AIPM: an Agile Inclusive Process Model

Rodrigo Bonacin *Marcos Antônio Rodrigues*
Maria Cecília Calani Baranauskas

Technical Report - IC-08-09 - Relatório Técnico

April - 2008 - Abril

AIPM: an Agile Inclusive Process Model

Rodrigo Bonacin[*] Marcos Antônio Rodrigues[†]
Maria Cecília Calani Baranauskas[‡]

Abstract

The Internet represents a new dimension to software development. It can be understood as an opportunity to develop systems to promote social inclusion and citizenship. These systems impose a singular way for developing software, where accessibility and usability are key requirements. This technical report proposes a process model for agile software development, which takes into account these requirements. This method brings together multidisciplinary practices coming from Participatory Design, and Organizational Semiotics with concepts of agile methods.

1 Introduction

The increase in software complexity and consequently development costs, as well as the demand for quality and productivity resulted in the need for organizing the software development tasks. Many software engineering research projects attempt to establish process models to make the software development a more predictable task. Some process models are very systematic, and advocate performing many extra activities in addition to the core development ones. These activities usually produce a lot of documentation and demand a lot of resources.

Another problem of many processes is the difficulty to deal with context changes during the software development. The agile methods [1], aim to be flexible to deal with these changes, focusing on the core development activities. The quality and productivity are achieved by focusing on the individuals, by working most of the time with the software itself, by collaborating with customers, and by being agile enough to respond to changes.

Nowadays, the Internet represents a new dimension to the software development, where systems, people, and business can globally communicate. However, the internet is not accessible for everyone, especially in the development countries with many illiterate people. According to the World Wide Web Consortium [21], the social value of the Web is that it enables human communication, commerce, and opportunities to share knowledge. These

[*]Centro de Pesquisas Renato Archer, Rodovia Dom Pedro I, km 143,6, 13069-901, Campinas, SP and CESET, University of Campinas, R. Paschoal Marmo, 1888, 13484-332, Limeira, SP

[†]Centro de Pesquisas Renato Archer, Rodovia Dom Pedro I, km 143,6, 13069-901, Campinas, SP

[‡]Institute of Computing, University of Campinas, 13081-970 Campinas, SP. This work is funded by Microsoft Research - FAPESP Institute for IT Research, grant #2007/54564-1

benefits should be available to all people, independently of their hardware, software, network infrastructure, native language, culture, geographical location, physical or mental ability. These aspects are related to both social and technological issues.

To produce systems that are accessible by everyone, represents a big challenge to the Human-Computer Interaction field. This work is situated as part of the project entitled: e-Cidadania: "Systems and Methods for the Constitution of a Culture mediated by Information and Communication Technology" [4]. The project investigates and proposes solutions for the diversity of users and competencies that constitute the scenario of the digitally excluded people in the Brazilian society (which include illiterate and impaired people). To reach this goal, the research group develops joint actions with a partner institution (network Jovem.com and communities around it) to conduct interaction and interface design of a pilot system to support inclusive social networks, to be implemented in the target community.

The work reported here deals with the problem of building those systems considering the quality and productivity during the software development process. The approach relies on bringing together multidisciplinary practices coming from Participatory Design (PD), and Organizational Semiotics (OS) with concepts of agile methods.

This report is organized as follows: the second Section presents the background for this proposal including Agile Methods, Human-Computer Interaction, and Organizational Semiotics; the third Section delineates the process model; the fourth Section details principles of the proposed process model, lifecycle and practices; the conclusion Section summarizes the work and points out further research.

2 Background

In this section we detail the main background work used to delineate the proposed process model. Section 2.1 introduces the agile methods, the main values, principles and practices. Section 2.2 presents the Human-Computer Interaction foundations and the Participatory Design. Section 2.3 presents Organizational Semiotics practices, which are used in a participatory way in the proposed process.

2.1 The Agile Methods

The term "Agile Method" is used to identify a set of methods which follow common agreements to respond to changes during the software projects. This term date from 2001, when leading proponents of "light" methodologies wrote the agile manifest [1], which states:

"We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more."

They also agreed in eleven agile principles, which give more detail and more concrete meaning to these values. So, the agile methods should follow the principles: "

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software;
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
4. Business people and developers must work together daily throughout the project;
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done;
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
7. Working software is the primary measure of progress;
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely;
9. Continuous attention to technical excellence and good design enhances agility;
10. Simplicity–the art of maximizing the amount of work not done–is essential;
11. The best architectures, requirements, and designs emerge from self-organizing teams;
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. "

With the popularization of the methods, the agile alliance [2], a Non-Profit Organization, was created to support agile software projects and help people start new projects. This alliance is formed by several intuitional and individual members from the industry and academy spread in the world. Its website becomes the main source of information about the agile methods, including hundreds of articles, events information, books, programs, and resources in general.

