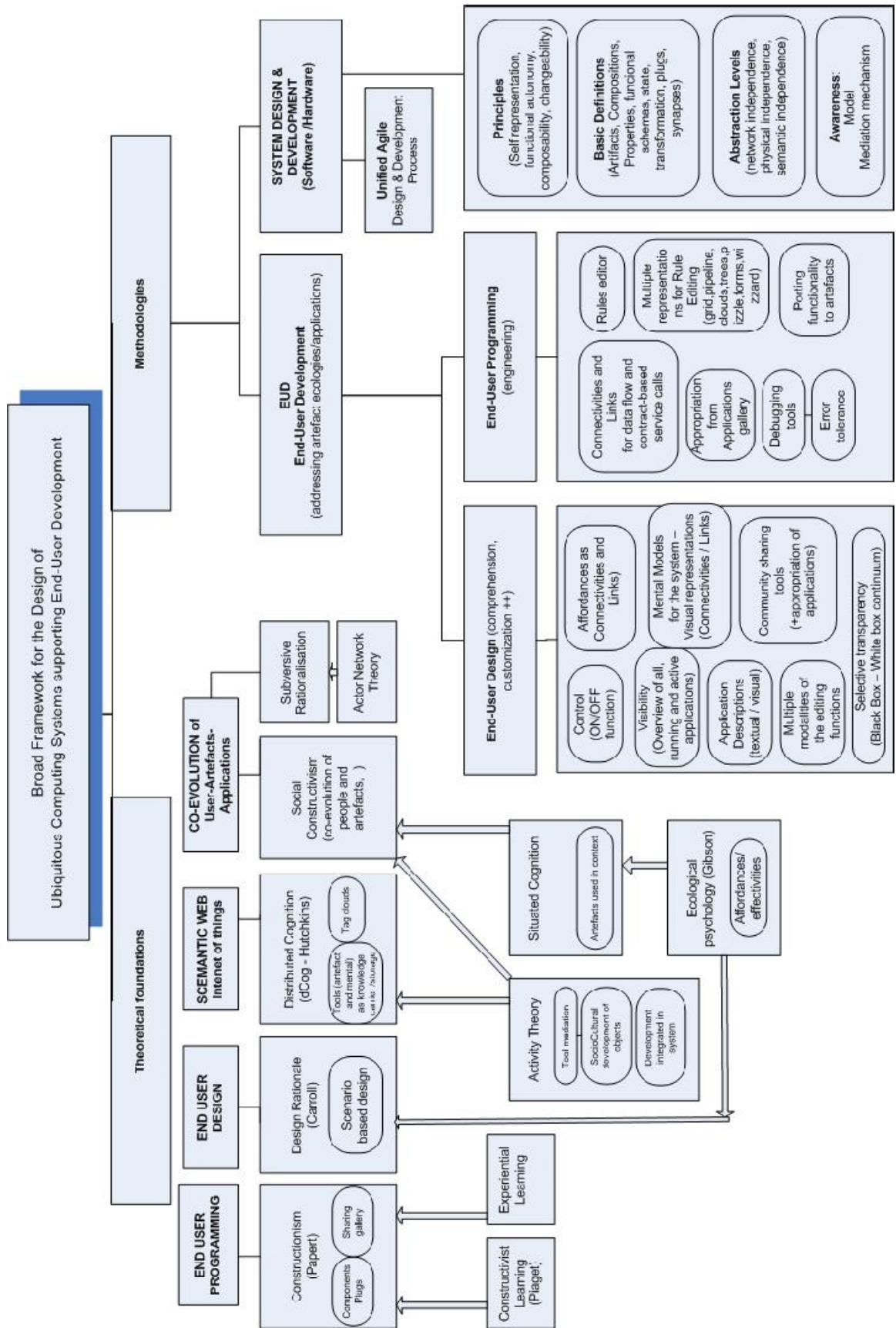
Figure 60: An outline schema of the framework, for the design of end user development in Ubiquitous environments

14.3. Theoretical Foundations

Concepts are things; they are evolving artifacts of the mind. Concepts evolve, "*new concepts originate as a blending of older ones, giving rise to new emergent properties*" (Fauconnier and Turner 2002) via (Imaz and Benyon 2007).

For reasons of clarification, the theoretical foundations for Ubiquitous Computing that support End User Devepoment tasks, are classified under four broad categories, that correspond to the relating concepts that are described in the Methodologies section (Mavrommati and Darzentas, 2011). These categories are: End User Design, End User Programming (both these form the bi-polar End User Development methodologies), the Semantic Web (that loosely corresponds to the System Design methodologies), and the Co-evolution of Users, Artifacts, and Applications (that relates to the evolution of the total system, and the perspective that is assumed in this research).

14.4. Foundations of End User Programming

Contructionism

Constructionism (Papert 1986) is an educational theory based on is the idea that people learn effectively through *making* things. It was inspired by Jean Piaget and his experiential learning ideas. Constructionism argues that learning happens more effectively when people are active in making tangible objects in the real world (these ideas connect to experiential learning).

Constructionism, according to Seymour Papert (Papert 1986) is combining the following: *"From constructivist theories of psychology we take a <u>view of learning as a reconstruction </u>rather than as a transmission of knowledge. Then we extend the idea of manipulative materials to the idea that <u>learning is most effective when part of an activity the learner experiences as constructing a meaningful product</u>."*

Construvist learning, (based on the ideas of Piaget) is inspired by constructivist theory according to which people construct mental models to understand the world around them. Piaget's theory of constructivist learning has had an impact in educational teaching methods and learning theories. According to him, children go through four development stages which are predetermined and in accordance with their age. Children are creating and updating their mental models by gradually moving towards higher level concepts while they grow through each phase. The mental development of children would have to be at the proper stage in order to to assimilate certain concepts. As a consequence of Piaget theory, the teacher's role is that of a guide to the child's own discovery of the world - rather than being the source and transmitter of knowledge.

Constructivism is a theory that stems from Piaget's work of cognitive development. Constructivism as a philosophy (not as a specific pedagogy) claims that people generate knowledge and meaning from their experiences, through assimilation, accommodation, and correction. Human cognitive development is a continually adaptive process (Piaget, 1968). Knowledge is considered as the outcome of experience that is depending on the experience of others and on prior knowledge. Since it is our thought that conceives the only reality known to us, each new conception of the world builds on our prior-constructed realities that we take for granted. The theory of development stages became also known as constructivism, as Piaget believed that children need to construct an understanding of the world for themselves. Piaget's theory addressed the active agency in children's learning, rather than the passive receiving of knowledge. (In the contrasting theory of behaviourism learning, on the other hand, children are considered as passive recipients, receiving influences from their environment which shape their behavior).

The soviet scholar Vygotsky, at an earlier time, has come to the same conclusions as Piaget regarding the constructive nature of development, yet his work remained unknown to the west until after 1970's, (after Piaget's influential work had been published). Vygotsky's contributions are described in his books: Mind in Society

(1930, republished in 1978) and Thought and Language (1934, republished in 1986). Piaget only became aware of Vygotsky's theories after the latter had died, and acknowledged Vygotky (Piaget, 1962) for having anticipated much earlier many of the important discoveries of development psychology, that are credited to Piaget.

Social Constructivism

Social constructivism refers to the individual making meaning of knowledge within a social context (Vygotsky, 1978). Social constructivists suggest that it is through the social process that reality takes on meaning and that our lives are formed and reformed through the dialectical process of socialization (Berger and Luckmann, 1966). People's understanding of science follows a similar dialectic. The artifacts that people invent are continually shaped in order to be adapted to the continually evolving context of the human life and environment.

> People and artifacts are interdependently shaped: People are shaped by their interactions with artifacts, tools and machines (of physical or conceptual nature) and at the same time artifacts and tools evolve and change in response to the use that is made of them by communities of humans.

The theory of Social Constuctivism, applies the general ideas of constructivism into social settings. According to social constructivism, social groups create knowledge for one another and share it, sharing artifacts and meanings, subsequently creating a culture by collaboratively sharing constructs and artifacts. At the same time, immersion in a culture of shared meanings, concepts, artifacts, and tools, pertains to learning, on many levels, on how to be a part of this cultural group. Origins of these notions are also attributed to Lev Vygotsky. Social constructivism, in effect, extends constructivism theory by introducing the role of other actors and culture in human development.

14.5. Theoretical Foundations of End User Design

Design Rationale

Carroll has extensively argued (Carroll and Rosson, 2003), (Carroll, 2002), (Carroll, 2000) for considering design rationale as theory, and provided a related framework. Carroll argues for the value of capturing design rationale for understanding system design.

Design Rationale is the reasoning that leads to design decisions. Documenting design rationale is important for understanding the context behind design decisions and validating design decisions. It helps those who are trying to interpret ambiguous design decisions or examples that don't fall clearly within a design principle, and to avoid going back and changing design decisions without knowing the original reasons in the first place. A design rationale can be an important tool in arriving at the initial design decision in the first place. Rationale should give advantages and disadvantages of a choice and include rejected alternatives (so that those alternatives don't keep popping up for reconsideration).

Design Rationale in the context of this framework is considered as a foundation that can be applied not only to the design of systems (by teams of engineers and experience design specialists), but can also be used also as a methodological foundation for end users to act as designers of their own applications. Scenario Based Design methods can provide insight and inspiration for mechanisms supporting scenario based development as a means for end users to conceptualize and express applications ideas. Design Rationale is a theory based on the *ecological approach* of Gibson (Carroll and Rosson, 2003, pp.440), (which serves as a theoretical foundation to the social constructivism approach as well, influenced equally by Activity Theory and Ecological Psychology).

14.6. Theoretical Foundations relating to SemanticWeb

Distributed Cognition - DCog

Distributed cognition is a framework that involves the coordination between individuals, artifacts and the environment , that was developed by Edwin Hutchins (Hutchins 1995), (Rogers, 1997). According to DCog human cognition is distributed in the environment, by placing reminders, knowledge and facts onto objects and tools (internalised or externalised) within the surrounding environment that can be readily accessed so that this knowledge is retrieved. It emphasizes the social aspects of cognition, taking its influences by activity theory and the work of Vygotski.

According to (Hollan, Hutchins, and Kirsh, 2000), *"distributed cognition views a system as a set of representations, and models the interchange of information between these representations. These representations can be either in the mental space of the participants or external representations available in the environment"*. Cognitive processes are distributed between different members of a social group and artifacts (internal or external). Cognition involves the coordination between the artifacts, processes and people, and have a causal relationship through time, with earlier events impact and transform events that come later in time.

Activity theory: a broad philosophical framework

Activity theory is a broader philosophical framework that has influenced education, organizational design, and interaction design. It emphasizes on social factors as shapers of interaction between agents and their environments. It originated in the 1920's from the work of Russian psychologist Lev Vygotsky, and his students and followers (his students' Leontiev and Luria role was crucial in transmitting his theory, while more recently researchers such as Nardi, Bannon, Bodker, Norman, and Carroll have applied this theory in their research in collaborative systems and interaction design research). According to Activity theory, consciousness is shaped by practice (Bonnie Nardi website). Human beings mediate their activity by artifacts; language and symbol systems were also considered by Vygotzky as tools

for developing the human condition. "*Consciousness is produced in the enactment of activity with other people and things – and is not confined inside the mental processes of the brain alone*" (Bonnie Nardi website).

Human activities are driven by certain needs where people wish to achieve a certain purpose. This activity is usually mediated by one or more instruments or tools (the concept of *mediation* is central to the whole theory). (Bannon and Bodker, 1991)

Activity Theory consists of a set of basic principles which constitute a general conceptual system which can be used as a foundation for more specific theories (Kaptelinin and Nardi, 1997), [(Liam Bannon 1997), Activity Theory, tutorial available online)]. As Liam Bannon (Bannon 1997) explains "*the basic principles of Activity Theory include object-orientedness, the dual concepts of internalization/externalization, tool mediation, hierarchical structure of activity, and continuous development. The principle of object-orientedness states that human beings live in a reality which is objective in a broad sense; the things which constitute this reality have not only the properties which are considered objective according to natural sciences but socially/culturally defined properties as well.*"

Activity Theory, as Bannon explains, separates internal from external activities. Mental processes are internal activities. Transformations between these two kinds of activities are mutual and are intertwined, in such a way that activities cannot be understood when analyzed in isolation, (separating the internal from the external ones). It is the general context of activity (consisting of both external and internal components) that determines the cycle of transformation of external activities to internal (Bannon 1997).

Tool mediation is a central concept in Activity Theory. As explained in (Bannon 1997), (Kaptelinin and Nardi 1997) tools shape the way human beings interact with reality. Bannon (Bannon, 1997), states that "*according to the internalization / externalization principle, shaping external activities ultimately results in shaping*

internal ones. Tools reflect the experiences of other people who have tried to solve similar problems earlier and invented or adapted and modified the tool in order to make it more efficient. This social accumulative experience is accumulated in the properties of tools (structure, shape, material, etc) and in the knowledge of how the tool is used. Tools are created and transformed during the development of the activity itself and are carriers of a particular culture - the historical remnants from that development. The use of tools is a means for the accumulation and transmission of social knowledge. It influences the nature, not only of external behavior, but also of the mental functioning of individuals". As noted by Bannon and Bodker, (Bannon and Bodker 1991): *"Artifacts are there for us when we are introduced into a certain activity, but they are also a product of our activity, and as such they are constantly changed through the activity. This 'mediation' is essential in the ways in which we can understand artifacts through activity theory".*

Vygotsky developed the theoretical foundation of language development - (his theories serve as a much broader foundation); it proceeded and anticipated Piaget's psychology; nevertheless his works were published after his death in 1934 but were then suppressed and only reached the West after 1958 (Lev Vygotsky Archive and biography, available online at http://www.marxists.org/archive/vygotsky/). Vygotsky referred critically to the early works of Piaget, (in Vygotski's book Thought and Language, written originally in 1934 (Vygotski 1962). Piaget has discovered Vygotski's work 25 years later, and recognized his contribution. (Piaget, 1962), Vygotski described the development of logical thinking and language in early age and of conceptual thinking later, in the course of children's interactions with adults and the world around them. He suggested knowledge is acquired from one's life experiences, through active contact with socially transmitted knowledge of adults in their environment. More recently, linguists and educationalists influenced by Piaget's Psychology have acknowledged Vygotsky's work, since it provides a broader theoretical understanding of the relationships that influence cognitive development. People must 'negotiate' with the children who have an active role in the learning process. Vygotsky's concept of a 'Zone of Proximal Development'

(Vygotsky 1978) addresses an all-round development of conceptual ability, whereby tuition and leadership is able to facilitate intellectual and social development.

Vygotsky's theory can be used as a conceptual framework addressing how human thinking could advance further via the use of (computer) tools. Vygotsky's theory, having human activity in its focus, has gained an international and multidisciplinary importance, and is especially influential in understanding Human Computer Interaction and Interaction Design (Kaptelinin and Nardi, 2006), (Aboulafia et all 1995) and in the areas of computer mediated communication and computer supported collaborative environments. Amont the HCI researchers influenced in their approach by Vygotski's Activity theory are Nardi (Nardi, 1996a) and (Nardi, 1996b), Kaptelinin, (Kaptelinin, 1996) (Kaptelinin and Nardi, 2006), Bodker (Bodker, 2000), (Bodker, 1991), Norman (in The Psychology of Everyday Things (Norman, 1988), and Cognitive Artifacts (Norman, 1991), (Norman, 1993)), Carroll (Scenario-Based design (Carroll, 2000), and Designing Interaction, (Carroll, 1991)), and Hutchins (distributed Cognition (Hutchins, 1995)), among others.

14.7. Co-evolution of Users, Artifacts, and Applications

Situtated Cognition and Gibson's Ecological psychology

The cognition approach typically approaches the perception and motor systems as input and output of humans. Embodied cognition on the other hand, considers the mind and body as a single entity, the two parts interacting with each other continually and 'on the fly'(Niedenthal, 2007). Knowledge happens via the body interacting with the world (Winkielman et al, 2009). This was founded in research that shows that movements of the hands or arms relate to human evaluation of concepts or words (Markman, & Brendl, 2005). Facial expressions have been shown to influence judgments (Mondillon et al, 2007).

In the Situated cognition approach knowledge is determined by both the agent and the context. *"Knowing is inseparable from doing"* (Brown et al, 1989); (Greeno, 1989) while *"knowledge is situated in activity bound to social, cultural and physical contexts"* (Greeno & Moore, 1993).

Gibson views on visual perception have influenced Situated Cognition Theory (Greeno, 1994). To Gibson, visual perception is not solely about input from the eyes providing the brain with symbolic representations, but more about people perceiving certain elements, by selectively viewing from a huge amount of information and identifying certain elements of the environment, that change or remain stable. Such perceptions of the environment, geared by people's intentions and evolving through time, co-determine the possibilities for use of the environment or the artifact. This process of perception evolves in time.

Ecological psychology: Situated cognition is influenced from the ecological psychology of the perception-action cycle (Gibson, 1986). Key principle of Ecological perspective, adopted by the Situated Cognition approach, is the notion of Affordances (Gibson, 1977). He defined the term as properties that present possibilities for action that are directly perceptible by people to act upon (Gibson 1979/1986). Gibson focused on the affordances of physical objects and suggested that affordances were directly perceived instead of mediated by mental representations such as mental models. Affordances are seen by Gibson as *"preconditions for activity"*, not determining behavior per se, but increasing the chances of certain actions to happen.

Objects can 'afford' certain types of actions to be done with them, as a result of their physical shape, their material, and the cues and cultural knowledge of usage. *Affordance*, a popular design conceptual construct, was originally introduced by Gibson (Gibson, 1977), describes the *relationship between objects and tasks that can be performed with them.* Don Norman (Norman, 1990) further elaborated on the concept of affordances by introducing *perceived affordacnces*, in parallel with

the actual properties of the object. To Norman, affordances stem from properties and signify *how the thing can possibly be used*. Technology "hidden affordances" by Bill Gaver (Gaver, 1991), was the further elaboration of the concept, signifying functions of interactive systems that are not always directly and immediately perceived. Gaver has addressed technology affordances (apparent as well as hidden) in the context of interactive systems design, considering them as '*properties of the world defined with respect to people's interactions with it*".

Subsequent to *affordances* the term *effectivities* has been introduced by Gibson, signifying the abilities of the person itself that determine what he/she could do, and the interaction taking place as a result. It is effectivities and affordances together, working simultaneously, that determine action and perception (Gibson 1979/1986; Greeno, 1994). Which affordances are picked up and used, is determined by the person interacting within the environment, and perceiving it based on his/her effectivities.

Based on Gibson's work, Donald Norman (Norman, 1988) developed a follow up theory of '*perceived affordances*', which emphasizes "*the agent's perception of an object's utility as opposed to focusing on the object itself*".

Subversive Rationalization, introduced by Andrew Feenberg (Feenberg 1992), describes the constructivist nature of technology. Subversive Rationalisation suggests that technologies evolve and change through being adopted, used over time, and adapted by people. Technology transformation is seen as an eventual result that is guided by social, democratic and human values. So, technology is shaped as a result of adaptation to the cultural logic of the people who use it (and is not solely defined by the technology designers). This view seems to have similarities to the perspective of Lev Vigotsky that was described earlier.

Actor – Network Theory

Actor-network theory is a 'material-semiotic' method, in that it maps material (between things) and semiotic (between concepts) relations. It is assumed that many relations are of dual nature, both material and semiotic. Actor-network theory tries to explain how material–semiotic networks come together to act as a whole, looking at explicit strategies for relating different elements together into a network so that they form an apparently coherent whole (for example an establishment that consists of networked operations between its agents and artifacts, can act as a system itself, such as for example a store or a bank). Actor-networks existence is seen as constantly in the re-making and thus their nature is transient. It has a constructivist approach in that it avoids essentialist explanations of events (e.g. explaining a successful event by saying it is 'true' and the others are 'false').

14.8. Methodologies for EUD enabling design and implementation

Methodologies for the end users to be able to configure ubiquitous computing systems are separated in two parts:

1. Methods, guidelines, tools and interface elements that enable *End User Development*
2. Methods that enable the *system mechanisms* that support the End User Development (such as the selection of a specific Architectural Style, and the related principles and mechanisms, ontologies, and system schemas that it uses).

14.9. End User Development

End User development is defined as having two distinctive parts in this framework:

- **End User Design:** this very important part of end user development signifies access, control, and the ability of customization of ubiquitous applications. It draws its influence from the experience and interaction design perspective.

- **End User Programming:** this part of end user development is closer to the system design for software systems perspective of EUD. Possibilities are provided to completely alter applications or create new ones, by giving end users syntactic tools (such as a rules editor).

End User Design

*In the context of End User Development in Ubiquitous computing applications, we introduce the term **End User Design**.* End User Design is one side of the spectrum of End User Development for Ubiquitous computing environments while the other side of the spectrum is End User Programming.

This term is used in order to refer to a significant step beyond the classic End User Customization, whereby User becomes a Designer, envisioning, planning, designing and adapting his/her own ambient experience, facilitated by shared application galleries empowered by shared knowledge. End User Design includes overview of active and running applications, manipulation functions such as the on/off switch (universal and per application), the customization and adaptation of given applications. It also includes the configuration of settings (by selecting options) in order to customize a given ubicomp application. This can be the most frequently used part, the easier to comprehend and manipulate by end users, and is defining the surface elements of the manifestation of the application rather than its core functionality. This concept also involves tools and mechanisms that support

collaborative sharing for applications that can be semi-automatically adapted to another environment.

Multimodal interfaces can act complementarily in systems that support End User Design. Text based scenarios and techniques from Scenario based development can help with application descriptions by end users.

A cascading degree of complexity, from the interfaces of the end user design functions, to more detailed end user programming (e.g. rules editing), is suggested.

Mental tools for reasoning on the End User Design of ubiquitous applications can be provided (i.e. by visual representations or promoting mental models of the system that correspond to the functional models). Such mental constructs assist users in gaining an understanding of the configurable components of the system, and the way they can be associated; it is an enabler to reason and assign functionality. Where mental models are not in place, easy and self explanatory interfaces are needed; nevertheless due to the complexity involved in end user development, and the subsequent difficulty to cater for all necessary functions with easy to use interfaces, we stress the need for straightforward and robust mental models which are very useful in the early days of this domain.

End User Programming

A cascading view of editing complexity can start from the overview and ON/OFF functions of End User Design and range to the manipulation of the specific connection rules and programming syntax. Multiple visuals for syntax methods can be tested for rule editing (i.e. Pipeline view, bubbles view, tree structure, grid). Provision of syntax can be coupled by automation or semi automation techniques for rule editing.

14.10.   System Design and development

The architectural framework for the system design of Gadgetware Architectural Style is addressing the ubiquitous computing hardware and software required. This technological framework is described by Goumopoulos in (Drossos et al, 2007) and extended in (Goumopoulos et al, 2009) to include sharing of awareness information in pervasive computing systems. This technological supporting framework is one that can allow for the end users to act as designers and developers, and as such is adopted as a concrete methodological framework regarding the Software and Hardware System Design. Key elements from the Gadgetware Architectural Style Feamework, that are adopted as related methodologies for system design, are outlined in the following sections.

Basic Principles

 (Drossos et al, 2007) describes a conceptual framework for the system architecture of ubiquitous computing systems that can be accessible to end users, The underlying principles, relating to artifacts, according to (Drossos et al, 2007), are:

- *Self-representation:* the artifact's physical properties (that are closely associated with the artifact itself) are available through digital representations.
- *Functional autonomy:* artifacts function independently –without pre-requiring other artifacts in order to function.
- *Composeability:* artifacts can be used as building blocks of larger and more complex systems.
- *Changeability:* artifacts that have access to digital storage can change the digital services they offer (and thus change themselves).

The ways that an object can be used is usually determined by its affordances and effectivities that stem from certain aspects of its characteristics (physical or digital). Affordances describe the perceived and actual properties of an artifact, that

determine how the user will handle it and the tasks that (s)he will perform with it. Artifacts get augmented by producing descriptions of their properties, abilities and services in the digital space, and by participating in broader group-compositions, they can enhance their functionality –they can be adaptive and context aware, or provide augmented or alternative functionality. Methods for collections of artifacts should be in place so that they can be configured to synergize in their functions.

Basic Definitions

The basic definitions as reported in the Gadgetware Architectural Style conceptual framework (Drossos et al, 2007) pertaining the Software and Hardware Ubiquitous system design are:

Artifacts: Tangible objects which can express their properties digitally are called artifacts. Artifacts can be augmented with sensors, actuators, processing, networking, etc., or a computational device that already has embedded some of the required hardware components. Software applications running on computational devices are also considered to be artifacts. Artifacts can be simple or composite, from a single sensor to an augmented building or furniture, clothes, central heating, a software digital clock, a software music player, etc.

