Agile-User Experience Design: an Agile and User-Centered Process?

Lou Schwartz
Public Research Centre Henri Tudor
Luxembourg, Luxembourg
lou.schwartz@tudor.lu

***Abstract*—Agile-User Experience Design, also called Agile-UX, is a trend of the last decade that mixes values and practices from the Agile software engineering methods and the User-Centered Design. Several practitioners have proposed different processes to organize the work between development and design. After a short reminder of the values of Agile and User Centered Design methods, this paper presents five processes proposed in the literature. The processes are discussed with regards to their respect of the Agile and User Centered Design values. This comparative study concludes that not one process totally covers the Agile and User Centered Design values: they all make a trade-off and could be completed by practices and by a state of mind and a willingness adopted by the team.**

Keywords-Agile; Agile-UX; Agile Software Techniques; Software Engineering; User-Centered Desing;

I. INTRODUCTION

Since a decade, several software companies, or only at the teams' level, try to integrate Agile software development methods and User Centered Design (UCD) [6][8][14][19]. This integration, called Agile-User Experience Design or Agile-UX, is bound on the one hand to the interesting performance of Agile methods to quickly provide software that answers the users' needs with a certain level of quality, and on the other hand it results in the observation that this software quality is relative, particularly related to Human Computer Interactions aspects [3][18]. Based on this observation, several practitioners tried to integrate UCD in their Agile process with various degrees of success. After a reminder of Agile and UCD methods in section II and III, this paper will present processes used to integrate Agile and UCD, often addressed in the literature in section IV and discuss them regarding their respect of the agile and UCD values in section V.

II. AGILE METHODS

The Agile methods' goal is to enhance the value of the delivered product in order to satisfy the customer's requirements. Agile methods adopt the following four values defined in the Agile Manifesto [1]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

The Agile movement was instigated and pioneered by software developers in reaction to a frustration emerging from history of delayed projects, budget overruns and stressful jobs [2]. For the Agile Manifesto founders, these problems have their origin in a too big analysis, specification and design done before code writing that enables volatile or useless requirements and incompleteness. With the Agile methods, customers would obtain faster working software that corresponds better to their real requirements, thanks to the flexibility provided to the development process [2].

Agile methods are focused on the developers' work and on the development quality [4]. Even if the aim of Agile methods is to satisfy the product owner's (who is the representative of stakeholders: customers and end-users) requirements, they define neither method nor good practices to achieve this objective, particularly for the needs elicitation or the design part. The needs elicitation is done by the product owner, based on his own knowledge of the domain or of the work done by users. He can use the methods he wants, including involving the users (e.g., by interviews, context inquiries, etc.). The user interface design depends on the openness to ergonomics of developers, customer and users. So there is no guarantee about it. [4]

The use of the UCD principles and methods is one way to ensure answering to users' needs. Based on these assessments, Agile teams can benefit from the integration of UCD methods with Agile to improve, in particular, the needs elicitation and the design part.

III. USER-CENTERED DESIGN

UCD focuses on producing usable software that satisfies real end-users and customers. This method, described by the standard ISO 9241-210 [9] defines the process to follow to produce software that meets the users' requirements. It includes in particular the design and the validation stages.

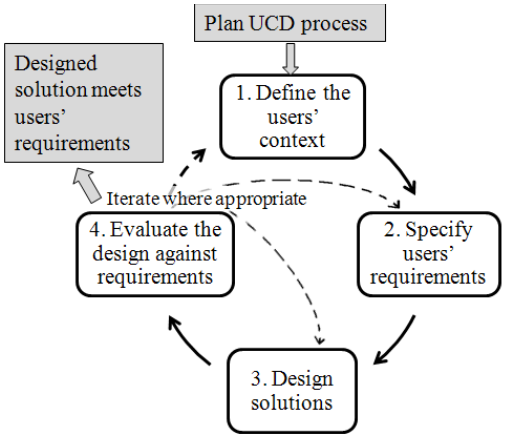

Figure 1. UCD process as described by the standard ISO 9241-210 [9].

