User-Centric Software Development Process

Juan Carlos Augusto
Research Group on Development of Intelligent Environments
Dep. of Computer Science, Middlesex University, London, UK
e-mail: j.augusto@mdx.ac.uk

***Abstract*—The Software Engineering community has produced plenty of recipes on how to build software systems. Through the decades, we have seen a shift from the more structured and organized to the most flexible approaches, lately with emphasis on speed. Although these methods in principle apply to all software, in being generic they may not address specific needs of some areas where systems have special features and demands. We argue in this paper that Intelligent Environments is one such area to be considered and in need of a tailored software development process. We introduce a new software development process and explain how is to be applied in systems of the Intelligent Environments field.**

Keywords - Intelligent Environments; Software Engineering, Software Development Process.

I. INTRODUCTION

"An Intelligent Environment is one in which the actions of numerous networked controllers (controlling different aspects of an environment) is orchestrated by self-programming pre-emptive processes (e.g., intelligent software agents) in such a way as to create an interactive holistic functionality that enhances occupants experiences".(Vic Callaghan, [4]) Do we need a different way to build these systems? Here we argue that we do. Systems in this area are made of a collection of pre-existing fields, which grew up mostly independently: sensors, actuators, networks, interfaces, intelligent software, and human-computer interaction. Hence, it is understandable that despite the significant effort invested by companies and research centres around the world, these systems still remain an engineering challenge, from the simple deployment and maintenance issues to the core algorithmic problems and decision-making challenges of system organization. A sign of the formidable difficulties we face is that Intelligent Environments are still predominantly built in labs instead as of commercial products massively consumed around the world.

The technical literature does not offer specialized solutions to this problem. A recent survey by Preuveneers and Novais [2] covered requirements, component re-use and system verification, and other Software Engineering areas. The paper highlights that from an SE perspective, there are specific needs teams developing Intelligent Environments have to consider, however does not report on the general process of driving the development strategy. Not their fault, there is not such a guidance yet. This paper tries to remedy that.

Established Software Development Processes [3] can be roughly split into plan-driven (i.e., waterfall) or agile (i.e., SCRUM [1]).

II. A USER-CENTRIC APPROACH TO BUILD IES

A. User-centred approaches

One driving force for this research is to provide a system that is tailored to developers of Intelligent Environment systems. Closely linked with that is the need for this process to be user-centric, as this is a core value of the Intelligent Environments community [4]. To put it in simpler terms: the focus is on *How to make final users happier with what they ordered*?

B. A New Development Software Process

A new software model will not necessarily change the way software is developed in an area; however, it offers those with the willingness to develop software a guide that will remind them along the process of the important stages which define the ethos of an area.

Figure 1 shows in a picture the essence of the *User-Centric Software Development Process*, or U-C SDP for short. That diagram includes several typical stages of software development that we will find present in traditional software development models, e.g. the waterfall model. This figure, however, highlights the importance given to the intervention of the stakeholders driving the process. Only one step is numbered, other steps are performed following the arrows. Solid arrows mean mandatory and dotted arrows mean non-mandatory (but desirable). There are three stages which can be mixed in any order although usually they will be performed clockwise: the right hand side loop is the *Initial Scoping*, the central loop is the *Main Development* and the left hand side cycle is the *IE installation*.

A thesis worth assessing in the future is that this process should be used for both creation and maintenance. Although the refinements and fixings which may come after the development and deployment may go faster through these cycles, users should be involved in co-designing and perfecting the system in all those stages.

The idea with the first circular path is that the stakeholders should be in control and that IE infrastructure can (ideally: should) be checked by stakeholders (see dotted arrows) to make sure they accept it before prototyping is started. Although some projects claim to involve stakeholders, actually this does not happen in practice so this method is explicitly reinforcing this aspect. In addition, systems are installed in already build houses so the same emphasis on stakeholders' involvement is put on the validation stage. Prototyping, as other stages of the process, can be 'zoomed in' and decomposed in several steps and iterations. Behind the spirit of this proposal is that teams often have a product and then try to create an excuse to sell it. By forcing the revision of technology and

the acknowledgment of the stakeholders, we increase the chances that the stakeholders can veto inadequate technology being imposed to them by engineers.