Properties: Properties are representations of the physical characteristics, capabilities, and services provided by artifacts. Some properties are artifact-specific (such as the physical characteristics), while others may be not (i.e. services). Properties can be modeled as functions (either evaluating an artifact's state variable into a single value or triggering a reaction, to an actuator artifact). Emergent properties are those that result from artifact's compositions. All properties of an artifact are encapsulated in a property-schema which can be send on request to other artifacts, or tools (e.g., during an artifact discovery).

Plugs: Plugs are the digital constructs representing artifact properties. Plugs in (Drossos 2007) approach are defined by their direction and data type. Plugs are

distinguished into output (O) plugs or input (I) plugs and Input/Output (I/O) Plugs. Plugs in this approach have a certain data type, that can be semantically primitive (e.g., integer, boolean, etc., relating to the syntactic level of programming), or semantically rich (e.g., image, sound, etc. –in general services that can be used as output content that can be streamed from actuator devices). Plugs are the constructs that make visible the artifacts' properties, capabilities, and services to people, agents, and other artifacts. (Expert review suggested that the name Plugs would be better replaced with the term "Capabilities" for the better explanation of the model to people).

Synapses: these are associations between two compatible plugs. When a property of a source artifact changes (a state transition of the source artifact causing change of value), the new value is propagated through the synapse to the target artifact, resulting in state transition to the target artifact(s). Synapses relate the functional schemas between artifacts, enabling the functional / context changes. (Expert review suggested that the term "Synapses" is unknown to many, and would be better replaced with the term "Links" for the better explanation of the model).

Artifact compositions: A collection of two or more artifacts properties that can be combined (composed) for a meaningful purpose. Ubiquitous computing applications can usually be handled as service compositions. The act of composing artifacts into certain applications can be assisted by end-user tools.

Functional schemas: An artifact is modeled in terms of a functional schema. Functions in a functional schema can be as simple or complex according to what is required to define the property. Functional schemas can cover from Sensor readings to rule-based formulas (with multiple properties), to first-order logic (quantifying over sets of artifacts and properties).

State: The concept of state is useful for reasoning about how things may change (there is no hidden internal state). The values for all property functions of an artifact at a given time are the state of the artifact.

Transformation: a transition from one state to another. Transformations are the results of internal events (such as the change in the state of a sensor) or they may result from changes in the artifact's functional context, happening via the synapses of the artifact.

Abstraction Levels

The model adopted here, is as proposed in (Drossos et al, 2007). Three abstraction levels are foreseen in this model:

- *Network independence:* the Plug/Synapse model is independent of the underlying protocols, needed for example to route messages or to discover resources in realization of an application.
- *Physical independence:* the services offered by an artifact are decoupled from the artifact itself (and thus are independent and can evolve, whereas its physical characteristics cannot be altered). Thus the creation of artifact compositions does not always require the continuous presence of an artifact.
- *Semantic independence:* the description of applications (compositions of artifacts) is based on the types of the plugs that are associated, but it is independent of how the plugs are realized within the artifacts.

Awareness model

For the context of Ubiquitous Systems that enable the peripheral awareness between individuals, the framework presented in (Goumopoulos et al, 2009) is adopted. The supporting technological framework in this approach views context from a user-centered perspective, giving emphasis to the presentation of information in unobtrusive ways, within pervasive awareness systems. Information presentation can move between the center and the periphery of people's attention. (Goumopoulos et

al, 2009) classify awareness according to the awareness situation of a user or a community is as follows:

- *activity awareness,* revealing what one or more community members are doing;

- *presence awareness,* representing the knowledge of who is around in a community;

- *group awareness,* representing the knowledge of activity and presence awareness of a community;

- *situated awareness,* representing contextual awareness;

- *social awareness*, referring to the information that a person maintains about others in a social context.

For any of the above categories an awareness system may handle, it needs to tackle the questions of what users should be made aware of and how they should be made aware of it.
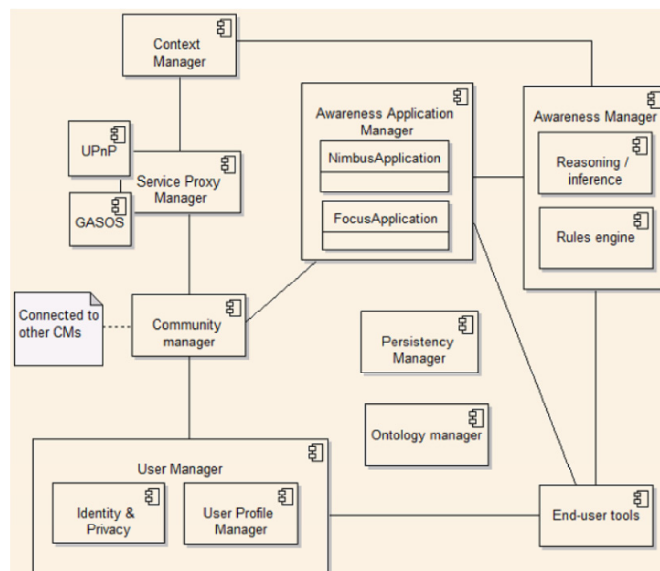
Figure 61: ASTRA Component Architecture. Source: (Goumopoulos et al, 2009)

Awareness Mediation Mechanism

(Goumopoulos et al, 2009) propose that an awareness management process be responsible to control and manage incoming and/or outgoing awareness information. Being able to control incoming awareness information of others implies the user is able to define the acceptable level of awareness detail to be presented or captured, in order to prevent disturbance. Controlling outgoing awareness information implies one is able to control privacy issues. Information can be filtered by introducing rules/triggers and constraints that can be configured by the user via a defined vocabulary (e.g. an Ontology) and the services of end-user tools.

Awareness information may be mediated in various ways, as (Goumopoulos et al, 2009) describes (see Figure 61, 62). Information can be explicitly generated by the user or collected by sensors, so that the user gets up to date information; this is achieved via *Synchronous Awareness*. *Synchronous Awareness* can be delivered via a server-push mechanism (i.e. transmitting notification services). Notification services realize the information distribution within the awareness manager. *Asynchronous information* on the other hand provides information about stored events (i.e. history of use or log files). *Asynchronous communication* relies on a client-pull mechanism, (i.e. the client retrieving the information when needed).

A model providing a loosely coupled form of interaction, as is required for dynamic awareness systems, is the publish/subscribe model, that is used by the ASTRA approach, as described in (Goumopoulos et al 2009). The publish/subscribe model is the foundation of the abstraction used in the Gadgetware Architectural Style: the Plug Synapse model (Kameas and Mavrommati 2005), (Drossos et al, 2007), (see Figures: 18 and 19).
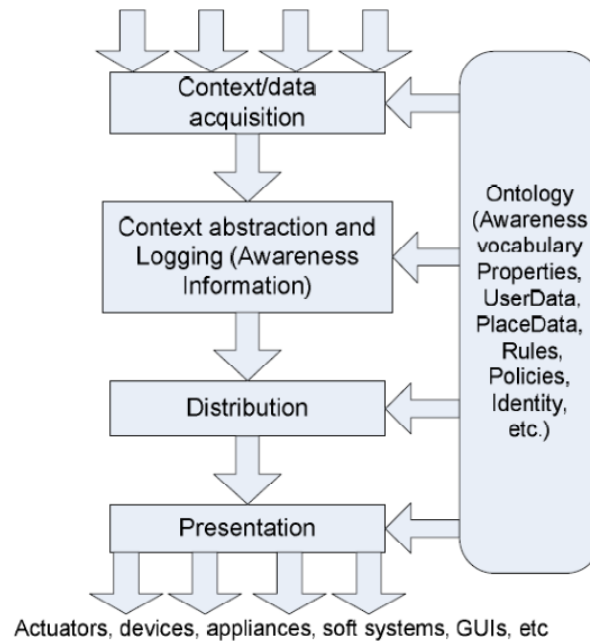
Figure 62: The Awareness Management Process. Source: (Goumopoulos et al, 2009)

System Design and Development Process:

The Agile process for system development is a process characterized by adaptive, incremental, iterative development, where cross functional self-organized teams work through system requirements and solutions. Agile development methods emphasize the importance of teamwork and face to face collaboration for development, with the participation of all stakeholders, and adaptability of the process throughout the project's span (by tasks broken into small increments and minimal, short term planning). Written documentation is minimized, due to emphasis on face to face communication in small teams (5-10 people), while working software is considered as the measure of progress.

Systems design and architecture emerge in increasing detail, as the team works through and through the design, in iterative development cycles (Figure 64), gradually focusing from the broader framework to the more specific and detailed

aspects of the system. The Agile Process steps to System Design (see Figure 63) broadly are:

1. Identification of System and Architecture Objectives (in each project phase).

2. Key Scenarios are defined, and used as a way to focus in priority issues when going through development and evaluate the implemented architectures at the end of each phase.

3. Overview of application: Understanding deployment, (such as the architecture's styles and technologies involved), so that the foreseen application types can operate.

4. Identify key issues (Hot Spots) based on quality attributes and the architecture framework (so as to prevent shortcomings of the design of the application).

5. Create possible system architecture solutions, which are then evaluated against the key scenarios, hot spots, and deployment constraints.
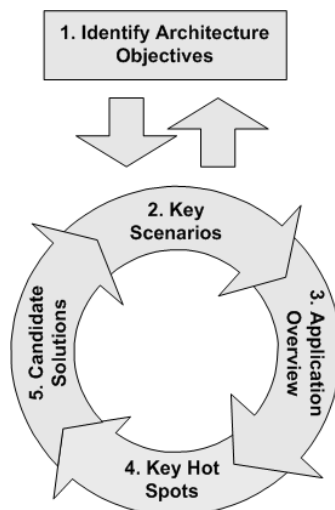
Figure 63: Core Architecture Design Activities in the Agile Process (Source http://www.guidanceshare.com/wiki/Agile_Architecture_Method_Explained)

The Agile Manifesto (Agile Aliance, 2001) puts mode value on (1) individuals and interactions over processes and tools, (2) working software over comprehensive documentation (3) customer collaboration over contract negotiation and (4) responding to change over following a plan. Designs stemming from the Agile Process (Figure 64) are therefore emergant and not defined up front, while Agile developers *'learn by building'*. In this aspect, the User Experience (UX) design process resembles that of an Agile iteration (Kreitzberg and Little, 2009): UX design (following the Iterative Design Process) is built intrinsically on of building a UI prototype, testing it with users and learning from user reactions, in iterative cycles between design, prototyping and evaluation, similar to the Agile Process whereby Design and Development proceed as an intertwined spiral (Figure 64) in line with four Agile values presented in the Agile Manifesto. [More details on the Agile process and its relationship with User Centered Design (UCD), can be found in the articles of (Shore and Warden, 2007), (McInerney and Maurer, 2005), (Cohen et al, 2004), (Hudson, 2003), (Hwong et al, 2004), (Detweiler, 2007), and (Blomkvist, S. 2005)]. Due to this existing relationship between the Iterative Design Process and Agile process, Agile is adopted in this framework as a process bridging the disciplines of Experience Design and System Development, a process that can be used by multidisciplinary teams in order to design and develop ubiquitous computing systems.
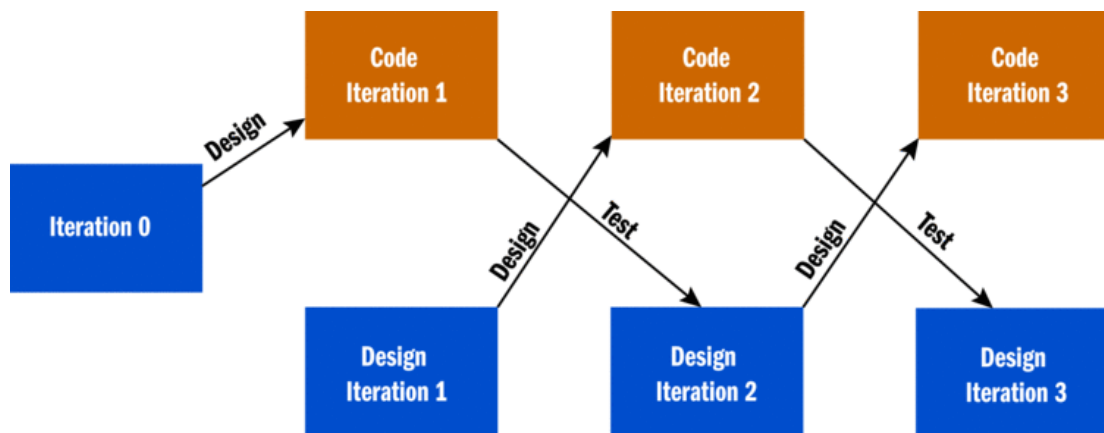
Figure 64: The Agile Process (Source: http://msdn.microsoft.com/en-us/magazine/dd882523.aspx)

14.11.   Open Issues

An issue that is not extensively discussed in the course of this thesis, is that of injecting (uploading) new functionality to artifacts. Nevertheless it is noted in the framework as a necessary concern that needs to be addressed. In low level artifacts, such as sensors, connectivities can be their mere output readings. Adding higher level 'Connectivities' that relate to the artifacts affordances or observed usage can be assisted by intelligent systems and ontologies. Software developers should also be able to port new connectivities in artifacts, automatically (in the form of software upgrades), or created intentionally and injected into artifacts.

Appropriation of ubiquitous applications into different environments, and debugging of applications are two very important issues for End User Development in the domain of ubiquitous computing. Appropriation may be assisted by ontologies, and tweaked by end user decisions and debugging mechanisms. How to know of errors and avoid them, recovery and debugging are very important issues that are not easily addressed in ubiquitous applications. Multiple representations in a step by step design-creation-explanation approach can help with reasoning about possible errors. But it is not easy to test the applications given that the conditions that apply for their triggering are not easy to reproduce. Many actions in the real world cannot be un-done or pre-simulated. Simulations and debugging tools are an open area that can be investigated in detail, addressed to the specifics of this domain. Aspects of error tolerance and handling errors in ubiquitous application creation is an open experience design issue that in time has to also be investigated in its own merit.

14.12.   Framework Walkthrough

The value of the proposed framework is in that it provides a unified view that can be accessed from the various teams and disciplines involved in the creation of ubiquitous systems. The goal of the framework walkthrough is to assess the

usefulness of the framework in providing a unified overview to the different disciplines involved. It aims to assess to what extent bridges are provided between such disciplines, for understanding the total aspects of the system and for having common concepts and vocabulary to cooperate and collaboratively design such systems, without having to get into the details of each other discipline.

The theories and concepts that are collected together under the framework umbrella stem from experience of working within multidisciplinary teams with different focus, in the course of research projects ranging from the Networked Home, to Ubiquitous Computing (UbiComp) and Ambient Intelligence (AmI) environments. Philips Design, with a focus on User Experience, and Computer Technology Institute, with a core focus in engineering, are two examples of such different institutional perspectives, both working towards the conceptualization and creation of ubiquitous systems, while the Products and Systems Design Engineering Department of the University of the Aegean typically covers the middle ground between the two perspectives.

A "walkthrough" is a heuristic usability inspection method that is used to validate and identify potential issues of usability, applied often in the context of software application development. It typically starts with a task analysis that specifies the sequences of steps or actions required in order to accomplish a task, and how the system corresponds to these actions.

In the case of the proposed framework, the task to be accomplished is building Ubiquitous Computing systems that support End User Development, by cooperating multidisciplinary teams. The walkthrough is a step by step approach, asking questions at each step, in order to report on potential input; in this case it is used as a means to provide a better understanding of how all the parts of the framework come together towards the better cooperation of multidisciplinary teams.

A Cognitive Walkthrough is a short usability method particularly useful to look at flow. A streamlined version of a walkthrough has been applied, in order to better explain how the proposed framework is to be used. Questions of the streamlined cognitive walkthrough method (Spencer, 2000) were adapted so that they relate not to applications but to the broader context of the framework – being more generic, rather than targeted to a specific application and action sequence.

The Streamlined Cognitive Walkthrough was developed by Rick Spencer (Spencer, 2000), proposing four questions to evaluate each step. Before the walkthrough one needs to clarify on the definitions of: (a) *the users*, (b) *sample tasks*, (c) *action sequence to complete the tasks*, and (d) *description of the systems usage and response*. These input questions to the walkthrough were adapted to fit the context of the framework, rather than that of software applications (that it was initially aimed at). In the case of the framework, providing these walkthrough definitions helps to describe more coherently the framework itself.

The definition of "Sample Tasks for evaluation" (that are applicable to applications) was thus altered to "Sample Thematic Areas" (indicating the range of areas and corresponding needs that are being targeted by such systems). Process of using the system (Action sequence), being a question typically relating to software applications, was replaced by Process of using (applying) the framework. Description of the implementation of the interface was no longer valid in this broader context, and was therefore replaced by checking the steps of the process, as these relate to certain parts of the framework. The questions, as they were adapted so that they relate to the broader context of the framework, are the following:

a) What are the thematic areas targeted by Ubiquitous Computing systems that support End User Development?
b) Who is the framework addressed to? (identification of users of the framework)

c) Process of using the framework (replacing the question on the action sequence -that related mostly to software applications)

d) Description of task sequences during the design & development process, as related to framework distinct parts.

The responses to questions of the streamlined walkthrough, as discussed by two subject experts, provide a better understanding of the framework itself:

a) What are the thematic areas targeted by Ubiquitous Computing systems that support End User Development?

The thematic areas that are being addressed based on the needs identified in various real-world research and development scenarios are:

1. Ambient Assisted Living (AAL), (i.e. see AAL website)
2. Home Automation and Context Awareness
3. Pervasive Awareness (i.e. see ASTRA project website), and
4. Ludic and playful leisure use (i.e. see (Divitini and Mavrommati, 2008), (Gaver, 2002))

b) Who is the framework addressed to?

The Framework is addressed to the different disciplines that collaboratively create Ubiquitous Computing systems and applications. The framework is to be used by System Developers, Engineers, User Experience (UX) Designers, User Interface (UI) and Interaction Designers, and Experts from Human Sciences (Psychology, Sociology). The Framework aims to bridge gaps in the communication between these disciplines, that collaboratively (but not always smoothly) are involved into the creation of ubiquitous computing systems. It provides a unified overview, rather than disconnected views and frameworks separate for each discipline.

c) Process of using the framework

A unified Agile process for system design and development (see previous section) is proposed, characterized by multidisciplinary face to face cooperation, and incremental, iterative development, with progress achieved through working system versions. Design and System Development progress are intertwined, while work proceeds in a stepwise approach, with each phase planned and recited anew, from the broader framework layout towards increasing implementation detail.

d) Description of task sequences during the design & development process, as related to framework distinct parts

There are two major tasks identified: User Experience Design (in its broad sense) and System Development. The respective framework elements for enabling End User Development get cross fertilized with elements of UbiComp System Design and Development (as previously described in the Framework), as the system development process proceeds (through the Agile development method steps). In the first phases of conceptualization via the Agile process, the process takes input from the theoretical foundations, that abridge understanding of the long term implications of the system, the end user experience design aspects and the system design and development aspects of it. As the phases get into more detail, elements of the relevant methodologies (i.e. Design Rationale, System Design and Development Methodologies –such as the Gadgetware Architectural Style Framework-, etc) are used and worked into further detail into system deployment as a functional prototype.

The Streamlined Cognitive Walkthrough method then proposes discussing two questions: (1) "*is there a plausible story that the system designers and developers (i.e. the framework users) will know what to do at each step?*" and (2) "*if the designer/developer does the right thing, is there a plausible story that they will know that they did the right thing?*". In the walkthrough that took place with two experts

broadly discussing the use the frameowork; a gap was noted regarding the process of using the framework; concequently the Agile process was added, as a necessary process describing the steps of how to use to the framework. Via the Agile process designers/developers know when and how to apply parts of the framework, at each Agile process step. They know if they made the right thing by marking stepwise progress in the form working prototypes so that they know if they were successful.

14.13.   Conclusions

In this chapter we described concepts, methodologies and their relationships towards a framework for Ubiquitous Systems that can support End User Development. We suggest for research to consider End User Development as an affordance of ubiquitous computing systems, in the sense that Ubicomp systems can provide the principle elements that enable people to perceive and handle ubiquitous applications. Affordances provide the implied use, -(affordances being the *relationship that develops between objects and the tasks that can be performed with them* (Gibson 1977))- which may or may not be realized, nevertheless the elements that can afford and effect it should be present in the ubiquitous system.

The perspective that should be adopted in ubiquitous systems is that of co-evolution and co-development between artifacts, applications, together with related EUD tools and people. End User Tools, structures and mechanisms provided are mediators for this development. EUD affordances in Ubiquitous Systems can be effected by relevant systems design that has included EUD elements and mental models for the user –which should be available to people even when the system is relying on agents and automations for realizing its ubiquitous functionality.

It is important to consider the system with a number of co-dependencies that it involves, from the different perspectives involved. An important split is to distinguish between *end user design* (the more creative 'designerly' aspects of

humans, but also the ones that are based on more intuition and direct physical involvement, by subsequent observation and manipulation), and *end user programming*, (that pertains the more systematic, analytical and syntactic mental tasks, which are facilitated by different abstractions and syntaxes, as seen fit per case of different users). The full functionality that is possible from the part of End User Programming may not be used by all users, or for complex applications cases, due to the cognitive load and the programming complexity involved. Nevertheless the possibility should be there, present and accessible to end users, whose culture and knowledge will co-evolve with ubiquitous systems.

Expansion on this framework and the concepts that it provides, and investigation into the issues that still remain open can provide valuable insight and guidance to the total design of Ubiquitous Computing Systems of the future, used by people and augmented human environments.

The perspective to be adopted in ubiquitous systems is that of co-evolution and co-development between people, EUD tools, and Systems –primarily drawing from Activity theory and Ecological Psychology. The ubiquitous system, its' End user tools, and people's perspectives on them, are developing together and are cross-influenced in an iterative cycle, in the context of socio-cultural developments. The Conceptual and methodological Framework developed in this direction should therefore be broader and multidisciplinary in its scope. This Framework addresses the relationships between three broader perspectives, those of a) understanding human cognition, b) defining a wider spectrum of interaction-concepts and methods for people to access the EUD ubiquitous system and c) understanding the system design of the ubiquitous system and how it can be constructed. It can be used by ubiquitous computing researchers as an umbrella framework, that can serve as the basis of integration of more detailed sub-frameworks (theoretical and methodological) from each of the three perspectives, while items that remain open can be addressed and outcomes of future research can be added.

15

15. Conclusions and Outcomes

15.1. Overview of Conclusions

Research presented in this thesis has made some headway in the effort to empower people to actively shape Ambient Computing environments. It has demonstrated the feasibility of letting end-users shape their ubicomp environments. Experience from system implementation case studies, as well as evaluation of expert and end-user trials (see chapter 11 and appendices 1 to 4), all suggest that an architectural model, where users act as composers of predefined components, is a worthwhile approach that can be further explored, and can act complementarily to artificial system intelligence.

Evaluation results show that people understand the split in the dual nature of artifacts: their tangible and sensory characteristics and their connectable software counterparts. Conceptual models and alternative information visualizations are needed to support people in creating their own applications. Such visualization methods should combine different syntax styles that act complementarily to each other, thus allowing people (including non-computer experts) to use different ways to describe to the system what they want to achieve. More intuitive/natural ways to

express user wishes should be provided in parallel with more formal structures that enable more detailed descriptions and advanced control.

For the creation of successful ubiquitous computing systems, that is both scalable and robust as well as user friendly, the cooperation of different disciplines is required, including User Experience Design and Cognitive Psychology. The overall User Experience is a key issue to the success and adoption of ubiquitous systems because it is the basis for their acceptance and trust, since these systems are involved in more aspects of the users life, are used over extended time, and have consequences on privacy and control, since the users are handing over part of their autonomy to the system.

From experiences gathered by working within teams in this research area, for EU basic research projects, we note a communication gap between the different disciplines involved (be them engineers, designers, psychologists, human sciences experts, domain experts, stakeholders) – an observation not new to HCI, since it resonates to the development of most user-centered systems. In the development of such systems the perspective of Computer Engineering is dominant, as it is system engineers that have to make decisions and build the system, without being able to integrate successfully input from other disciplines - such dialogue or input often being inappropriately communicated or having different priorities. Many important elements and crucial nuances to the user experience are thus 'lost in translation'. Building systems this way often results in over-engineering, adding unnecessary complexity, and increasing the mental load of users. Typically, End User Development is being treated - perhaps due to the computer science inertia - as a programming activity done by end users; in order to balance this approach, the perspective of End User Design is introduced as its "softer" counterpart. End User Programming and End User Design are envisioned as two different but complementary perspectives and approaches of End User Development, with the former being functionality oriented and the latter focusing on user experience

For the disciplines involved to be able to work together, better processes and methodologies are needed, targeting specific research and development areas (such as for example pervasive awareness, ambient assisted living, leisure and ludic use, home automation and monitoring, resources optimization, applications for enhancing the life of people with special abilities, etc). The framework proposed in chapter 14 can be seen as an abridging tool, providing a conceptual basis and theoretical foundation, as common ground for the communication and cooperation between the disciplines involved in user-focused ubiquitous computing research.