 ISBN: 978-1-61208-304-9

Four activities compose the UCD process (see Figure 1.):

- Understanding and specifying the context
- Specifying the user needs
- Produce design solutions to meet user requirements
- Evaluate the designs against requirements

The principles of the UCD are listed below [9]

- The design is based upon an explicit understanding of users, tasks and environments
- Users are involved throughout the design and the development
- The design is driven and refined by user-centered evaluation
- The process is iterative
- The design addresses the whole user experience
- The design team includes multidisciplinary skills and perspectives

Even if some Agile concerns could prevent a UCD attitude [4] (the focus is often more on programming techniques and programmers, automated tests, very short iterations and fast increments) a reconciliation of both approaches is possible and has often been implemented [6][8][11][12][14][15][19]. The integration of both methods implies focusing more on design activities. It results to a redefinition of the process to organize the activities dedicated to the design and the process dedicated to the development.

IV. REVIEW OF THE AGILE-UX PROCESSES PRESENTED IN THE LITERATURE

A major issue listed in the literature about Agile-UX is the organization of the work between development tasks and UCD tasks in respect to the Agile and UCD values. Among the existing work, five propositions of process design are studied in this section.

A. *Parallel tracks*

To manage exchanges and to organize the work to carry out between developers and usability experts, Sy [19] proposes that they work in parallel tracks after the planning iteration also called iteration "0". It enables usability experts to keep ahead of the developers, to have enough time to gather users' data, to analyze that data and to propose design solutions. For that, designers and developers work with one to two iterations of delay (see Figure 2.). During the iteration *i*, designers:

- Gather user and context data for the iteration *i*+2
- Work on the designs for the iteration *i*+1
- Help developers for the implementation of the designs of the iteration *i*
- Evaluate the software developed during the iteration *i*-1

The principle of parallel tracks is well acclaimed by usability experts who test it [6][15][19] thanks to the proactive attitude given to them. As any method, the Sy's process has advantages and potential issues.

The advantages of working ahead of the development team [14] are:

- Better definition of the conditions of satisfaction (test acceptance criteria)
- Better planning the design
- Better inclusion of designs in the global users' process
- Designers can be more concentrated on exceptions rather than trying to produce the best design right the first time.

The potential issues of parallel tracks are [14]:

- Sensation of not being one team that can give a vision of inequality
- Exclusion or self-exclusion of usability experts of some meetings
- Risks of the lack of communication which could lead to misunderstanding and resentment
- Forget to rectify issues noted during previous iteration's tests.

To avoid these issues two solutions are proposed [14]: encourage communication, build common channels of communication; and give helpful assistance to developers as soon as possible when a design is not understood.

This iterative process covers the four UCD activities and it also respects the following UCD principles:

- Understanding of users, tasks and environment: the activities of gathering data on user and context are scheduled.
- Users' involvement: users can be involved for the gathering of data and for design, but they are particularly consulted to test the developments.
- Evaluation: software tested by users.
- Iterative: intrinsic to the process.
- Multidisciplinary: by the involvement of designers, developers and stakeholders.

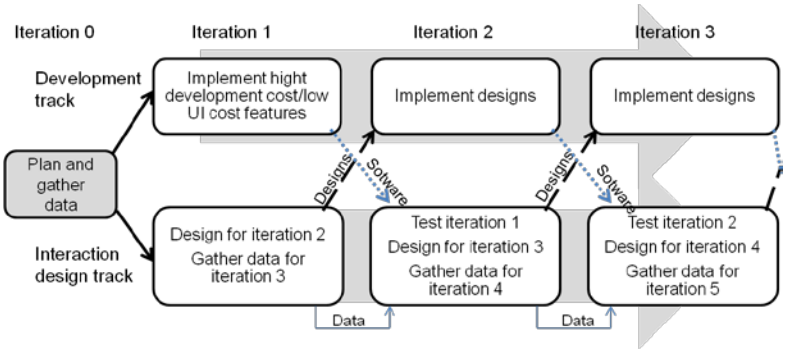

Figure 2. Sy's parallel tracks of work [19].

B. Design work done on parallel levels

Armitage [2] proposes another type of parallel work that concerns only the designers' work organization. The design work is done on three parallel levels (see Figure 3.) from unit to global level:

- Provide detailed designs for the requirement developed in the current or next iteration.
- Redesign software developed in previous releases (a release is a set of several iterations).
- Provide overall product vision, to keep a global coherence throughout the project and developed software.

