Identifying Guidelines for Designing and Engineering Human-Centered Context-Aware Systems *(Position paper)*

Emilian Pascalau

Conservatoire National des Arts et Métiers,
2 rue Conté,75003 Paris, France
emilian.pascalau@cnam.fr

Abstract. In the "future internet" environment that is generative and fosters innovation, applications are simplifying, are becoming mobile, are getting more social and user oriented. Software design capable of coping with such a generative environment that drives and supports innovation is still an issue. Some of the challenges of such systems include: empowering end-users with the necessary tools to model and develop applications by themselves, while in the same time hiding the technical layer from them. This paper introduces a set of guidelines for designing and engineering human-centered context-aware systems from a human computer interaction and meta-design perspective.

1 Introduction

Recent years have brought rapid technological advances both hardware and software: increasing pervasive computing paradigm, embedded sensor technologies and wide range of wireless and wired protocols. Applications are simplifying, are becoming mobile, are moving to the cloud, are getting more social and user focused [14]. "Future Internet" is emerging as a new environment, an environment that is generative, that fosters innovation, through the advances of technologies and a shift in people's perception about it and a paradigm shift in how people act and react in this environment [22].

In this context two directions get highlighted: context together with context -awareness and human-centered computing. Studied for more than 30 years in the field of artificial intelligence, computer and cognitive science, context has still been identified by Gartner, alongside cloud computing, business impact of social computing, and pattern based strategy, as being one of the broad trends that will change IT and the economy in the next 10 years [21].

We observe a paradigm shift in terms of users of context-aware systems. For example a user is no longer a specific individual or organization. It is often a community of collaborating, or performing similar tasks groups of users. Therefore, there is a growing need for systems meeting expectations of massive distributed

user base of pervasive ubiquitous devices as well as distributed cloud-based web services.

Design and deployment of such software capable of coping with a generative environment that drives and supports innovation through direct interaction and empowerment of the end-user is still an issue. Not only the development of such systems should be agile - the boundary is pushed even further - we need systems that are designed to be agile on run-time. It has been already argued (see for instance [15]) that designers and programmers can not foresee and anticipate what end-users will need. Users know better what they need and future internet environment clearly underlines this fact.

In consequence some of the challenges of such systems that arise in the future internet environment include: empowering end-users with the necessary tools to model and develop applications by themselves, while in the same time hiding the technical layer from them. This paper is part of a work in progress, and identifies and introduces a set of guidelines for designing and engineering human-centered context-aware systems.

The rest of the paper is organized as follows: Section 2 discusses a use case that is based on a concrete end-user problem (managing and tracking online purchases) arising from the future internet environment; the use case will support us in identifying a set of guidelines, for designing and engineering human-centered context-aware systems, in Section 3; Section 4 is reserved for related work. We conclude in section 5.

2 Application Scenario - Slice.com

In the future internet environment email communication is part of daily activities. Many of the business processes that take place in a company and not only are started and / or integrate the actions of receiving / sending emails. In some situations entire processes are comprised in email communications. This type of email based use cases are important both for academia as well as for industry. For academia from a perspective oriented towards methodology, for industry from a practical perspective, addressing very specific problems. Emails contain knowledge that is bundled in an unstructured way but which has meaning to end-users, and which could be used to address end-user specific problems.

A human-centered context-aware system dealing with such a use case would be required to provide at least the following capabilities:

- provide the means and allow the end-user to model and represent context;
- allow the modeling of relationships between context elements;
- allow the modeling of interactions between different contexts, this implies both in the form of conditions and sequences of events and actions (more precise business rules and business processes)
- based on the provided models have the capabilities to discover in the environment the modeled context(s)
- sense events and actions that are performed in the system
- perform actions according to models defined.

We support our previous assertions by discussing the Slice application scenario, in the next paragraphs.

Commius[1] is a European research project that tackles systems for small and medium enterprise systems (SMEs). The final goal of Commius, as argued in [5], is to turn existing email-systems into a management framework for structured processes. Each incoming email is autonomously matched to a process that is enhanced with proactive annotations.