15.2. Achievements of the reported research

Research reported in this thesis has achieved the following

- End User Development as being a combination of End User Design and End User Programming in the context of Ubiquitous Computing environments has been introduced (chapters 7, 13, 14). This research questioned the assumption of embedded artificial intelligence and automation, and promoted end-user empowerment and understanding as a means to ensure adoption, adaptation and emergent functionality (stemming from the creativity of end users themselves) in ubiquitous computing systems (see chapter 4). From the outset of research in 2002, the engineering approach of creating ubiquitous computing artifacts by attaching screens or feedback LED lights to them was questioned. A more flexible and inclusive model of augmented artifacts was proposed instead, based on affordances, that is, more suited to the tangible physical nature of the particular artifact [(Mavrommati and Kameas, 2002), (Mavrommati and Kameas, 2003b), (Mavrommati et al, 2004) (Kameas and Mavrommati, 2005), (Kameas, Mavrommati et al, 2005)]. This research position was among the very first to promote End User Development for Ubiquitous Computing applications [(Mavrommati and Kameas, 2002), (Edwards, Newman et al, 2002), (Newman

et al, 2002), (Humble et al, 2003), (Mavrommati and Kameas, 2003b) (Rodden et al, 2004), (Kameas, Mavrommati et al, 2003)].

- The Capabilities and Links model has been proposed (chapter 9) that bridges the gap between people's perception of artifacts and the actual manufacturing and software engineering constructs of those artifacts. The Capabilities and Links model expanded on publish-subscribe concepts and semantics, introducing the use of affordances to ubiquitous computing systems (at a higher conceptual level than the usual treatment of services, sensors and actuators).

- The Capabilities and Links model was used in the context of EU funded IST Future Emerging Technologies (FET) research projects, validating it in terms of technological appropriateness and scalability (chapter 10) (Kameas and Mavrommati, 2005), (Drossos, Mavrommati, Kameas, 2007).

- The proposed Capabilities and Links model was evaluated in expert and user trials (see Appendices 1 to 4), using case study implementations at FET projects, as a means for assessing the comprehensibility and applicability of End User Development in Ubiquitous Computing (Chapter 11). End User Development was validated as a worthwhile approach for Ambient Computing (Mavrommati, et al 2003c), (Mavrommati et al, 2004), (Markopoulos et al 2004). It is an approach that complements Ambient Intelligence –the two approaches not being mutually exclusive, but acting complementarily.

- The Cognitive Dimensions framework (see Appendix 4 and Chapter 11) has been trialed and proved [(Mavrommati et al, 2003d), (Mavrommati et al, 2004), (Markopoulos et al, 2004)] to be an appropriate tool for assessing Ambient Computing concepts in the early stages of their development.

- An overview control interface, running on a separate device called an "Editor", was adopted (see chapter 12) as the most appropriate approach for assisting End

User Development. This permits better overview of applications lists and nodes, and higher level programming by allowing more complex syntax. A separate interface, rather than either embedded interfaces in the artifacts themselves (some of which may be too small or too remote to access) and programming-by-example techniques, offers the advantages of a broader approach, one that can cater for the different granularity of artifacts. Moreover, it can be combined with other tools, such as community-sharing tools or awareness application mechanisms. Lastly, it allows the easier distribution of the Editor's functionality and upgrades. Functionality of the Editor was outlined and possible interaction modules were proposed in Chapter 12.

- Graphical User Interfaces were explored, using various programming syntaxes. These were seen as scenarios in visual form that help further to explore End User Development (EUD) parameters in the context of the editor. Internet-based editor GUIs were proposed as most likely to be the most widespread and commonly used, promoting End User Design as well as End User Programming functionality. Concretely exemplified versions of possible Graphical User Interfaces were sketched out (see chapter 13), showing different semantics and modalities for interaction with the Editor.

- Existing User Interface paradigms and semantic representations for End User Development in the context of Ubiquitous Computing were described (from various well known project examples) (see chapters 5, 8, 13).

- A broad Framework for the design of Ubiquitous Computing Systems supporting End User Development has been outlined. It maps theoretical, methodological, and engineering concepts, from the disciplines of Cognitive Science, Interaction Design, and Computer Engineering. This broad perspective will be beneficial to the cross-disciplinary research community working on ubiquitous systems, as it provides a deeper and broader understanding of theory and issues concerning End User Development (chapter 14).

- End User Design has been introduced, which, together with End User Programming, enables end-users to imagine, rationalize, create and manipulate ubiquitous applications. The former corresponds to being able to conceptualize and describe application ideas or to apply basic control and customization over existing applications and the related User Experience (specify parameters, such as "look and feel"), while the latter corresponds to the realization of advanced customization of the specifics of the applications, in their detailed structure and elements, using more advanced programming constructs. These two different but complementary parts make up the different aspects of End User Development in Ubiquitous Computing environments (see chapter 14).

It should be noted that the research reported here has had an influence in design and development of Ambient Systems. Overview (Mavrommati and Darzentas, 2007), and perspectives on End User Tools for Ambient Computing Environments has been recognized internationally (see citations in appendix 5, and invited speech in Doors of Perception 7 (Mavrommati 2002) – an internationally acclaimed curated design conference).

Work in the area of Ambient Systems is by definition multidisciplinary and has to happen in teams with mixed backgrounds, due to the complexity of research issues involved. The (broad) disciplines involved in interactive and ubiquitous systems are Human Sciences, Design, and Computer Engineering. Underlying assumptions and goals differ greatly between teams from these synergizing fields (Mackay, 2003). For this reason, Interaction Design techniques, approaches and perspectives were introduced into a Computer Engineering team's culture, most notably the extended use of the scenario-based development method, 'Design Rationale', as well as the 'Iterative Design process' that includes system design evaluation feedback cycles. Specifically regarding this research, introducing a consideration of the notion of "Affordances" and encouraging an abstract and simplified formulation of the "Capabilities and Links" model has influenced the development of middleware for Ambient Systems in a strongly multidisciplinary manner [(Kameas et al, 2003),

(Kameas et al, 2005) and also see related citations in appendix 5], and supported putting people at the center of System Architecture developments instead of down at the receiving end. This collaboration and cross-fertilization between the perspectives of People-Centered design and Computer Systems design was an achievement in the first part of this research route. This work has subsequently informed and influenced the development of middleware for AmI (within the context of EU funded e-Gadgets project as well as internationally). This was recognized by a number of cross-references to the outcome research publications that were produced from this synergy (Appendix 5).

15.3. Directions for future research

It has to be noted that the User Experience Design (UXD) perspective to this day fails to be integrated successfully within technology teams. Experience Design is seldom even introduced into technology-development teams, and is even more rarely integrated, because of cultural and methodological differences. As a consequence, the Computer Science perspective continues to lead System Design and often defines, from that perspective, the Human Computer Interaction (HCI) experience. A prevailing focus on functionality and performance is thereby maintained, since Systems Design is often failing to pose questions regarding the vision and the original assumptions of the system, and it often does not address the system's all-round requirements (including additional / non-functional requirements) and eventually the target user needs and other stakeholders' investment in the project. Even when minimally integrated, Experience Design questions existing assumptions regarding a system's requirements, provides a view from different perspectives, and gives a broader, richer view of the system and its requirements (often from the perspectives of different disciplines that come into play). Experience Design enhances the output of System Design process when combined with processes that bring users and system stakeholders into continuous dialogue with the system design development team (such as the Agile Process).

Interaction/Experience Design Research for End User Development in the area of Ubiquitous Computing systems can further explore, evaluate and analyze a set of paradigm syntaxes and different interface instantiations that can be used for End User Programming.

Human Computer Interaction research needs to further explore error prevention and error handling (including fault tolerance) in the context of End User Development in Ubiquitous Computing applications. Cases of application creation by use of Editor Interfaces need to be studied in order to explore what is missing and what can go wrong in the process of programming in the context of an augmented environment. Special effort will be required to categorize types of errors, their causes and to what these causes may relate (that is, an error may relate to what the interface affords, what the system allows, or the causal effect on the augmented environment). More systematic effort towards error tolerance, error handling, and preventing errors is needed in ubiquitous computing systems research.

In the context of End User Development in Ubiquitous Computing, the proposed broad Ubicomp Framework provides bridges between a number of theories and perspectives, from Cognitive Science to Design to Systems Engineering. This Framework opens up a wide area for future research; it provides a broad multidisciplinary view to researchers for understanding End User Development in Ubiquitous Computing, and could be further explored (ie with a systemic approach), in order to produce a holistic framework for Ubiquitous Systems that allow for End User Development.

16. References

1. Aarts, E., Harwig, H., M. Schuurmans (2001). Ambient Intelligence, in: J. Denning (ed.) The Invisible Future, McGraw Hill, New York, pp. 235-250.

2. Aarts, E., Marzano S. (eds.) (2003). The New Everyday: Visions of Ambient Intelligence, 010 Publishing, Rotterdam, The Netherlands.

3. Aboulafia, A., Gould, E., Spyrou, T. (1995) Activity theory vs cognitive science in the study of human-computer interaction. Proceedings of the IRIS (Information Systems Research Seminar in Scandinavia) conference, Gjern, Denmark.

4. Agile Alliance (2001). Manifesto for Agile Software Development. Technical Report by Agile Alliance, http://www.agilealliance.org

5. Alexander, C. (1964). Notes on the Synthesis of Form. Harvard University Press, Cambridge.

6. Alexander, C., Ishikawa, S., Silverstein, M. (1977). A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York.

7. Andrews, A., Bueno, M., Cass, J.(2002). Open Tools Definitions, Internal Paper, Philips Design, www.design.philips.com

8. Baldus, H., Baumeister, M., Eggenhuissen, H., Montvay, A., Stut, W. (2000). WWICE: an architecture for in-home digital networks. In Proc. SPIE Vol. 3969, p. 196-203, Multimedia Computing and Networking 2000, Klara Nahrstedt, Wu-chi Feng, Eds.

9.   Bannon, L. (1997). Activity theory. [Tutorial available Online]. Available at: http://www.irit.fr/ACTIVITES/GRIC/cotcos/pjs/TheoreticalApproaches/Act vty/ActivitypaperBannon.htm

10.   Bannon, L., Bødker, S. (1991). Beyond the Interface: Encountering Artifacts in Use Book Chapter in J.M. Carroll (Ed.) Designing Interaction: Psychology at the Human-Computer Interface, New York, Cambridge University Press, pp. 227-253.

11.   Barkhuus, L., Vallgårda, A. (2003). Smart Home in Your Pocket. In: Adjunct Proceedings of UbiComp 2003, pp. 165-166

12.   Becker C., Handte M., Schiele G., Rothermel K. (2004). "PCOM-A Component System for Pervasive Computing," in Proceedings of the 2nd International Conference on Pervasive Computing and Communications, Orlando, Florida.

13.   Bell, G., Dourish P. (2007). "Yesterday's Tomorrows: notes on ubiquitous computing's dominant vision. Pervasive Ubiquitous Computing (11), pp. 133-143.

14.   Bellotti, V., Back, M., Edwards, W.K., Grinter, R.E., Henderson, A., Lopes, C. (2002). Making Sense of Sensing Systems: Five Questions for Designers and Researchers. In: Proceedings of the Conference on Human Factors and Computing Systems (2002) pp. 415-422.

15.   Berger, P.L., Luckmann, T. (1966). The social construction of reality: A treatise in the sociology of knowledge - New York.

16.   Birbilis, G., Koutlis, M., Kyrimis, K., Tsironis, G., Vasiliou, G. (2000). "E-Slate: A software architectural style for end-user programming", presented at

the 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland.

17. Blackwell, A., Green, T. (2003). Notational Systems-the cognitive dimensions of notations framework. In HCI models, theories and frameworks, ed: J. Carroll, pp. 103-132.

18. Blomkvist, S. (2005). Towards a Model for Bridging Agile Development and User-Centered Design. Published as a book chapter: Seffah, A., Gulliksen, J., and Desmarais, M., (eds.). Human-Centered Software Engineering – Integrating Usability in The Development Process. Springer, Dordrecht, The Netherlands, 217-243.

19. Bodker, S. (1991). Through the Interface: a Human Activity Approach to User Interface Design. L. Erlbaum Associates Inc.

20. Bødker, S. (2000). Scenarios in user-centred design—setting the stage for reflection and action in: Interacting with Computers. Volume 13, Issue 1, pp. 61-75.

21. Brown, J. S., Collins, A., Duguid, P. (1989). Situated Cognition and the Culture of Learning. Educational Researcher, January 1989, vol. 18, no. 1, pp. 32-42.

22. Calemis, J., Mavrommati, I. (2007). Preliminary requirements and approach for Tools that configure pervasive awareness applications: the ASTRA case. Poster in: HCI International 2007, Beijing, China.

23. Carroll, J. (2000). Making Use: Scenario based design of Human Computer Interactions MIT press.

24. Carroll, J., Rosson, M.B. (2003). Design Rationale as Theory" in HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science (Interactive Technologies), Eds: John M. Carroll, published by Morgan Kaufmann, pp. 432.

25. Carroll, J.M. (2000). Five reasons for scenario-based design, in: Interacting with Computers, Elsevier, Volume 13, Issue 1, pp. 43-60.

26. Carroll, J.M. (2002). Making use is more than a matter of task analysis, in: Interacting with computers, vol. 14,– Elsevier, pp. 619-627.

27. Carroll, J.M. (2006). Dimensions of Participation in Simon's Design. In: Design Issues, Spring, Vol. 22, No. 2, MIT Press, Pages 3-18.

28. Carroll, J.M. ed. (2003). HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science (Interactive Technologies), Morgan Kaufmann.

29. Carroll, J.M., ed. (1991). Designing interaction: Psychology at the human-computer interface, Cambridge University Press, New York.

30. Chong J., See, S., Leng-Hiang Seah, L., Ling Koh, S., Theng, Y., Duh, H. (2008). in: Chapter1: Ubiquitous Computing History, developments and scenarios in: book TBA ICI Global.

31. Christopoulou, E., Garofalakis, J. (2010). Enabling the User to Setup Ubiquitous Computing Applications Based on Intelligent Context, cisis, pp.87-92, 2010 International Conference on Complex, Intelligent and Software Intensive Systems.

32. Christopoulou, E., Goumopoulos, Ch., Kameas, A. (2005). An ontology-based context management and reasoning process for ubicomp applications. In sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart

Objects and Ambient Intelligence, pp. 265-270, New York, NY, USA,. ACM.

33.   Christopoulou, E., Kameas A. (2003). GAS-Ontology: Conceptualizing Gadgetware Architectural Style TALES of the Disappearing Computer, CTI Press / Ellinika Grammata.

34.   Christopoulou, E., Kameas, A. (2005). Gas ontology: an ontology forcollaboration among ubiquitous computing devices. Int. J. Hum.-Comput. Stud., 62(5):664-685.

35.   Cohen David, Lindvall Mikael, Costa Patricia (2004) An introduction to agile methods Advances in Computers - AC , vol. 62, pp. 2-67.

36.   Cohen, N.H., Lei, H., Castro, P., Davis, J.S., Purakayastha, A. (2002). Composing Pervasive Data Using iQL. In Proc. Fourth Workshop on Mobile Computing Systems and Applications - WMCSA 2002. pp. 94-104.

37.   Costabile, M.F., Ed. (2002). D4.1 Evaluation report from EUDnet, REF IST 2001037470, available from EUD-Net Network of Excellence website. http://giove.isti.cnr.it/projects/EUD-NET/deliverables.htm (last accessed June 2010)

38.   Cotterell, S., Vahid, F. (2005). A Logic Block Enabling Logic Configuration by Non-Experts in Sensor Networks. In Extended Abstracts of the Conference on Human Factors in Computing Systems, CHI 2005. pp.1925-1928

39.   Crabtree, A., Rodden, T., Hemmings, T., Benford, S. (2003). Finding a place UbiComp in the home. In: Proceedings of the 5th International Conference Ubiquitous Computing, Seattle. Berlin, Heidelberg: Springer, pp. 208-226.

40. Cross, N. (2008). Engineering Design Methods: Strategies for Product Design, Willey publishers.

41. Crutzen, C.K.M. (2006). 'Ambient intelligence between heaven and hell; A transformative critical room?', in: Information society technology from a gender perspective - Epistemology, construction and empowerment, VSVerlag, series 'Interdisciplinary gender research', edited by the Centre for feminist Studies of the University of Bremen and the Centre of Womens and Gender Studies of the University of Oldenburg.

42. Daskala, B., Maghiros, I. (2006). Digital Territories. Proceedings of the 2nd IET International Conference on Intelligent Environments (IE06), Athens, Greece. Published by IET pp. 221 – 226, volume 2.

43. Daskala, B., Maghiros, I. (2007). Digital Territories: Towards the protection of public and private spaces in a digital and Ambient Intelligence environment. EUR 22765 EN http://www.jrc.es/publications/pub.cfm?id=1474 (last accessed March 2010)

44. Détienne, F. (1990). Difficulties in designing with an object-oriented language: An empirical study. In Proceedings of the IFIP Tc13 Third interational Conference on Human-Computer interaction (August 27 - 31, 1990). D. Diaper, D. J. Gilmore, G. Cockton,, B. Shackel, Eds. North-Holland Publishing Co., Amsterdam, The Netherlands, pp. 971-976.

45. Détienne, F. (1990). Expert Programming Knowledge: a Schema-Based Approach Published in J-M Hoc, T.R.G. Green, R. samurcay, & D. Gilmore (Eds) : Psychology of Programming. Academic Press, People and Computer Series, pp. 205-222.

46. Détienne, F. (1990). Program understanding and knowledge organization: the influence of acquired schemata. In Cognitive Ergonomics: Understanding,

Learning and Designing Human-Computer interaction, P. Falzon, Ed. Academic Press Series In Computers And People. Academic Press Professional, San Diego, CA, 245-256.

47. Detweiler, M. (2007). Managing UCD within Agile Projects. ACM Interactions May-June 2007, 40 – 42.

48. Dey, A.K. (2005). End-User Programming: Empowering Individuals to Take Control of their Environments, in Proceedings of the CHI 2005 workshop on the Future of User Interface Design Tools.

49. Dey, A.K., Hamid, R., Beckmann, C., Li, I., Hsu, D. (2004). a CAPpella: Programming by Demonstration of Context-Aware Applications. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2004). ACM Press, New York pp. 33-40.

50. Dey, A.K., Salber, D., Abowd, G.D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In Special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), pp. 97-166.

51. Divitini, M. & Mavrommati, I. (2008) Pervasive Awareness Applications: Aesthetics and Ludic Aspects, 3rd ACM International Conference on Digital Interactive Media in Entertainment and Arts DIMEA 2008 ACM Press, Athens, Greece, 2008, pp. 499-500.

52. Dix, A., Finlay, J., Abowd, G., Beale, R. (2004) Human Computer Interaction, 3rd edition, Pearson / Prentice Hall publications.

53. Drossos, N., Goumopoulos, C., Kameas, A. (2007a). "A Conceptual Model and the Supporting Middleware for Composing Ubiquitous Computing Applications". Journal of Ubiquitous Computing and Intelligence (JUCI),

special issue on Ubiquitous Intelligence in Real Worlds, vol 1, pp. 1-13, American Scientific Publishers.

54.   Drossos, N., Mavrommati, I., Kameas, A. (2007b). Towards ubiquitous computing applications composed from functionally autonomous hybrid artifacts. In the Disappearing Computer book (eds: Streitz N. Kameas A. Mavrommati I.), Springer Verlag, LNCS.

55.   Dunn, T., Raby, F. (2002) Design Noir: The secret life of electronic objects, ed. August/Birkha.

56.   Edwards, W..K, Newman, M.W., Sedivy J., Izadi S. (2002). Challenge: recombinant computing and the speakeasy approach, Proceedings of the 8th annual international conference on Mobile computing and networking, September 23-28, Atlanta, Georgia, USA.

57.   Engestrom, Y. (1987). Learning by expanding: An activity-theoretical approach to developmental research. Helsinki, Orienta-Konsultit. Available at http://communication.ucsd.edu/MCA/Paper/Engestrom/expanding/toc.htm

58.   España, S., Pederiva I., Panach J.I. (2008). Integrating Model-Based And Task-Based Approaches To User Interface Generation in Book: Computer-Aided Design Of User Interfaces V, pp253-260, eds: Gaëlle Calvary, Costin Pribeanu, Giuseppe Santucci, Jean Vanderdonckt, Springer, Netherlands.

59.   Fauconnier G., Turner M. (2002). the Way we Think: Conceptual blending and the Mind's hidden complexities. New York: Basic books.

60.   Feenberg, A. (1992). 'Subversive rationalization: Technology, power, and democracy', Inquiry, 35: 3, pp. 301-322.

61. Fernando, O.N.M, Adachi, K., Duminduwardena, U., Kawaguchi, M., Cohen, M. (2006). Audio narrowcasting for multipresent avatars on workstations and mobile phones, in IEICE Trans. Inf. & Syst.Vol. E 89–D, N0.1 January 2006, 73-87.

62. Fischer, G., Giaccardi, E. (2006). Meta-design: A Framework for the Future of End-User Development. In: End User Development, edited by Henry Lieberman, Fabio Paternò, Volker Wulf. Springer Netherlands, pp 427-457.

63. Fischer, Giaccardi, Ye, Sitcliffe, Mehandjiev (2004). Meta-Design a manifesto for end user development, Communications for the ACM, 2004, vol.47, no.9.

64. Fokidou, T. Romoudi E., Mavrommati, I. (2008). Designing GUI for the User Configuration of Pervasive Awareness Applications, T.. IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2008), Freiburg.

65. Gardien, P. (2007). Breathing Life into delicate ideas. white paper at http://www.design.philips.com/ (last accessed Jan 2010).

66. Gaver, W. (1991). Technology affordances. In Proceedings of CHI 1991, ACM Press: New York, 79 – 84.

67. Gaver, W. (2002). Designing for Homo Ludens. i3 magazine issue 12, June 2002

68. Gedenryd, H. (1998). How designers work, phD thesis, online at http://www.lucs.lu.se/henrik.gedenryd/HowDesignersWork/ (last accessed March 2010).

69.   Gibson, J. (1986). The Ecological Approach to Visual Perception. Lawrence Erlbaum Associates, Inc. (originally published in New York: Houghton Mifflin, 1979).

70.   Gibson, J.J. (1977). The Theory of Affordances (pp. 67-82). In R. Shaw & J. Bransford (Eds.). Perceiving, Acting, and Knowing: Toward an Ecological Psychology. Eds. Robert Shaw, John Bransford, ISBN 0-470-99014-7. Hillsdale, NJ: Lawrence Erlbaum .

71.   Go, K., Carroll, J. (2004). The blind men and the elephant: views of scenario-based system design. ACM Interactions, Vol.11, Issue 6, pp. 45-55.

72.   Goumopoulos, C., Kameas, A. (2009). Ambient ecologies in smart homes, in: The Computer Journal, Oxford University Press, vol. 52, no. 8, pp. 922-937; doi: 10.1093/comjnl/bxn042.

73.   Goumopoulos, C., Kameas, A., Berg, E., Calemis, I. (2009). A Service-Oriented Platform for Pervasive Awareness Systems, Proc. of the International Conference on Advanced Information Networking and Applications Workshops, IEEE CS CPS, ISBN 978-0-7695-3639-2, DOI 10.1109/WAINA.2009.197, pp.470-475, 26-29 May, Bradford UK.

74.   Graves, Petersen Marianne (2004). 'Remarkable Computing - the Challenge of Designing for the Home', CHI 2004, Vienna, Austria, pp. 1445-1448

75.   Green, J. (2007). Democratizing the future, towards a new era of creativity and growth, 2007 Philips report, pp. 46-48, accessed Jan.2010 from www.design.philips.com/.../**democratizing-the-future**-14324.pdf

76.   Green, T. R.G., Petre, M. (1992). When Visual Programs are Harder to Read than Textual Programs. In G. C. van der Veer, M. J. Tauber, S. Bagnarola, & M. Antavolits (Eds.), Human- Computer Interaction: Tasks and Organisation,

Proceedings of ECCE-6 (6th European Conference on Cognitive Ergonomics)

77.   Green,T.R.G., Petre, M. (1996). Usability analysis of visual programming environments: a cognitive dimensions framework. Journal of Visual Languages and Computing. J. Visual Languages and Computing, 7, 131-174

78.   Greenberg, S., Fitchett, C. (2001). Phidgets: Easy Development of Physical Interfaces Through Physical Widgets. In Proc. ACM Symposium on User Interface Software and Technology, UIST 2001, pp. 209-218.

79.   Greene, S. L., Devlin, S. J., Cannata, P. E., Gomez, L. M. (1990). No IFs, ANDs, or ORs: A Study of Database Querying. International Journal of Man-Machine Studies, 32(3), 303-326.

80.   Greeno, J. G., Moore, J. L. (1993). Situativity and symbols: Response to Vera and Simon. Cognitive Science, 17, 49-60.