Nowadays, there are many agile methods. Although the agile methods follow the same basic principles, they have expressive differences. Koch [8] presents a systematic way to evaluate which method is more adequate to a specific organization. In his approach, evaluators rank features of each method in worksheets concerning the organizational context and the agile method values. The evaluation is summarized and the results point out the organizational changes and the appropriated methods to be applied. The methods considered by Koch [8] are:

- *Adaptive Software Development (ASD)*: This method views a software project team as a complex adaptive system that consists of agents, environments, and emergent outcomes. It is based on the following cycle: Speculate, Collaborate, and Learn;
- *Dynamic Systems Development Method (DSDM)*: This method leaves he details of software writing relatively undefined and instead focuses on system development. The most important part is not the process flow itself but the set of nine principles on which that process was built: active user involvement is imperative, the teams must be empowered to make decisions, the focus is on frequent delivery of products, fitness for business purpose is the essential criterion for acceptance of deliverables, iterative and incremental development is necessary to converge on an accurate business solution, all changes during development are reversible, requirements are baselined at a high level, testing is integrated throughout the life cycle, and a collaborative and cooperative approach between all stakeholders is essential;
- *eXtreme Programming (XP)*: It is a widely recognized method, and adopt the practices, which became reference to many agile projects. The adopted practices are the planning game, small releases, use of metaphor, simple design, test first, refactoring, pair programming, collective code ownership, continuous integration, work 40-hour week, on-site customer, and use of coding standards;
- *Feature-Driven Development (FDD)*: FDD differs from other agile methods in its focus on upfront design and planning. FDD is defined by eight practices: domain object modeling, developing by feature, Class (code) ownership, feature teams, inspections, regular build schedule, configuration management, and reporting/visibility of results;
- *Lean software Development (LD)*: It is not really a method; LD can be understood as a set of principles and tools that an organization can employ in making its software development projects "more lean". The principles of LD are: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole. These principles are supported by 22 tools, based on lean production concepts;
- *Scrum*: It is a method for managing product development that can be wrapped around any specific technology, including software. The scrum practices includes: the scrum master, product backlog, scrum teams, daily scrum meetings, sprint planning meeting, sprint, and sprint review.

2.2 Human-Computer Interaction and Participatory Design

Researchers and practitioners in the HCI field have developed and assessed various methods and tools for the promotion of quality in use of interactive systems. Aspects related to human factors in the use of these systems have been given special emphasis. In order to refer to the design of systems that are allied with the user's point of view and expectations, Norman and Draper [14] introduced the term "User-Centered Design" (UCD). Nowadays, this is a very active line of research in HCI field.

Almost twenty years after the introduction of the “User-Centered Design”, Norman [15] argues that this approach has a limited view of design. He argues that the design process must also consider all human activities in an activity-centered view. Another alternative for user-centered design approaches is usage-centered design, proposed by Constantine et al [5]. His approach, which focuses on the task instead of the user, involves successive refinements of models (e.g. user role model, user task model, and interface content model) to fit the final user interface design.

A strategy for achieving a broader view of design is to encourage the potential users to participate during the whole engineering life cycle. By using this approach, the interface is designed with the users, so that they can participate in design decisions, and express their opinions about activities, practices, tasks and usage context. This participatory approach in system development has its roots in work done in Scandinavia during the seventies and promotes direct user involvement in various phases of the design process, including problem identification and clarification, establishment of requirements and analysis, as well as high level design, detailed design, evaluation, user customization and redesign [13]. In PD, collaboration between designers and users is considered essential to achieve democracy in decision making, quality in use and acceptance of the product. PD also promotes mutual learning [7] from the combination of different experiences.

A large amount of research in the PD field has been conducted to establish meaningful practices for the provision of a common ground for discussion among those directly involved in the design and use of the technology [18]. Participatory techniques are useful instruments for the discussion of the social context of users, since they promote active participation. PD researchers have developed techniques that explore different approaches to promote productive worker-designer co-operation. These techniques have the aim of providing designers and workers with a way of connecting current and future work practices with envisioned new technologies. Participatory design techniques do not suppose to be a straightforward sequence of well-understood steps that produce a guaranteed outcome, but a scaffold or an infrastructure for a complex group process.

Nevertheless, as argued by Gronbaek et al [6], PD techniques seldom go beyond the early analysis/design activities of project development. Pekkola et al. [16] argue that a multi-methodological approach using prototyping and a set of various means of communication is necessary to stimulate end-user participation in development processes. It is important that the people involved share a model for the representation of the work domain which will be involved in the prospective system.

Meaning-making is constructed as a result of cooperation among designers, developers, interested parties and prospective users of the technology being designed. Therefore extensions and adaptations of the techniques are necessary for inclusive environments (e.g. use of tactile cues, use of high contrast solutions, sign language interpretation, and assistive technology compatibility) to enable the participation of people with different needs, skills and interests, including people with disabilities [11]. The participation of each and every individual must be promoted so that he/she can perceive, understand, communicate, be understood and interact with his/her peers [10].

2.3 Participatory Practices based on Organizational Semiotics

Semiotics has been an area actively attended to by scientists in linguistics, media studies, educational science, anthropology, and philosophy, to name a few. Semiotics as a whole, or some of its branches, has influenced many studies in these fields. Computing, including the development and use of computer systems, is another field in which Semiotics has shown a great relevance.

