J Ambient Intell Human Comput (2010) 1:31–42
DOI 10.1007/s12652-009-0004-4

ORIGINAL RESEARCH

Towards a new human-centred computing methodology for cooperative ambient intelligence

Tom Gross

Received: 22 June 2009 / Accepted: 26 September 2009 / Published online: 4 December 2009
© Springer-Verlag 2009

Abstract Cooperative ambient intelligence aims to improve users' work and private life by analysing their current situation with a special focus on their social interaction and to adapt the environment accordingly. As technology is seen as highly embedded in physical environments and the social fabric of users, a reconsideration of methods for its design, implementation, and evaluation is vital. In this paper we characterise cooperative ambient intelligence, revisit existing methods, and discuss what constitutes a novel human-centred computing methodology.

1 Introduction

Cooperative ambient intelligence is intrinsically human-centred—it aims to improve users' work and private life through the creation of novel information and communication technology, and through the analysis of the users' situation with a special focus on their social interaction as well as adaptation of technology. It can be seen as a further evolution of technological environments based on ubiquitous computing technology towards support for both face-to-face as well as remote communication and cooperation. Several such evolutions of concepts of information and communication technology have been witnessed over the last decades. Thereby, the evolution of the interaction concepts of the technology always goes hand in hand with the evolution of the approaches, methods, and techniques for the design, development, and evaluation of technological innovation. The following two examples of Grudin, and Boehm illustrate this co-evolution of technology and processes for their development.

Grudin studied the evolution of the *focus on user interface research* from the 1950s to the beginning of the 1990s. Grudin identified five foci of interface development: interface as hardware in the 1950s used by engineers and programmers; interface as software in the 1960s and 1970s used by programmers; interface as terminal from the 1970s to the 1990s used by end-users; interface as dialogue from the 1980s onwards used by end-users; and interface as work setting from the 1990s onwards used by groups of end-users (Grudin 1990, p. 265). Grudin points out that 'as the focus shifts, the approaches to design and the skills required of practitioners changes' and 'the principal focus of activity in computer development has moved gradually from hardware to software and is now shifting toward the user interface. Corresponding shifts are present within the domain of user interface research and development itself'. And he continues that 'this in turn has led to new approaches to design and evaluation. And so it shall continue. We can extrapolate that new approaches, responding to the user interface's move into the workplace, will require new skills, supplementing current approaches. They may not graft easily—or at all—onto existing development' (Grudin 1990, p. 261).

Boehm introducing his spiral model specifically emphasises the shift of the focus from *software challenges to end-users*. He points out that the traditional waterfall model had been suitable for traditional software applications, but not for later interactive applications. He writes (1988, p. 63):

T. Gross (✉)
Faculty of Media, Bauhaus-University Weimar,
Weimar, Germany
e-mail: tom.gross@medien.uni-weimar.de

> 'The waterfall model's approach helped eliminate many difficulties previously encountered on software projects. The waterfall model has become the basis for most software acquisition standards in government and industry. Some of its initial difficulties have been addressed by adding extensions to cover incremental development, parallel developments, program families, accommodation of evolutionary changes, formal software development and verification, and stagewise validation and risk analysis. […] For some classes of software, such as compilers or secure operating systems, this is the most effective way to proceed. However, it does not work well for many classes of software, particularly interactive end-user applications.'

The evolution characterised by Grudin and Boehm some twenty years ago has continued. In particular, the trend towards support for groups of end-users has continued in *computer-supported cooperative work* for cooperative applications based on the traditional desktop, and in *social software* based on cooperative Web applications (Gross and Fetter 2009). An even bigger step in the evolution is the emergence of ubiquitous computing—Mark Weiser (1993, p. 75), who coined the term *ubiquitous computing*, writes that it 'enhances computer use by making many computers available throughout the physical environment, while making them effectively invisible to the user'. *Ambient intelligence* has additional characteristics such as 'simple and effortless interactions, attuned to all our senses, adaptive to users and context-sensitive, and autonomous' (Weber et al. 2005, p. 1). Ambient intelligence requires that the environment is embedded, adapts to the presence of people and objects, and assists users smartly while preserving security and privacy. Ambient media (Lugmayr et al. 2009) present information embedded in the environment. And ambient intelligence supports human contacts. Only recently, *cooperative ambient intelligence* departs from social settings with the assumption that the presence of a single person is possible, but that the presence of a group of persons is more likely and is the primary focus (Markopoulos et al. 2005).

This continual evolution of the technology entails questions on the nature of a fruitful evolution of the processes for their design, development, and evaluation. We explore questions subsequently. We start with a characterisation of the evolution of technology until today. We then analyse the approaches for the processes for design, development, and evaluation. Finally, we introduce a human-centred computing methodology in the form of a software lifecycle for the design, development, and evaluation of cooperative ambient intelligence.

2 Towards cooperative ambient intelligence artefacts

In this section we characterise the evolution towards cooperative ambient intelligence, and narrow the focus on interactive systems providing functionality for end-users who interact with the technology and through the technology (Baecker et al. 1995; Dix et al. 2004; Preece et al. 2007).

2.1 Traditional GUI and WIMP

After a first period of batch mode processing in the 1950s and 1960s and a second of time-sharing in the 1970s, now the third period of *Windows-Icons-Menus-Pointing Devices (WIMP) paradigm with the Graphical User Interface (GUI)* has been prevalent (van Dam 1997) ever since. Software and hardware are based on keyboard, mouse, and computer screens and their technical capabilities. The WIMP and GUI often use a desktop metaphor and allow users the direct manipulation of electronic artefacts—that is, the system provides a continuous representation of the electronic artefacts and the possible actions and users can perform rapid, reversible, and incremental actions, and get immediate feedback on them (Hutchins et al. 1985; Shneiderman 1983). Considerable progress has been made towards a better understanding of users and tasks, design and implementation of computer software and hardware technology, and evaluating it. Yet, human–computer interaction is still a very active research field for academia and industry likewise.