81.   Greeno, J.G. (1989). A Perspective on Thinking, American Psychologist.

82.   Greeno, J.G. (1994). Gibson's affordances Psychological Review, vol101, no2, pp336-342, Elsevier.

83.   Gross, T. Marquardt, N. (2007). CollaborationBus: An Editor for the Easy Configuration of Ubiquitous Computing Environments. In Parallel, Distributed and Network-Based Processing, 2007. PDP '07. 15th EUROMICRO International Conference on Digital Object Identifier: 10.1109/PDP.2007.29, pp. 307–314.

84.   Gu, M., Aamodt A. (2005). A Knowledge-Intensive Method for Conversational CBR in Case-Based Reasoning Research and Development,

Springer LNCS, ,Proceedings of the 6th International Conference on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, pp. 296-311..

85.   Habermas, Jurgen (1984). The theory of communicative action I and II (trans. T. McCarthy).

86.   Hague, R., Robinson, P., Blackwell, A. (2003). Towards Ubiquitous End-User Programming. In: Adjunct Proceedings of UbiComp 2003, pp. 169-170.

87.   Harwig, Rick, Emile Aarts (2002). Ambient Intelligence: Invisible Electronics Emerging, Proceedings of the 2002 International Interconnect Technology Conference, San Francisco.

88.   Hollan, J., Hutchins, E., Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. ACM Trans. Comput.-Hum. Interact. 7, 2, 174-196. DOI= http://doi.acm.org/10.1145/353485.353487

89.   Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H.W. (2001). Smart-Its friends: a technique for users to easily establish connections between smart artifacts. Proceedings of UBICOMP 2001, pp. 116-122.

90.   Hudson, S. (2005). "Leveraging 1,000 and 10,000-Fold Increases: Considering the Implications of Moore's law on Future UI Tools Research". Proceedings of the CHI 2005 workshop on the Future of User Interface Design Tools.

91.   Hudson, W. (2003). Adopting User-Centered Design within an Agile Process: A Conversation. Cutter IT Journal, (16), 10 available from http://www.suntagm.co.uk/design/articles/ucdxp03.pdf

92.   Humble, J., Crabtree, A., Hemmings, T., Åkesson, K., Koleva, B., Rodden, T., Hansson, P. (2003). Playing with the Bits, User-Configuration of Ubiquitous Domestic Environments. In: Proceedings of UBICOMP 2003. Springer-Verlag, Berlin Heidelberg New York, pp. 256-263.

93.   Hutchins, E. (1995). Cognition in the Wild, MIT Press. ISBN 0-262-58146-9.

94.   Hutchins, E., Hollan, J., Norman, D. (1986). Direct Manipulation interfaces. In User Centered System Design, Norman D, Draper S. (ed). Lawrence Erlbaum Associates, Hilsdale, NJ pp. 87-124.

95.   Hwong, B., Laurance, D., Rudorfer, A., and Schweizer, A. (2004). User-Centered Design and Agile Software Development Processes. Siemens Corporate Research available from http://www.scr.siemens.com/en/pdf/se_pdf/rudorfer-1.pdf.

96.   Imaz M., Benyon D. (2007). Designing with blends, conceptual foundation of human computer interaction and software engineering, MIT press, preface.

97.   IST website: Projects launched by the Disappearing Computer Proactive Initiative. http://cordis.europa.eu/ist/fet/dc-sy.htm#dc2fp6 (last accessed Jan 2010).

98.   ISTAG Scenarios for Ambient Intelligence in 2010. Final report. (2001). K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten & J-C. Burgelman, (Eds). IPTS-Seville.   ftp://ftp.cordis.europa.eu/pub/ist/docs/**istag**scenarios2010.pdf (last accessed January 2010).

99.   Jacquet, C., Bourda, Y., Bellik, Y. (2005). An Architecture for Ambient Computing, The IEE International Workshop on Intelligent Environments.

100.   Jones, J.C. (1991). Designing Designing (London: Architecture Design and Technology Press).

101. Kameas, A. Mavrommati, I. Markopoulos, P. (2005). Computing in tangible: using artifacts as components of Ambient Intelligent Environments" In: Riva, G./Vatalaro, F./Davide, F./Alcañiz, M. (eds) Ambient Intelligence. IOS Press , chapter 7, p.121-141

102. Kameas, A., Bellis, S., Mavrommati I., Delanay, K., Colley, M., Pounds-Cornish, A. (2003). An Architecture that Treats Everyday Objects as Communicating Tangible Components. in IEEE international conference on Pervasive Computing and Communications, (PERCOM2003), Texas. Proc. PerCom03, IEEE, Forth Worth.

103. Kameas, A., Mavrommati, I. (2001). Interacting with ubiquitous computing applications: issues and methodology. Proceedings Panhellenic Conference on Computer Human Interaction (PCCHI) 2001, Greece.

104. Kameas, A., Mavrommati, I. (2005). Configuring the e-gadgets, Communications of the ACM (CACM), special issue section on The Disappearing Computer, vol. 48, issue 3: ACM, pp. 69.

105. Kaneko, N., Onisawa, T. (2005) An Experimental Study on Computer Programming with Linguistic Expressions. In Knowledge-Based Intelligent Information and Engineering Systems (Proceedings of KES - Lecture Notes in Computer Science), vol. 3684, Vol. 1: Springer, pp. 911-917.

106. Kaptelinin, V. (1996). Activity Theory: Implications for human-computer interaction. In B. Nardi, (ed), Context and Consciousness: Activity theory and human-computer interaction. Cambridge, MA: MIT Press. Excerpt online at: http://www.quasar.ualberta.ca/edpy597mappin/readings/m15_kaptelin.htm

107. Kaptelinin, V. (1996). Computer-Mediated Activity: Functional Organs in Social and Development Contexts. Context and Consciousness: Activity

Theory and Human-Computer Interaction. ed B. A. Nardi. Cambridge, MA, MIT Press, pp. 45-68.

108. Kaptelinin, V., Nardi, B. (1997). Activity theory: basic concepts and applications in CHI '97 extended abstracts on Human factors in computing systems: looking to the future, Tutorials, pp. 158–159.

109. Kaptelinin, V., Nardi, B. (2006). Acting with Technology: Activity Theory and Interaction Design. Cambridge, MIT Press.

110. Klann, M., Paterno, F., Wulf, W. (2006): Future perspectives in EUD, in Lieberman, H., Paterno, F., Klann, M., Wulf, W. (eds): End User Development, Springer Netherlands, pp. 475-486.

111. Klemmer, S.R., Li, J., Lin, J., Landay, J.A. (2004). Papier-Mâché: Toolkit Support for Tangible Input. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2004). ACM Press, New York, pp. 399-406.

112. Kreitzberg, C.B., Little (2009), A. Usability in Practice: Agile Ux Development in MSDN magazine, issue of June 2009. Available from http://msdn.microsoft.com/en-us/magazine/dd882523.aspx

113. Kuutti, K. (1996). Activity Theory as a Potential Framework for Human-Computer Interaction Research. Context and Consciousness: Activity Theory and Human-Computer Interaction. B. A. Nardi. Cambridge, MA, The MIT Press, pp. 17-44.

114. Kyffin, S., Gardien, P. (2009). Navigating the innovation matrix: An approach to design-led innovation. International Journal of Design, 3(1), 57-69.

115. Lave J. (1988). Cognition in practice: mind, mathematics, and culture in everyday life. Cambridge University press, Cambridge.

116. Lawson, B (1980). How Designers Think. Architectural Press, London.

117. Leadbeater, C. (2008). We-think: Mass innovation, not mass production. Published by Gardner books.

118. Li, Y., Hong, J.I., Landay, J.A. (2004). Topiary: A Tool for Prototyping Location-Enhanced Applications. UIST 2004, CHI Letters, 6(2), pp. 217-226.

119. Li, Y., Landay, J.A. (2005). Rapid prototyping tools for context aware applications. In Proceedings of the CHI 2005 workshop on the Future of User Interface Design Tools.

120. Lieberman, H., Paterno, F., Klann, M., Wulf, W. (2006). End User Development: an emerging paradigm. In: End User Development, edited by H. Lieberman, F. Paternò, V. Wulf. Springer Netherlands, pp. 1-8.

121. Lieberman, H., Paterno, F., Klann, M., Wulf, W., Eds., (2006). End User Development, Springer Netherlands.

122. Lohmann, S., Ziegler, J., Tetzlaff, L. (2009) Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration, T. Gross et al. (Eds.): INTERACT 2009, Part I, LNCS 5726, pp. 392–404.

123. Mackay, W. (2003). Educating multidisciplinary teams. In: Design Education at the age of the Disappearing Computer, Section editors I. Mavrommati, J. Darzentas), to be found in: Tales of the Disappearing Computer (volume editors: Kameas A., Streitz N. Greek Letters, pp. 105-117.

124. Mankoff, J, Hudson, S., Abowd, G. (2000). Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. Proceedings of the SIGCHI conference on Human factors in computing systems, pp.368-375.

125. Marc, Rl. (1995). Scenarios as engines of design. In Carroll, J. M., ed.. Scenario Based Design: Envisioning work and technology in system development. New York: John Wiley and Sons, Inc..

126. Markman, A.B., Brendl, C.M. (2005). Constraining theories of embodied cognition. Psychological Science, pss.sagepub.com

127. Markopoulos, P. et al. (2001). Interaction design for home information appliances. PC-CHI 2001, Patras, Greece.

128. Markopoulos, P., Mavrommati, I., Kameas, A. (2004). End-User Configuration of Ambient Intelligence Environments: Feasibility from a User Perspective. EUSAI, European Symposium on Ambient Intelligence, Eindhoven, published in: Ambient Intelligence ISBN 3-540-23721-6 Springer Lecture Notes on Computer Science (LNCS3295), pp. 243-254.

129. Mavrommati I., Kameas A., Markopoulos, P. (2004). 'An Editing tool that manages the devices associations'. Personal and Ubiquitous Computing. ACM, Springer-Verlag London Ltd. ISSN: 1617-4909, Volume 8, Numbers 3-4.pp. 255–263.

130. Mavrommati, I., Calemis, J. (2010). ASTRA awareness connectivity platform based on service oriented concepts. in: Gerhäuser, H.; Hupp, J.; Efstratiou, C.; Heppner, J. (Eds.) Constructing Ambient Intelligence - AmI 2008 Workshops, Nuremberg, Germany, Revised Papers, Communications in Computer and Information Science, Vol. 32, ISBN: 978-3-642-10606-4, pp.70-74.

131.   Mavrommati, I., Darzentas, J. (2006). An overview of AmI from a User Centered Design perspective, IET Proceedings of IE06, Athens.

132.   Mavrommati, I., Darzentas, J. (2007). End User Development in AmI: a user centered design overview of issues and concepts. eMinds: International Journal on Human-Computer Interaction (ISSN: 1697-9613), Vol.1, Issue 3.

133.   Mavrommati I., DarzentasJ. (2011) Towards pervasive systems that can support end user development. Hybrid City Symposium, Athens, 4-5 March 2011.

134.   Mavrommati, I., Kameas, A. (2002). e-Gadgets case description. Doors of Perception7 @flow, Amsterdam. http://flow.doorsofperception.com

135.   Mavrommati, I., Kameas, A. (2003a). End-user programming tools in ubiquitous computing applications. In Proceedings of the 10th International Conference on Human - Computer Interaction, (HCI International), Krete, Greece. (2003a).

136.   Mavrommati, I., Kameas, A. (2003b). The evolution of objects into Artifacts: will it be mostly harmless?   Personal and Ubiquitous Computing (PUC), Volume 7 , Issue 3-4, pp. 176–181.

137.   Mavrommati, I., Markopoulos, P., Calemis, J., Kameas, A. (2003c). Experiencing Extrovert Gadgets. Johnson, H., Gray, P. and O'Neil, E. (Eds) Proceedings of HCI 2003, Research Press International, Volume 2, pp. 179-182.

138.   Mavrommati, I., Markopoulos, P., Kameas, A. (2003d). Visibility and accessibility of a component-based approach for Ubiquitous Computing applications: the e-Gadgets case. In Proceedings of the 10th International

Conference on Human - Computer Interaction, (HCI International), Krete, Greece.

139. McInerney, P., and Maurer, F. (2005). UCD in agile projects: Dream team or odd couple?. ACM Interactions, 12(6), 19 - 23.

140. Merrill D., Kalanithi, J., Maes, P. (2007). Siftables: Towards Sensor Network User Interfaces. In the Proceedings of the First International Conference on Tangible and Embedded Interaction (TEI'07), Baton Rouge, Louisiana, USA.

141. Ministry of Public Management Home Affairs Posts and Telecommunications of Japan, Economic Research Office, General Policy Division, Tokyo (2004). MPHPT, Information and Communications in Japan: Building a Ubiquitous Network Society that Spreads Throughout the World, White Paper. Available from http://www.johotsusintokei.soumu.go.jp/whitepaper/eng/WP2004/2004-index.html

142. Moggridge, B. (2007). Designing Interactions, MIT press

143. Mondillon, L., Niedenthalb, P.M., Gila, S., Droit-Voleta, S. (2007). Imitation of in-group versus out-group members' facial expressions of anger: A test with a time perception task, Social Neuroscience, Volume 2, Issue 3 & 4, pp. 223–237.

144. Moran, Carroll (1996). Design Rationale, Concepts, techniques and Use

145. Mori, G., Paternò, F., Santoro, C. (2004). Design and development of multi-device user interfaces through multiple logical descriptions, IEEE Transactions on Software Engineering, IEEE Press, August, Vol. 30, No. 8, pp.507–520.

146. Mugellini, E., Rubegni, E., Gerardi, S., Khaled, O.A. (2007). Using personal objects as tangible interfaces for memory recollection and sharing. In Proceedings of the 1st international Conference on Tangible and Embedded interaction (Baton Rouge, Louisiana, February 15 - 17, 2007). TEI '07. ACM Press, New York, NY, pp. 231-238.

147. Murakami, T. (2003). Establishing the Ubiquitous Network Environment in Japan: From e-Japan to U-Japan. NRI Paper 66. Tokyo: Nomura Research Institute.
http://www.nri.co.jp/english/opinion/papers/2003/pdf/np200366.pdf.

148. Myers, B.A., Ko, A.J., Burnett, M.M (2006). Invited research overview: end-user programming, CHI '06 extended abstracts on Human factors in computing systems, Montréal, Québec, Canada.

149. Nardi, B. (1993). A Small Matter of Programming: Perspectives on End User Computing, Cambridge, MIT Press.

150. Nardi, B. (1996). Studying context: a comparison of activity theory, situated action models, and distributed cognition. In Context and Consciousness: Activity theory and Human-Computer interaction, B. A. Nardi, Ed. Massachusetts Institute of Technology, Cambridge, MA, pp. 69-102.

151. Nardi, B**.,** ed. (1996). Context and Consciousness: Activity theory and human-computer interaction. Cambridge, MA, MIT Press.

152. Nardi, B.. website, http://www.artifex.org/~bonnie/#pr1

153. Newman, M., Sedivy, J., Neuwirth, C.M., Edwards, W.K., Hong, J.,I., Izadi, S., Marcelo, K., Smith, T.F. (2002). Designing for serendipity: supporting end-user configuration of ubiquitous computing environments (ACM SIGCHI DIS2002), London, England. NY: ACM, pp. 147-156.

154. Newmann, W.M., Lamming, M.G. (1995). Interactive System Design, Addison Wesley.

155. Nichols J., Faulring A. (2005). Automatic Interface Generation and Future User Interface Tools: in Proceedings of the CHI 2005 workshop on the Future of User Interface Design Tools.

156. Niedenthal, P. M. (2007) Embodying emotion. Science 316:1002–05.

157. Norman D.A. (1988). The Psychology of Everyday Things, New York, Basic books.

158. Norman D.A. (1990). The design of everyday things, Doubleday, N.Y.

159. Norman D.A. (1999). The Invisible Computer. MIT Press.

160. Norman, D. (1993). Things that make us smart: Defending human attributes in the age of the machine. Reading, MA: Addison-Wesley.

161. Norman, D.A. (1991). Cognitive artifacts. In: J.M. Carroll, Editor, *Designing interaction: Psychology at the human-computer interface*, Cambridge University Press, New York.

162. Norman, Don (2010). Natural User Interfaces Are Not Natural. ACM Interactions magazine, XVII.3. Article also donwloadable from: http://jnd.org/dn.mss/natural_user_interfaces_are_not_natural.html

163. Olsen, D, Klemmer, S. (2005). Proceedings of the CHI 2005 workshop on the Future of User Interface Design Tools.

164. Papert, S. (1986). Constructionism: A New Opportunity for Elementary Science Education. A proposal to the National Science Foundation.

165. Paternò, F., Santoro C., Mäntyjärvi J., Mori G., Sansone S., Moruzzi G. (2008). Authoring pervasive multimodal user interfaces. In: International Journal of Web Engineering and Technology, IJWET 2008.

166. Piaget, J. (1962). Comments on Vygotsky's critical remarks concerning The Language and Thought of the Child, and Judgment and Reasoning in the Child, by Jean Piaget, MIT press, available online from: http://www.marxists.org/archive/vygotsky/works/comment/piaget.htm

167. Piaget, J., Duckworth, E. (1968). On the development of memory and identity - Clark University Press.

168. Punie, Y. (2005). The future of Ambient Intelligence in Europe: The need for more Everyday Life. In: Communications and Strategies 57, pp. 141-165.

169. Rakers, G. (2001). "Interacting Design Process. In: User Interface Design for Epectronic Appliances, Baumann K., Thomas B. (Eds), Taylor and Francis, pp. 32.

170. Rauterberg, M. (2003). Human Computer Interaction Research: a paradigm clash?. In: *Design Education at the age of the Disappearing Computer,* Section editors I. Mavrommati, J. Darzentas), to be found in: Tales of the Disappearing Computer (volume editors: Kameas A., Streitz N. Greek Letters, pp. 157-164.

171. Reeder, K. (2002). Primary Techniques for Concept generation in the product development process. In proceedings of the IDSA National Design Education Conference, San Jose, CA, p.p. 285-290

172. Repenning, A., Ioannidou, A. (2006). What makes End-User Development Tick? 13 design guidelines. End-User Development. H. Lieberman, F. Paterno, V. Wulf. Springer pp.51-86.

173. Ringas, D., et al (2002). An Architecture that Supports P2P Networking among Ubiquitous Computing Devices. 2nd IEEE international conference on Peer to Peer Computing, (P2P 2002), Linkoping, Sweden.

174. Rodden, T, Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Åkesson, K-P., Hansson, P. (2004). Between the dazzle of a new building and its eventual corpse: assembling the ubiquitous home. Proceedings of the 2004 ACM Symposium on Designing Interactive Systems, Cambridge, Massachusetts: ACM Press.

175. Rodden, T. Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Åkesson, K., Hansson, P. (2007). "Assembling connected cooperative residential domains", The Disappearing Computer: Interaction Design, System Infrastructures and Applications for Smart Environments (eds. Streitz, N., Kameas, A., Mavrommati, I.), pp. 120-142, Heidelberg: Springer.

176. Rodden, T., Benford, S. (2003). The evolution of buildings and implications for the design of ubiquitous domestic environments. In: Proceedings of the CHI 2003 conference on human factors in computing, Florida, USA.

177. Rodden, T., Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Åkesson, K-P., Hansson, P. (2004). "Configuring the ubiquitous home", Proceedings of the 6th International Conference on Designing Cooperative Systems, May 11th-14th, French Riviera: IOS Press.

178. Rogers, Y. (1997). A brief introduction to distributed cognition. Available: http://www.cogs.susx.ac.uk/users/yvon-ner/dcog.html.

179. Rolland, C., Achour, C. B., Cauvet, C., Ralyte, J., Sutcliffe, A., Maiden, N. A. M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P. (1996). A proposal for a scenario classification framework. *Requirements Engineering 3*(1), pp. 23-47.

180. Roman, M., Cerqueira, C.K.H.R., et al. (2002). "Gaia: A middleware infrastructure to enable active spaces," IEEE Pervasive Computing, pp. 74–83.

181. Roschelle, J., Koutlis, M., Reppening A. (1999). "Developing Educational Software Components", IEEE Computer, September 1999 Special Issue on Web based learning and collaboration, pp. 2-10.

182. Rosson, M.B., Carroll, J.M. (2002). Usability Engineering; scenario based design of human computer interactions, Morgan Kaufmann.

183. Saffer, D. (2007). Designing for Interaction: Creating Smart Applications and Clever Devices. New Riders, AIGE design press.

184. Schmidt, A. (2005). Interactive Context-Aware Systems Interacting with Ambient Intelligence.In: Riva, G./Vatalaro, F./Davide, F./Alcañiz, M. (eds) Ambient Intelligence. IOS Press part 3, chapter 9, p.159-178.

185. Schneider, J.G., Nierstrasz, O. (199). Components, scripts and glue, in Software architectures – advances and applications (J. Hall, P. Hall – eds), Springer-Verlag, pp. 13-25.

186. Scholtz, J., Consolvo, S. (2004). Towards a discipline for evaluating ubiquitous computing applications. INTEL white paper, IRS-TR-04-004. http://www.intel-research.net/Publications/Seattle/022520041200_232.pdf

187. Schon, D.A. (1983). the reflective practicioner: how professionals think in action. New York, Basic Books.

188. Sharp, D., Salomon, M. (2008). User Led Innovation: a New Framework for Co-creating Business and Social Value. Published by CRC & Swinburne University of Technology.

189.   Sharpe, B. (2003). Information Appliances, an introduction. Downloadable white paper, http://www.appliancestudio.com

190.   Shore, James and Warden Shane (2007): The art of Agile Development. O'Reilly Editions.

191.   Simon, H. (1969). The Sciences of the Artificial, MIT press.

192.   Sohn, T., Dey, A.K. (2003). iCAP: An Informal Tool for Interactive Prototyping of Context- Aware Applications. In: Extended Abstracts of ACM Conference on Human Factors in Computing Systems (CHI 2003). ACM Press, New York, pp. 974-975.

193.   Souza de, C.S., Barbosa, S.D.J. (2006). A semiotic framing for end-user development. In End-User Development. H. Lieberman et al., Eds. Kluwer, Dordrecht, the Netherlands, pp. 401-426.

194.   Spencer R., "The streamlined cognitive walkthrough method," Proceedings of the Conference of Computer Human Interaction, (CHI 2000), ACM Press, Hague Netherlands, April 2000, pp 353-359.

195.   Streitz, N., Kameas, A., Mavrommati, I., eds. (2007). The Disappearing Computer: Interaction Design, System Infrastructures and Applications for Smart Environments. Heidelberg, Springer.

196.   SWAMI D1: Safeguards in a World of Ambient Intelligence (SWAMI) (2006). IST Deliverable D1: the brave new world of ambient intelligence: A state of the art review. Friedewald M. Vildjiounaite E., Wright D. (Ed). Accessed September 2007 from http://swami.jrc.es

197. SWAMI final report: Safeguards in a World of Ambient Intelligence: Final report (SWAMI) (2006). Wright David. (Ed). Commissioned by Joint Research Center (JRC). Accessed September 2007 from http://swami.jrc.es

198. Terry M., Mynatt E., Nakakoji K., Yamamoto Y. (2004). "Variation in element and action: supporting simultaneous development of alternative solutions", Proceedings of the 2004 conference on Human factors in computing systems, pp. 711-718.

199. Truong, K.N., Huang, E.M., Abowd, G.D. (2004). "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home.", Proceedings of Ubicomp 2004, pp. 143-160.

200. Vygotsky, L. (1896-1934), archive, available online at http://www.marxists.org/archive/vygotsky/

201. Vygotsky, L. (1934). Piaget's Theory of Child Language and Thought (chapter 2), in: Thought and Language. Edited and translated by Eugenia Hanfmann and Gertrude Vakar; Publisher: The M.I.T. Press, 1962; (originally written in 1934). Available online from: http://www.marxists.org/archive/vygotsky/works/words/index.htm

202. Vygotsky, L. S. (1978). Mind in Society: The Development of Higher Psychological Processes. Cambridge,MA, Harvard University Press.

203. Weis, T., Handte, M., Knoll, M., Becker, C. (2006). "Customizable Pervasive Applications". International Conference on Pervasive Computing and Communications (PERCOM), Pisa

204. Weiser, M. (1991). The Computer for the Twenty-First Century. Scientific American, pp. 94-10.

205. Weiser, M. (1993). Some Computer Science Issues in Ubiquitous Computing. Communications of the ACM, (CACM) 36(7), pp. 75-84.

206. Weiser, M. (1994). The world is not a desktop. Interactions, pp. 7-8.

207. Weiser, M., Seely Brown, J. (1996). Designing Calm Technology. PowerGrid Journal, v 1.01. (Also appeared as Chaper 6 - The Coming Age of Calm Technogy in the book Beyond Calculation - The Next Fifty Years of Computing by Peter J. Denning and Robert M. Metcalfe, Copernicus/An Imprint of Springer-Verlag).