Slice.com is an industry project. Similar to Commius uses emails to tackle a very specific end-user related problem, keeping track of online purchases, that emerged from the future internet dynamic and generative environment. This project it is even more specific from the perspective of an end-user.

Slice is an online purchase management tool that gets hooked into your email account. Whenever a new email is received Slice automatically analyzes the content of the email. If the email contains order information from one of your online shops, then Slice via pattern-based recognition techniques extracts order related contextual information and organizes this information for you. Hence all your purchases will be gathered in one place, you will be able to keep track of your shopping history, amount of money that you spent, type of products, time related information i.e. when a shipment is about to arrive and so forth.

We analyze from an end-user perspective what this use case is about.

- **Problem faced by users:** keeping track of the purchases made online.
- **Applications involved, Services**: Email as a legacy system; Services: online shops (Amazon, EBay), shipment services (FedEx, UPS); Geolocation services (Google Maps); other type of services i.e. topic extraction
- **Concepts**: shop, service, user, invoice, email, time and date, amount of money, product, type of product, location, address, tracking number. The list of concepts is not exhaustive, and is up to the each user; however these are the most obvious ones. Concepts are all those entities that are used in taking decisions and / or involved in any way in the process of resolving the end-user's problem.
- **Context**: For example one context, from the perspective of an end-user in the Slice use case, could comprise: a specific service such as FedEx; concepts associated with it, i.e. shipment, location, address. Further more interaction related to this specific context could be provided, as what to do with this information and so forth.

Figure 1 depicts a general interaction process with respect to this use case.

3 Identifying Guidelines

Fisher and Giaccardi argue in [11] that in a world that is not predictable, improvisation, evolution and innovation are a necessity. There is a shift from processes towards users. Users want their problems and their requirements to be taken into

[1] `http://www.commius.eu/`

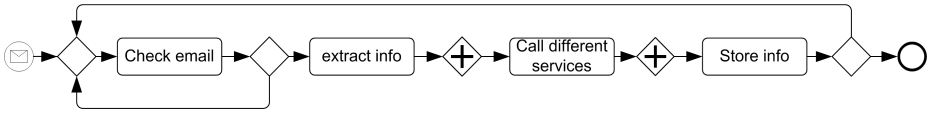

Fig. 1. General interaction process to solve the problem of keeping track of online purchases

account; they want to be part of the conversation. Continuously changing business models, do not fit anymore the old and stiff approaches. Processes must be in accordance with the reality. For instance process mining techniques [20] that look at event logs emphasize the fact that processes which actually get executed are different compared to the original blueprints. Companies need to change to what customers/users actually do.

Context greatly influences the way humans or machines act, the way they report themselves to situations and things; furthermore any change in context, causes a transformation in the experience that is going to be lived, sensed [4]. Many psychological studies have shown that when humans act, and especially when humans interact, they consciously and unconsciously attend to context of many types as stated in [12].

Traditionally context has been perceived in computer science community as a matter of location and identity, see for instance [9]. However interaction and problems concerning interaction require more than just the environmental context (location, identity) used traditionally in context-aware systems [12]. Lately the notion of context has been considered not simply as state but as part of a process in which users are to be involved [7].

Nardi underlines this aspect clearly in [15] stating that *"we have only scratched the surface of what would be possible if end users could freely program their own applications... As has been shown time and again, no matter how much designers and programmers try to anticipate and provide for what users will need, the effort always falls short because it is impossible to know in advance what may be needed... End users should have the ability to create customizations, extensions and applications..."*.

From a human-centered computing perspective this type of system is what Donald Norman calls in [16] the type of system, where the system itself disappears from sight, and humans (end-users) can once again concentrate upon their activities and their goals.

Grundin [12] continues and argues that aggregation or interpretation done by software systems are different than aggregation and interpretation done by biological, psychological and social processes.

Meta-design is a conceptual framework defining and creating social and technical infrastructures in which new forms of collaborative design can take place [11]. Meta-design originates in human computer interaction field and tackles end-user development.