2.2 Ubiquitous computing

Ubiquitous computing is a different paradigm that goes beyond the WIMP paradigm. Basically, ubiquitous has the meaning of 'existing or being everywhere at the same time: constantly encountered: widespread'; and computing comes from 'to compute' and is 'to determine especially by mathematical means <compute your income tax>; also: to determine or calculate by means of a computer' (Merriam-Webster 2009). So, literally ubiquitous computing means 'computing everywhere'. More specifically, Mark Weiser (1993, p. 75), who coined the term ubiquitous computing, writes:

> 'The goal is to achieve the most effective kind of technology, that which is essentially invisible to the user. […] To bring computers to this point while retaining their power will require radically new kinds of computers of all sizes and shapes to be available to each person. I call this future world "Ubiquitous Computing" (UbiComp).'

Hand in hand with ubiquitous computing goes *Calm Technology* as the paradigm for the interaction of users with ubiquitous computing technology. According to Weiser and Brown (1997, p. 79):

> 'The most potentially interesting, challenging, profound change implied by ubiquitous computing era is a focus on calm. […] Calmness is a new challenge that UC brings to computing. […] But when computers are all around, so that we want to compute while doing something else and have more time to be more fully human, we must radically rethink goals, context and technology of computer and all the other technology crowding into our lives. Calmness is a fundamental challenge for all technological design of next fifty years.'

Abowd points out that ubiquitous research is experimental and good research in this area should have a *motivating application* and 'should address scale (i.e., space covered, number of people involved, number and variety of devices supported, amount of time over which an application is run)' and 'should be subjected to real and everyday use (Abowd 1999, p. 75). Also according to Lyytinen and Yoo (2002, p. 64) the 'main challenges in ubiquitous computing originate from integrating large-scale mobility with the pervasive computing functionality'. So, ubiquitous computing entails requirements related to mobile computing and its motivation to make devices more capable of being physically moveable; and of pervasive computing and its motivation to develop hand devices that can capture information from its environment and environments that can detect the nearby devices (Lyytinen and Yoo 2002).

Abowd and Mynatt (2000) identify three *core themes* that are being addressed in ubiquitous computing: natural interfaces, context-awareness, as well as automated capture and access to live experiences. Natural interfaces refer to an interaction of users with the environment beyond keyboard, mouse, and computer screen that is increasingly based on speech and gesture. Context-awareness allows environments to capture the presence and activities of users, to infer on these data, and to adapt accordingly. Automated capture and access to live experiences provides technology for capturing various types of media such as projected slides and spoken language via multiple sensors, smart merging of the streams captured, and flexible later access.

So, to sum up the characterisation of ubiquitous computing, thus far in this paper: realising the original vision of Mark Weiser of ubiquitous computing entails a considerable amount of requirements relating to technology of hardware and software and networks including mobility as well as relating to the intuitive interaction of users with this technology.

Beyond these technical and software engineering aspects, also *social and organisational* issues are vital—Lyytinen and Yoo (2002, p. 64) continue:

> 'The shift toward ubiquitous computing poses multiple novel technical, social, and organisational challenges. At the technology level, there are several unresolved technical issues concerning the design and implementation of computing architectures that enable dynamic configuration of ubiquitous services on a large scale. New challenges will also emerge in terms of how one should design and develop ubiquitous services. This may require rethinking of feasible architectures, design ontologies and domain models, requirements and interactions scenarios, and analysing new families of non-functional requirements (such as configurability and adaptability). Anticipated new ways of dynamically configuring services will also shift the line between proactive design and tailoring during use. Previously, unexplored challenges will also emerge at the border between the technical and the social: some issues are to be left outside the technical implementation to be addressed by social negotiation and due process; other issues should be addressed during technical design. Finally, the emergence of truly integrated sociotechnical systems will create a wide array of research and policy issues that deal with social organisation, impact, and the future of work, organisations, and institutions.'

2.3 Ambient intelligence and social interaction

Ambient intelligence takes a strong focus on such social and organisational issues and requires that the environment is *embedded*, adapts to the presence of people and objects, and assists users smartly while preserving security and privacy. Abowd and Mynatt (2000) suggest everyday computing as a new area of applications research. Based on concepts and technology of the previously described requirements, everyday computing suggests seeing technology as a 'constant companion' supporting continuous interaction between users and the environment, allowing for interruption and resumption and dealing with passages of time. From an everyday computing perspective users' activities have the following characteristics (Abowd and Mynatt 2000, p. 42): (1) they rarely have a clear beginning or end; (2) interruption is expected; (3) multiple activities operate concurrently; (4) time is an important discriminator; and (5) associative models of information are needed.

As a consequence, Abowd and Mynatt (2000, p. 45) suggest the following research directions for ubiquitous computing: (1) design for a continuously present computer interface; (2) concept for the presentation of information at various levels of the periphery of users' attention; (3) support for connections of the physical and virtual worlds; and finally, but very importantly (4) methods for the support of developing such informal, peripheral, and opportunistic behaviour. Referring to the latter point, they (Abowd and Mynatt 2000, p. 46) write: 'there is no one methodology for understanding the role of computers in our everyday lives. However, combining information from methods as different as laboratory experiments and ethnographic observations is far from simple'.

Taking care of this embeddedness is important and has been particularly researched by Dourish (2001). That means that on the one side the users and their social interaction and on the other side the technology are mutually embedded and interdependent. At the same time there is synergetic potential if the social situation is adequately understood, appropriate conclusions are taken, and the technology adapts accordingly.

Cooperative ambient intelligence has a strong focus on this embeddedness and can be defined as environments that aim 'to improve users' work and private life by analysing and adapting to the current situation with a special focus on the social interaction among users' (Gross 2008, p. 270). As Gross (2008) points out, technology is continually evolving and consequently its adaptation to users needs to progress. There has been a constant evolution of foci towards interactive systems: from single-user WIMP to cooperative systems to single-user ubiquitous computing to single-user ambient intelligence, and to cooperative ambient intelligence (cf. Table 1).

As mentioned above, there are challenges from a technical perspective, from a social perspective, and from an organisational perspective. These challenges refer to the concepts and environments per se as well as challenges relating to adequate methods for those who develop such concepts and environments. Whereas Gross (2008) primarily addresses the consequences of these foci on technological challenges of cooperative ambient intelligence, in the following we primarily address consequences on methodological challenges.