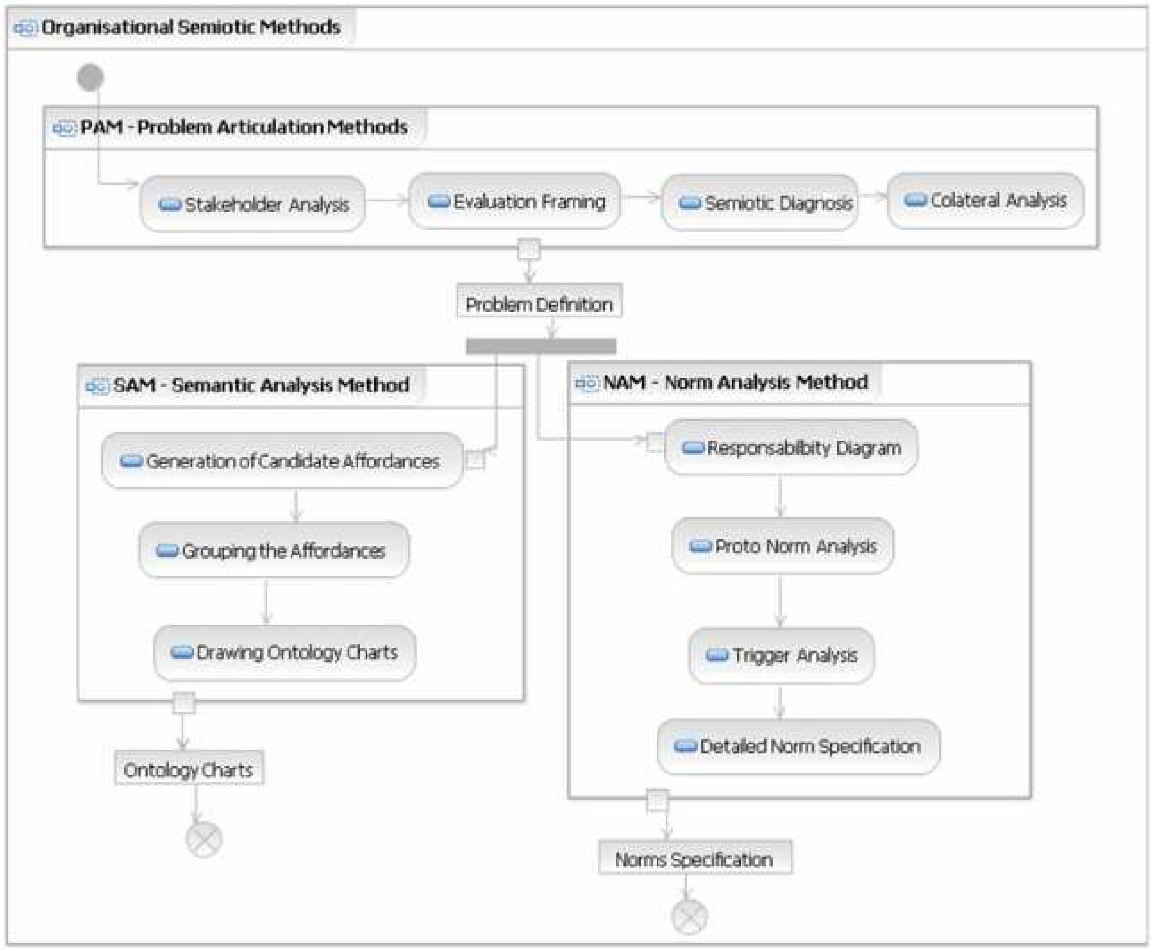


Figure 1: An Overview of the Selected Organizational Semiotic Methods

Organisational Semiotics (OS), a branch of Semiotics which understands the whole organization as a semiotic system [9], aims at studying organizations using concepts and methods based on the Semiotics developed by Peirce [17] and Morris [12]. OS understands that each organized behavior is affected by the communication and interpretation of signs by people, individually or in groups. As a "doctrine of signs", Semiotics encompasses several disciplines, facilitating our understanding about the ways people use signs for every type of purposes. Organizational Semiotics considers the internal work of the organization, including its information systems and its interactions with the environment, aiming at finding new ways of analyzing, describing and explaining the structure and behavior of the organization.

Organizational Semiotics presents theories and methods to allow the analysis and design of information systems in terms of three human information functions: expressing meanings, communicating intentions and creating knowledge [20]. Studies in OS are not restricted to information expressed in written or graphical discourse, but take into account the semiotic aspects of human interaction in the organization. In the philosophical stance underlying OS, reality is seen as a social construction based on the behavior of agents participating in it; people share a pattern of behavior governed by a system of signs.

OS Methods have been found useful for dealing with the influence of social aspects in organizations and in the elicitation of system requirements. Among the methods employed by the OS community is a set of methods known as MEASUR (Methods for Eliciting, Analyzing and Specifying Users' Requirements) [19], which deal with the use of signs, their function in communicating meanings and intentions, and their social consequences. MEASUR involves the analysis of stakeholders in a focal problem, as well as considering their needs and intentions, and the constraints and limitations related to the prospective software system. Figure 1 summarizes the logical sequence of the main methods of MEASUR.

In the process model presented in this report we propose "participatory workshops"; these workshops are inspired by and conducted using Organizational Semiotics artifacts to organize the discussion and support the meaning making. The diagrams are made into large posters and hung on the wall to mediate discussions during the workshops. A facilitator introduces the discussion based on the Semiotic artifacts. In a brainstorming format, the participants express their ideas using post-its that are stuck on the posters. During brainstorming the discussion includes the position of the post-its on the diagram, so that related ideas are grouped together constituting a certain structure. The participants are also invited to discuss conflicting ideas, even though some conflicts will only be resolved during later stages of the project. After the brainstorming, each participant is invited to synthesize the themes discussed, and these are then presented.