Table 1, introduced in [11] compares side by side traditional design vs. meta-design. However, in our perspective a human-centered context-aware system, in order to provide a high degree of generality and to avoid re-implementation of common things related to infrastructure, should be a layered system as discussed in [18]. A low level that is very technical and should be hidden from the end-user. This low level would follow to great extent traditional design. The high level on the other hand should follow mainly meta-design. A translation mechanism has to be put into place to assure translation between these two layers.

Table 1. Traditional Design vs. Meta-Design [11]

Traditional Design	**Meta-Design**
guidelines and rules	exceptions and negations
representation	construction
content	context
object	process
perspective	immersion
certainty	contingency
planning	emergence
top-down	bottom-up
complete system	seeding
autonomous creation	co-creation
autonomous mind	distributed mind
specific solutions	solutions spaces
design-as-instrumental	design-as-adaptive
accountability, know-what (rational decisioning)	affective model, know-how (embodied interactionism)

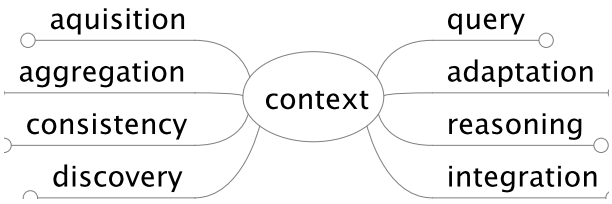

Fig. 2. Context requirements

Several definitions for the concept context have been enunciated; Fischer, however gives a definition that takes into account the human-centered computational environments. He defines context in [10] as being *the 'right' information, at the 'right' time, in the 'right' place, in the 'right' way to the 'right' person.*

Figure 2 depicts aspects that have been identified in [1] as requirements for dealing with context.

Table 2, introduced in [10] depicts adaptive and adaptable systems. Context aware systems traditionally position themselves, according to Table 2 in the category of adaptive systems. These systems employ users' profiles information and other type of contextual information, like location to improve users' experience. These approaches, although they provide a degree of flexibility, are however still stiff approaches because they are still based on predefined designs and very specific.

Table 2. Adaptive vs. Adaptable Systems [10]

	Adaptive	**Adaptable**
definition	dynamic adaptation by the system itself to current task and current user	users change the functionality of the system
knowledge	contained in the system; projected in different ways	knowledge is extended by users
strengths	little (or no) effort by users; no special knowledge of users is required	users are in control; users know their tasks best
weaknesses	users often have difficulties developing a coherent model of the system; loss of control	users must do substantial work; complexity is increased (users need to learn adaptation components); systems may become incompatible
mechanisms required	models of users, tasks, dialogs; incremental update of models	support for end-user modifiability and development
application domains	active help, critiquing systems, recommender systems	end-user modifibiality, tailorability, design in use, meta-design

In our vision a human-centered context-aware system is a system where adaptivity and adaptability are blended together. By such a method users will be able to directly model how the context should look like for a particular problem, and afterwards the system would be required only to verify that the specified context really exists in a given environment. Moreover while for adaptive systems as stated in [3] the architecture of such systems comprises a user model (user perspective on the problem) and a domain model (system perspective as it has been designed), for a human-centered context-aware system there should be only one conceptual model of the problem, that should be shared and understood in the same way both by the end-user and the system.

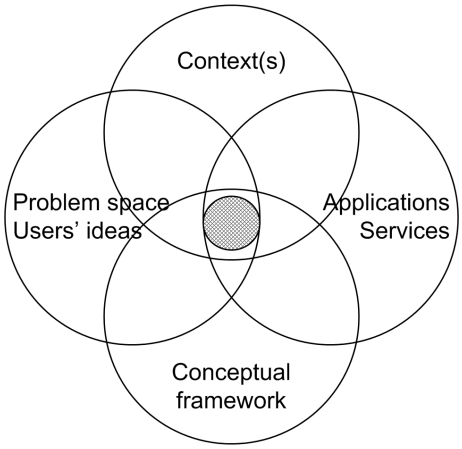

Fig. 3. Interrelated topics

Figure 3 depicts the main perspectives that need to be blended together in order to design a human-centered context-aware system. This particular view has its roots in the meta-design conceptual framework [11].