3 Towards a human-centred computing methodology

Similar to the evolution of information and communication technology, an evolution of the processes for developing this technology is vital, and can be identified. The evolution of processes can be witnesses in two areas: software engineering, and human–computer interaction. We first characterise the relationship between usability and process models, and then analyse the process models from software engineering and from human–computer interaction.

3.1 Usability and processes

In general, *usability* refers to effective, efficient, and satisfactory interaction of users with artefacts, from everyday things such as furniture and appliances to advanced technology such as computer hardware and software (Norman 1988). The usability of software and hardware has been an important topic in computer science, since the 1970s (Hansen 1971). Many major associations of engineering and computer science include usability in their suggestions for curricula of computer science education at universities [(e.g., the Institute of Electrical and Electronics Engine IEEE has (Computer Society 2001), the Association of Computing Machinery ACM in (ACM SIGCHI 2008), the German Informatics Society GI has (Maass et al. 1996)].

Table 1 Summative overview of five foci with respect to time, key concepts awareness well as autonomous and adaptive behaviour

	Single-user WIMP	Cooperative systems	Single-user UbiComp	Single-user AmbInt	Cooperative AmbInt
Time	1970s	1980s	1990s	1990s	2000s
Key concepts	Graphical user interfaces; windows, icons, menus, pointing devices	Graphical user interfaces; computer-based communication, screen sharing, tele-pointers	Vast availability of embedded and mobile technology; periphery of user attention	Attuned to users' senses; adaptivity part of any environment	Adaptivity to groups of users part of any environment
Autonomous and adaptive behaviour	Adaptability through user customisation; adaptivity of user interface	Adaptability through articulation works support; adaptivity of resource allocation	Adaptivity of information and functionality	Highly adaptive to (mostly single-users') presence and activities	Highly adaptive to presence and activities of local and remote groups, and to their needs for local group interaction and remote group–group interaction

From (Gross 2008, p. 275)

And for usability, the process perspective is important. Jim Foley writes (Preece et al. 2007, p. XI):

> 'It is the process that I believe is the most important part of UI design, and the hardest for technology-oriented students to appreciate. […] When I first started teaching a full semester-long HCI course to computer science students about thirty years ago, I spent much more time than now on details of interaction devices and interaction techniques and software structures. That was at a time when GUIs were just becoming popular, and students had limited experience and exposure to their concepts, capabilities, and underlying software architectures. While I did preach 'know thy user', task analysis etc., I did not spend enough time helping students really understand how to do it, so their projects were often too technology-centric rather than user-centric. […] A well-known mantra is "*users* perform *tasks* using *computers*." This implies that the user interface designer needs to know about users, tasks, and computers. …learn about users and the tasks they perform and then to apply that knowledge in creating, evaluating, and refining designs that do indeed allow users to perform their tasks.'

Despite the fact that it is rather easy to identify issues with usability, it is difficult to design and implement usable systems. Nielsen (1994, p. 23) writes: 'in reality, different users have different needs, and a system that is "friendly" to one may feel very tedious to another'. Therefore, usability requires a good structure of the processes—that is, a structured process, involving the actual users of the system is essential for producing artefacts that have usability (Nielsen 1992).

Good process models are vital for the development of successful technology that is usable for their users. They can traditionally be found in software engineering, but also human–computer interaction.

3.2 Software engineering

The field of *software engineering* has a long tradition of dealing with the organisation of processes for the development of software. The traditional waterfall model, which was initially published in 1970, already provided a detailed list of steps to follow. The waterfall model of Royce [1987 (reprint from 1970)] suggested the phases: system requirements; software requirements; analysis; program design; coding; testing; and operation (cf. Fig. 1).

And—despite many rumours on the waterfall model about lacking feedback loops—Royce already in the original paper pointed out that 'the ordering of steps is based on the following concept: that as each step progresses and the design is further detailed, there is an iteration with the preceding and succeeding steps but rarely with the more remote steps in the sequence' [1987 (reprint from 1970), p. 328].

Later, Boehm published the famous 'Spiral Model of Software Development and Enhancement' in Boehm (1988). Boehm announced the spiral model with considerable criticism for the waterfall model; he writes (Boehm 1988, p. 63):

> 'However, even with extensive revisions and refinements, the waterfall model's basic scheme has encountered some more fundamental difficulties, and these have led to the formulation of alternative process models. A primary source of difficulty with the

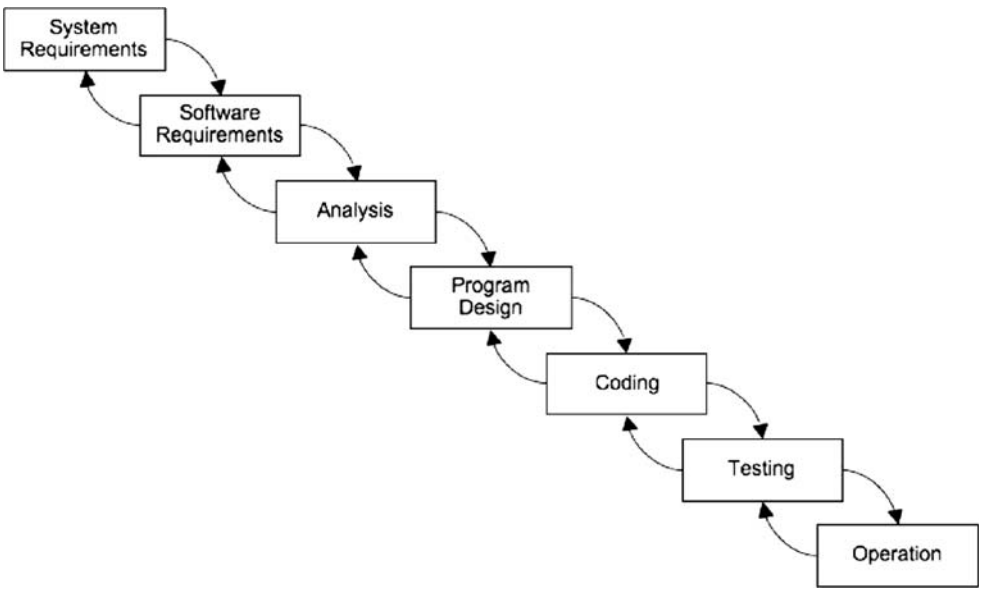

Fig. 1 Waterfall model. Adapted from: [Royce 1987 (reprint from 1970), p. 330]

waterfall model has been its emphasis on fully elaborated documents as completion criteria for early requirements and design phases.'