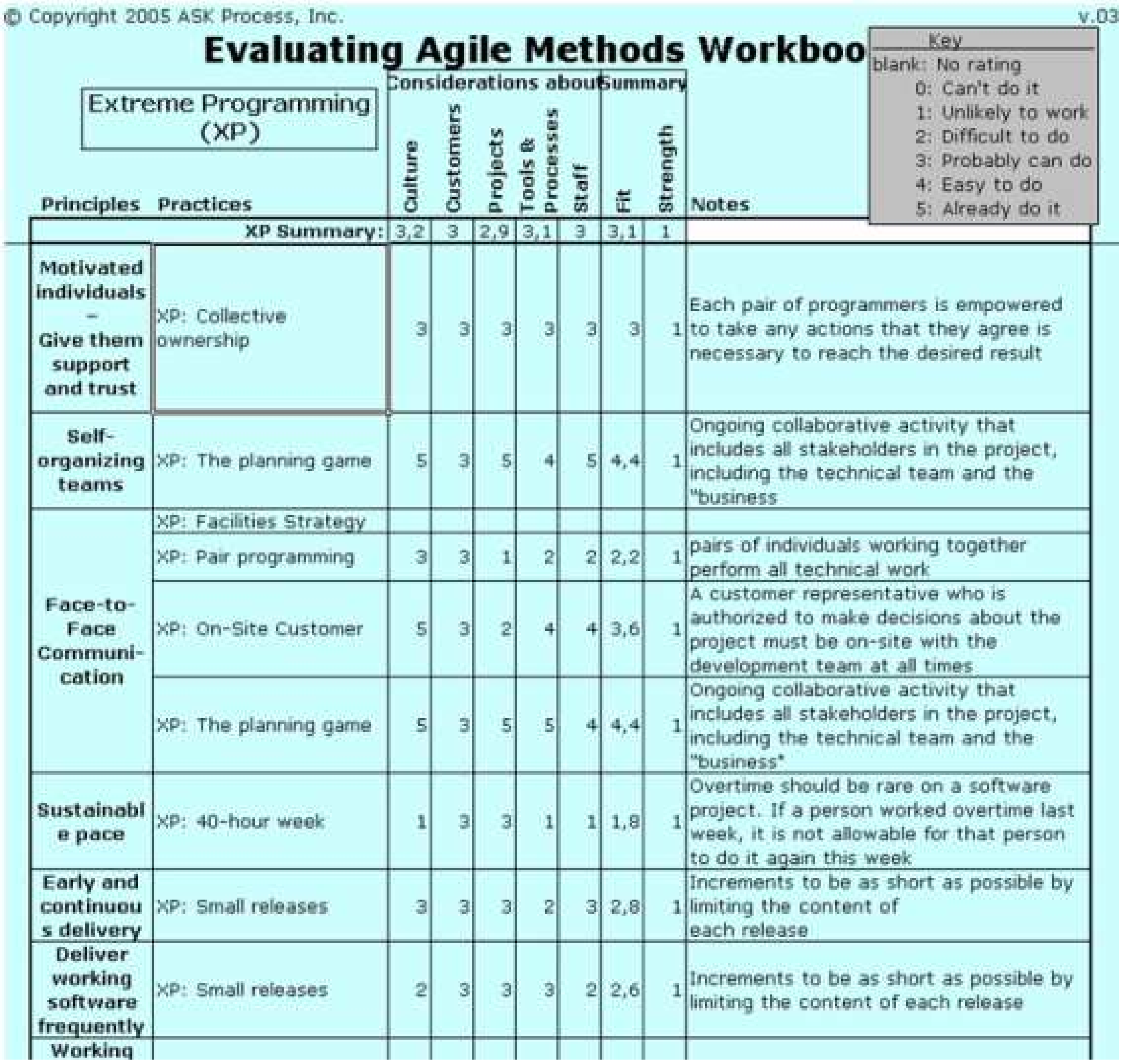
3 Delineating a Process Model: From eCidadania requirements to the process model

The initial proposal was to establish a disciplined way to develop the systems proposed in the "e-Cidadania" project. From the specific process requirements for this project the process model was generalized to deal with accessibility and usability issues.

The first step conducted to delineate the model was to analyze the agile principles in order to verify the compatibility of these principles with our development context. The main aspects of the "e-Cidadania" context include:

- It is a research project that involves development activities;
- The team is composed of researchers, students, and other professional contracted to codify the system;

- The team is geographically distributed;
- The software is innovative in terms of accessibility and usability;
- The development team follows a flexible timescale.


© Copyright 2005 ASK Process, Inc. v.03

Evaluating Agile Methods Workboo

Extreme Programming (XP)

Key
blank: No rating
0: Can't do it
1: Unlikely to work
2: Difficult to do
3: Probably can do
4: Easy to do
5: Already do it

		Considerations about					Summary		
Principles	Practices	Culture	Customers	Projects	Tools & Processes	Staff	Fit	Strength	Notes
	XP Summary:	3,2	3	2,9	3,1	3	3,1	1	
Motivated individuals – Give them support and trust	XP: Collective ownership	3	3	3	3	3	3	1	Each pair of programmers is empowered to take any actions that they agree is necessary to reach the desired result
Self-organizing teams	XP: The planning game	5	3	5	4	5	4,4	1	Ongoing collaborative activity that includes all stakeholders in the project, including the technical team and the "business
Face-to-Face Communi-cation	XP: Facilities Strategy								
	XP: Pair programming	3	3	1	2	2	2,2	1	pairs of individuals working together perform all technical work
	XP: On-Site Customer	5	3	2	4	4	3,6	1	A customer representative who is authorized to make decisions about the project must be on-site with the development team at all times
	XP: The planning game	5	3	5	5	4	4,4	1	Ongoing collaborative activity that includes all stakeholders in the project, including the technical team and the "business"
Sustainabl e pace	XP: 40-hour week	1	3	3	1	1	1,8	1	Overtime should be rare on a software project. If a person worked overtime last week, it is not allowable for that person to do it again this week
Early and continuou s delivery	XP: Small releases	3	3	3	2	3	2,8	1	Increments to be as short as possible by limiting the content of each release
Deliver working software frequently	XP: Small releases	2	3	3	3	2	2,6	1	Increments to be as short as possible by limiting the content of each release
Working									

Figure 2: Example of Method Evaluation

The next step involved an analysis of the most popular methods, using the Koch [8] approach. The intention was to identify process features that could be interesting to our context. These were considered in the construction of the proposed process model. Figure 2 shows an example of part of the worksheet constructed to evaluate principles and practices of the XP method.