The freedom of interplay between the second and third loops is deliberate. The rationale for that is that in some areas of Intelligent Environments, e.g., Ambient Assisted Living (AAL) [5], there is a lot of emphasis on testing systems in as close as possible to final deployment contexts. The so-called "Living Labs" [1]are labs which are inhabited by real final users (including proper homes kitted up). This usually complements the more traditional software development done in a university lab or in a company. These can co-evolve in many different ways and some development teams will put more emphasis in one or the other, they may start with one or the other and some will perform back and forth iterations amongst lab and real home development. No strategy is guaranteed to work better in all cases and hence they are purposefully left somehow detached so that our model remains flexible to adapt to different strategies.

C. *Ethics*

Ethics is mentioned in several system descriptions but rarely given any attention at system development level. To increase the chances that ethics is embedded in the product being developed the U-C SDP explicitly includes it as part of the process. Underlying all this development there should be an ethical framework which should be taken into consideration from beginning to end of the process and at all stages there should be specific actions taken to ensure the ethical layer of the system is translated accurately from one stage to the other [6].

III. U-C SDP EXPLAINED

This section explains the different elements in Figure 1. The depiction of the process has three main loops: Initial Scoping (on the right hand side), Main Development (low centre loop) and the IE installation (left hand side loop). Each of the sub-sections below describes the component elements and their connections.

A. *Initial Scoping*

1) Interview Stakeholders: each project should start by gathering the expectations of the stakeholders. The very essence of systems in this area is to serve humans. This principle shapes U-C SDP as through the different stages they can continuously monitor and influence the development with their opinions.

2) Define Required Services: the technical teams translate the information gathered from stakeholders into services the system will aim to provide.

3) Define Required IEs Infrastructure: the technical teams select the sensors, actuators and other devices and interfaces which will allow the materialization of the IE in the real world. The figure shows dotted arrows indicating it is suggested the technical team checks whether the proposed infrastructure to materialized the services is acceptable to the stakeholders and the stakeholders can accept or make alternative suggestions (which should be taken on board by the team).

4) Initial Design and Prototyping: this involves the use of methods and tools which allow the technical team to make an initial approach to the system. The result of this stage is shown and explained to the stakeholders, their suggestions for changes should trigger another iteration through the Initial Scoping cycle. This iteration will go very fast if changes are minimal or should be devoted the necessary time if they are more fundamental.

B. *Main Development*

1) Design: having the approval of the stakeholders at this stage implies a more detailed design analysis which should be strongly connected with the other stages of this cycle (i.e., create technical material which can feed testing and verification). Notice there is also a smaller loop between design and stakeholders indicating the desirability of making this step not an isolated stage but an interactive one with the stakeholders.

2) Implementation and Testing: this is about coding and testing that code. Testing should consider software, hardware and human-computer interfaces.

3) Verify Correctness: verification (e.g., through model checking) is one the most rigorous soundness checks available which can [8] and should be performed on systems in this area. Notice the dotted arrows to indicate verification and testing are complementary and should be used in conjunction to make sure the system built is correct.

4) Interview Stakeholders: stakeholders should be also involved on testing and approving the final functionality obtained through the interfaces and other aspects of the system they will experience.

C. *Intelligent Environment Installation*

1) Equipment Validation: deploying starts with the infrastructure (hardware, network, devices, interfaces) and this step should have a separate safety and reliability check. Users can check if the infrastructure deployed is acceptable for them (location, maintenance required, and other practical aspects of its presence).