The spiral model suggests to go through the phases in a spiral from inside out and to continually expand the results of each phase in each circle (Boehm 1988, p. 65):

'Each cycle of the spiral begins with the identification of the objectives of the portion of the product being elaborated (performance, functionality, ability to accommodate change, etc.); the alternative means of implementing this portion of the product (design A, design B, reuse, buy, etc.); and the constraints imposed on the application of the alternatives (cost, schedule, interface, etc.).'

3.3 Human–computer interaction

In the field of *human–computer interaction* and *user-centred design* also process models for the development of interactive systems have been suggested. Despite the fact that the basic goal is the same—that is, the organisation of processes for the development of the system—there are quite some differences. In fact, Stone et al. (2005, p. 16) write:

'User-centred design and traditional software engineering take very different approaches to computer system design. Traditionally, software developers have treated each phase of the software design life cycle as an independent part of software development, which must be completely satisfied before moving onto the next phase. This is particularly so in relation to the classic life cycle (also known as the waterfall model, so named because of the cascade from one phase to another…).'

Yet, Stone et al. (2005, p. 16) also write that some software engineers already saw the iterations:

'In practice, however, the development stages overlap and feed information to each other. During design, problems with requirements are identified; during coding, design problems are found; and so on. The software process is not a simple linear model but involves a sequence of iterations of the development activities [Sommerville 1992, p. 7].'

And, so, Stone et al. summarise the difference between user-centred design and software engineering as follows (Stone et al. 2005, p. 16):

'The essential difference between the classic life cycle and user-centred interface design is that user interface design and development is based on the premise that users should be involved throughout the design life cycle. Additionally, the process should be highly iterative, so that the design can be tested (or evaluated) with users to make sure it meets the users' requirements.'

This distinction can best be seen in the 'star life cycle' (cf. Fig. 2) of Hartson and Dix. Introducing it, the authors write (Hartson and Hix 1989, p. 52):

'These results suggest a 'star' life cycle for human–computer interface development […]. This star life cycle, with evaluation at its centre, supports iterative refinement and rapid prototyping. Because of its high interconnectivity, it allows almost any ordering of development activities and promotes rapid alternation among them.'

The International Standardisation Organisation (ISO) has developed several standards. Most important are the ISO 9241 standard with the traditional title 'Ergonomic Requirements for Office Work with Visual Display Terminals' that has been partly renamed by ISO to 'Ergonomics of Human–System Interaction' (ISO/IEC 2009) primarily referring to qualities of the artefact, and the ISO 13407 standard with the title 'ISO 13407: 1999—Human-Centred Design Processes for Interactive Systems' (ISO 2009) primarily referring to qualities of the process of developing the artefact. This latter standard ISO 13407 names similar steps and requires that the overall process iterates through these steps until the interactive system has reached the required quality. The individual steps are the identification of the need for human-centred design; the understanding and specification of the context of use; the specification of the user and organisational requirements; the production of the design solutions; and the evaluation of the design against the requirements (cf. Fig. 3).

So, overall in both software engineering and in human-centred computing there are process models including

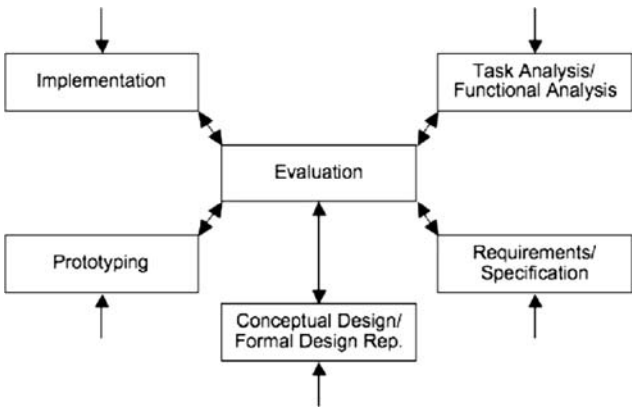

Fig. 2 Star life cycle. Adapted from: (Hartson and Hix 1989, p. 52)

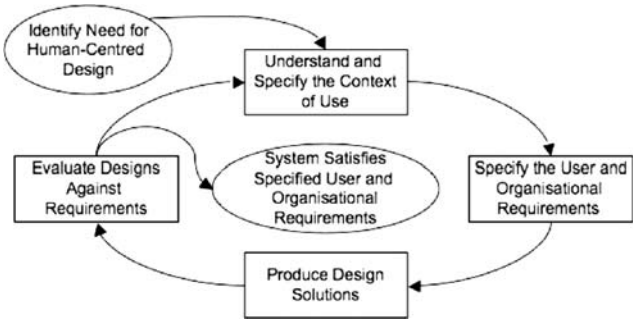

Fig. 3 ISO 13407: 1999–human-centred design processes for interactive systems. Adapted from: (ISO 2009)

multiple steps from analysis to design to implementation to evaluation, with iterations between the steps and within each individual step. These steps have been successfully applied to single-user WIMP and cooperative systems over decades. Yet, with the considerable shifts from single-user WIMP and cooperative systems to ubiquitous computing and cooperative ambient intelligence new and innovative process models are needed.

4 Human-centred computing methodology for cooperative ambient intelligence

Cooperative ambient intelligence is embedded in multifarious circumstances and contexts and needs to be designed from a human-centred perspective. In this section we present a novel process model based on the insight gained from the preceding two sections: the evolution of technology towards cooperative ambient intelligence and the evolution of process models. We want to advance the design, implementation, and evaluation processes to provide a basis for successful cooperative ambient intelligence. We first revisit technology and its requirements concerning process models; and then we develop a novel process model for cooperative ambient intelligence.