The methods practices and principles were discussed in a first meeting with the developers, project researchers and representatives of the target audience. From the discussion, we extracted some examples of issues to be addressed and possible solutions, for example:

- Problem: "We have distributed teams, resulting in difficulties to practice pair programming and daily meetings". Discussed solution: To investigate the use of collaborative development tools and to use practices adopted by the "open source" development communities;
- Problem: "We need models". Discussed solution: Investigate the use of agile modeling [3];
- Problem: "Accessibility and usability are key requirements". Discussed solution: promote the user participation with PD methods; adopt the W3C standards, use low-fidelity prototype, etc.

After initial meetings, the process principles were synthesized. From these principles a lifecycle model and practices were defined. The principles, lifecycle and practices were discussed with developers, researchers and representatives of the target users in another meeting. The process model was also revised by specialists in accessibility and usability, and finally the model was instantiated in the e-Cidadania project context to carry a case study.

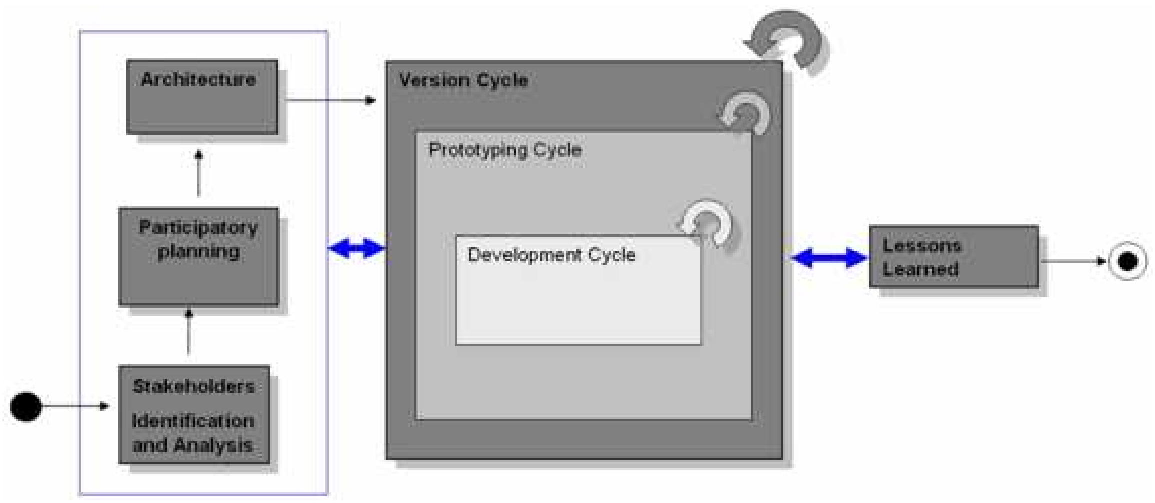


Figure 3: An Overview of AIPM Lifecycle

4 The Agile Inclusive Process Model

The Agile Inclusive Process Model (AIPM) follows principles relatively distinct from other process models. While the other process focused mainly on the production of software and its quality, the AIPM focus on accessibility and usability of the final product. The AIPM principles are as follows:

- Promote the participation of the users and other stakeholders' with the universal access and inclusive design values;
- Construct a shared vision of the social context;
- Include more than just technical issues in the development of the system;
- Promote the digital inclusion through participatory activities.

In order to develop the software in an agile way, the general principles and values of the agile alliance are also considered in addition to the AIPM principles. Figure 3 presents an overview of the AIPM lifecycle. This lifecycle follows the XP idea of adopting nested lifecycles.

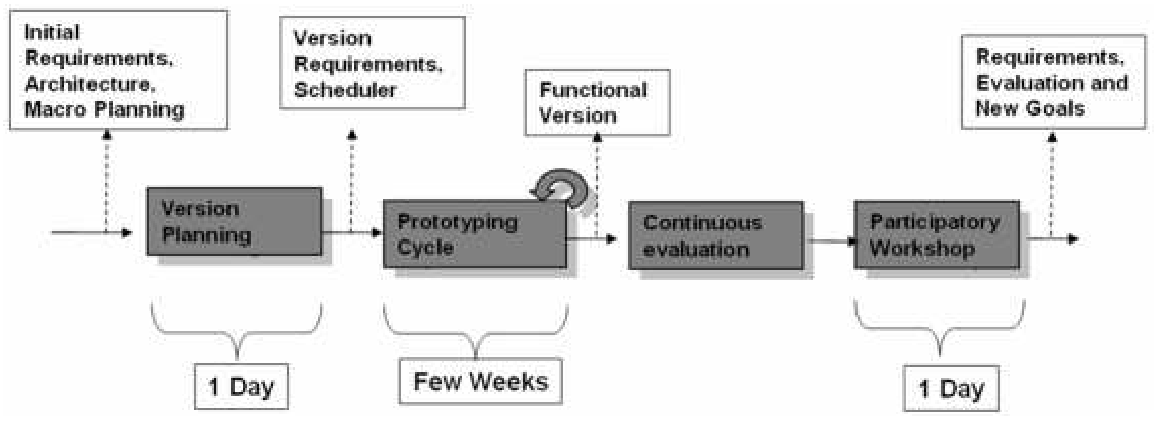

Initial Requirements, Architecture, Macro Planning
Version Requirements, Scheduler
Functional Version
Requirements, Evaluation and New Goals
Version Planning
Prototyping Cycle
Continuous evaluation
Participatory Workshop
1 Day
Few Weeks
1 Day

Figure 4: Version Cycle Details

As Figure 3 shows, the first practice is the "stakeholder identification and analysis", where the designers, developers, specialists on digital inclusion, and community members participate in workshops to discuss the possible stakeholders and the project context. The next steps are the "participatory planning" and the definition of the "architecture". After few days, the version cycle is initiated, but these three practices still occur in parallel till the end of the project. Each version cycle takes no more than one or two months and involves 3 or 4 prototyping cycles. Each prototype cycle takes three to four weeks involving 3 or 4 development cycles that takes around seven days each. According to Figure 3 the lessons learned are also synthesized in parallel during all the lifecycle.