2) Software Validation: software is deployed on the infrastructure and the behaviour of the system can be experienced and tested by users. As in equipment testing the user can object on individual parts of infrastructure, here users can do the same on specific functions of the systems.

3) Services Validation: this involves the stakeholders experiencing the system for significant periods of time on a continuous basis (e.g., through Living Labs).

4) Interview Stakeholders: Their feedback after equipment or software testing goes to the development team. A problem in any of those system components at this stage can lead to redesigning and redevelopment of the system by going back to any of the other main loops. After the system has been improved it comes back to another installation exercise.

These main loops and secondary loops can be executed as many times as needed. The different combinations of

[1] http://www.openlivinglabs.eu/

paths can be exercised in a specific rigid order and with the full breath of services in mind from the very beginning, more like in the Waterfall model, or with smaller objectives in faster cycles more like in incremental and agile models. However, as most systems in this area are safety critical, the slower and more careful approaches are strongly advisable.

D. Ethical Framework

Frequently papers within this field refer to ethical issues, rarely these are implemented. Within our research group we have considered the problem of finding a suitable ethical framework for the development of systems in this area and our resulting proposal is eFRIEND (see [4]). This ethical framework is based on the principles that AAL systems should provide services which collectively are consistent with the following higher level ethical principles: non-maleficence and beneficence, user-centred multiple user groups, privacy, data protection, security, autonomy, transparency and openness, equality, dignity and inclusiveness of provision. These principles are designed to protect users from informal and rushed system development. These generic principles are translated in eFRIEND to specific AAL features of a specific system. We provide examples for these: 1) *Non-Maleficence and Beneficence:* the system should avoid causing harm to any of the users and the system should proactively seek for opportunities to help and this help should be agreed by the users in advance, 2) *User-Centred*: the type of technology and associated services should be agreed with the users in advance, 3) *Multiple User Groups:* The system should be aware of the different needs and preferences of all individuals, 4) *Privacy:* Users decide on the level of acceptable monitoring, tracking and recording of activities, 5) *Data Protection:* Users have access to the sensitive information stored about them, 6) *Security:* The system should protect the individuals whom it is helping, 7) *Autonomy:* User can selects degree of protection, 8) *Transparency:* All users should be clearly informed of the pros and cons of the services offered by the system, and 9) *Equality, Dignity and Inclusiveness:* The system should provide help regardless of age and technical background and ability.

We believe these principles should be taken into account when developing Ambient Assisted Living systems and this ethical framework overall should be used to inform development from early stages of gathering requirements and planning to later stages of validation and deployment.

IV. U-C SDP IN PRACTICE

Our User-Centred Software Development Process model is more specific than other well-known models which have been used for decades in software engineering; still it keeps a degree of generality to be applied to any area where user-centred systems are built. Intelligent Environments is one such area where users are at the core of systems conception and U-C SDP is well-suited to guide systems throughout all stages.

As an example let us consider a project from the AAL area which was funded by the UK government in collaboration with government-funded healthcare organizations (NHS trusts) and a company (Fold) focused on the provision of services for elderly people. The project was called NOCTURNAL (Night Optimized Care Technology for UseRs Needing Assisted Lifestyles) and its goal was to produce a commercially viable technological infrastructure which, based in sensors, can increase safety of elderly people at night time. Movement sensors were deployed in each room of the house and pressure pads were used in the main bed to allow the system to know where the person being cared for was. Lights and a bedside unit which was capable to provide calming music and images were used as actuators. For an overview of this project see [7].

During the *Initial Scoping* phase different group of stakeholders were interviewed and their views taken into account (the financial constraints from the company, what the NHS organizations and the users considered acceptable services and technology where amongst the most important issues to be balanced). These interviews were all face to face. As a result a non-intrusive platform was defined based on existing technology from the company. This information allowed us to create an initial infrastructure which was capable of delivering the acceptable services agreed by the main stakeholders. The team has clear that the first goal to achieve was to be able to deliver the expected services and then subsequent iterations will be used to do that in the optimal way for the company (e.g. maximizing reliability and minimizing cost).