4.1 Cooperative ambient intelligence

In the epilogue of (Weiser et al. 1999), John Seely Brown (Weiser et al. 1999, p. 695) that Mark Weiser had a transdisciplinary perspective on ubiquitous computing:

> 'Is as much a study in phenomenology as it is of user and community interface design […] boundaries between the social and the technical, between the artistic and the scientific, and between work and play never existed.'

Weiser also commented methodological aspects; he talks about the research method for ubiquitous computing (Weiser 1993, p. 75) as follows:

> 'The research method for ubiquitous computing is standard experimental computer science: the construction of working prototypes of the necessary infrastructure in sufficient quantity to debug the viability of the systems in everyday use, using ourselves and a few colleagues as guinea pigs. This is an important step towards insuring that our infrastructure research is robust and scalable in the face of the details of the real world.'

Abowd and Mynatt particularly point to the new challenges that arise from the fact that the technology will be available anytime and everywhere and facilitate distant connections of people. This new permanent presence and availability of other users and technology has great advantages, but entails novel challenges for the design (Abowd and Mynatt 2000, p. 31):

> 'Today we are just starting to understand the implications of continuous immersion in computation. The future will hold much more than constant availability of tools to assist with traditional, computer-based tasks. Whether we wear computers on our body, or have them embedded in our environment, the ability of computers to alter our perception of the physical world, to support constant connectivity to distant people and places, to provide information at our fingertips, and to continuously partner with us in our thoughts and actions offers much more than a new "killer app"—it offers the possibility of a killer existence.'

4.2 Human-centred computing

Putting users as human beings with their specific strengths and weaknesses and with technological compensation in terms of pushing the limits of the strengths and compensating weaknesses in the centre of the focus is a vital perspective. Jaimes et al. (2007, p. 31) write on this human-centred computing (HCC):

> 'HCC facilitates the design of effective computer systems that take into account personal, social, and cultural aspects and addresses issues such as information design, human–information interaction, human–computer interaction, human–human interaction, and the relationships between computing technology and art, social, and cultural issues.'

So, from what has been identified so far, it is important to take a human-centred computing perspective and organise the whole process of developing cooperative ambient intelligence truly interdisciplinary, and to really make users a part of the cooperative ambient intelligence and use it and

experience the short- as well as mid- to long-term consequences and give feedback on them. Addressing these human-centred computing challenges and providing a methodology for it, is a great challenge that was already tackled by the traditional Bauhaus in Weimar.

4.3 Lessons learned from the traditional Bauhaus

The traditional Bauhaus had the paradigm of bringing creative and systematic approaches together into what the founder Walter Gropius called 'Neue Einheit von Kunst und Technik'—a new unity, where the German word *Kunst* can mean both arts and skill, and where the German word *Technik* can mean both technique and technology. So, in fact it is a real combination of arts and creative concepts, with experience and knowledge on perceiving the world, on the technology and on the basic materials, as well as with a systematic organisation of the process (Lupfer and Sigel 2005). The Bauhaus was founded in 1919 in Weimar as a new kind of art school—an 'art school of modernism'—intended 'to reconstruct the unity of art and the culture of production that had broken down as a result of industrialisation, to reintegrate art and life, to undo the splintering of artistic genres, and thus to use art itself as an instrument of cultural and social regeneration' (Wick 2000, p. 15).

The Bauhaus had a clear approach: to find prototypical solutions; to systematise existent, and develop new tools, techniques and materials; to develop concepts and methods; and to create an educational framework for the work and the training of such a 'new artist'. As mentioned above, the Bauhaus was an art school intended to be a testing field for prototypical solutions and to develop an educational framework to support the elaboration of a new profession concerned with the reintegration of art in everyday life in order to develop an adequate aesthetic culture of mass production from a human-centred perspective. The Bauhaus had two distinct phases. In *the early phase* of the Bauhaus, its founding director Walter Gropius proclaimed in 1919 the paradigm of *art and crafts, a new unity* to create the living environments of a new era (later called the Modernity) by collaborative work of all art and crafts professions. In the further development of the Bauhaus, this paradigm shifted in *the later phase* towards a more technology-oriented focus. In 1923 Walter Gropius consequently proclaimed in the later phase a modification into art and technology, a new unity as an art-driven approach to socially embedded technology taking the human being as well as society into account, and to emphasise the importance of aesthetic in every day life experience.

Overall the Bauhaus made: 'outstanding contributions to aesthetic education' (2000, p. 15); 'substantial contributions to foundation of what we now generally characterise as design' (2000, p. 15); developments of new aesthetic and functional concepts from artwork to every day products, architecture and urban planning; and various artefacts that embody and transport their aesthetic and functional concepts from the well-known Bauhaus buildings to furniture and print products.

What is similar between the traditional Bauhaus in Weimar and today's cooperative ambient intelligence is the fact that building and construction have new challenges. Whereas the Bauhaus was facing a new Zeitgeist of modernism, cooperative ambient intelligence faces new unprecedented technological opportunities. For instance, Kandinsky aimed at developing a dictionary [quote from (Kandinsky 1955 (reprint from 1926), p. 90) translated in (Lupton and Miller 1993, p. 23)]:

> 'The progress won through systematic work will create a dictionary which, in its further development, will lead to a "grammar" and, finally, to a theory of composition that will pass beyond the boundaries of the individual art expressions and become applicable to "Art" as a whole.'

4.4 Human-centred computing lifecycle

The Human-Centred Computing Methodology for Cooperative Ambient Intelligence tries to achieve this by combining and extending the strengths of existing process models (cf. Fig. 4). It departs from the need for cooperative ambient intelligence—that is, the need for an environment that provides smart functionality for the social interaction in local as well as between remote groups based on ubiquitous computing software and hardware.

Since cooperative ambient intelligence scenarios in general include multiple stationary and mobile settings, several analyses of the respective *Situation of Use* need to be made and models for the respective situation need to be developed [please note that here we use the term situation, rather than context as in other process models, since here we refer to the users' situation, rather than the context in terms of the boundary between the system and the environment such as in (Sommerville 2007, p. 171)]. These situation models need to represent both the statics of a given setting including the users, their tasks, and their physical environment as well as the dynamics of a given setting in the form of adaptation results that the environment should make based on the users and their tasks.