Figure 4 presents details of the version cycle; the grey boxes represent practices and activities, the upper boxes the main artifacts consumed or produced, and the boxes below, the time required for each grey box. This cycle starts with the "Version Planning" which delineates the main version requirements and a macro schedule for the prototyping cycle. After the first prototypes a "continuous evaluation" is conducted, and at the end of the cycle a "Participatory Workshop" is performed. OS methods and tools support these workshops.

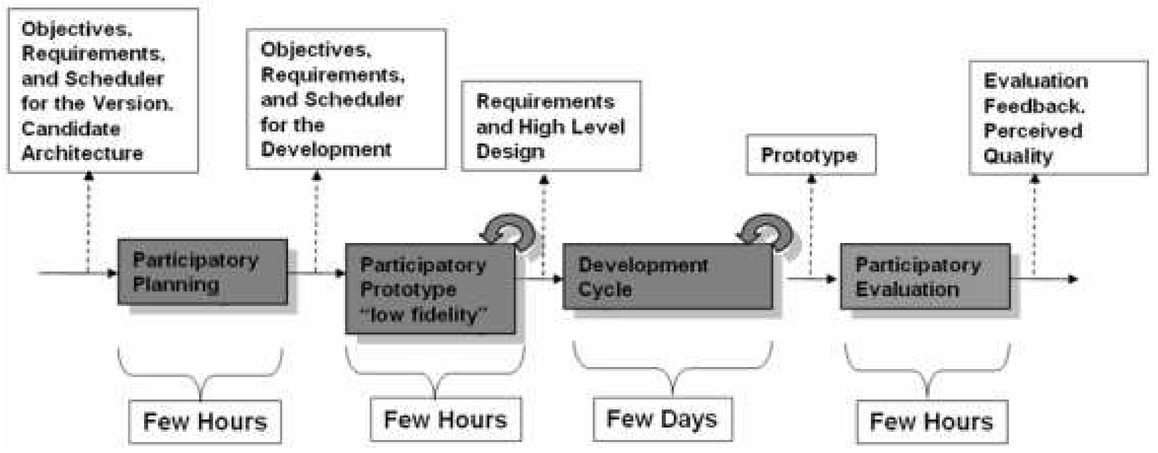

Objectives, Requirements, and Scheduler for the Version. Candidate Architecture
Objectives, Requirements, and Scheduler for the Development
Requirements and High Level Design
Prototype
Evaluation Feedback. Perceived Quality
Participatory Planning
Participatory Prototype "low fidelity"
Development Cycle
Participatory Evaluation
Few Hours
Few Hours
Few Days
Few Hours

Figure 5: Prototype Cycle Details

Figure 5 presents the details of the prototyping cycle. This cycle starts with the "participatory planning" for the prototype and development. The "Participatory prototype" practice is conducted in order to define the requirements and the high level design of the interface using "low fidelity" prototypes. After that, the prototype is implemented during the development cycle and evaluated using PD techniques.