Aspect Oriented Programming [19] is a relatively new methodology for software development that aims at providing software modularity by means of separation of cross cutting concerns. This is an approach for requirements engineering that focuses on customers concerns to be made consistent with aspect oriented software development. In terms of software engineering throughout the code there are specifically designed points that support adaptation based on defined aspects.

The authors in [8] discuss how aspect orientation and can be used in context aware systems design. Furthermore because in aspect oriented programming direct user input is taken into account this is an example of human-centered context aware systems. However this approach although it goes into the right direction it is restricted by the predefined points that support aspect orientation in the implemented code.

Based on the analysis made the design and engineering process of human-centered context-aware systems should follow the following guidelines:

1. such a system should be as general as possible and should be able to tackle as many problems as possible [11];
2. such a system should should provide the right level of representation such that a problem representation could be automatically translated into the core constructs of the underlying programming language in which the overall system is implemented;
3. such systems should not be domain specific and therefor closed system, but an open system that could be used to address a wide range of problems and applications;

4. in such a system the focus should be on the needs of the end-user and not on the system itself; the system should be hidden from the end-user;
5. such a system should be designed for evolution and should provide the means to evolve through the input of users, as well as by itself;
6. such a system should support both skilled domain workers as well as novice users;
7. such a system should be a co-adaptive environment where users change because they learn and systems change because users become co-developers;
8. such a system should allow active participation and empowerment of end-users;
9. in such a system the end-user should be the coordinator of how the system works.

Some of the guidelines refer to the relationship between system and the end-user, and some concern just the system. The system that we envision is similar to an operating system in terms of being general and not domain specific. The system will be an intelligent one and will accept problem descriptions given by the end-users. These descriptions will act as application models. Such a description together with the intelligent system will make the application. We have already made initial steps towards implementing such a system in [18], [17].

4 Related Work

Context awareness has been studied for several decades from many perspectives. Development of context-aware application, however, is still very difficult and time consuming and aside location-based services, not too many applications have been put into real use.

A series of surveys addressing different aspects (i.e. context modeling and context-based reasoning, context-aware frameworks and applications, context-aware web services) of development of context-aware systems and their applications have been developed. We observe that the present approaches to context reasoning and context-aware system design are only partial, in most of the cases being too specific, or too generic.

Types of context used and models of context information systems that support collecting and disseminating context, and applications that adapt to the changing context have been addressed by Chen and Kotz in [6].

Bolchini et al. discuss in [4] general analysis framework for context models and a comparison of some of the data-oriented approaches available in the literature. The framework addresses: modeled aspects of context (space, time, absolute/relative, history, subject, user profile), representation features (type of formalism, level of formality, flexibility, granularity, constraints) and context management and usage (construction, reasoning, information quality monitoring, ambiguity, automatic learning features, multi-context modeling).

A unified architectural model and a new taxonomy for context data distribution has been discussed in [2]. Authors identify 3 major aspects: (1) context

data distribution should take into account node requests and quality of context requirements to reduce management overhead; (2) context data distribution requires adaptive and crosscutting solutions able to orchestrate the principal internal facilities according to specific management goals; (3) informed context data distribution can benefit from their increased context-awareness to further enhance system scalability and reliability.

The impact of context-awareness on service engineering has also been noticed. A classic and relatively recent survey [13] by Kapitsaki et al. considers context as constituting an essential part of service behavior, especially with the interaction of users. They observe that "at the heart of every context-aware service, relevant business logic has to be executed and (. . .) adapted to context changes".

Related work concerning this human-centered context-aware perspective, as it was analyzed in this paper is to the best of our knowledge only in an early stage.