Then, the *User, Ambient, and Cooperation Requirements* are specified. They again contain the statics as well as the dynamic adaptation rules and heuristics based on the respective situation and its changes measured by sensors in the electronic and physical environment.

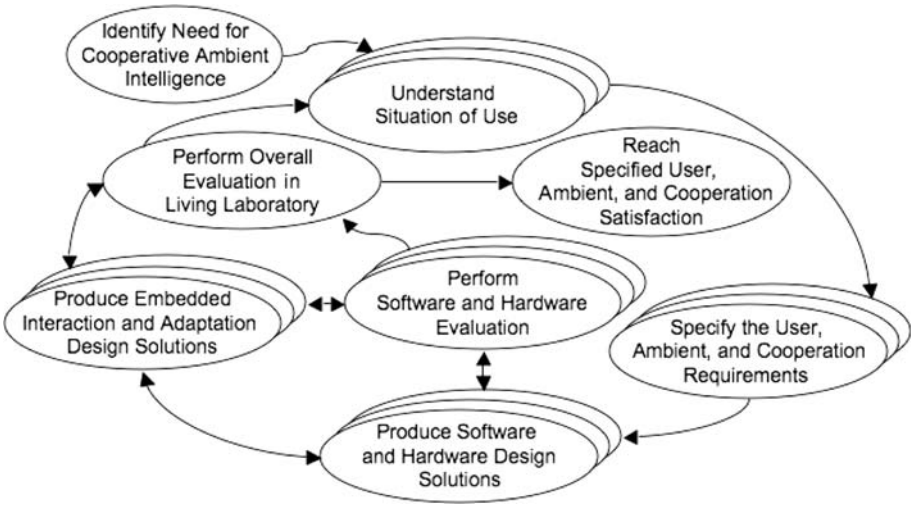

Fig. 4 Human-centred computing methodology for cooperative ambient intelligence

Software and Hardware Design Solutions are produced that provide the functionality the users need for their interaction with the technology and with each other through the technology as well as the capturing of the data of the environments.

Embedded Interaction and Adaptation Design Solutions provide the means for embedded interaction of the users in the respective surrounding.

All design solutions are evaluated in the *Software and Hardware Evaluation* specific to the situation they are situated in against the requirements of the respective technology unit.

Based on successful individual evaluations, an integrated evaluation of the design of the whole cooperative ambient intelligence environment can take place in an *Overall Evaluation in Living Laboratory*.

In the inner iteration cycle, these latter four steps—that is, the production of the software and hardware design solutions, the production of the embedded interaction and adaptation design solutions, the specific evaluations, and the overall evaluation—are all highly interwoven and feedback and feedforward are very likely to occur between them. This is the case since the different design solutions can affect each other and can trigger the need for modifications; since the individual evaluations can affect both types of design solutions (software and hardware); and since the overall evaluation can affect the individual evaluations as well as the overall embedded interaction and adaptation design solutions.

Since the adaptations in cooperative ambient environments have considerable influence on the situation of use, it is quite likely that after an overall evaluation the processes need to be run again in the outer iteration cycle, starting with an analysis of the *Situation of Use*. Once the overall evaluation has produced a satisfactory overall result and reaches the *Specified User, Ambient, and Cooperation Satisfaction*, the process can terminate.

Overall all phases can take considerable time—yet, the evaluation of the overall design in the living laboratory can take by far the longest time, since this phase really aims to gain long-term feedback of users that are really using the cooperative ambient intelligence in their everyday life.

5 Discussion

This paper has introduced cooperative ambient intelligence, it has elaborated on existing approaches for organising software engineering and user-centred design processes, and it has suggested a new human-centred computing methodology for cooperative ambient intelligence. The individual phases of the human-centred lifecycle have been broadly characterised.

Many research issues remain for each phase in terms of the methods to be applied, the adaptation of the method concerning the characteristics of the targeted technological innovation, and the properties of the results of each phase. The following list provides some suggestions for methods and tools to be used in each phase:

- *Understand Situation of Use* in order to get a better understanding of the situation of use, ethnography (and particularly ethnomethologically informed ethnography) can be applied (e.g., Crabtree et al. 2006; Intille et al. 2005; Nova et al. 2005; Randall et al. 2007) for studying the situation, and various approaches can be taken for modelling the situation (e.g., Chalmers 2004; Gross and Prinz 2004; Schmidt et al. 1999)
- *Specify the User, Ambient, and Cooperation Requirements* traditional methods from software engineering

(cf. Sommerville 2007) or later methods (e.g., UML Fowler 2004 from the OMG 2009) can be used here
- *Produce Software and Hardware Design Solutions*: middleware and software platforms can be used for programming complex ambient software solutions (e.g., Gaia, Ranganathan and Campbell 2003 or Sensation, Gross et al. 2006); and since developing custom hardware is often tedious standard hardware can be used (e.g., Arduino, Arduino 2009 or Phidgets, Greenberg and Fitchett 2001)
- *Produce Embedded Interaction and Adaptation Design Solutions* editors can be used to prototype and configure ambient scenarios [e.g., CollaborationBus (Gross and Marquardt 2007), iCAP (Sohn and Dey 2003), or jigsaw (Humble et al. 2003)]; living laboratories can be used for producing complex environments [e.g., MIT's Oxygen Project (Mitchel 2004), or Philips' HomeLab (Philips 2008)]; such living laboratories need to go beyond the traditional phase models [including also the rational unified process; cf. (Sommerville 2007, p. 82)]
- *Perform Software and Hardware Evaluation*: tools can be used to evaluate the functionality of the technology developed [e.g., Momento (Carter et al. 2007)]; and methods have been introduced to address the specifics of ambient technology (e.g., heuristic evaluation by Mankoff et al. 2003)
- *Perform Overall Evaluation in Living Laboratory*: living laboratories have become prominent in many institutions [e.g., see the European Network of Living Laboratories (ENoLL 2009)]; the approach of living laboratories promises innovative ways of participatory design and evaluation in real-life scenarios, helping to address multifarious ambient evaluation areas (a broad range of evaluation areas can be found in Scholtz and Consolvo 2004)

This list shows that in each steps of our lifecycle there has been research and development; and specific methods and tools are already available. There has also been some work towards a better understanding of the overall picture of human interaction in physical and mixed-reality settings. For instance, (Hornecker and Buur 2006) developed a framework addressing the following four themes: tangible manipulation (i.e., material representations of artefacts); spatial interaction (i.e., existence and movements in real space); embodied facilitation (i.e., influence of real space on behaviour); and expressive representation (i.e., system's properties for tangible interaction). Such contributions can be highly complementary to the processes, methods, and tools described in this paper.