208. Winkielman, P., Niedenthal, P.M., Oberman, L.M. (2009). Embodied Perspective on Emotion-Cognition Interactions. In Mirror Neuron Systems. Contemporary Neuroscience, 4, pp. 1-23, DOI: 10.1007/978-1-59745-479-7_11

PROJECT WEBSITES AND SOURCES:

209.   Accord project website: http://www.sics.se/accord/

210.   Ambient Assisted Living (AAL) joined programme website: http://www.aal-europe.eu/

211.   Ambient   Intelligence,   Philips   Research   Password
        Issue   13,   Oct.   2002,   (last   accessed   Jan   2010   from
        http://www.research.philips.com/newscenter/pictures/password_13.html

212.   ASTRA Deliverable D4 (2009a), sections 3,4,5, (Markopoulos P. ,
        Mavrommati I.) available from ASTRA project website: http://www.astra-project.net/ , pp.5-65

213.   ASTRA Deliverable D4 (2009b), Section 9 "Communities and End User
        Development: The ASTRA repository and application manager" (Divitini
        M.), available from ASTRA project website: http://www.astra-project.net/ ,
        p.p.91-111.

214.   ASTRA project website: http://www.astra-project.net/

215.   e-Gadgets project website: http://www.extrovert-gadgets.net

216.   e-Slate website: http://e-slate.cti.gr/

217.   EUD-Net Network of Excellence website. http://giove.cnuce.cnr.it/eud-net.htm

218.   KINECT website: www.kinect.com .

219.   Oxygen project website: http://oxygen.lcs.mit.edu/

220. OXYGEN project website: http://www.oxygen.lcs.mit.edu/

221. Philips Research website: http://www.research.philips.com (pictures of Ambient Intelligence projects at: http://www.research.philips.com/newscenter/pictures/systsoft-ambintel.html

222. PLANTS project website http://daisy.cti.gr/plants/

223. SIFTABLES website: http://sifteo.com

224. SIFTABLES website: http://sifteo.com/

225. SMART-ITS website: http://www.smart-its.org/

226. The Disappearing Computer (DC) initiative: http://www.disappearing-computer.net/

17. Appendix 1 - Expert review

17.1. Introduction

This report describes the early evaluation of e-Gadgets from the perspective of an end-user. It details the process and results for the evaluation meeting held at the Technical University of Eindhoven during the first project year, and, in the next appendix (appendix 2) a review of the concepts discussed using the Cognitive Dimensions framework of (Green and Petre, 1996). The aim of the meeting has been to evaluate the e-Gadgets concepts and the proposed interaction concepts with respect to end-users. In general we are interested in the perceived usefulness and ease of use of the technologies envisaged, but more particularly focus on two issues:

- Are the targeted users likely to comprehend and use the e-Gadgets (G) and GadgetWorlds (GW)?
- Are the targeted users likely to be willing to use them?
- The workshop tried to address these questions systematically, eventually collecting recommendations for the project.

In the following sections we describe:

- The process followed in the evaluation meeting.
- The comments collected during the evaluation meeting.
- The conclusions drawn for the project.

17.2. Method

At the time of the evaluation, most technology was still under development and too immature to withstand a test with users. Nevertheless, it was felt that issues of usability and perceived usefulness, needed to be considered at this stage of the project, in order to enable e-Gadgets to realize the ambition of end-users architecting

their Gadgetworlds. Consequently, 3 experts in user system interaction were invited for a structured workshop to evaluate related user issues. The experts were:

- M.Bekker (MB), TU/e, Assistant Professor and experienced researcher in user system interaction, with expertise in participatory design and designing for children.
- C.Huijnen (CH), TU/e expert in user centered design with a background in Cognitive Psychology.
- A.Gritsenko (AG), TU/e expert in user centered design with a background in Industrial Design.

All are familiar and friendly to technology but not computer programmers. This also fits the profile that we had in mind for the end-user of e-Gadgets for this evaluation (technophile but not programmer). Further participants to this meeting were:

- Panos Markopoulos (PM), facilitated the process and at times when appropriate also acted as an expert evaluator.
- Irene Mavrommati (IM) acted as a facilitator and presented the concepts of the project to the participants.

The agenda of the meeting is added as an appendix. The intention was that first the concept would be presented in an abstract form by IM. A first round of comments was collected to warm up the discussion, let experts familiarize with the concept and focus on the usage aspects.

A collection of four scenarios was then discussed that highlighted different usage issues. A small discussion followed each scenario, that had elements of a focus group (trying to get all participants vocal, not aiming for consensus, trying to get at gut reactions for the concepts proposed, helping participants build on each other's ideas).

A problem solving exercise was given to see the extent to which these experts could build their GadgetWorld (GW) and further, when they did so, to reflect on what they consider as problems for the end-user (based on this first hand experience).

Finally, the current interface for the GW editor and video-prototypes created by IM, for Tangible Interfaces for constructing GWs were shown. Opinions were solicited from all experts. In the following section we discuss the various points raised by the experts as they occurred in sequence (as opposed to thematic units).

The meeting ended with a round the table request for global level feedback for the project.

A note has to be made for the role of the evaluation. First the evaluation did not aim to assess the concepts or products of e-Gadgets so far. <u>It deliberately focused discussion on problem areas and on hypothetical scenarios that exposed the pitfalls users might confront through the technological vision of e-Gadgets.</u> The aim of the evaluation was formative, i.e., it aimed to provide a direction for the next steps of the project, that will ensure user needs are taken into account. As a formative evaluation, it did not focus on detailed interaction (look and feel), as the relevant end-user interfaces, have not yet been developed. Rather we focused on the concepts and on the role the e-Gadgets technology can play in fulfilling user needs.

17.3. Comments (feedback) from experts

Reactions to basic concept: A general problem for ubiquitous computing environments is the visibility of their boundaries. It makes sense if these boundaries coincide with physical boundaries, e.g., of a room. When we draw GWs as connected graphs of digital artifacts, and we draw boundaries around groups of gadgets forming a GW, we are drawing a boundary that is invisible to the end-user. This nesting of GWs is necessary for scaling them up, but <u>needs to be made visible to the end-user.</u>

In the drawings of GWs, synapses are associated with different meanings. There is a tension here, between needing a simple uniform mechanism for connecting e-Gadgets, and the different meanings such synapses will have for the end user. A first reaction to the notion of synapses, is that they should not be loaded with semantics, as users might interpret these inconsistently. The notion of a synapse should thus be of a single type, (eg, as entering a plug into a socket), at least for the 'naive' end user.

It important for the user to be able to check whether a synapse works. This is one of the many examples of observability that need to be engineered into the gadgetworld editors. There is an important trade-off: you need the system to be observable to repair, to understand, but you don't want the work that comes with it.

The semantics of synapses were not clear to the experts from the first introduction. Several possibilities can be envisaged: e.g., control flow, data flow, stimulus-response, bi-directional constraints between values, etc. Depending on the choice made here, there will be different abilities of the user to predict what the outcome of a synapse will be. Consider for example, trying to predict the execution of a program from its control-flow specification or to predict the output value of an electronic circuit, from its input signal. There is a clear gap between a static structure and its runtime behaviour and restraint has to be applied in e-Gadgets concerning the ability of users to construct GWs and predict the behaviour of these worlds. Simple, non-complex and hierarchical structures of GWs should be preferred. More complex structures are anticipated to be beyond the capabilities of the average target user.

The role of intelligence was discussed a lot by experts. In this first stage, two common caveats of adaptive systems were noted: knowing and controlling when the system is learning, e.g., if the system is your room how do you stop the adaptation of it, and how do you cope with multiple users? These issues are not specific to gadgetworlds.

Regarding how users will perceive GWs and e-Gadgets, a fear was voiced as to whether augmenting every-day objects with computing capabilities will be a shock to users or not. An alternative suggested, was that a complete new breed of objects, clearly distinguishable should constitute gadgetworlds. At this stage though, how users will accept the computational enhancement of every day-objects cannot be conjectured without some long-term user trial.

Discussion on Scenaria

The scenaria are included at the end of this section. Comments are listed per scenario. We do not duplicate issues arising in later scenarios when they have been addressed in earlier ones.

Scenario 1

The experts in the meeting recognized the gadgetworld construction or modification as a programming activity. Some were skeptical about the end-user acceptance for programming activities. However, other researchers, (eg, Newman et al 2002) consider end-user programming as a crucial element for the advent of a ubiquitous computing environment. In the following, we shall not adopt the skeptical view about end-user acceptance of end-user programming, but shall focus on the potential hurdles that users have to overcome for GW construction.

The end-user programming is by its nature a secondary task (Card Moran and Newell, 1980). Therefore the effort required for GW construction should be commensurate with the value and complexity of the primary task. Thus, a large GW construction effort will only be justified if the value or the complexity of the primary task is large. Constructing a GW to support a simpler task, e.g., switching on the light as you enter a room, should be straightforward to support. Even though the product that will be delivered to end-users was not in the scope of this evaluation, we note some remarks made that concern the deployment and marketing of gadgetworlds (and not the concepts of concern to this research project). E.g.,

- A GW should work out of the box – no set up or programming should be needed.

- Decoration is above functionality in the home. People will want to choose on the basis of aesthetics or extend their furniture with e-Gadget capabilities.

- Providing energy to all of these devices may be problematic (a maintenance headache and an environmental threat).

- It should be possible to add personalisation to the capabilities that come out of the box.

- Finally, the search for gadgets or gadgetworld configurations (e.g. search on the internet) should be flexible. Search mechanisms should support both strategies: starting from both what you have (gadgets) and from the function you want to do (task).

Scenario 2

The second scenario gave rise to discussions about adaptivity. Some possible needs for the user were discussed:

- It should be possible to override system adaptation and automation. This touches upon a central problem for ubiquitous computing research, i.e., the ability to control an augmented environment that adapts to the user and is designed for invisibility.

- Adapting to conflicting needs of multiple users is seen as a major challenge, probably a major research issues on its own right.

- Auto-configuration helps in this scenario, but in general, users would like to veto or control when a new item is added to the GW. When will this stop? E.g., who asked it to connect to the GW of the next house, or of the bedroom, etc. The user must stay in control of system behaviour.

- A challenge in designing GWs is deciding what user habits are relevant to watch. It seems that this must be included in the 'programming' tasks for end-users.

Scenario 3

This scenario took the discussion beyond the first time user. The following requirements were conjectured:

1. Complexity of context sensing should be allowed to grow as the user gets more experienced with e-Gadgets.

2. <u>Debugging/test mode facility necessary.</u> Just like programmers need it, the users will need it to, since you ask them to construct a type of program. For example, the users will need to verify that the light switches on when it gets dark without waiting for the sun to set.

3. How do you tell instances of the same e-Gadget (e.g., many lights, many chairs, cups). <u>The interaction with GWs has to be based on some sort of deixis (e.g., "this gadget"") or reference (e.g., gadget XXXX at YYYY).</u>

4. A dependency to physical spaces was noted. The description of gadgets and their links on the one hand is facilitated by reference to the physical world, e.g., the light by the desk, but needs to be orthogonal to that, e.g., consider what happens when you rearrange the furniture.

Scenario 4

Scenario 4 was deliberately constructed to expose the feasibility of writing short scripts for gadget worlds. This is not an impossible feature; indeed a large number of people are accustomed to writing such scripts for their professional activities. However, the implicit application context for gadgetworlds has been non-professional activities, where we can expect reticence by users to act as programmers.

The following recommendations can be drawn from the discussing.

- Harder programming tasks should be supported by the system, e.g., by wizards or templates.

- Plugs are architectural abstractions, but the users start from task related abstractions. Wizards are a way to bridge the gap unless we expect some serious programming effort by the users.

Problem solving exercise

In this part of the meeting IM presented GWs of increasing complexity (see appendix 3), to show a diagrammatic way of modelling them and to convey how different behaviours are achieved through different GWs. CH, AG, MB and PM made their own examples based on the floor-mat e-Gadget collection. Their solutions are included in appendix 4.

Problems identified are:

1. Arrows indicate directed-ness, and connections seem to imply stimulus-response pairs. This is intuitive, but the synapse concept was introduced as undirectional. If there is direction in the synapse, what should the mental model of the user be? Should it be a flow of information, control, etc? The project must commit to one meaning for a synapse, the desired mental model for it and choose graphical representations accordingly.

2. During problem solving, it was not clear to the participants whether concepts should be operationalised as plugs or as synapses. For example, why is a study a plug in figure x of appendix 1 and not a synapse. Why is the plug a combination of basic functions?

3. If the software creates new higher level plugs, what guarantees do you have that the user will understand them as fitting their mental model of the GW? The user sees and understands behaviour, not the GW structure. Templates can be provided to support the mapping between tasks of the user and the GW structures that will support them.

4. It was found confusing that sensors were attached to everyday objects. E.g., if you want to know the light intensity this shouldn't be a plug of the book e-Gadget, but from a light intensity object. Sensing objects can be 1st class

objects, as opposed to attributes of an augment object, like a book. Decisions like these are very hard for the end-user. Professional programmers know from experience or rely on patterns or styles to make these decisions. GAS should make explicit and communicate through the design of the GW editor how the user can best map concepts from their world to the GW.

5. People are an important component in describing context and activity. They are completely missing from the GW. Currently, their effect on e-gadgets describes parts of their activity indirectly. Rather, a GW should include re-usable abstractions of people that the user can utilise directly in the GWs they architect. For example, if you have an abstraction "Small group of people", this could provide plugs to different kinds of sensors that would detect its existence and its activities. This construction could be supported directly in the GW editor rather than be left upon the user, or be implicit in the connections made between e-Gadgets.

Agent Based Behavior

As a follow up to the problem solving, IM presented a scenario where an agent adds an object after watching the behavior of the user. The reactions were:

1. Which is the most "important" lamp? The one on the table or the one on the floor? It seems that you control through the one you made first. This brings up an important issue in the design of GAS: is history important, are GWs associative? The experts did not provide any direction to this decision.

2. The experts treated the autonomy of the system in making reconfigurations with scepticism. They pointed out common caveats with adaptive systems: how does a user know what changes are happening? Are the decisions made by the system meaningful to the user? How can the user observe the status of the system at any moment?

These are difficult and very broad research questions. The experts thought that a tool for inspecting and monitoring GWs, are a necessary addition.

GUI Interface

A screen mock-up of a GW editor was shown to participants (see figure 1), and it was explained to them how GWs could be created or modified with it.

Discussion with the experts, focused on the bridge between the on-line information that is mostly context independent, and the real GW that is installed and affects your own living environment. The following recommendations were made:

1. From an abstract description it is hard to know what a GW will do. <u>A test mode is needed</u> where the installed behaviour is simulated, before it is actually installed. A potential solution would be a virtual room to simulate the operation of a GW.
2. Users will need to store, retrieve and redeploy end-user programs made for GWs was identified. For example, you could travel taking preferred '; light settings' with you.

3. The graphical interface (figure 1) seems to differ significantly from the conceptual diagrams, used in the problem solving. The graphical interface does not show the connections shown on the conceptual diagrams.

Figure 1. The screen design for the graphical GW editor, that was shown during the evaluation meeting.
Figure x. The screen design for the graphical GW editor, that was shown during the evaluation meeting. The user can browse e-Gadgets through (bottom left), or complete gadgetworlds (bottom right). A search facility is foreseen (top right), where the user enters task related information. Once a gadgetworld or a gadget has been selected, a matrix including gadgets and their attributes is presented at the top left window.

17.4. Advanced interface options

IM presented video prototypes of a magic wand interface, a PDA interface, and one with tangible representations of plugs.

The reactions of the experts were positive for the overall concepts. Concerns expressed were about detailed design issues not shown on the video prototypes: e.g., the want helps only select e-Gadgets, but the problem of specifying desirable behavior has not been solved.

<u>In some cases, a more text based and programmer oriented interface can be more understandable than a gesture based interface.</u>

Global comments

At the end of the meeting a round-table was done, where general remarks were collected. The experts expressed the following opinions:

AG- Extending traditional objects with a digital self is confusing.

The comments of CH and MB concerned mostly the perceived usefulness of GWs. Contrasting the light scenario with Norman's thinking about every day things (DOET), she commented on how ordinary things cause hassle, e.g., opening doors. E-gadgets should have a value proposition for solving some of these problems instead of complicating them. E-gadgets would benefit by trying to address some of these daily hassles.

AG focused on how much behaviour of the e-Gadgets remains invisible to the user and on the breaking of expectations one has from every day objects. This hidden complexity can be positive as long as the user does not have to repair the system behaviour. In other cases, more observabilty of e-Gadget behaviour has to be ensured.

17.5. Summary of evaluation session

There were many comments raised and there is little objective means of prioritising them. Perhaps the remaining impressions of issues that recur throughout the evaluation discussion recorded above are the following:

1. Ubiquitous computing technologies embedded in physical objects effectively add hidden behavior and complexity to them. Problems may arise if this behavior is not observable, inspectable and predictable for the user.

2. The example applications chosen to demonstrate the project concepts are almost as important as the concepts themselves, to ensure higher perceived usefulness.

3. Intelligence causes problems of observability and of unpredictability for users. It must be used with caution and this should be reflected in the demonstrations built.

4. Constructing and modifying GadgetWorlds is a problem solving activity performed by end-users. As such, it has an algorithmic nature and thus good programming support should be offered, (e.g. Inspectors, debugging, search mechanisms, unambiguous syntax), rather than hiding the complexity of the activity.

5. Some clarity about the meaning of a synapse is needed. It was felt that a simple directed relation, e.g., event broadcast and event listening would be a useful convention. Currently, a synapse could be interpreted in many inconsistent ways.

Related material used for the session is shown in the following sections

17.6. Problem solving: Applications created in the evaluation workshop.

The figures below show how experts connected their own gadgetworlds. Broadly speaking, they expressed themselves in these diagrams, though not in a consistent

manner. This shows that without tool support to suggest well-formedness of the configurations diagrammatic constructs may be used arbitrarily by users.
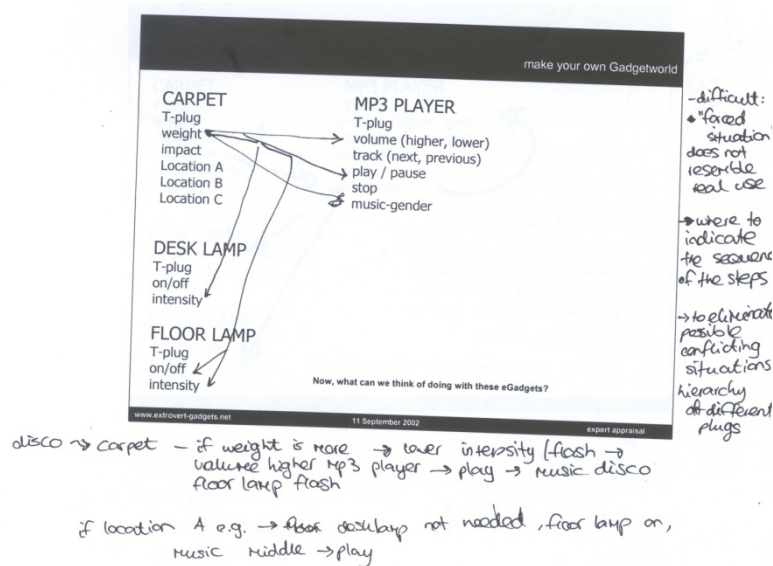
Gadgetworld 1. A carpet that plays disco for heavy footsteps.

This expert found it difficult to articulate the exact task she wanted with the constructs in hand. It was further confusing for her that she had to describe architectural configurations, while she was thinking of defining behavioral abstractions (Sequence of steps). It would be useful to provide libraries of re-usable templates that encapsulate behavioral abstractions. Then users could re-use and modify these ready-made configurations.
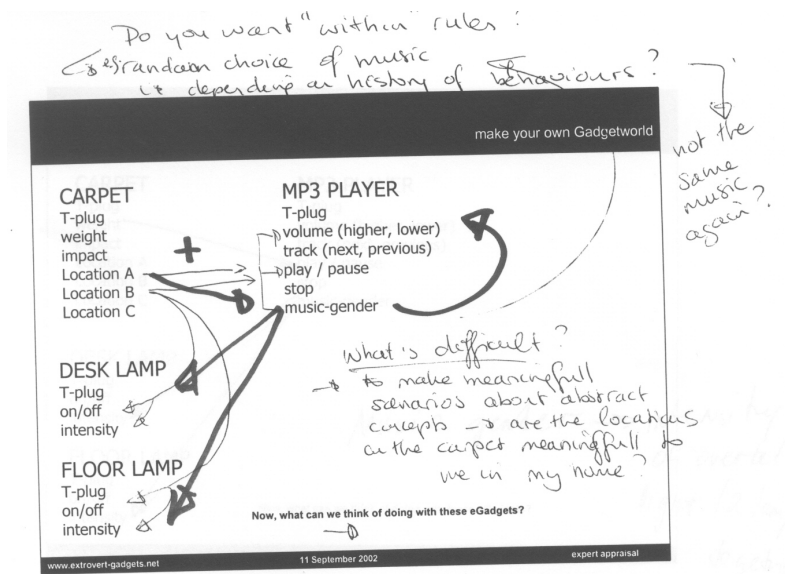
Gadgetworld 2. A carpet that sets music and lighting for romantic situations or decides it is study time.

This expert commented on the gap between concepts that are meaningful for the system (E.g., the definition of locations A,B,C) and how the user identifies concepts meaningful to them (e.g., how does the user understand the concept of a location in the carpet, is it in the centre or periphery, is in left or right). In other words the mapping of concepts from the mental model to the system model is a programming activity (e.g., defining the borders of locations A,B, and C) that is not per se supported by the e-Gadgets architecture and remains an open issue in terms of end-user programming of a GW.
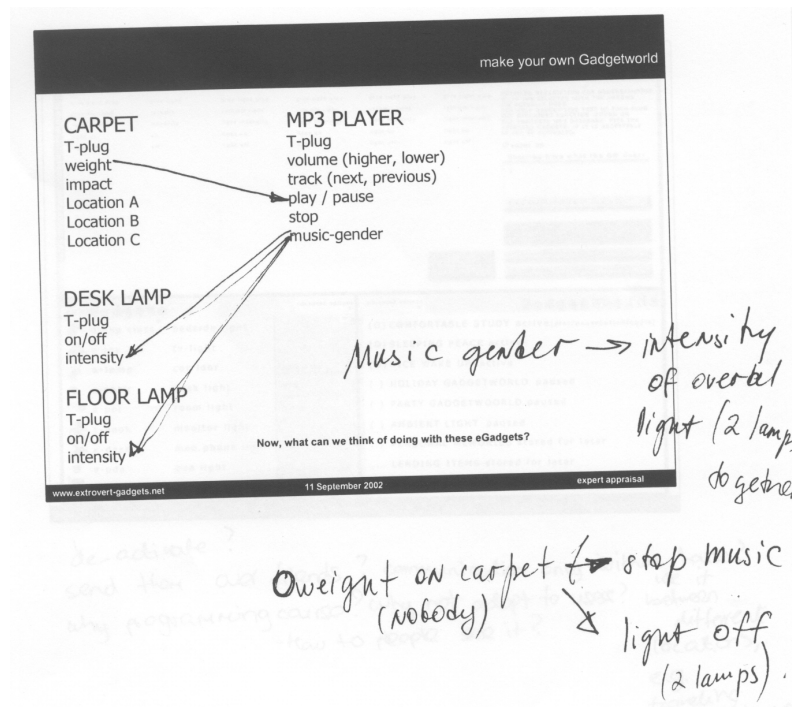
Gadgetworld 3. A carpet that turns off the music if you are not on it.

This expert commented on the need to model people inside the programmatic environment. The example shows how ordinary objects are enhanced with sensing capabilities, and therefore become elements in expressing a program. However, it seems that situations we would wish to program for are more directly expressed in terms of the people inside them, rather than in terms of what the sensors sense. In conclusion if we want people to program their own living environment as a GW, then we have to provide re-usable ready made abstractions to represent the system's model of the people inside the environment.

17.7. Example configurations shown to participants

A series of gadgetworld architectures were shown diagrammatically to participants by IM. These had increasing difficulty, and introduced increasingly higher level constructs, e.g., a synapse, creating a new plug, creating a new object. We show here two of these configurations. Gadgetworld 1 shows a simple set of plugs and some associated rules, for controlling a light depending on whether a person sits on a desk, or has a book open or places an object of some weight on the desk.

Gadgetworld 2, shown below, is an example of introducing a plug, to model architecturally an abstract concept 'study' which is meaningful for the user.