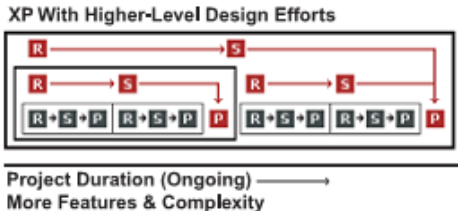

Figure 3. Parallel design tasks presented in [2].

This process covers the four UCD activities. The evaluation of designs against requirements is supported by the redesign activities and an overall product vision.

This process respects the following UCD principles:

- Understanding of users, tasks and environment: the focus is on the design in this process, that encloses the respect of this principle
- Iterative: the process is iterative on each level
- Multidisciplinary: involvement of designers.
- However it is not clear if the evaluation of the designs are driven and refined by users (third UCD principle) or if the users are involved (second UCD principle) but there is no counter-argument to respect these principles.

C. Sequence of an iterative design phase and an iterative development phase

Deuff et al. [8] present another proposition of process for Agile-UX that gives a good place to an upfront designing (see Figure 4.). They justify this iterative design phase by the fact that time is necessary before development to build the context (gather data on users, their tasks, the context, etc.) and make the first design propositions. But, the time is not available in classical Agile processes. So usability experts have to juggle between too much tasks (gather the necessary data, define the design, test) while trying to maintain a global vision during iterations. To resolve this issue they propose to cut the project in 3 phases: Design, Development and Final test. The design and the development phases are iterative. Even if a Final test is planned, several users' tests are done during the first and second phase.

This process covers the four UCD activities if the phase 1 is dedicated to understanding and specifying the context of use, specifying the user and organizational requirements and producing design solutions. But the description of the process is not deep enough to ensure that phase 1 covers these activities. The evaluation of designs against requirements is covered by Phase 3 and by the regular tests done throughout the project.

Figure 4. Deuff's process proposal [8].

This process respects the following UCD principles:

- Understanding of users, tasks and context: notably through phase 1 of designing
- Users' involvement: users are involved throughout the project in particular thanks to regular testing.
- Evaluation: design and software are iteratively evaluated by users and it is enhanced by phase 3 which plans a final users' test
- Multidisciplinary: designers and users involvement.

The fourth (iterative) UCD principle is more or less respected since the first and second phases are iterative but a global loop is missing.

D. Big upfront design

Agile methods do not encourage a big upfront design [4][14][15]. Or more precisely this upfront design is out of the scope of the Agile methods. In fact an analysis conducted by the product owner is necessary to define the product backlog, but no best practice is defined to support the product owner for this task, which is done before the start of the development Agile process. To support the product owner for this task, some usability experts propose to conduct a big analysis up front. Others are against this practice and prefer to use the iteration called "zero" to conduct a short analysis and then go deeper throughout the project according to the needs of analysis. Big upfront design in Agile-UX has supporters and opponents (see TABLE I.), their arguments are presented bellow.

1) Supporters of a big upfront design: Chamberlain [6] in his principle 4 for integration UCD and Agile development insists on a big upfront design before any development: "UCD practitioners must be given ample time in order to discover the basic needs of their users before any code gets released into the shared coding environment." This time is necessary to capture users' needs, usability goals, context of use and design criteria. It is also used to define users or to build personas. In some cases, at least a part of the designs is defined in this step which is not recommended by Nodder [14]. It's even risky according to Blomkvist [4] and Deuff [8] to engage a project in a development without this initial analysis and design. Agile methods are intensive during iterations, so that usability experts do not always have time to ask questions or to take a global view and ensure the homogeneity and consistency of the solution.

For Brown [5], long research projects are sometimes necessary to devote more time in analysis in order to gather the necessary data.