Yet, it is clear that in this emerging field of cooperative ambient intelligence more research is needed from many directions—top-down with broad contributions of frameworks of embodiment, human-centred computing methodology, and technology platforms and environments; but also bottom-up with solid individual methods and tools. With this paper we aim to provide a broad contribution towards a human-centred methodology for cooperative ambient intelligence targeting at the big picture. Covering more details of our lifecycle would be important, but goes beyond the scope of this paper.

Acknowledgments I would like to thank all the members of the Cooperative Media Lab—special thanks to Christoph Beckmann, Mirko Fetter, Arkadiusz M. Frydyada de Piotrowski, Stefanie Koch, Thilo Paul-Stueve, and Maximilian Schirmer for inspiring discussions.

References

Abowd GD (1999) Software engineering issues for ubiquitous computing. In: Proceedings of the 21st international conference on software engineering—SE'99 (Los Angeles, CA). IEEE Computer Society Press, Los Alamitos, pp 75–84

Abowd GD, Mynatt E (2000) Charting past, present, and future research in ubiquitous computing. ACM Trans Comput Hum Interact 7(1):29–58

ACM SIGCHI (2008) ACM SIGCHI curricula for human–computer interaction. http://sigchi.org/cdg/. Accessed 13 October 10 2008

Arduino (2009) Ardiuno—HomePage. http://www.arduino.cc/. Accessed 18 September 2009

Baecker RM, Grudin J, Buxton WAS, Greenberg S, eds (1995) Readings in human–computer interaction: toward the year 2000, 3rd edn. Morgan Kaufmann Publishers, San Francisco

Boehm BW (1988) A spiral model of software development and enhancement. IEEE Comput 21(5):61–72

Carter S, Mankoff J, Heer J (2007) Momento: support for situated Ubicomp experiments. In: Proceedings of the conference on human factors in computing systems—CHI 2007 (April 28–May 3, San Jose, CA). ACM, NY, pp 125–134

Chalmers MA (2004) Historical view of context. Comput Support Cooper Work J Collab Comput 13(3–4):223–247

Computer Society (2001) Computer curricula 2001—appendix A CS body of knowledge. IEEE & ACM, http://www.sigcse.org/resources/cs-2001/appendixa/?searchterm=Computer%20Curricula%202001%20-%20Appendix%20A%20CS%20Body%20of%20Knowledge. Accessed 18 September 2009

Crabtree A, Benford S, Greenhalgh C, Tennent P, Chalmers M, Brown B (2006) Supporting ethnographic studies of ubiquitous computing in the wild. In: Proceedings of the conference on designing interactive systems—DIS 2006 (June 26–28, University Park, PA). ACM, NY, pp 60–69

Dix A, Finlay J, Abowd GD, Beale R (2004) Human–computer interaction. Pearson, Englewood Cliffs

Dourish P (2001) Where the action is: the foundations of embodied interaction. MIT Press, Cambridge

ENoLL (2009) Open living labs—the European network of living labs. http://www.openlivinglabs.eu/. Accessed 18 September 2009)

Fowler M (2004) UML distilled: a brief guide to the standard object modelling language. Addison-Wesley, Reading

Greenberg S, Fitchett C (2001) Phidgets: easy development of physical interfaces through physical widgets. In: Proceedings of the ACM symposium on user interface software and technology—UIST 2001 (November 11–14, Orlando, FL). ACM, NY, pp 209–218

Gross T (2008) Cooperative ambient intelligence: towards autonomous and adaptive cooperative ubiquitous environments. In: International journal of autonomous and adaptive communications systems (IJAACS), vol 1, suppl 2, pp 270–278

Gross T, Fetter M (2009) Computer-supported cooperative work. In: Stefanidis C (ed) The universal access handbook, vol 43. Lawrence Erlbaum Associates, Hillsdale, NJ, pp 1–22

Gross T, Marquardt N (2007) CollaborationBus: an editor for the easy configuration of ubiquitous computing environments. In: Proceedings of the fifteenth Euromicro conference on parallel, distributed, and network-based processing—PDP 2007 (February 7–9, Naples, Italy). IEEE Computer Society Press, Los Alamitos, pp 307–314

Gross T, Prinz W (2004) Modelling shared contexts in cooperative environments: concept, implementation, and evaluation. Comput Support Cooper Work J Collab Comput 13(3–4):283–303

Gross T, Egla T, Marquardt N (2006) Sensation: a service-oriented platform for developing sensor-based infrastructures. Int J Internet Protoc Technol (IJIPT) 1(3):159–167

Grudin J (1990) The computer reaches out: the historical continuity of interface design. In: Proceedings of the conference on human factors in computing systems—CHI'90 (April 1–5, Seattle, WA). ACM, NY, pp 261–268

Hansen W (1971) User engineering principles for interactive systems. In: Proceedings of the fall joint computing conference—FJCC'71 (December, Montvale, NJ). AFIPS Press, Washington, pp 523–532

Hartson HR, Hix D (1989) Human–computer interaction development: concepts and systems for its management. ACM Comput Surv 21(1):5–92

Hornecker E, Buur J (2006) Getting a grip on tangible interaction: a framework on physical space and social interaction. In: Proceedings of the conference on human factors in computing systems—CHI 2006 (April 22–27, Montreal, Canada). ACM, NY, pp 437–446

Humble J, Crabtree A, Hemmings T, Akesson K-P, Koleva B, Rodden T, Hansson P (2003) Playing with the bits—user-configuration of ubiquitous domestic environments. In: Fifth international conference on ubiquitous computing—UbiComp 2003 (October 12–15, Seattle, WA). Springer, Heidelberg, pp 256–264

Hutchins EL, Hollan JD, Norman DA (1985) Direct manipulation interfaces. Int J Hum Comput Interact 1:311–338