Gadgetworld 1: Diagrammatic representation of a gadgetworld that controls light intensity

DESK
T-plug
weight
proximity

LAMP
T-plug
on/off
intensity

BOOK
T-plug
open/closed
luminosity

CHAIR
T-plug
occupancy
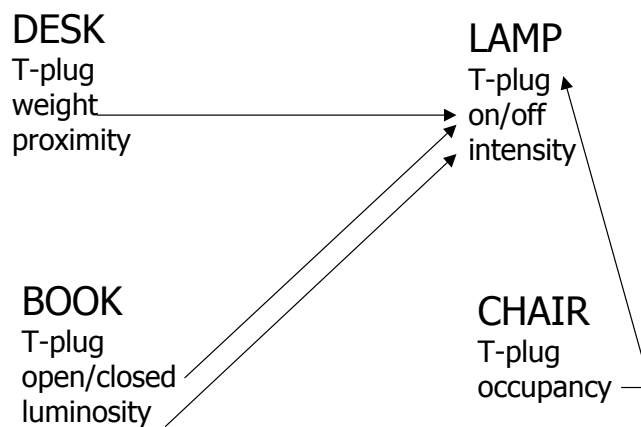
…and is that what we really want?
Rules:
when something (WEIGHT) is on the DESK, then turn LAMP ON
When the BOOK is OPEN, then turn LAMP ON
When someone is sitting on the CHAIR, then turn LAMP ON
Adjust LAMP INTENSITY according to the level of light on the
BOOK (LUMINOCITY).

Gadgetworld 1: Representation of a gadgetworld that controls light intensity

DESK
T-plug
weight
proximity
study

LAMP
T-plug
on/off
intensity
study

BOOK
T-plug
open/closed
luminosity

CHAIR
T-plug
occupancy
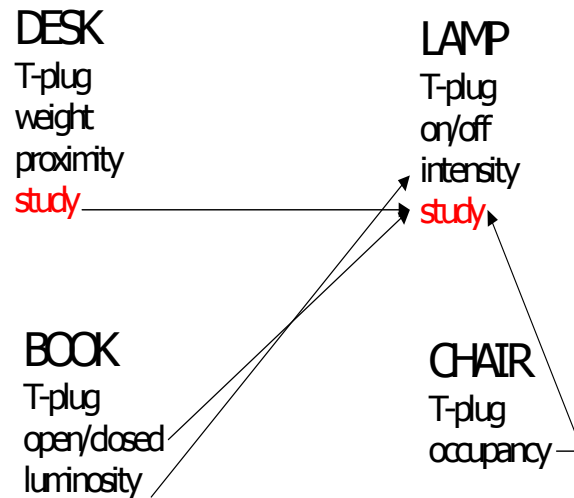
Advanced Rules:
The state of STUDY plug (DESK) functions with rule:
If (WEIGHT >0 and PROXIMITY >=2) then STUDY = 1
The state of STUDY plug (LIGHT) functions with the rule:
If (STUDY=1) then (TURN LAMP ON)

description:
When something is NEAR the DESK **AND** something is ON the DESK
AND SOMEONE is sitting ON the CHAIR **AND** The BOOK is OPEN
THEN *ADJUST INTENSITY ACCORDING TO LUMINOCITY*

Gadgetworld 2: Introducing a plug to represent the more complex concept of a study.

17.8. Material Used for Expert Appraisal (agenda, scenarios, schemes)

Agenda for evaluation meeting:

IPO 1.24, room 1.18, 9:00

1. Opening
2. Confidentiality agreement
3. Introduction to e-Gadgets concepts (Irene)
   Round table with initial reaction.

   Panos minutes dialogue.

4. Scenarios presentation and claims analysis
   Panos presents scenarios

Participants write on post-it notes, any of the following:

- Problems relating to using e-Gadgets
- Issues relevant for user acceptance
- Trade-offs, e.g., <u>feature</u> **causes** <u>desirable consequences</u> **but also** <u>undesirable consequences</u>

Irene collects notes on flipchart. Panos summarises on paper

5. Problem solving exercise. Modify your gadget world.
   Irene sets exercise that participants solve. Again comments are invited and inventoried. Panos keeps notes.

6. Irene presents the scenario for agents for configuring gadgetworlds.
   Comments and suggestions offered. Panos collects them.

7. Participants point out situations and concerns not covered in scenarios.
   Panos keeps minutes notes.

8. Irene shows programmer's GW editor and ideas for the user's editor.
   Participants & Panos note comments down on post-its. Irene collects notes and collects comments.

9. If time permits, the role of intelligence will be discussed.
   End.

 Scenarios Used are mentioned below:

Joe Bloggs + no script

Joe is an English literature student and an early adopter of new technologies. He's recently purchased, the desk-lighting package from the local e-Gadgets supplier. This includes an e-Gadget cushion for his seat, an e-Gadget lamp and an e-Gadget desk mat. Joe wonders what the cushion does. He sits on it and notices that a small LET indicator lights up.   He guesses it is supposed to sense if he is seated or not. After putting them in place, he gets to the e-Gadgets product site and uses the search facility. He enters the following description:

"Adjust the light on my desk when I study"

The search facility identifies a few packages that he can download. He inspects them on his e-Gadget editor. One of them includes a collection of items he has purchased. He downloads it and activates it by pressing the activate button.

In the beginning nothing happens. He opens the book and puts it on his desk hoping for the light to go on. Nothing happens though. He looks at his gadgetworld editor, selects the gadgetworld and from the context-menu selects the option 'show me yourself'. A soft glow illuminates the rim of each object included in the package. He decides to sit on the chair with the cushion on. At that moment, the light goes on, because the logic

implemented by the gadgetworld he has downloaded is that the light will switch on, when it is dark, the book is open and someone sits on the cushion.

Joe checks again on his gadgetworld editor and re-reads the description of the selected gadgetworld. He now notices better the role the cushion was supposed to play.

Joe Bloggs + Agents

Joe, a 21-year-old Law student lives in a student dormitory, in the University campus. He is familiar with PC use, (he uses it mostly for text editing and web searching), but by no means a programming expert.

Joe has recently created his Study Gagetworld, with a new "extrovert-Gadgets" system that he recently bought, and he has been using for a week. He has installed the Study GadgetWorld as he downloaded it from the e-Gadgets web site. This turns on the light automatically, when he is studying on his desk, since the desk light switch is at the back of the shelves and it is often a hustle to find it and switch it on. This has been a busy week for him, as he had to put long hours studying for his exams. Now, a week later, he notices that now the eGadgets related to his study have more plugs available. Moreover now the automated behavior is working better than a week ago, with the light intensity dimming to provide him enough light, without him needing to regulate it so often as in the first few days. Now, at night, the light adapts to the distance that the book is from the light source.

A few days later, he gets a new floor lamp in his room. It is also GAS-enabled. He puts it in the side of the room, between the bed and his study chair. A few times at first of times he prefers to turn it on as well, as he studies. Then one Saturday evening, his desk lamp bursts. He is happy that he has his floor lamp, until he would manage to go and buy another halogen-bulb at the shops. When he sits on his desk later on in the same desk to study, he is happy to find out that the floor lamp turns on, providing enough light to replace his broken desk lamp. It saves him having to stand up again and search the floor a few meters away from desk to find the switch, and he hasn't even needed to connect the floor lamp to the rest of his extrovert things!

Stella sets the rules

Stella has set up the study gadget world to light up her desk when she studies. The Gadgetworld she got ready made from the web. In the beginning she enjoyed the fact that the light would switch on automatically, and she does not have to reach for the switch when she needed light. However, she now has to switch off the light for every time she sits on the desk, even if it is not a book on the desk but a beer-glass. Sometimes she likes reading by her window, so she drags the chair there. However, the e-Gadget world she has is amazingly unperceptive, and keeps lighting the desk-lamp.

Stella is no wimp for technical stuff. She gets her e-Gadgetworld editor and tries to see how she can fix it. She adds a new plug "STUDY" for the desk and the lamp. For this plug she enters a rule: if there is weight on it, if the book is open and if the chair is near, then STUDY is set to TRUE. For the Lamp gadget, she selects the on/off plug and she adds the rule if STUDY = TRUE, the lamp is turned on. Things improve immediately. She tests by moving the chair at different locations and it still works. That evening, her neighbour comes in and sits

on the chair by the desk. The book is still open. Then Stella sits on her bed. They both prepare to drink some beer, when the desk lamp lights up.

Stella adds a clock

Stella has now gotten accustomed to her gadgetworld. She has started to depend on it, even anticipating the moment that the lights will go on or off. She goes out to buy some new gadgets. As she is a bit tired of studying long hours, and finds it hard to wake up in the morning, she decides to create another function, connecting her alarm clock to the light.

She goes to the local e-Gadget supplier and buys an e-Gadget enabled alarm. She inserts the alarm in the gadgetworld by using the editor. She drags the alarm description to the window with the other gadgets. The alarm has several plugs. She wants to tie the ALARM plug with the lamp. She decides for a real shake-up in the morning, so she adds a plug to the lamp, which is the 'FLASH' plug. For this she has to write a small programme segment:

"While (FLASH == 1) then repeat (Lamp.onOff)"

With this new plug in place, she only has to connect Clock.ALARM to Lamp.FLASH.

So now, when her alarm clock rings, her desk light flashes. So the next morning it really isn't pleasant but the flashing of the lamp wakes her up.

18. Appendix 2 - Cognitive Dimensions Evaluation

18.1. Assessment with respect to Cognitive Dimensions

Cognitive Dimensions (Green and Petre 1996) is a broad-brush technique for the evaluation of visual notations or interactive devices. It helps expose trade-offs that are made in the design of such notations with respect to the ability of humans to translate their intentions to programs and to manage and comprehend the programs they compose. As GWs are essentially programs that are composed in non-textual manner, it is helpful to consider this theoretically based technique to discuss some of the potential choices for e-Gadgets.

Below, we examine each of the 13 cognitive dimensions identified by Green, and discuss some of the choices that the e-Gadgets project can make.

Abstraction Gradient

Abstraction gradient discusses the minimum and maximum levels of abstraction supported by the notation. A programming language can be abstraction hating, tolerant, loving or hungry, depending on the degree to which the programmer needs to define their own abstractions.

End-users adapt easier to abstraction hating or tolerant environments (e.g., menu based systems, scripting languages). Abstraction hungry environments, e.g., Smalltalk, Java, require considerable expertise.

At this point several possibilities seem to be open for e-Gadgets. The conceptual

diagrams and the problem solving activity seemed to suggest an abstraction loving, or abstraction hungry environment difficult for end-users.

The graphical user interface appeared more like a browser of components that is abstraction tolerant. It seems that for non-sophisticated programmers, a graphical user interface, like the one shown in the evaluation session, would be the most appropriate. However, as was discussed in the review meeting, this should be enhanced with programming aids for the end-user-programmer.

Closeness of Mapping

Closeness of mapping refers to the tricks that the programmer has to learn, e.g., how to write a loop in a declarative language like Prolog, or how to create a list in a language like C, etc. The mapping is needed to translate concepts from the intended task domain to that of the programming environment.

E-gadgets support an architectural abstraction. Users might conceptualise their tasks in a variety of ways, stimulus-desired response, rules, sequences, and constraints between entities. In this way there will always be a gap. A graphical editor can bridge this gap, by allowing several different ways of expressing the user's goals. For example, the graphical editor discussed in the evaluation meeting supports the user describing their task in natural language. More ways of expressing the desired criteria for a configuration would do a great deal in minimising the mapping distance, and thus requiring less programming skills by the end-user.

Consistency

When some language has been learnt, how much more can be inferred? This is not clear at this stage of the project. Internal consistency is of course an important requirement (similar things done in similar ways), but external consistency with the expectations of users from the real life artifacts seems also to be an issue for e-Gadgets. However, the latter concerns the design of individual e-Gadgets, rather than the essential concepts that the project investigates and the nature of GAS.

Diffuseness / Terseness

A notation is terse, if it has many symbols that need to be learnt. At this stage e-Gadget appears to be diffuse, i.e., it has few conventions that need to be learnt. These concern e-gadgets, plugs and synapses (a very economic collection).

Error-proneness

Does the notation avoid careless mistakes? Without the concrete syntax for the notation, we cannot assess along this dimension. Care should be taken, in the way that detailed behaviours are described (as for example the rules used in the conceptual diagrams, or the comments that are presented in the browser to describe system behaviour).

Hard Mental Operations

This dimension concerns the extent to which the user needs auxiliary representations to support his thinking of what is happening or will happen, e.g., representing the logic on paper for writing a conditional. As far as simple synapses are concerned, "e-Gadgets" does not seem to impose too hard operations upon the user. However, it is unclear yet, how rules will be associated with synapses or how users will cope with complicated nested structures. This is an area where user testing would show which are acceptable levels of complexity for the structure of GWs.

Hidden Dependencies

Hidden dependencies concern the well-known effect to programmers of side effects. A change in one component has implications on the function of another, which are not visible in the system.

In the diagrams used for the discussion at a conceptual level, dependencies are directly visible. As e-Gadgets are modular and independent software entities, there are no other dependencies than the ones shown. However, in the graphical user interface, shown to the experts in the evaluation connections are not shown and their

rules are not shown. Some way of visualising and inspecting such connections needs to be added to the graphical user interface.

Premature Commitment

This dimension refers to the extent where the user is forced to make a decision, before relevant information is available. E.g., Pascal, where procedure declarations have to be written as placeholders, to render syntactically correct procedure calls. As a conceptual architecture E-Gadgets does not require premature commitment, though the extent to which this will be true in the eventual implementation is an issue for the future.

Progressive Evaluation

Can a partially complete program be executed to obtain feedback? This issue has not been addressed yet in e-Gadgets: How will the environment behave with incomplete configurations?

Role expressiveness

This dimension relates to the extent to which users can discern the relation between parts of the program and the whole. Object oriented languages present considerable difficulties to the user, as the structure of the whole is not the static class and object structure, but is effected at run time. On the contrary sequential programs, e.g., in Fortran or C, would show much better a link between static structure and the dynamic behavior. E-Gadgets is closer as a concept to object oriented languages. Future developments of the editor will need a way to illustrate to the user how the specification of the parts influences the dynamic behaviour of the GW (similar to debuggers in OO environments).

Secondary notation

Secondary notation concerns habits of programmers like naming conventions, indentation, that help the reader comprehend the structure of the program written. It seems premature to discuss this dimension with respect to the E-Gadgets project.

Viscosity: resistance to change

Viscosity concerns the resistance to change: the cost of making small changes. There are a couple of nuanced variations on this theme:

Repetition viscosity: a single goal-related operation on the information structure (one change 'in the head') requires an undue number of individual actions,

Knock-on viscosity: one change 'in the head' entails further actions to restore consistency.

Viscosity has a clear trade-off with abstraction gradient. The more that abstraction is afforded by a language the smallest the viscosity. Further it relates to the mapping of concepts from the users intentions to those supported by the computational environment.

"E-Gadgets" has a medium to high level of abstraction; so small knock on viscosity should be expected. (A positive issue).

However, this depends upon the closeness of the mapping. The closer the mapping, the smallest the repetition viscosity we should expect. E.g., if there is a task template describing a particular behaviout that the user wants to modify, then the user can access and modify that directly, rather than all the individual objects involved. Thus, low repetition viscosity requires a very mature development of the graphical interface that is probably not feasible in the context of a research project.

Visibility and juxtaposability

This concerns the extent to which it is possible to view concurrently and compare two different parts of the system influencing each other. It used to be that programmers printed their code to be able to do this. This issue has not yet been addressed by the e-Gadgets project, and does not appear to be so important for the validation of the concept itself.

Summary of views about the CD evaluation

E-Gadgets is on the right track with respect to the choice of abstraction levels. A considerable challenge for the usability by end-user is the need to minimize the mapping distance, which requires several different strategies to be combined, e.g., searching for GWs by task, by gadgets, defining constraints in respective behaviours.

Many other dimensions discussed above, concern more mature products and should not concern the research team.

18.2. Conclusions

The feedback to the project is captured as the following 4 tentative design principles:

- The Gadgetworld behavior should not surprise the user, i.e. automation or adaptation actions should be predictable (or at least justifiable).
- Simple tasks should remain simple even in an intelligent Gadgetworld. Intelligence should be applied to simplify complex tasks.
- End-users should be supported with at least as good tools as programmers have at their disposal, e.g., debuggers, object browsers, help, etc.
- Multiple means to define user intentions should be supported by the graphical editor, as the users tasks tend to be comprehended and expressed in a variety of ways.

These are recommendations for the future developments of the e-Gadgets concept. Currently "e-Gadgets" seems to be pitching at a level appropriate for end-users, but much depend on the quality of the tool support provided to them. At the following stages of the project, it remains to be verified through end-user testing of working prototypes, whether users are capable of forming GWs that will support their intentions.

References

Green,T.R.G., Petre, M. (1996). Usability analysis of visual programming environments: a cognitive dimensions framework. Journal of Visual Languages and Computing. J. Visual Languages and Computing, 7, 131-174.

19. Appendix 3: Evaluation at conferences DC-Tales and BCS-HCI

During the 2rd year of the Gadgets project's technology produced results sufficiently robust, so as to enable a hands-on demonstration at conferences. The project exhibited a hands-on demonstrator at the two conferences in 2003: DC-Tales, and BSC-HCI 2003 (Mavrommati et al, 2003c). This annex is part of the e-Gadgets project evaluation report, authored by Markopoulos and Mavrommati, in 2005 (see e-Gadgets project website).

19.1. Set-Up

A proof of concept demonstrator (Figure 1 in Appendix 3), has been created and was shown at the two events by the e-Gadget project members. The demonstration scenario below illustrates the steps for setting up a GadgetWorld.

First, the GadgetWorld editor, which is also an eGadget, is initiated. This editor runs on an iPAQ computer. Through wireless communication and by use of the GAS-OS middleware, the editor finds other eGadgets in the network vicinity and displays their names on the screen.

In this demonstration the Gadgetworld editor finds:

- A MATHMOS lamp converted to an eGadget (MATHMOS lamps resemble luminous bricks that illuminate in different colours depending on which side they are standing).
- An MP3 player, which for the purposes of the demonstration will be the WinAmp running on a laptop.
- An eCarpet, which is a small mat equipped with a pressure sensitive grid.

All the above objects have been made GAS compatible. Their digital part offers their capabilities through their plugs, that are listed on the editor display as a drop down list associated with each eGadget. In the demonstration these plugs were possible to link to each other and synapses were created so that several functions were demonstrated step by step. The demonstration showed the following:

- On the GadgetWorld editor the plugs offered by the eGadgets were shown. The MATHMOS eGadget had only one plug, the 'side' plug, which 'knew' on which side the lamp has been placed. The mp3 player has several plugs associated with it, i.e.: a pause, play, track number and music gender plugs.
- Through the GadgetWorld editor we create a "synapse" between the carpet and the MP3 player and between the MATHMOS and the MP3 player.
- We turn the MATHMOS on its side and, according to the user has set-up, it will play a different type of music (classic, rock, etc), or play/pause, stop, etc.
- Stepping at different areas of the mat causes the MP3 player to play, pause, the volume to change, etc according to what the user has set it up to do.

At both conferences a questionnaire form was used to collect expert opinions of the delegates regarding the concepts presented. Figure 2 of this Appendix includes the questionnaire items which were specifically chosen to target the concepts from an end-user perspective rather from a technological perspective and which were written in such a way as to invite delegates to be as critical as possible.

19.2. Participants

The Disappearing Computer Tales event (held in June 2003, Santorini, Greece,) was attended by delegates who are expert in various disciplines relating to the study of Ambient Intelligence (e.g., computer scientists, industrial designers, human-computer interaction experts). These are representatives of universities and industries working on research. The stand of the e-Gadgets project was visited by 30 delegates approximately. Ten (10) completed forms were received.

The British HCI (BSC-HCI) conference is a specialized event, the delegates (approximately 200) are experts in human computer interaction from both industry and academia. BSC-HCI 2003 included mostly researchers and fewer practitioners. The stand of the e-Gadgets project was visited by many delegates (approximately 50-70), of whom 29 completed the questionnaires.

Figure 1. The set-up of the e-Gadgets hands-on demonstrator shown in exhibitions (here in IST2003)

19.3. Results

In this section we combine answers from both events. Contrived answers are omitted, such as when delegates report as a nice thing about the eGadgets demo that the people giving it are kind or when they say "no bad things", as a response to the request to name a "bad thing about the demo". We omit answers we did not understand or are equivalent to "no comment". We also omit answers that do not help evaluate the concepts directly, but rather refer to the way the demonstration was delivered (E.g., "did not see the value of the carpet"). However, it is not always easy to make this distinction, as we believe most delegates describe the impression they got from the particular demonstration set up and sometimes found it difficult to refer directly to the concepts shown.

Remarks by users are rephrased to be understood out of context. The number indicated in parentheses after each comment describes how many delegates made a

particular remark. Comments have been regrouped, e.g., when a "bad thing" (answer to the last question) describes a possible obstacle to adoption, we cluster it with similar answers to the first question.

It has to be noted that we specifically invited negative comments in order to spot weak points (see the questionnaire, figure 2 of this appendix). The critical answers should be taken for what they are: *attitudes of experts towards a particular concept, rather than experimentally validated truths*. For each collection of remarks we add our own conclusions.

Possible obstacles to the end user adoption of e-Gadgets technology:

- Fear of technology, users may be reluctant to change their current ways of doing things (2).
- If it is not your own gadgetworld it is difficult to know what is happening.
- The concept is too not accessible (too futuristic, threatening and complex) (2)
- Complexity of setting up a system may prohibit users (13).
- Users would prefer to interact with objects directly rather than as a group.
- Gadgetworlds need only be configured rarely, so users will not be willing to learn to use a purpose specific system (2)
- People may find it difficult to guess what is possible and what benefits can be obtained (5).
- Gadgetworlds cause too much unintended invocation of functionality.
- A PDA is not a good device to run an interface on (5)
- The concept refers to the structure of the system rather than the tasks of the user.
- People may prefer to have things ready rather than do it themselves.
- Cost can be too high (2)
- Privacy and security fears.
- Level of abstraction of synapses too low.
- People will try it out of curiosity but will only make simple gadgetworlds.

This collection of comments can be expected particularly from experts in human computer interaction. Some delegates found that the operation would be too complex. Others (see below) found it very easy and simple. Clearly this is a judgement call and one that depends on the targeted user group. An empirical evaluation can settle this division of opinion.

The PDA was not welcomed as an interface in it's current form, but again this is something that empirical testing should resolve. PDA users might be less reticent about using them than those who do not. However, the e-Gadgets concepts are not tied to the PDA itself, but to the concept of editing tools with modalities.

The fact that something futuristic is considered threatening can be assumed to portray an attitude towards technology that may or may not align with the eventual end users' attitude. Clearly, we expect early adopters of such technologies to be technophiles and technologically versed (e.g., using mobile phones, PDA's, or PCs).

The remarks about lack of task orientation, people preferring ready-made solutions and understanding someone else's gadgetworld are serious research and interaction design challenges that call for empirical testing and further work to improve the deployment of e-Gadgets concepts.

Possible obstacles to professional designers adopting eGadgets technology

- eGadgets help create only boring "serious" scenarios.
- Designers may find eGadgets useful for professional applications.
- Safety critical side effects.
- The designers would rather design themselves rather than let the users design.
- Designers would need a different interface than the PDA (2).
- Inappropriate editor to enable design (2)

The fact that automation scenarios demonstrated were considered boring by one person, does not to our view reflect on the underlying concepts, the same as

programming languages may be used to create boring or exiting effects. <u>The issue of *side effects* is important and is one that needs to be studied in follow up research for providing interaction and technological solutions.</u>

Obstacles and motives for end user acceptance of agent technology

- End-user must be able to control the agent (7).
- Resistance to agent as a 'big brother' monitoring user (2)
- Benefits must be very clear and compelling for agents to be adopted (2)
- May help overcome handicaps of people.
- Will make operation of the system easier (2).
- Adoption of the agent is an issue of getting accustomed to it (2)
- ..."Agents don't work" "AI doesn't work"...(sic)
- Agents should recommend rather than take action.
- Agents will be acceptable provided they make correct inferences (4).
- Agents can correct omissions by users.
- Agents would work better if they could perceive users'emotions.
- People's behaviour may be irregular and therefore difficult to support with agents (2)
- Learning from people requires their identification, without any explicit action by the user. (This person assumed this is not possible).
- Agents are a good idea because they introduce adaptivity
- Agents can provide explanations.

The comments above show a lot of skepticism about agent technology and in some cases outright dismissal of a whole research field. This is we believe caused by flaws in currently deployed user adaptive systems (in one case a delegate described his/her annoyance from MS Word). We agree with the judgment that end-user control is important but is one that can be designed into the operation of the agent. Demonstrating the usefulness of the agent is also critical when we consider either the positive or the critical suggestions above.

Positive impressions

- Easy to create/modify gadgetworlds (11)
- Easy to learn, simple (4)
- Enjoyable, relaxing demo.
- The demo worked (5)
- Flexibility of gadgetworlds.
- Departs from windows style interaction.

Clearly some delegates were more convinced about the others about the ease of use and ease of learning of the system. We expect such divisions to exist also among the general population of intended users. The flexibility of Gadgetwolds, a major benefit of this technology, was remarked only by one delegate. This is perhaps due to the limited number of eGadgets shown during the two conferences (figure 1), which does not portray well the potential combinations of functionalities that a user can achieve.