 ISBN: 978-1-61208-304-9

TABLE I. REPARTITION OF OPPONENTS AND SUPPORTERS OF A BIG UPFRONT DESIGN AND THEIR ARGUMENTS

		[2]	[4]	[6] Prj. I	[7]	[5]	[8]	[11]				[14] Prj. PV	[15]
								Prj. 1	*Prj. 2*	*Prj. 3*	*Prj. 4*		
Supporters	Do first analysis and design		X	X			X		X	X	X		X
	Avoid risks		X				X						
	Have a global vision		X		X	X	X					X	X
Opponents	Avoid risks (time & money consuming)	X		X		X		X					
	Respect Agile values: accept changes	X				X		X					X
	Big upfront analysis reduce quality	X											

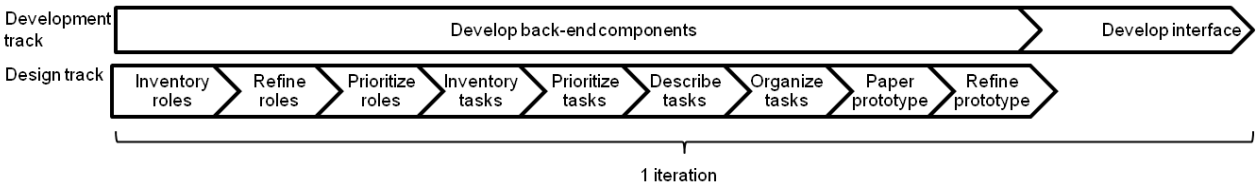

Figure 5. One iteration in Usage-Centered Design adapted to Agile methods [7].

2) Opponents of a big upfront design: In opposition, for Armitage [2] it is too risky and time and money-consuming to design deeply beforehand and it is totally against Agile practices which encourages "trial and error to reduce the risk of building the wrong thing". A big upfront design might reduce quality of the software and its design [2]. Another problem is the difficulty to accept changes later when a big upfront design was done, which goes against the Agile values "Responding to change over following a plan" and "Working software over comprehensive documentation" [2][11][15]. For Brown [5], gathering data or needing design validation is not a justification for an upfront design phase: these tasks can be conducted throughout the project thanks to the planning of regular meetings with users. These meetings can serve to discuss all the elements already built (wireframes, personas, software, etc.) with users but also to gather data on their tasks etc.

3) Conclusion: For Brown [5], it is a myth that no upfront design is allowed in Agile-UX. In fact, Agile developers all work with a kind of high-level plan also called a roadmap. It is also necessary for usability experts to develop a kind of roadmap in the form of, e.g., a simple sketch, a workflow diagram, wireframes or Post-its. This way the team has to take the time to build this global vision while taking care not to spend too much time and fall into the track of a design phase that never ends. This is necessary to identify proactively technical impediments.

This process covers only three of the four UCD activities: understanding and specifying the context of use, specifying the user and organizational requirements, and producing design solutions. Furthermore it depends on the tasks done in this big upfront design phase: in fact the proposition of designs is not always included, sometimes it is diluted in the iterations following this first phase. So a big upfront design is not enough to ensure that designs will meet the users' requirements.

This process does not ensure the second (users' involvement), the third (evaluation), the fourth (iterative) and the sixth (multidisciplinary) UCD principles even if they are recommended to ensure a better design. In fact the goal of this process is to answer to the first UCD principle: understanding of users, tasks and context.

E. Usage centered design

Constantine [7] proposes another approach, which is the integration of Usage-Centered Design, and not User-Centered Design, and Agile (see Figure 5.).

Usage-Centered Design is more focused on roles than on users and on usage scenarios also knew as task cases. Roles and tasks are identified by stakeholders (domain experts, business people, designers, developers, users, etc.) thanks to brainstorming. The process is composed of iterations that are all composed of these succeeding steps: (1) Inventory roles; (2) Refining roles; (3) Prioritizing roles; (4) Inventory tasks; (5) Prioritizing tasks; (6) Describing tasks; (7) Organizing tasks; (8) Paper prototype; (9) Refining of prototype. During this time developers develop the back-end components. When the prototype is refined, they develop the interface.

This process covers only three of the four UCD activities: understanding and specifying the context of use, specifying the user and organizational requirements, and producing design solutions. The evaluation of designs against requirements is not covered; it goes against the third UCD principle [16].

As stakeholders are consulted to define roles and tasks, the second (users' involvement) and sixth (multidisciplinary) UCD principles are respected. The process is intrinsically iterative (principle 4.). The good definition of roles and tasks

 ISBN: 978-1-61208-304-9

answers to the first UCD principle, even if the process does not ensure the understanding of the environment.

V. DISCUSSION

The facing of the presented Agile-UX processes to the UCD activities shows that they respect generally the four UCD activities (see TABLE II.). However the activity of evaluation is not covered by the big upfront design or by the Constantine's process: they have to be completed.