Intille SS, Larson K, Beaudin JS, Nawyn J, Tapia ME, Kaushik P (2005) Late breaking result: a living laboratory for the design and evaluation of ubiquitous computing technologies. In: Extended abstracts of the conference on human factors in computing systems—CHI 2005 (April 2–7, Portland, OR). ACM, NY, pp 1941–1944

ISO (2009) ISO 13407: 1999—human-centred design processes for interactive systems. ISO—International Organisation for Standardisation, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21197. Accessed 18 September 2009

ISO/IEC (2009) ISO 9241-1: 1997—ergonomic requirements for office work with visual display terminals—part 1: general introduction. ISO—International Organization for Standardization, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=21922. Accessed 18 September 2009

Jaimes A, Gatica-Perez D, Sebe N, Huang TS (2007) Human-centred computing: towards a human revolution. IEEE Comput 40(5):30–34

Kandinsky W (1955) Punkt und Linie zu Flaeche (Point and Line to Plain; in German). Freiburger Graphische Betriebe, Freiburg (reprint from 1926)

Lugmayr A, Risse T, Stockleben B, Laurila K, Kaario J (2009) Semantic ambient media—an introduction. Multimedia Tools Appl 44(3):337–359

Lupfer G, Sigel P (2005) Walter Gropius, 1883–1969: the promoter of a new form. Taschen, Cologne

Lupton E, Miller JA (eds) (1993) The ABC of triangle, square, and circle: the Bauhaus and design theory. Thames & Hudson Ltd, London

Lyytinen K, Yoo Y (2002) Issues and challenges in ubiquitous computing. Commun ACM 45(12):63–65

Maass S, Ackermann D, Dzida W, Gorny P, Oberquelle H, Roediger K-H, Rupietta W, Streitz N (1996) Recommendations for software ergonomics education in informatics curricula at german universities. German Informatics Society GI. http://www-cg-hci.informatik.uni-oldenburg.de/GI-Recommendations/. Accessed 18 September 2009

Mankoff J, Dey AK, Kientz J, Lederer S, Ames M (2003) Heuristic evaluation of ambient displays. In: Proceedings of the Conference on Human Factors in Computing Systems—CHI 2003 (April 5–10, Minneapolis, Minnesota). ACM, NY, pp 169–176

Markopoulos P, de Ruyter B, Privender S, van Breemen A (2005) Case study: bringing social intelligence into home dialogue systems. ACM interactions, pp 37–45

Merriam-Webster I (2009) Merriam-Webster Online. http://www.m-w.com/. Accessed 13 March 2009

Mitchel K (2004) MIT project oxygen. Computer Science and Artificial Intelligence Laboratory. http://oxygen.lcs.mit.edu/. Accessed 19 March 2008

Nielsen J (1992) The usability engineering life cycle. IEEE Comput 25(3):12–22

Nielsen J (1994) Usability engineering. Academic Press, London

Norman DA (1988) The psychology of everyday things. Basic Books, NY

Nova N, Girardin F, Dillenbourg P (2005) 'Location is not enough!': an empirical study of location-awareness in mobile collaboration. In: Proceedings of the IEEE international workshop of wireless and mobile technologies in education—WMTE 2005 (November 28–30, Tokushima, Japan). IEEE Computer Society Press, Los Alamitos, pp 21–28

OMG (2009) Object management group—UML. http://www.omg.org/. Accessed 18 September 2009)

Philips (2008) Philips Research—HomeLab. Koninklijke Philips Electronics NV, http://www.research.philips.com/technologies/misc/homelab/index.html. Accessed 19 March 2008

Preece J, Rogers Y, Sharp H (2007) Interaction design: beyond human–computer interaction. Wiley, NY

Randall D, Harper R, Rouncefield M (2007) Fieldwork for design: theory and practice. Springer, Heidelberg

Ranganathan A, Campbell RH (2003) A middleware for context-aware agents in ubiquitous computing environments. In: Proceedings of the ACM/IFIP/USENIX international middleware conference—Middleware 2003 (June 16–20, Rio de Janeiro, Brazil). Springer, Berlin, Germany, pp 143–161

Royce WW (1987) Managing the development of Larage software systems. In: Proceedings of the ninth international conference on software engineering—ICSE'87 (March 30–April 2, Monterey, CA). IEEE Computer Society Press, Los Alamitos, (reprint from 1970). pp 328–338

Schmidt A, Beigl M, Gellersen H-W (1999) There is more to context than location. Comput Graph J 23(6):893–902

Scholtz J, Consolvo S (2004) Toward a framework for evaluating ubiquitous computing applications. IEEE Pervasive Comput 3(2):82–88

Shneiderman B (1983) Direct manipulation: a step beyond programming languages. IEEE Comput 16(8):57–69

Sohn T, Dey AK (2003) Interactive poster: iCAP: an informal tool for interactive prototyping context-aware applications. In: Extended abstracts of the conference on human factors in computing systems—CHI 2003 (April 5–10, Fort Lauderdale, Florida). ACM, NY, pp 974–975

Sommerville I (1992) Software engineering, 4th edn. Addison-Wesley, Reading

Sommerville I (2007) Software engineering 8. Pearson Education Limited, Harlow

Stone D, Jarrett C, Woodroffe M, Minocha S (2005) User interface design and evaluation. Morgan Kaufmann Publishers, San Francisco

van Dam A (1997) Post-WIMP user interfaces. Commun ACM 40(2):63–67

Weber W, Rabaey JM, Aarts E (eds) (2005) Ambient intelligence. Springer, NY

Weiser M (1993) Some computer science issues in ubiquitous computing. Commun ACM 36(7):75–84

Weiser M, Brown JS (1997) The coming age of calm technology. In: Denning PJ, Metcalfe RM (eds) Beyond calculation: the next fifty years of computing. Springer, NY, pp 75–85

Weiser M, Gold R, Brown JS (1999) The origins of ubiquitous computing research at PARC in the late 1980s. IBM Syst J Spec Issue Pervasive Comput 38(4):693–696

Wick RK (2000) Teaching Bauhaus. Hatje Cantz Publishers, Ostfildern-Ruit