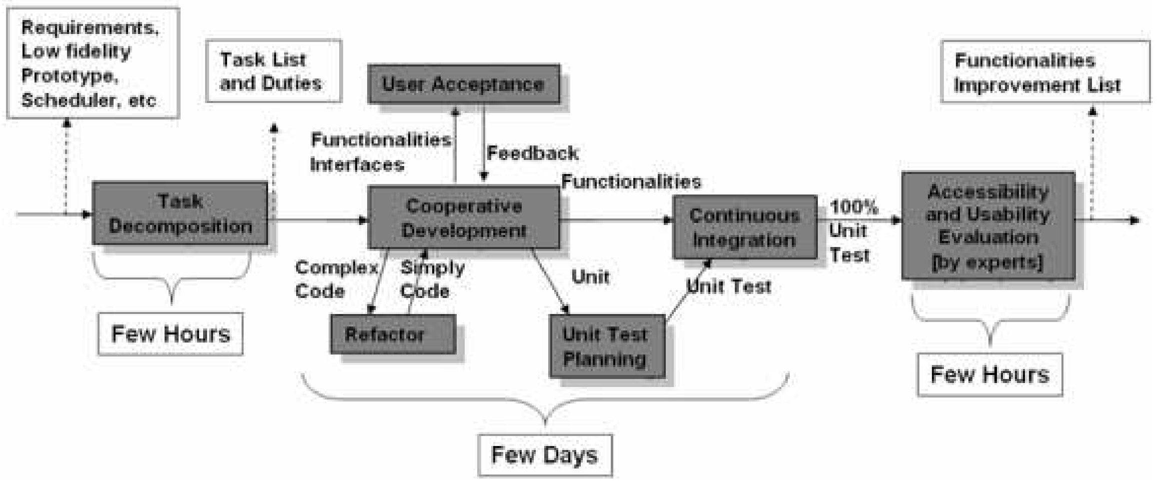
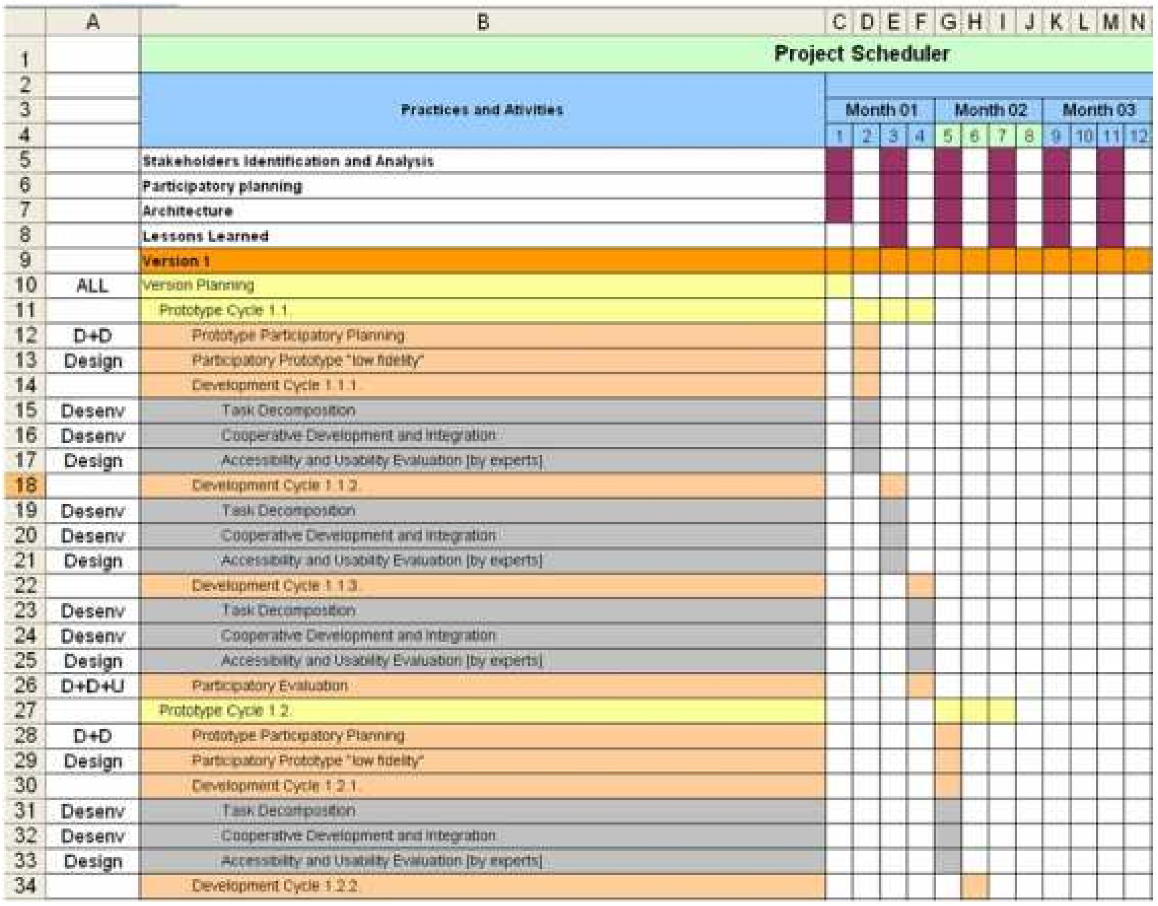


Figure 6: Development Cycle Details

The development cycle (Figure 6) is based on XPs "Refactor and Continuous Integration" practices, but the proposed cycle also aims at producing accessible and inclusive software. The "task decomposition" is the first step proposed; it is a meeting between the developers and designers to identify tasks and to attribute duties over each task. During the "cooperative development" the functionalities, navigation models, screens, and other elements of the interface are implemented using tools for supporting the cooperative development. The new functionalities and interfaces are frequently evaluated by the users, as well as the code frequently refactored and continually integrated. When all the unit text is completed the code is inspected by accessibility and usability experts.

Figure 7 shows a fragment of the eCidadania project schedule constructed using the proposed development cycle. In this schedule the deadline for each lifecycle is defined, for example: the "Task Decomposition" for the development cycle 1.1.2 is performed in the third week. This schedule is used to guide the project but it can be adapted to changes if necessary. By the time of this report was written, the eCidadania project was at the end of the first version cycle.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Project Scheduler												
2														
3		Practices and Ativities	Month 01				Month 02				Month 03			
4			1	2	3	4	5	6	7	8	9	10	11	12
5		**Stakeholders Identification and Analysis**	■		■		■		■		■		■	
6		**Participatory planning**	■		■		■		■		■		■	
7		**Architecture**	■		■		■		■		■		■	
8		**Lessons Learned**			■		■		■		■		■	
9		**Version 1**	■	■	■	■	■	■	■	■	■	■	■	■
10	ALL	Version Planning	■											
11		Prototype Cycle 1.1		■	■	■								
12	D+D	Prototype Participatory Planning		■										
13	Design	Participatory Prototype "low fidelity"		■										
14		Development Cycle 1.1.1		■										
15	Desenv	Task Decomposition		■										
16	Desenv	Cooperative Development and Integration		■										
17	Design	Accessibility and Usability Evaluation [by experts]		■										
18		Development Cycle 1.1.2			■									
19	Desenv	Task Decomposition			■									
20	Desenv	Cooperative Development and Integration			■									
21	Design	Accessibility and Usability Evaluation [by experts]			■									
22		Development Cycle 1.1.3				■								
23	Desenv	Task Decomposition				■								
24	Desenv	Cooperative Development and Integration				■								
25	Design	Accessibility and Usability Evaluation [by experts]				■								
26	D+D+U	Participatory Evaluation				■								
27		Prototype Cycle 1.2					■	■	■					
28	D+D	Prototype Participatory Planning					■							
29	Design	Participatory Prototype "low fidelity"					■							
30		Development Cycle 1.2.1					■							
31	Desenv	Task Decomposition					■							
32	Desenv	Cooperative Development and Integration					■							
33	Design	Accessibility and Usability Evaluation [by experts]					■							
34		Development Cycle 1.2.2						■						

Figure 7: Fragment of the Project Development Schedule

5 Conclusion

The Internet represents a new dimension to the software development. It can be understood as an opportunity to develop systems that support the social inclusion and citizenship. However, those systems impose a singular way for developing software, in which accessibility and usability are the key requirements.