None of the presented processes ensures the fifth UCD principle (see TABLE II.), this principle aims to improve the whole user experience by addressing the support of users in their use of the product. This can be addressed by all processes if the willpower to care about it exists in the project and in the team. Sy's, Deuff's and Constantine's processes clearly involve users at least to support the evaluation of designs and software and/or to define the context and the needs (see TABLE II.). Armitage's and the big upfront design processes do not ensure this involvement and evaluation, but the respect of these UCD principles is recommended to improve the designing and the meeting of the users' needs (see TABLE II.). Sy's, Armitage's and Constantine's processes are strongly iterative (see TABLE II.). Deuff's process is more or less iterative, this is due to the introduction of an upfront analysis separated from the development phase, but each phase is iterative (see TABLE II.). For the big upfront process it is recommended to make it iterative but it is not ensured (see TABLE II.). Finally, all presented processes involve at least designers (see TABLE II.), and even if some of them do not ensure the involvement of developers or stakeholders, including the end-users, it is at least recommended.

Evaluate these Agile-UX processes under the Agile values is not an easy task. Firstly, as they are processes they can go instead of the first Agile value (Individual and interactions over processes and tools) (see TABLE II.). We can understand that processes promote a separate analysis and design phase, as Deuff's and big upfront design, are certainly more rigid and thus do not encourage the third Agile principle (Customer collaboration over contract negotiation) by fixing the designs before development and the discovery of impediments (see TABLE II.). As the separate design phase aims to produce designs, we can deduct that both processes do not promote the second (Working software over comprehensive documentation) (see TABLE II.). The more iterative attitude of the Sy's, Armitage's and Constantine's processes respects better the third Agile principle (see TABLE II.). Sy and Constantine both insist on the necessity to reduce documentation by doing designs (as paper prototypes) but only when it is essential for communication and exchange or to support specification of the user stories, in the respect of the second Agile value (see TABLE II.).

Finally, respecting the Agile values is more a question of attitude adopted by the team, a question of culture, that something intrinsic to the Agile-UX emerging processes.

TABLE II. AGILE-UX PROCESSES FACING TO UCD ACTIVITIES AND PRINCIPLES AND TO AGILE VALUES

		Sy's process	Armitage's process	Deuff's process	Big upfront design	Constantine's process
UCD Activities	1. Specify context	X	X	X	X	X
	2. Specify users' needs	X	X	X	X	X
	3. Design	X	X	X	X	X
	4. Evaluate	X	X	X	NO	NO
UCD principles	1. Design based on explicit understanding of users, tasks and environment	X	X	X	X	X
	2. Users involved	X	Not ensured	X	Not ensured but recommended	X
	3. Design driven and refined by user-centered evaluation	X	Not ensured	X	Not ensured but recommended	NO
	4. Iterative process	X	X	More or less	Not ensured but recommended	X
	5. Process addresses the whole user experience	Not ensured	Not ensured	Not ensured	Not ensured	Not ensured
	6. Team includes multidisciplinary skills	X	X	X	Not ensured but recommended	X
Agile Values	1. Individual and interactions over processes and tools	Not ensured	Not ensured	Not ensured	Not ensured	Not ensured
	2. Working software over comprehensive documentation	Not ensured but promoted	Not ensured	Not ensured	Not ensured	Not ensured but promoted
	3. Customer collaboration over contract negotiation	Not ensured	Not ensured	Not ensured	Not ensured	Not ensured
	4. Responding to change over following a plan	X	X	More or less	NO	X

 ISBN: 978-1-61208-304-9

VI. CONCLUSION AND FUTURE WORK

Even if the parallel tracks process is generally accepted, some other processes are proposed. This echoes Brown [5] who explains that one myth of Agile-UX is to believe that there is only one way to do it. Every team has to find its proper way to process Agile-UX because "different challenges require different solutions". This corresponds perfectly with Agile values, notably "Individuals and interactions over processes and tools".