Points for improvement

- Scaling up the number of eGadgets and synapses is an important future challenge (covers both technical and user interaction perspective) (2)
- Could be faster.
- Interface to editor was poor (4).
- Pen input was annoying.

Most answers to this question have been merged with the first question (obstacles and motivations for adoption). We note that indeed improving the underlying technology in efficiency and speed would be a reasonable next step after the first concept demonstrators have been built.

Questionnaire for expert feedback at DC-Tales and HCI

Please help us evaluate the e-Gadgets demo

We would like you to focus on the concepts put forward with this demo (rather than the rough state of the technology we present) and give us your own opinion for the items below:

The non-trained persons will not use ready Gadgetworlds or modify them to suit their needs because....

The designer will not find eGadgets (the objects, the concepts and the editors) an appropriate framework for constructing .....

Intelligent agents could be modifying Gadgetworlds to suit our pattern of use. Would this be welcomed by users or not? Please suggest your reasons why?

What was a nice thing about the demo you saw:

What was a bad thing about the demo you saw:

Figure 2: The above image shows the questionnaire used for feedback from conference participants.

20. Appendix 4: the iDorm user test

20.1. Introduction

With the iDorm evaluation (2004) we have not attempted an acceptance or rejection test of the concepts of the e-Gadgets project, rather, an account of the problems that users encounter with these concepts and their current state of realisation. However, we did try to test for an overall 'non acceptance' of the concept and to test whether potential users are not willing or able to understand make and edit Gadgetworlds.

The iDorm evaluation tried to answer the following questions:

- Can they understand the basic concepts with only a brief introduction.
- Can participants predict the behaviour of a gadgetworld from its architecture?
- Can they modify an existing gadgetworld to do something slightly different?
- Can they create the synapses to make a behaviour they wish?
- Do they feel restricted by the editor in making the behaviours they want?
- Do the feel this gadgetworld making is something they would not want to do?

The study was a combination of conventional short tests and a single test that took place overnight. The short tests aimed to gauge how potential users grasp the concepts and the overnight test aimed to get a relatively longer term and more realistic test of e-Gadgets when it is used in anger.

20.2. Participants

3 pairs of participants were recruited for the short test locally at the University of Essex. We had asked for technology-friendly ('technophile') participants, with some familiarity with computers; we invited diversity of subjects, e.g., subjects from all walks of life, e.g., students, librarians, shop assistants, as long as they would be

familiar with computing technology (at least users of PCs). It was required that participants would not be involved with the project itself.

On the day one non-computer science participant did not turn up and was replaced by a computer science PhD student, generally familiar with the concepts of Agents and Intelligence.

From the pre-test questionnaire it comes out that only one participant was over 35 years of age. Two were younger than 25 and the rest were between 26-35y.o. All participants except one were frequent users of personal computers, e-mail, SMS and mobile phones. This one participant was not a mobile phone user. They were a rather biased sample as they all hold a university degree and with the exception of two they were knowledgeable in computer science.

Participants worked in pairs during the short tests as will be explained below. A single participant stayed overnight: we had requested someone who would be pretty comfortable with computing to increase the chances of him having a comfortable stay (it was thought that someone with less chances of programming their own gadgetworld would probably have a less comfortable night).

Figure 1. The experimenter instructs one of the two participants how to execute the first scenario. Then she takes over to explain it to the second participant. The second participant looks over the situation.

20.3. Materials

The following e-Gadgets were made available for the user test:

1. **Occupancy**. A gadget that senses if the room is occupied (true) or not (false).
2. **LightLevel**. A gadget that measures the ambient light in the room.
3. **Chair**. A gadget that senses if someone is sitting on it (true) or not (false)
4. **Bed**. A gadget that senses if someone is on the bed (true) or not (false).
5. **Temperature**. A gadget that senses the room temperature.
6. **RoomLights**. A gadget that lets you switch room-lights on and off.
7. **DeskLight**. A gadget that lets you switch desk light on or off.
8. **BedLight**. A gadget that lets you switch bed-light on or off.
9. **Blinds**. This gadget tells you fully if blinds are fully open or fully shut (openclose = true if the blinds are shut) . This gadget also tells you which angle do the blind blades have, which can be in positions –0,1,2,3,4.
10. **MP3Player**. A gadget that plays music. It lets you start = or stop the music, set the volume and choose a genre of music.
11. **Clock**. A software gadget that tells the time or raises an alarm.

A list of the above e-Gadgets was given to participants plus a brief explanation of their nature. As the system was unstable on the day of the testing the Blinds and the BedLight eGadgets were not operational. This instability was due to our decision to include in the test very recently integrated agent functionality which meant that, due to insufficient prior testing of the latest version within the iDorm environment, in some cases some e-Gadgets would fail unpredictably. As some of the eGadgets would sometimes not work as expected (for example, the function to discover eGadgets would not find all the present ones, due to a bug related to network timing), we adapted the tasks given to participants on the fly. All substitutions though preserved the level of complexity and tried to gauge the extent to which users would understand concepts and gain control of the editor.

Participants were given the list of tasks that is shown in 'Material and Questionnaires' section, at the end of this appendix. Participants were given the Gadgetworld editor running on an iPaQ. A pre-task questionnaire was given to participants to fill in after basic introductions. A post-session questionnaire was given to them at the end of the session. Both questionnaires can be found at the end of this appendix, (Material and Questionnaires section).

20.4. Method for the short usability tests

The evaluation technique will be a combination of co-discovery learning (Van Kemp and van Gelderen, 1996) and peer tutoring (Hoÿsniemi et al 2003) with post-task interviews. The process followed was as follows:

- The experimenters introduced the subjects to the experiment, explained the set-up and the nature of their involvement and obtained informed consent for video-taping.
- Participants filled-in the pre-session questionnaire.
- The experimenters provided a brief oral explanation plus a minimal demonstration of the system, performing experimental task 0.
- One of the two participants would take control of the editor and was given advice how to operate it (see Figure 2). After the experimental task 1 was completed in this way, this participant explained the operation to the other participant who performed tasks 2-5 for the experiment. The table below shows how tasks were given to the participant pair:

Scenario	Participation structure
0	Experimenter shows
1	Experimenter instructs participant 1
	Participant 2 watches
2	Participant 1 instructs participant 2
3	Participant 2 operates, participant 1 discusses and observes
4	Participant 1 operates, participant 2 discusses and observers
continue alternating observer and performer in this manner	

Table 1: the sequence of tasks given to each pair of participants.

- A mini-structured interview was conducted at the end of the session on the basis of the questionnaire (see 'post session interview' at the end of this appendix). We asked participants to fill-in the questionnaire after our discussion. We decided to include a discussion before letting them fill-in the questionnaire in order to make sure questions were understood and to encourage them to bring out opinions in the open (the discussion was a facilitated as a mini-focus group). We asked them to record answers to the questions for efficiency (two users at a time) and so that they would use the opportunity to formulate their thoughts more succinctly and clearly than during the discussion (Figure 2).

- After the test participants were thanked and paid their appreciation.

Figure 2. Snapshots from the short evaluations. In all cases, one test participant is being instructed by the other test participant.

20.5. Overnight stay.

One participant stayed overnight after the tests. In the evening he was invited to play around with the gadgetworld, not as a programmer or an experimenter which tries to reach boundary conditions, but trying to get it to a state reasonable to live with (even if that was only for one night). On the day after in the morning we asked him his opinion and invited him to make changes to the Gadgetworld once more.

Experimenters had an open discussion with him trying to establish the following:

Whether he felt comfortable now with the e-Gadgets concepts?

1. Whether he felt comfortable with the e-Gadgets editor?
2. Is there some way in which the editor is restrictive?
3. Is there some way that the editor is unclear?
4. Whether he felt he could predict how things would go with the agent?
5. Whether he felt confused or surprised at any times because of the system behaviour?
6. Whether he was worried that an agent is adapting to your actions and taking initiative?
7. Whether he saw any potential problems with such an agent in his house in a few years time?
8. Then the participant was paid and thanked.

20.6. Results

For the first group of subjects the e-Gadgets technology worked well. But the function gradually deteriorated with time. Group2 could see only a few eGadgets while the blinds did not work. Group3 experienced major problems with e-Gadgets technology not working; mostly they were not able to see the gadgets and to activate the synapses.

The results of the evaluation were two-fold. On the one hand the short evaluation went very smoothly, with users easily accomplishing their tasks and grasping the underlying concepts, despite the occasional failing of the technology. On the other hand the overnight stay, where the iDorm was controlled by the agent showed that the technology is not yet robust enough for such a test. The user did not manage to control the system and had an unpleasant night in the iDorm.

Some general impressions of the experimenter are recorded below:

Subjects 1 and 2 were very enthusiastic. As they were working in the same laboratory they did have an expressly positive attitude towards the project. They did tune into the whole concept extremely fast and were very happy when things worked as they planned. Their facial expression and their words portrayed sincere enthusiasm. Their interest in computer science, potentially coloured their views. On the same token, their comments portrayed a good insight into some of the pitfalls of this technology. (e.g. vulnerability to hacker attacks, need for manual override, etc.). Subject1 would like more variety in control (more attributes in the synapses).

Subject 3 was a technical person, while subject 4 a law student. Both succeeded in their tasks and made all the gadgetworlds required. They seemed to enjoy the experience. Like subjects 1 and 2, they commented on the limited feedback given: i.e. it was difficult during the tasks to tell the state of the gadgetworld construction and the necessary follow up states. Subjects 3 and 4 commented that it seemed very logical and clear to distinguish 3 levels of components (gadgets, synapses and attributes). They kept noticing and being puzzled by message 'offline', which was not self-explanatory. Both subjects liked the idea of "messing about with their furniture".

Subjects 6 studies international relations and subject 5 (a substitute for a participant who did not turn up) was a computer scientist just becoming member of the Essex research team (but not related to the e-Gadgets project). They had a good positive

attitude. They got to try out the agent, but were unfortunate in that their most complex gadgetworld creation did not work (the system became unstable). *A critical observation here is that as soon as the agent was part of the system, it was not necessary any more for the end-user to think of what is an object and what is a synapse. They simply have to enact the behaviours they want and the agent would create the synapses for them: this seems to address the requirement for a task oriented language to communicate with the system and to simplify the whole ontology*. However subject 5 was worried about how much control should the agent have: "I don't really know how much control over it and if I cannot control it I would be afraid to use it. If I don't understand it I cannot control nor understand what it is doing". For this subject the problem of the agent learning was not the major problem: it is how much has the agent learned at any moment that was not clear and worrying. Subject 6 would like an *on-off switch* for the whole gadgetworld. *These subjects found the terminology used unnecessarily complex.*

In light of our earlier hesitance as to whether the whole concept would be understandable and comprehensible to the test users, the most compelling comments came from subjects 5 and 6, who said "it is very logical, one thing leads to another", and "we do it with our mobile phones today". The younger generation are much more adept with handheld technology than earlier generations and much more capable at handling an amount of technology that seems daunting at first sight. What was difficult for these subjects was to know what the 'boundaries'of the system were: there was nothing visible to show where the gadgetworld begins or ends.

Subject 7, who stayed overnight is a computer science post-graduate student. He had worked in the idorm in the summer, for devices such as blinds and MP3 player (but not on the e-Gadgets project). He experienced the 1 hour session (same as the other subjects) and could evaluate e-Gadgets concepts and technology then. Overnight the network failed and he could not experience much of the system anymore. The system got back to functioning overnight and as it did the lights

switched on (including the ceiling spot lights above the bed) so he woke up. He could not manually switch the lights off, so he had to cover his eyes with the pillow to sleep. He was happy that the blinds could manually close. <u>He felt a manual override is needed for all devices</u>. In the morning he attempted to work with the agent that had in the meantime been restored. He seems a calm and well-tempered character who took this technical failure very well.

He pointed out several usability bugs that are listed below:

- A gadgetworld becomes activated but when it is recalled you cannot see an overview of it.
- When you pause or stop the gadgetworld, it still continues to do the last thing it was doing, and then you need to manually override it (i.e. switch the light off, the MP3 player off, etc).
- It was frustrating: trying to get the system work, while there were no error messages at all, no feedback on what was going wrong. As he did not know what was wrong he had to restart each time.
- Had to restart it over and over again, and played a bit with the system before the demonstrator went down.
- Frustrating that there was no explanation of what has happened, once you completed an action; also frustrating that there was no help to guide the user through.
- The interface should use tangible interaction and multi-modality.
- The editor is a quick and easy way of setting things. Focus of editor seems limited (only 4-5 devices). How to select and reselect a gadget is not clear.
- When the subject got back after dinner out, he attempted to set up a gadgetworld in the idorm; due to a network problem he could not go ahead with that. When he attempted to set up a gadgetworld, there was no feedback, e.g. as to whether an action had been successfully achieved. As he put it "Information coming up is not relating to the action you did, so you cannot pin it (=failure) down to what you have done".

- Interaction with the agent was confusing. When the agent was activated it was not clear which device has dominance/precedence?

- Confusing pop up boxes for activating agent separately. If you activate agent control in one, it should also appear in the other.

- Overnight he had to turn off the PC because of the noise. System is not dependable due to network being undependable.

- The agent can see only one synapse. A second synapse would not work with the agent between two objects. A synapse between the chair and Mp3 player was working adaptively (with the agent), but only the on/off off the player could get influenced by the on/off chair. When volume or gender was affected, the subjects felt this should adapt too, but it did not (as the agent was not based on a preexisting synapse between gender and on/off chair, -but was working only between on/off Mp3player and on/off chair).

- Programming is oozing through the interface: The possible values for music gender should not be 1, 2, 3, 4, but certain types (i.e., rock, classic, Irish folk, jazz, greek folk, drum'n'base, etc).

- No connections can be made sometimes

- The list of eGadgets was constantly changing in the Editor. Sometimes there are no egadgets at all, or just one or two, then after refreshing it can see almost all of them.

- Lights could not work, and could not get connected (synapse was not established).

We examine the findings in more detail in the next paragraphs, by examining each question of the post-task interview.

Did you like using the system?

To the question of whether they enjoyed using the system, 4 participants selected the answer "very much", two the answer "so and so", while the person staying overnight commented that he didn't manage to use the system. We note that this is consistent

with our impression of them. Participants were boyant and exited when trying things out and were cheering when they achieved what they set out to do. We also note that they succeeded in their tasks with an ease that surprised us as well.

What did you like about the system?

To this question participants reported that they liked the simplicity of the interface, the ability to connect and control devices, the context awareness supported by the sensors. One participants found it a bit like a game and suggested that "only imagination puts a limit to its possibilities". One person anticipated gains in efficiency, i.e., not wasting time with controlling the environment or the music.

What did you not like about the system?

The overnight user was the most critical (understandably). The reliability of the system was the major flaw: "Once the network went down, I was powerless to control some devices manually". Clearly this is a reflection of the experimental nature of the demonstrator but also a useful reminder: when people live with technology, there will not always be a technical support at hand to repair problems. A graceful degradation of performance, similar to safety critical systems becomes an important requirement.

The short term users were more puzzled about the observability of the system and the interface: It is not clear what is possible, what is the range of possibilities for the user at any moment. The delay between doing something and observing the result was also found too much. One person did not enjoy the PDA interface for the gadgetworld editor. We note that there was a running version of the editor on the fridge display, but we avoided using it because it was slightly more complex in its operation than the iPaQ model. One usability problem spotted on the editor was the need to always select a synapse before activating it. Participants expected a default selection to have taken place.

Do you feel you understand how objects are interconnected?

All but one participant said they did. The one who didn't still enjoyed using the system and was very effective in achieving her tasks. One person felt that she needed more practice to grasp the concept. We note that she was one of the least positive users about the whole experience.

Do you feel you can achieve the functionality you want with such connections?

One person noted that little information is available to the user as to whether a synapse has been created effectively. Three users expressed the need to control more aspects of the behaviour of the eGadgets than those already implemented.

Was the system adapting or changing its behaviours in ways that surprised you?

This question was relevant to only the last 2 participants of the short tests and the overnight participant who all tried using the agent. At the short test, they got the general idea and seemed to understand how it worked. During the longer test (the overnight stay), the agent appeared inconsistent and unpredictable to the subject. We consider this a natural outcome of the fact that the agent was still too fresh out of development and needs more refinement to improve its robustness.

Controlling everyday objects.

All but one user felt they would use such a system to control their every day devices. The one who didn't finds it too complex a way of doing things and is worried about potential costs. She also would like a larger touch screen interface (she didn't know it already exists). The reasons for using it were efficiency (doing things easier) and the potential cost savings from automation (thinking of domotica equipment).

Controlling everyday objects with the agent.

This question also was answered only by the 3 subjects who experienced the agent. Despite the 'teething problems'of this technology that they experienced they were all affirmative in their answers.

General comments and suggestions.

Technology should not take over. Fewer steps should be needed for setting up a synapse and perhaps some hints should be offered on the way. One user seemed to suggest the need for a 'wizard' type interface for setting up connections. One (computer savvy) participant spotted the potential problems of network security, which are beyond the focus of this study.

Not enough explanation is available for deactivating a gadgetworld once it is activated. The terminology on the user interface seems too technical. A visual representation for synapses was also requested.

Some subjects expressed an appreciation for having degrees of transparency into the workings of the Ubiquitous computing system. When the users architect their gadgetworld they get to know it's structure, so we are talking about a transparent approach (a 'white box' as opposed to a 'black box'). When the users train an agent without knowing the architecture, then we have a black box approach. In this project the two ideas have co-existed, with users gradually getting to know more and more about the white box, (in the form of the system architecture). For this to happen, the agents have to be easier and more robust than the white box solution, so that the beginner-level users can start with an agent. (Subjects 5 and 6 seem to point to that direction). Agents can return for the very advanced users when more complex patterns of behavior need to be defined, that are too detailed to describe in architectural terms: in such cases, rules have to be associated with synapses, which means that the simplicity experienced by the users will be lost.

Overall the impression by the users was very positive with some obvious shortcomings noted. However, we note that the sample of users was not representative of the user population we had aimed for: people trained in computer science have an extreme talent for comprehending and manipulating abstractions. So while it is encouraging that these participants did so well, we cannot expect that for the majority of people similar results will hold. It has to be noted though, that even for two of the participants who were not computer scientists, the test went quite well, which is a very encouraging result.

20.7. General comments regarding the evaluation

On retrospect the evaluation could have been better organised. The selection of the subjects should have been a bit broader and the experiment should have limited its scope to parts of the project that were more robust. We decided to take the risk and user test a piece of software of which the version on test was released a day before the test, so that it integrated the agent functionality; nevertheless this version was not sufficiently tested on site to foresee possible problems during the test sessions. Nevertheless, if we had taken the step to freeze it, as initially planned, the version for the user-test would not have included the agent, and although there might have been fewer problems, that were judged as shortcoming of the evaluation in the iDorm.

The verbal protocol (peer tutoring) did not work as intended. The small screen of the iPaQ did not allow the experimenter to know what was going on unless he would come very close to the subjects (in which case he could easily be influencing their performance). In this way, most of the time a post-task interview was conducted instead of the peer tutoring protocol. Also, subjects tended to point too much on the screen which made it difficult to get a clear record of the verbalisation itself.

Subjects 3 and 4 were very quiet during the evaluation; subject 6 was very vocal and very capable of carrying out her tasks.

20.8. Conclusions of the evaluation

We entered this evaluation very aware that the technology under test was not very mature and robust and that much development work needs to be done before eGadgets concepts would be possible to convey to end-users. The remarks by experts in the DC Tales event and the HCI conference seem to raise many of the possible objections to and limitations of such technology. Especially important was the skeptisicm towards agent technology. The opinions of experts regarding the understandability of e-Gadget concepts, the ease of use and learnability of the system were almost perfectly divided among the experts and in many ways seemed to come from a pre-disposition to this technology.

The iDorm test, seems to provide conclusive evidence towards the viability of the concept and at least rest the fears of putting too complex tasks on the shoulders of the end-user. The system was clearly understandable and users could do their tasks despite some technical failures of the system.

While the iDorm evaluation seems to be a clear success for the project, there are some limitations: the set of users examined is rather small and homogeneous. This clearly prevents us from generalizing our positive findings for the general public, but shows the clear potential of the concepts presented.

Some clear pointers for future research seem to emerge from this evaluation:

- Agents can potentially simplify the construction of gadgetworlds (by eliminating the need for the user to understand the ontology of e-Gadgets). It seems that agents are a complementary modality approach to making end-users into gadgetworld architects. It is likely that different types of users will enjoy constructing and others might enjoy training an agent. Some important and large

research questions remain. Designing interaction with the agent to foster trust and to enable manual override is as important as the agent working robustly and correctly.

- The user interface in the current implementation of the gadgetworld editor on the iPaQ needs to be iteratively designed to eliminate the usability faults.

- As soon as users get a few gadgets with a few attributes, they want more gadgets and to control as many facets of their behaviour as possible. It is not clear what the technical and usability ramifications of such a scaled up use of e-Gadgets will be, but it seems to be what the users want.

- Users need to be able to choose how much transparency of the system they want to have. Being able to understand, in a first level, how the Ubiquitous computing environment worked was appreciated. The study seems to point to a direction of adopting varied levels of transparency, (rather than a black box approach). When the users create their gadgetworld they get to know it's structure, so a transparent approach (white-box) is adopted by the project (as opposed to a 'black box'). Agents are needed for two very different groups: a) to help novice users to initially familiarize themselves with the system b) for the very advanced users when more complex patterns of behavior need to be defined, that are too detailed to describe in architectural terms.

- The users should enabled to choose how much active an agent should be in designing their environment. It can be argued that after they are socialized to accept the agent (accepting it as trustworthy), they would see the advantages of training the agent instead of programming. An exploration of this design space should be the topic of further research, e.g., providing solutions for the user to control the gradual engagement of an agent; also understanding the segmentation of the target user population with respect to their readiness to design and to train agents. For the latter part, broader and longer term user studies are needed where a larger set of e-Gadgets is available and different ways of interacting with agents are supported.

- The iDorm evaluation addressed only end-users and not professional designers. Clearly an interface for professional use was beyond the scope of the current

project, but seems a necessary component towards their eventual deployment. The assertions made by delegates at the two conferences where e-Gadgets has been demonstrated remain to be tested in an empirical evaluation study.

20.9. References

Kemp, J. A. M. and van Gelderen, T. (1996). Co-discovery exploration: an informal method for iterative design of consumer products, in Usability Evaluation in Industry (Eds, Jordan, P. W., Thomas, B., Weerdmeester, B. A. and McClelland, I.) Taylor and Francis, London, p139 - 146.

Höysniemi, J., Hämäläinen, P., and Turkki, L. (2003). Using Peer Tutoring in Evaluating the Usability of a Physically Interactive Computer Game with Children. Interacting with Computers, Vol. 15/2, May 2003: Special Issue: on Interaction design and children. pp. 203-225.

20.10.   Material and Questionnaires

List of scenarios for the iDorm user test

The scenarios used in the evaluation test were adapted on the day, to make use of eGadgets that we knew were working robustly at the time (rather than the complete list of eGadgets mentioned in the report)

Scenario 0: musical alarm

a.   Discover gadgets
b.   Create synapses between eClock and eMP3Player
c.   Check how it works
d.   Reset the system

Scenario 1: Chair light

Make the desklamp switch on if someone is sitting on the chair

a.   Drag eChair and eDeskLamp to the drawing area
b.   Select the two eGadgets so that they are hilited
c.   Press the "set synapses" on the synapse info box
d.   Select a cell of this table for the synapse you want to make
e.   Click on "plus sign" at the info box to expand synapse description
f.   Select a synapse and press "Synapse Properties"
g.   Add a row
h.   Use pull down menus to fix the mapping, so that the light will go on when the chair senses someone sitting on it
i.   Do the same for switching the light off when getting up from the chair
j.   Reset the system

Scenario 2: musical room

Use the eOccupancy gadget and the eMP3Player gadget to make the music start playing when the room is occupied and stop when it is empty.

Scenario 3: Study

a.   Drag the Bed and the Chair gadgets inside the drawing area.
b.   Let sitting on the bed set the genre of music for the MP3Player to 1
c.   Let sitting on the chair set the genre of music for the MP3Player to 4

Tip: From the info box, you can select a synapse and click on "Synapse properties". You can use Add Row to see the details of a synapse and use the pull down menus to make the mappings you want.