The *Main Development* phase focused on materializing the initial conception of the system. The technical teams in the consortium used formal methods to model the system at different levels of abstraction, to simulate the possible emergent behaviour of different alternative solutions, and to verify that specific requirements were present in the behaviour emerging from those modelled solutions. This was interleaved with coding so the models and code were expanding in synchrony [10]. In the first iteration different stakeholders tried different parts of the system which were closer to their interest and we gathered feedback. In subsequent iterations they assessed the whole infrastructure, to the extent their technical knowledge naturally allowed them.

The *Intelligent Environment Installation* phase was very important in all three iterations and in all cases it involved final users experimenting with the system in their own living spaces (homes or sheltered accommodation). The first iteration was a bit dissociated in the sense that equipment was experienced partially by different stakeholders. Whilst the technical partners were experimenting with sensors at their working places the users and NHS partners were assessing the interfaces. Iterations 2 and 3 involved a more holistic validation, just that with different sensing technology. Stakeholder's feedback was used to inform each of the next iterations.

The first iteration was based on a system which was provided by another company to Fold and although it allowed to define the services a platform which was shaped from the hardware onwards was preferred. Two iterations were performed which subsequently focused on a new hardware platform and the last one focused on improving the efficiency of the platform. The interesting part of the way the system progressed was that the initial stage required a few iterations on the first circle (Initial

Scoping) with three separate elements: services, infrastructure and interface. The first iteration consisted of those three loosely integrated parts to make sure they were acceptable. Once these were approved the system progressed into the main development stage changes were less radical and the infrastructure changed but not the philosophy and way of service delivery. Hence iterations 2 and 3 were more related to the company's technological offer and the algorithms the university team used to materialize the services with different technology. The modelling, design, simulation and verification stages [9] remained very much the same and the adjustments were more at the interface between service programming and sensors/actuators. The overall lessons learnt from that experience were that the U-C SDP was organized but at the same time flexible enough to accommodate for changes and refinements which delivered the best compromise for the stakeholders within the resources available.

V. CONCLUSIONS

We have introduced a new process to guide the development of Intelligent Environments. This is needed because traditional methods do not specifically focus on the importance of stakeholders and the different technological components of IEs. We have provided an initial validation with the development of an AAL system. There are noticeable advantages of flexibility which allow developers to follow different strategies akin to more traditional strategies. Our method has been engineered to provide more specific support to the IE community.

REFERENCES

[1] M. Cohn (2009). Succeeding with Agile: Software Development Using Scrum. Addison-Wesley. ISBN-13: 9780321579362

[2] D. Preuveneers, P. Novais (2012). A survey of software engineering best practices for the development of smart applications in Ambient Intelligence. Journal of Ambient Intelligence and Smart Environments 4:3 (149-162).

[3] I. Sommerville (2011) Software Engineering, 9th Edition, Pearson. ISBN 13: 9780137053469.

[4] J. C. Augusto, V. Callaghan, A. Kameas, D. Cook, I. Satoh (2013). Intelligent Environments: a manifesto. Human-centric Computing and Information Sciences, 3:12, 2013. Springer.

[5] J.C. Augusto, M. Huch, A. Kameas, J. Maitland, P. McCullagh, J. Roberts, A. Sixsmith, and R. Wichert. (Eds.) (2012). Handbook on Ambient Assisted Living. Volume 11 of the Ambient Intelligence and Smart Environments Book Series, IOS Press.

[6] S. Jones, S. Hara, J. C. Augusto (2013). eFRIEND: an Ethical Framework for Intelligent Environment Development. To appear in Proc. of 7th Int. Conference on PErvasive Technologies Related to Assistive Environments (PETRA 2014), 27-30 of may, 2014.