5 Conclusions

In this paper we have started an initial discussion about the design and engineering of human-centered context-aware systems. Aspects discussed in this paper are part of a work in progress. Our previous experience [18], [17] with developing human-centered context-aware systems proved to be not trivial. This discussion comprised aspects from human computer interaction, meta-design, context and context-awareness. We have emphasized the fact that systems predesigned can not foresee all aspects and facets of a problem. Therefore end-user should be given the necessary tools to design and develop their own solutions based on existing services. We provide a set of guidelines and properties that should characterize such human-centered context-aware systems.

Next steps include formalizing a conceptual framework, methodology and implementation guidelines for developing such a system that is capable of tackling in a unified way the problem of development of human-centered context-aware applications.

References

1. Christos B. Anagnostopoulos, Athanasios Tsounis, and Stathes Hadjiefthymiades. Context awareness in mobile computing environments. *Wireless Personal Communications*, 42(3):445–464, 2007.
2. Paolo Bellavista, Antonio Corradi, Mario Fanelli, and Luca Foschini. A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys (CSUR)*, 44(4):50 pages, 2012.
3. David Benyon and Dianne Murray. Applying user modeling to human-computer interaction design. *Artificial Intelligence Review*, 7(3-4):199–225, 1993.
4. Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, and Letizia Tanca. A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26, 2007.

5. Thomas Burkhart, Dirk Werth, and Peter Loos. Context-sensitive business process support based on emails. In *WWW 2012 – EMAIL'12 Workshop*, April 2012.
6. Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Dartmouth College Hanover, NH, USA, 2000.
7. Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. Context is key. *Communications of the ACM - The disappearing computer*, 48(3):49–53, 2005.
8. Abhay Daftari, Nehal Mehta, and Shubhanan Bakre Xian-He Sun. On design framework of context aware embedded systems. In *Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation*, 2003.
9. Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.
10. Gerhard Fischer. Context-aware systems - the 'right' information, at the 'right' time, in the 'right' place, in the 'right' way, to the 'right' person. In *AVI'12*. ACM, 2012.
11. Gerhard Fischer and Elisa Giaccardi. *End User Development - Empowering People to Flexibly Employ Advanced Information and Communication Technology*, chapter Meta-Design: A Framework fo the Future of the End-User Development. Kluwer Academic Publishers, 2004.
12. Jonathan Grudin. Desituating action: digital representation of context. *Human-Computer Interaction*, 16(2):269–286, 2001.
13. Georgia M. Kapitsaki, George N. Prezerakos, Nikolaos D. Tselikas, and Iakovos S. Venieris. Context-aware service engineering: A survey. *The Journal of Systems and Software*, 82(8):1285–1297, 2009.
14. Charles McLellan, Teena Hammond, Larry Dignan, Jason Hiner, Jody Gilbert, Steve Ranger, Patrick Gray, Kevin Kwang, and Spandas Lui. *The Evolution of Enterprise Software*. ZDNet and TechRepublic, 2013.
15. Bonnie A. Nardi. *A Small Matter of Programming: Perspectives on End User Computing*. MIT Press, 1993.
16. Donald A. Norman. *The Invisible Computer*. MIT Press, 1999.
17. Emilian Pascalau. Mashups: Behavior in context(s). In *Proceedings of 7th Workshop on Knowledge Engineering and Software Engineering (KESE7) at the 14th Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2011)*, volume 805, pages 29–39. CEUR-WS, 2011.
18. Emilian Pascalau. Towards TomTom like systems for the web: a novel architecture for browser-based mashups. In *Proceedings of the 2nd International Workshop on Business intelligencE and the WEB (BEWEB11)*, pages 44–47. ACM New York, NY, USA, 2011.
19. Ian Sommerville. *Software Engineering 8*. Addison Wesley, 2007.
20. W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267, 2003.
21. Min Wang. Context-aware analytics: from applications to a system framework. http://e-research.csm.vu.edu.au/files/apweb2012/download/APWeb-Keynote-Min.pdf, 2012.
22. Jonathan Zittrain. *The Future of the Internet And How to Stop It*. Yale University Press New Haven and London, 2008.