Following the analysis of the different Agile-UX processes proposed in literature, we can observe that no one covers entirely all the UCD activities, UCD principles and Agile values. To ensure the respect of all these principles, each analyzed process should be completed by practices or by cultural aspects. For instance Constantine's process should be completed by tests. Armitage's process works more on the global vision than the other processes, it may be associated with Sy's process to improve it. Deuff's process makes a major contribution on the organization of the tests (not detailed in this paper) but the separation of an analysis phase and a development phase are in contradiction with Agile that fights against upfront analysis and design phase by its fourth principle (Responding to change over following a plan). This analysis brings out questions to investigate in future work:

- Which practices are necessary to complete the Agile-UX processes?
- What can be an Agile-UX process that respects all UCD and Agile principles?
- How may the people and the cultural question enhance the Agile-UX processes?
- How to ensure the respect of the fifth UCD principle: process addresses the whole user experience?

ACKNOWLEDGMENT

Many thanks to my colleagues (in particular Sylvain Kubicki) who support my work on Agile-UX through exchanges and experimentations and those who have helped me in particular to write this paper: Muriel Foulonneau and Jocelyn Aubert.

REFERENCES

[1] A. Alliance, Agile manifesto, 2001, Online at http://www.agilemanifesto.org, 08.01.2013.

[2] J. Armitage, Are Agile methods good for design?, interactions, 11(1), 2004, pp. 14-23.

[3] A. Bankston, Usability And User Interface Design In XP, 2003, White Paper, http://www.ccpace.com/resources/documents/usabilityinxp.pdf, 08/01/2013.

[4] S. Blomkvist, Towards a model for bridging Agile development and user-centered design, In Human-Centered Software Engineering—Integrating Usability in the Software Development Lifecycle, 2005, pp. 219-244, Springer Netherlands.

[5] D. D. Brown, Five Agile UX Myths, Journal of Usability Studies, 8(3), 2013, pp. 55-60.

[6] S. Chamberlain, H. Sharp, and N. Maiden, Towards a framework for integrating Agile development and user-centred design, In Extreme Programming and Agile Processes in Software Engineering, 2006, pp. 143-153, Springer Berlin Heidelberg.

[7] L. L. Constantine and L. Lockwood, Process agility and software usability: Toward lightweight usage-centered design, Information Age, 8(8), 2002, pp. 1-10.

[8] D. Deuff, M. Cosquer, and B. Foucault, Méthode centrée utilisateurs et développement Agile: une perspective «gagnant-gagnant» au service des projets de R&D, In Conference Internationale Francophone sur l'Interaction Homme-Machine. Sept. 2010, pp. 189-196, ACM.

[9] I. DIS, 9241-210: 2010, Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems, 2009, International Organization for Standardization (ISO), Switzerland.

[10] Extreme Programming: a gentle introduction, http://www.extremeprogramming.org/, 08.01.2013.

[11] J. Ferreira, J. Noble, and R. Biddle, Up-front interaction design in Agile development, In Agile Processes in Software Engineering and Extreme Programming, 2007, pp. 9-16, Springer Berlin Heidelberg.

[12] Z. Hussain, W. Slany, and A. Holzinger, Current state of Agile user-centered design: A survey. In HCI and Usability for e-Inclusion, 2009, pp. 416-427. Springer Berlin Heidelberg.

[13] D. Kane, Finding a place for discount usability engineering in Agile development: throwing down the gauntlet, In Agile Development Conference, 2003, ADC 2003, Proceedings of the, Jun. 2003, pp. 40-46. IEEE.

[14] P. McInerney and F. Maurer, UCD in Agile projects: dream team or odd couple?, Interactions, 12(6), 2005, pp. 19-23.

[15] C. Nodder and J. Nielsen, Agile usability: best practices for user experience on Agile development projects, Nielsen Norman Group, 2010.

[16] A. Nummiaho, User-Centered Design and Extreme Programming, In Software Engineering Seminar, 2006, pp. 1-5.

[17] D. Rawsthorne and D. Shimp, Scrum In A Nutshell, SCRUM alliance, http://www.scrumalliance.org/articles/151-scrum-in-a-nutshell , 08.01.2013.

[18] A. Seffah, J. Gulliksen, and M. C. Desmarais, Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle, 2005, p. 32, Springer

[19] D. Sy, Adapting usability investigations for Agile user-centered design, Journal of usability Studies, 2(3), 2007, pp. 112-132.

 ISBN: 978-1-61208-304-9