[7] J. Augusto, M. Mulvenna, H. Zheng, H. Wang, S. Martin, P. McCullagh, J. Wallace (2014). Night Optimised Care Technology for Users Needing Assisted Lifestyles. Behaviour and Information Technology. doi:10.1080/0144929X.2013.816773 Taylor&Francis.

[8] J. Augusto, M. and Hornos (2013): Software Simulation and Verification to Increase the Reliability of Intelligent Environments. Advances in Engineering Software, Vol. 58, Pages 18-34, Elsevier.

[9] J. Augusto, H. Zheng, M. Mulvenna, H. Wang, W. Carswell, P. Jeffers (2011): Design and Modelling of the Nocturnal AAL Care System. Proc. 2nd Int. Symposium on Ambient Intelligence (ISAmI 2011), pp. 109-116. Salamanca - Spain. Springer Verlag.

[10] A. Gravell, Y. Howard, J. C. Augusto, C. Ferreira, and S. Gruner (2003): Concurrent Development of Model and Implementation. Proc. of 16th Int. Conference on "Software and Systems Engineering and their Applications". Paris, France. .

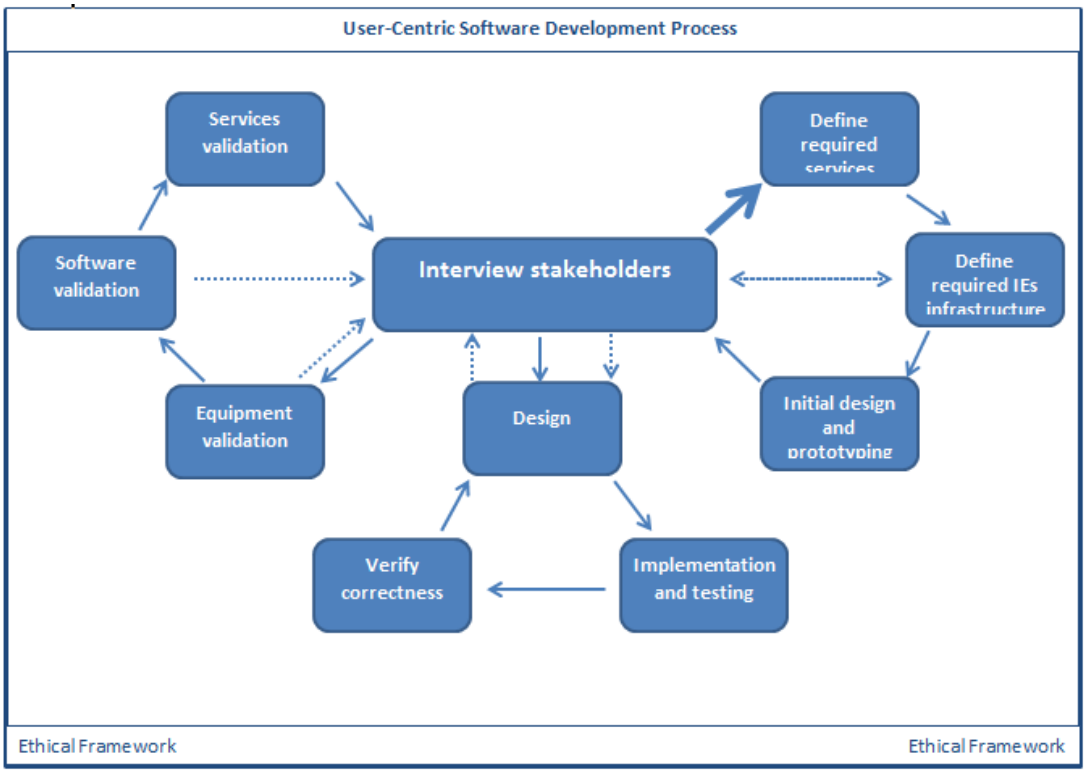

Figure 1. holistic view of the User-Centred Development Software