The increase in software complexity and consequently development costs, demands high quality and productivity of the software development. The software engineering community has proposed process models to increase the productivity and quality of the software systems; one of the current achievements are the agile software methods.

This technical report presented a process model, based on agile methods, as well as practices, methods and theories from Human-Computer Interaction, Participatory Design and Organizational Semiotics. The process model was defined to meet conditions and requirements of the e-Cidadania project based on principles, a lifecycle model, and fundamental practices. The process model was instantiated in the “e-Cidadania” context, and is in use. The refinement of this model will be possible in the near future with feedback of the usage in the project.

Acknowledgments

This work is funded by Microsoft Research - FAPESP Institute for IT Research (processo n. 2007/54564-1). The authors also thank Elaine C. S. Hayashi, Heiko Horst Hornung, Leonelo Dell Anhol Almeida, Maria Cecília Martins and Vânia Paula de Almeida Neris and other colleagues from NIED, InterHAD, Casa Brasil, IC Unicamp and Cenpra for insightful discussion.

References

[1] Agile Alliance, *Manifesto for Agile Software Development.* http://agilemanifesto.org/, Accessed 08 April 2008 (2001).

[2] Agile Alliance, *How The Agile Alliance Operates.* http://www.agilealliance.org/show/1646, Accessed 08 April 2008 (2008).

[3] Ambler S. W. *The Object Primer: Agile Model-Driven Development with UML 2.0.*, Third Edition, Cambridge University Press (2004).

[4] Baranauskas, M.C.C., *e-Cidadania: Systems and Methods for the Constitution of a Culture mediated by Information and Communication Technology.* Proposal for the Microsoft Research-FAPESP Institute. (2007).

[5] Constantine L., Biddle R., Noble J. *Usage-Centered Design and Software Engineering: Models for Integration.* In: IFIP Working Group 2.7/13.4, ICSE 2003 Workshop on Bridging the Gap Between Software Engineering and Human-Computer Interaction, Portland, Oregon, USA, 3-10, May 2003. (2003).

[6] Gronbaek K, Kyng M, Mogensen P (1997) *Toward a Cooperative Experimental System Development Approach.* In: Kyng M, Mathiassen L (eds) Computers and Design in Context. MIT Press, Boston Massachusetts.

[7] Kyng, M. *Designing for Cooperation: Cooperating in Design.* Communications of the ACM. Doi: 10.1145/125319.125323 (1991).

[8] Koch, A. S. *Agile Software Development: Evaluating the Methods for Your Organization.* Artech House, Norwood, MA, USA (2005).

[9] Liu K. *Semiotics in information systems engineering.* Cambridge University Press, Cambridge (2000).

[10] Melo, AM *Design Inclusivo de sistemas de informação na web.* PhD Thesis, University of Campinas, Campinas, Brazil. In Portuguese, (2007).

[11] Melo AM, Baranauskas MCC *An inclusive approach to cooperative evaluation of the web user interfaces.* In: Proceedings of the 8th International Conference on Enterprise Information Systems, Paphos, Cyprus, 23-27, May 2006, (2006).

[12] Morris, C. W. *Foundations of the theory of signs.*, International Encyclopedia of Unified Science, University of Chicago Press, 1 (2) (1938).

[13] Muller M. J., Haslwanter J.H., Dayton T.*Participatory Practices in the Software Lifecycle.* In: Helander MG, Landauer TK, Prabhu PV (eds) Handbook of Human-Computer Interaction, 2nd edition, North-Holland, Elsevier (1997).

[14] Norman, D. A., e Draper, S. W. *User centered system design: New perspectives on human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum. (1986).

[15] Norman, D. A. *Human-Centered Design Considered Harmful.*, Interactions, 12. 4, (July + August, 2005). Pp. 14-19, (2005).

[16] Pekkola S, Kaarilahti N, Pohjola P *Towards Formalised EndUser Participation in Information Systems Development Process: Bridging the Gap between Participatory Design and ISD Methodologies.* In: Proceedings of the 9th Participatory Design Conference, Trento, Italy, 1-5, August 2006 (2006).

[17] Peirce, C.S. *Collected Papers*, Cambridge, Mass: Harvard University Press, (1931-1958).

[18] Schuler, D, Namioka, A. (eds.) *Participatory design: principles and practices.* Lawrence Erlbaum Associates, USA, 312 p (1993).

[19] Stamper R.K. *Social Norms in requirements analysis - an outline of MEASUR.* In: Jirotka M, Goguen J, Bickerton M. (eds) Requirements Engineering, Technical and Social Aspects. Academic Press, New York (1993).

[20] Stamper, R. K. *Organisational Semiotics: Informatics without the Computer?* In Information, Organisation and Technology: Studies in Organisational Semiotics, eds. K. Liu, R. Clarke, P. B. Andersen and R. K. Stamper. Kluwer Academic Publishers (2001).

[21] World Wide Web Consortium, http://www.w3.org/. Accessed 26 March 2008.