Scenario 4: Study

a.   Create synapses between the eDeskLamp and the eChair
b.   Activate an agent by selecting "Active Agent" from the menu bar of the eChair
c.   Click at "Device Info" and "Rulebase" within the Agents' Display Panel and start the agent by clicking at the green play button
d.   You should be able to see the eDeskLamp and the eChair in the "Display-Devices" panel
e.   Sit on the eChair and switch on the eDeskLamp
f.   Now stand up and switch off the eDeskLamp
g.   When you sit down again you should see that the agent has learnt your actions and switches on the eDeskLamp automatically
h.   Repeat and change your actions as you wish and see if the agent has learnt your preferences

Scenario 5: Leisure

a.   Create synapses between the:-
        i.   eDeskLamp and eChair
        ii.   eBed and eDeskLamp
        iii.   eBed and eBedLight
        iv.   eBed and eMP3Player
b.   Activate an agent by selecting "Active Agent" from the menu bar of the eBed

c.   Click at "Device Info" and "Rulebase" within the Agents' Display Panel and start the agent by clicking at the green play button
d.   You should be able to see the eGadgets in the "Display-Devices" panel
e.   Sit on the eChair and switch on the eDeskLamp
f.   Now stand up and switch off the eDeskLamp
g.   Sit on the eBed and switch on the eBedLamp and activate the eMP3Player
h.   Now stand up and switch off the eBedLamp and the eMP3Player
i.   When you sit down again on the eChair you should see that the agent has learnt your actions and switches on the eDeskLamp automatically
j.   The agent should have learnt also that as soon as you sit on the eBed, it should switch on the eBedLamp and eMP3Player
k.   Repeat and changed your actions as you wish and see if the agent has learnt your preferences

Scenario 6: Overnight

Create synapses between the

a.   eDeskLamp and eChair

b.   eBed and eDeskLamp

c.   eBed and eBedLight

d.   eBed and eMP3Player

e.   eBed and eBlind

f.   Activate an agent by selecting "Active Agent" from the menu bar of the eBed

g.   Click at "Device Info" and "Rulebase" within the Agents' Display Panel and start the agent by clicking at the green play button

h.   You should be able to see the eGadgets in the "Display-Devices" panel

i.   Sit on the eChair and switch on the eDeskLamp

j.   Now stand up and switch off the eDeskLamp

k.   Sit on the eBed and switch on the eBedLamp and activate the eMP3Player and close the eBlind

l.   Now stand up and switch off the eBedLamp and the eMP3Player and open the eBlind

m.   When you sit down again on the eChair you should see that the agent has learnt your actions and switches on the eDeskLamp automatically

n.   The agent should have learnt also that as soon as you sit on the eBed, it should switch on the eBedLamp and eMP3Player and close the eBlind

o.   Repeat and changed your actions as you wish and see if the agent has learnt your preferences

Pre session questionnaire

1. What is your age?
   O 18-25
   O 26-35
   O 35 and older

2. Do you use frequently any of the following (cross more than one if appropriate)
   O Computer
   O E-mail
   O Mobile Phone (GSM)
   O SMS service (Short Message Service)

3. Do you use frequently any of the following (cross more than one if appropriate)
   O Computer
   O E-mail
   O Mobile Phone (GSM)
   O SMS service (Short Message Service)

4. What is your highest acadmic degree?
   O high school
   O university
   O other (please describe)

5. How do you describe your expertise with computer programming?
   O None or too little
   O Understand the concepts
   O Knowledgeable but rusty
   O Fluent

Post-Session Structured Interview

1. Did you like using the system?

o Not at all   o A little   o So and So   o Very Much   o Extremely

2. What did you like about the system?

_____
_____
_____

3. What did you not like about the system?

_____
_____
_____

4. Do you feel you understand the editor and how to make synapses well now?

_____
_____
_____

5. Did you feel restricted by the editor at any point?

_____
_____
_____

6. Was the system adapting or changing its behaviours in ways that surprised you?

_____
_____

7. If every day objects were controllable in this way, would you see yourself arranging their behaviours with the scheme shown to you (using an editor, synapses, etc..)

o Yes   o No
Please say why

_____
_____

If every day objects were controllable in this way, would you see yourself arranging their behaviours with the help of the agent?

o Yes   o No
Please say why

_____
_____

8. How would the adaptation need to be improved to suit your preferences?

_____
_____
_____

21. Appendix 5 – Citations

Papers from the research work are shown below, in text boxes, while their corresponding cross references are listed below them.

---

1. Mavrommati, I., Darzentas, J. (2007). End User Tools for Ambient Intelligence Environments: an overview. In Human-Computer Interaction, Part II (HCII 2007), LNCS 4551, pp. 864-872, Springer. Available at: http://www.springerlink.com/content/tr246k71112041mh/fulltext.pdf

---

    1.1.   Davidyuk, O., Georgantas, N., Issarny, V., Riekki, J. (2010). *MEDUSA: Middleware for End-User Composition of Ubiquitous Applications*, Mastrogiovanni, F., Chong, N.Y. (Eds.), Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives, IGI Global. Available at: http://www.mediateam.oulu.fi/publications/pdf/1288.pdf

    1.2.   Davidyuk, O., Sanchez Milara, I., Riekki, J. (2010). *CADEAU: Supporting Autonomic and User-Controlled Application Composition in Ubiquitous Environments*, Pervasive Computing and Communications Design and Deployment: Technologies, Trends, and Applications, IGI Global (Ed.). At: http://hal.inria.fr/docs/00/50/91/08/PDF/davidyuk_et_al_cadeau_chapter_2010.pdf

---

2. Kameas, A., Mavrommati, I. (2005). Extrovert Gadgets. Configuring the e-Gadgets, Communication of the ACM (CACM), special issue section on "The Disappearing Computer", ACM, vol. 48, no. 3, p.69.

    2.1.   Hyeonsang, E. *Topics in Distributed Systems - Information Protection & Use and Performance Engineering*, Department of Computer Science & Engineering, Seoul National University.

    2.2.   Obrenovic, Z., Nack, F.M., Hardman, L. (2007). *Information Systems Designing interactive ambient multimedia applications: requirements and implementation challenges*, Centrum voor Wiskundeen Informatica , Report INS-E0703.

    2.3.   Caire, P. (2007). *Conviviality for Ambient Intelligence*, In: Olivier, P., Kray, C. (eds.) Proceedings of Artificial Societies for Ambient Intelligence, Artificial Intelligence and Simulation of Behaviour (AISB 2007), pp. 14-19. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.4987&rep=rep1&type=pdf

    2.4.   Caire, P., van der Torre, L. (2009). *Convivial Ambient Technologies: Requirements, Ontology and Design*, The Computer Journal Advance Access, Oxford University Press, British Computer Society. Available at: http://icr.uni.lu/pubs/cai09c.pdf

2.5.   Urmetzer, F., Hoyer, V., Rivera, I., Aschenbrenner, N., Lizcano, D. (2010). *State of the art in gadgets, semantics, visual design, SWS and Catalogs*, Deliverable D2.1.2, FAST (*FAST AND ADVANCED STORYBOARD TOOLS), FP7-ICT-2007-1-216048, http://fast.morfeo-project.eu*. Available at: http://files.morfeo-project.org/fast/public/M24/D2.1.2_StateOfTheArt_v1.pdf

3.   Kameas, A., Mavrommati, I., Markopoulos, P. (2005). *Ambient Intelligence: the evolution of technology, communication and cognition towards the future of human-computer interaction*. IOS Press, 2005 Riva, G., Vatalaro, F., Davide, F. and Alcaniz, M.: (Eds). Emerging Communication series. IOS press. (Mavrommati, I. - chapter 10, titled "*Computing in tangible: using artifacts as components of Ambient Intelligent Environments*").

3.1.   Azodolmolky, S., Dimakis, N., Mylonakis, V., Souretis, G. (2005). *Middleware for In-door Ambient Intelligence: The PolyOmaton System*, Proc. of 2nd NGNM workshop, Networking.

3.2.   Brey, P. (2005). *Freedom and Privacy in Ambient Intelligence*, Journal of Ethics and Information Technology, vol. 7, no. 3, Springer, pp. 157-166.

3.3.   Kovács, K., Kopácsi, S. (2006). *Some aspects of ambient intelligence*, Acta Polytechnica Hungarica, Budapest, Hungary, 3(1), pp. 35–60.

3.4.   Arroyo, R.F., Garrido, J.L., Gea, M., Haya, P.A. (2006). *A Design Model Applied to Development of AmI Systems*, International Conference on Ubiquitous Computing ICUC06.

3.5.   Miao, Z., Yuan B., Yu M. (2006). *A Pervasive Multimodal Tele-Home Healthcare System*, Journal of Universal Computer Science, vol. 12, no. 1, pp. 99-114.

3.6.   Correal, R., Jardón, A., Martínez, S., Cabas, R., Giménez, A., Balaguer, C. (2006). *Human-Robot Coexistence in Robot-Aided Apartment*, 23rd International Symposium on Automation and Robotics in Construction (ISARC 2006), Tokyo, Japan.

3.7.   Cepa, V. (2005). *Product-Line Development for Mobile Device Applications with Attribute Supported Containers*, Dissertation, Fachbereich Informatik, Technische Universitaet Darmstadt.

3.8.   Crutzen, C.K.M. (2005). *Intelligent Ambience between Heaven and Hell: A Salvation?*, Journal of Information Communication and Ethics in Society, vol. 3, no. 4, Troubadour Publishing, Great Britain, ISSN 1477-669X, pp. 219-232.

4.   Mavrommati, I., Kameas, A., Markopoulos, P. (2004). *An Editing Tool That Manages Device Associations in an in-Home Environment*, Personal and Ubiquitous Computing 8, p.p.255-263, Springer.

4.1.   Gross, T., Marquardt, N. (2007). *CollaborationBus: An Editor for the Easy Configuration of Ubiquitous Computing Environments*, *pdp*, pp. 307-314, 15th Euromicro International Conference on Parallel, Distributed and Network-Based

Processing (PDP'07). Available at: http://www.nicolaimarquardt.com/research-documents/CollaborationBus_TechnicalReport.pdf

4.2. Mugellini, E., Rubegni, E., Gerardi, S., Khaled, O.A. (2007). *Using personal objects as tangible interfaces for memory recollection and sharing*, Proceedings of the 1st international conference on Tangible and embedded interaction, Baton Rouge, Louisiana.

4.3. Balka, E., Wagner, I., Jensen, C.B. (2005). *Reconfiguring critical computing in an era of configurability*, Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility, Aarhus, Denmark. Available at: http://www.informatik.uni-trier.de/~ley/db/conf/critical/critical2005.html

4.4. Goumopoulos, C., Christopoulou, E., Drossos, N. (2004). *The PLANTS System: Enabling Mixed Societies of Communicating Plants and Artifacts*, in P. Markopoulos et al. (Eds.), EUSAI 2004, LNCS 3295, Springer-Verlag, pp. 184–195.

4.5. Schirmer, M., Gross, T. (2008). *CollaborationBus Aqua: Finden und Bearbeiten ubiquitärer Umgebungskonfigurationen*, Fakultät Medien, Bauhaus-Universität Weimar, Mensch & Computer 2008.

5. Markopoulos, P., Mavrommati, I., Kameas, A. (2004). *End-User Configuration of Ambient Intelligence Environments: Feasibility from a User Perspective*, EUSAI, European Symposium on Ambient Intelligence, Eindhoven, published in: Ambient Intelligence, ISBN 3-540-23721-6, Springer Lecture Notes on Computer Science (LNCS3295), pp. 243-254.

5.1. Mosveen, C.H., Brustad, A. (2005). *UbiCollab: Evaluation and requirements re-engineering*, TDT4735 Systemutvikling, fordypning.

5.2. Coutaz, J. (2007). *Meta-User Interfaces for Ambient Spaces*, in book: Task Models and Diagrams for Users Interface Design, Lecture Notes in Computer Sciences (LNCS), Springer, Vol. 4385/2007, pp. 1-15.

5.3. Roudaut, A., Coutaz, J. (2006). *Méta-IHM ou comment contrôler l'espace interactif ambient*, Proc. Ubimob06.

6. Mavrommati, I., Kameas A. (2003). *The evolution of objects into Hyper-objects, will it be mostly harmless?*, Personal and Ubiquitous Computing ACM, Springer-Verlag, vol. 7, 3-4, pp. 176–181.

6.1. Ogata, Y. (2004). *Building Highly Interoperable Home-Computing Middleware Based on REST Architectural Style*, Master's thesis, Waseda University.

6.2. Atia, A., Takahashi, S., Tanaka, J. *Smart Gesture Sticker: Smart Hand Gestures Profiles for Daily Objects Interaction*. tsukuba.ac.jp - iplab.cs.tsukuba.ac.jp

7. Kameas, A., Bellis, S., Mavrommati, I., Delanay, D., Colley, M., Pounds Cornish, A. (2003). *An Architecture that Treats Everyday Objects as Communicating Tangible Components*, IEEE international conference on Pervasive Computing and Communications (PERCOM2003), Texas, Forth Worth.

7.1. Pellegrino, P., Bonino, D., Corno, F. (2006). *Domotic house gateway*, Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France.

7.2. Mottola, L., Murphy, A.L., Picco, G.P. (2006). *Pervasive games in a mote-enabled virtual world using tuple space middleware*, Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, Singapore.

7.3. Goumopoulos, C., Christopoulou, E., Drossos, N. (2004). *The PLANTS System: Enabling Mixed Societies of Communicating Plants and Artifacts*, in P. Markopoulos et al. (Eds.), EUSAI 2004, LNCS 3295, Springer-Verlag, pp. 184–195.

7.4. Liu, R., Yang, H., Wang, Y., Pan, W. (2004). *An evolutionary system development approach in a pervasive computing environment*, 2004 IEEE International Conference on Cyberworlds.

7.5. Ambient Intelligence: Second European Symposium, EUSAI 2004, Eindhoven, Netherlands. Panos Markopoulos Berry Eggen Emile Aarts James L. Growley (Eds.) Ambient Intelligence. Lecture Notes in Computer Science 3295.

7.6. Gritti, M., Broxvall, M., Saffiotti, A. (2007). *Reactive self-configuration of an ecology of robots*, ICRA-07 Workshop on Network Robot Systems, IEEE International Conference on Robotics and Automation, Rome, Italy.

7.7. Roj, M. (2005). *Smart Artifacts as a Key Component of Pervasive Games*. Proceedings of the 2nd International Workshop on Pervasive Games (PerGames 2005).

7.8. Ronai, M.A., Fodor, K., Biczok, G., Turanyi, Z., Valko, A. (2005). *MAIPAN: middleware for application interconnection in personal area networks*, 2nd International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005), ISBN 0-7695-2375-7.

7.9. Landini, E. (2004). *Un sistema di comunicazione wireless per l'integrazione di robot di servizio in architetture domotiche*, Thesis, University of Parma.

7.10. Zimmer, T.H. (2007). *Verbesserung der Kontexterkennung in Ubiquitären Informationsumgebungen*, PhD Dissertation, Institut für Betriebssysteme und Rechnerverbund, TU Braunschweig.

8. Mavrommati, I., Kameas, A. (2003). End user programming tools in ubiquitous computing applications, In Stephanidis C. (Ed.), Proceedings of 10th International Conference on Human-Computer Interaction (pp. 864-872). London, UK: Lawrence Erlbaum Associates.

8.1. Rullo, A., Marti, P., Grönvall, E., Pollini, A. (2006). *End-user composition and re-use of technologies in the Neonatal Intensive Care Unit*, Proceedings of the Pervasive Healthcare.

8.2. Pollini, A., Grönvall, E., Marti, P., Rullo, A. *Constructing assemblies in the health care domain: two case studies*, hcilab.uniud.it

8.3. Davidyuk, O., Georgantas, N., Issarny, V., Riekki, J. (2010). *MEDUSA: Middleware for End-User Composition of Ubiquitous Applications*, Mastrogiovanni, F., Chong, N.Y. (Eds.), Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives, IGI Global. Available at: http://www.mediateam.oulu.fi/publications/pdf/1288.pdf

9. Mavrommati, I. (2002). *E-Gadgets case description*. Doors of Perception 7 @ flow, Amsterdam.

9.1. Crutzen, C.K.M. (2006). *Invisibility and the Meaning of Ambient Intelligence*, International Review of Information Ethics (IRIE), ISSN 1614-1687.

9.2. Crutzen, C.K.M. (2005). *Intelligent Ambience between Heaven and Hell: A Salvation?*, Journal of Information Communication and Ethics in Society, vol. 3, no. 4, Troubadour Publishing, Great Britain, ISSN 1477-669X, pp. 219-232.

9.3. Crutzen, C.K.M., Hein, H.W. (2007). *Dekonstruktion und Konstruktion: Beiträge zu einer Theorie der Informatik*.

10. Kameas, A., Mavrommati, I., Ringas, D., Wason, P. (2002). *eComP: an Architecture that Supports P2P Networking Among Ubiquitous Computing Devices*, 2nd IEEE international conference on Peer to Peer Computing (P2P 2002), Linkoping, Sweden.

10.1. Kurmanowytsch, R. (2004). *Omnix: An Open Peer-to-Peer Middleware Framework. Engineering Topology- and Device-Independent Peer-to-Peer Systems*, Ph.D. Thesis, Faculty of Informatics, Technical University Wien. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.378&rep=rep1&type=pdf

10.2. Gross, T., Paul-Stueve, T., Palakarska, T. (2007). *SensBution: A Rule-Based Peer-to-Peer Approach for Sensor-Based Infrastructures*, 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007), pp. 333-340. Available at: http://www.computer.org/portal/web/csdl/doi/10.1109/EUROMICRO.2007.54

11. Mavrommati, I., Kameas, A (2007). Crisis rooms are ambient intelligence digital territories - Universal Access in Human-Computer, – Springer

    11.1. Étude de comportements coopératifs pour l'intelligence ambiante: application à la gestion de crises. S Suils… - 2009 –e-archivo.uc3m.es

12. Calemis, I., Mavrommati I. (2009). Preliminary requirements and approach for Tools that configure pervasive awareness applications: the ASTRA case

    12.1. Domingo, DR., Divitini, M. Development and integration of an awareness applications manager into ASTRA from perseum.com

13. Overall contribution to related research (in contributing to Digital Territories study, in assisting multidisciplinary design research, in the e-Gadgets research, and in considering the End User as part of the system) has been also respectively acknowledged by:

    13.1. Daskala, B., Maghiros, I. (2007). Digital Territories, towards the protection of public and private space in a digital and Ambient Intelligence Environment, JRC, no: EUR 22765 EN.

    13.2. Wendy Mackay: The Interactive Thread: Exploring Methods for Multi-disciplinary Design Designing Interactive Systems - DIS 2004 , pp. 103-112, 2004 (acknowledgments).

    13.3. Hagras H., Callaghan V., Clarke G., Colley M., et al. (2002). Chapter 2, Incremental Synchronous Learning for Embedded Agents Operating in Ubiquitous Computing Environments. In Soft Computing Agents, Ed. V. Loia, IOS Press.

    13.4. Demazeau, Y. (2003). Créativité Emergente Centrée Utilisateur (keynote), 11ème Journées Francophones sur les Systèmes Multi-Agents, pp. 31-36, Hermès, Hammamet.

    13.5. Several media programmes and interviews (such as TVs NET 14/10/2003, ET1 2/2004, Euronews 29/10/2003, and newspaper articles Ta Nea, e-Typos, etc)

22. Appendix 6: an example EUD scenario in a ubicomp home

A scenario on how Ubicomp Systems that support EUD could be used by End Users, to create several applications for their environment.

Let's assume the following example scenario, of end user development of an application within the ubiquitous home.

Helen is the 82 y.o., the grandmother of Dimitris and Alex. She likes to live independently, but is often thinking she would like to have someone checking on her in case she has any age related trouble at home. In order to maintain her independent living she has decided to get some sensors and artifacts in her house, thus making it a ubiquitous one). In order to create some applications that she wanted she has asked the help of her grandson Dimitris, a 15 year old who has grown up into a computerized environment and is an enthusiastic early adopter of technology developments.

Dimitris has grown up with computers; he had a computer since he was 5 years old, and he learned how to type, browse, and use the mouse at the same time (if not a bit earlier) that he learned to write. Dimitris has started learning some programming skills when he was 7, initially setting up his accounts in social systems chat with his friends and playing FARM-VILLE (his favorite social network game). Programming was a part of the school curricula when Dimitris was growing up, so he developed an affinity with computers and programming, not as a professional programmer, but as a keen side interest that helped him to get by in his assignments and past time hobbies, as many other kids of his age.

Helen wants not to be disturbed by the telephone ringing at inappropriate times, but on the other hand she is concerned that not answering the phone may cause concern on her family, regarding her well-being. She expresses her wish to Dimitris, who in turn explains the basic functionality to his grandmother, using the Capabilities and Links model to make her understand the underlying principles. They then decide that they should set up the following applications for her:

- She wants the telephone to go on the answering machine when she has her friends over for the occasional card-playing night and when she is asleep or having a bath.
- She wants her telephone ringer to go louder when she is in the veranda, in the kitchen, or watching TV, so that she could hear it better. Her hearing has deteriorated and especially when there are other sounds (like the TV or Cooking) she sometimes does not notice the ringing.
- She wants to share information about her location and movement in the house, with her daughter and her son, so that they know that she is well and they are not worried when they call her and she does not respond.

Dimitris checks that her phone is an augmented artifact that can offer its services through a ubiquitous system, and then places some extra sensors (wireless pressure sensor mats and movement sensors) under the sofa's cushions and the veranda-mat and a couple of movement sensors in the bathroom and kitchen.

He then downloads on his portable laptop the Ubicomp editor, to set up the application. He first assigns the sensors to the respective rooms and furniture (so that he can handle them as artifacts), then puts in some keywords and free text describing the application, and presses "search similar" in order to find if other similar applications have been created, so that he does not start from scratch. The search points out at a similar application that exists for checking out someone's location in the house, although it needs to be adapted for the specifics of grandma's home.

Dimitris connects certain groups of sensors to the telephone, and then goes on to select the specifics of this application. He sets a group of sensors (from the chairs of the dining table, the bed, and the bath), so that when they sense presence (pressure or movement), then, this is connected to the telephone and affects the ringing volume. When there is pressure / movement on these sensors and in parallel the phone receives an incoming phone, then (he goes on selecting from a pull down menu from the telephone properties the ringer), the answerphone picks up, and the ringer does not ring. Nevertheless he makes the addition of a small lamp, that in this case turns on instead of the ringer, so that his grandmother has in the periphery of her attention a visual signal that the phone is ringing when they are playing a card game.

He goes on to make the second setting that his grandma asked for, and copies the same configuration to start from. He changes the sensor inputs to that of the sofa, the kitchen and the veranda, and changes the telephone response to being increasingly louder than normal. He keeps the lighting of the lamp and sets it to turn on an off, as he considers that this would be more useful for his Grandmother to notice when she is in the kitchen or watching TV.

When Dimitris has finished setting up these two applications for his grandma's telephone, he goes on to test them. He asks his grandma to go to several places in the house, while he calls her land-line with his mobile phone. He notes a couple of glitches and corrects them: more than one chairs should be occupied, when they are playing bridge. There is a more serious glitch at the functionality of the TV watching part : When his grandma has left the sofa, the system indicates that the sofa is still occupied…the system takes input to start the application when a location is identified, but does not have input when to stop the application. He continually swaps editing views between the tree-line and pipeline editing view and the natural-text entry that makes questions and suggestions, in order to figure out what is best to do. He introduces sensor checks over 30 seconds, to be able to end that part of the application. The bath and kitchen sensors are newer and let the system know on the change of their state, in order to reconfigure the applications reaction.

Dimitris has corrected the glitches and is happy to receive a cup of hot chocolate from his Grandother, along with some pocket money as a gift of appreciation for his help. As he moves about in the house, the telephone rings; but the ring tone is going on and off randomly, and the lights are flashing, because now two people are in the house: he is in the veranda, while his grandma is on the sofa….he makes a point to himself to add some identification sensors and correct this in the future.

On the plasma TV he ports an 'idle mode' of the editor. He now can see a scrolling view of all the applications that he just made. He turns the last application OFF, while the other two are ON. Dimitris explains the overview control as seen on the TV plasma screen to his grandmother, so that she can turn the applications on and off if she wants - and hopes that she does not forget how to do this by next week.

For setting up the last application, Dimitris decides to use the application that is being shared on the community applications site. He re-configures it automatically, to take input from the sensors on his Grandma house, and he responds to a few system questions

regarding dubious points on where the sensors are. He asks his sister Alex to link up and check from their house if the movement in the house is spotted correctly. He gives in very strict rights, to his mother and uncle only so that they can use a text based interface in the idle mode of their mobile phone, that points at the grandma's location in the house (it displays messages such as "bed", "bath", "kitchen", "veranda"). He thinks to himself that this is a very useful application, but could do with a better interface for visualizing the information, and makes a point to make this at home and also add it to the communal repository of applications.

Dimitris thinks that he could install the same application to his girlfriends house, so that he can see when she is alone at home to call her or call by; but he is deterred by the fact that the idle screen of the system shows all the currently running applications, so it will be noticed by her parents and they may end up in trouble. Perhaps good old Tweet messaging may be better for their frequent status updates after all!

It is already dark when Dimitris leaves Helen's house; but before he leaves, he makes sure that Helen grants him remote control over her ubicomp system, so that he can service it remotely and cater for any bugs that he may not have foreseen, that, he is sure will occur in the next few weeks use.