On the Agile Development of Virtual Reality Systems

F. Mattioli[1], D. Caetano[1], A. Cardoso[1], and E. Lamounier[1]
[1]Faculty of Electrical Engineering, Federal University of Uberlândia, Uberlândia, Minas Gerais, Brazil

Abstract— *Processes for Agile software development present an iterative and incremental approach to computer systems, which focus on users' needs and embraces changes. Virtual Reality projects are strongly tied to rapid evolution of technology, and to the need for clients' feedback, during the whole project's life-cycle. In this work, a comparative evaluation of existing methodologies is presented and the application of agile software development methodologies in Virtual Reality projects is argued. Then, a proposal for an agile software development process for Virtual Reality systems is presented and its benefits are discussed.*

Keywords: Agile Development, Virtual Reality, Software Engineering

1. Introduction

Agile Software Development has, among its main features, an iterative and incremental approach to Software Engineering principles. This approach is suitable for Virtual Reality projects and offers, by its evolving nature, many benefits associated to risk management in software projects [1].

The word "agile" was first used in Software Engineering at 2001, by a consortium of software development methods specialists, who have written, at that time, the "Agile Manifesto" [2]. This manifesto highlighted some principles, shared by many different software development methods, which were thereafter called "Agile Methods" or "Agile Processes" [2], [3]:

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

Virtual Reality based systems require knowledge in different subjects, such as Computer Graphics, geometric modeling, multimodal interaction among others [4]. Some characteristics of these applications reveal the need for continuous improvement in their development process. Some of these characteristics can be highlighted:

- Rapid evolution of visualization and graphical processing technology [5].
- Customer's indecision and change of opinion, a critical concern when high-cost equipment is used [5].
- Need for implementation of prototypes, used to help customers in the solution's evaluation process.

Therefore, evolutionary development, adaptive planning and response to requirements' changes are major improvements to be considered on agile development of Virtual Reality systems.

2. Agile Software Development

Agile Software Development is an approach of software production focused on adaptability, which can be understood as the process' capability of responding to changes in markets, requirements, technology and development teams [6].

Sections 2.1 and 2.2 present a brief description of two agile methods: XP and Scrum.

2.1 Extreme Programming (XP)

Extreme Programming (XP) had its origins guided by the needs of small software development teams, working on projects with highly volatile requirements. XP is a light development method, which fundamentals include [7]:

- Unit tests are written before the code being tested. These tests are executed throughout the project life-cycle.
- Integration and testing are performed continuously, many times a day.
- The project begins with a simple architecture, that constantly evolves in an effort to increase flexibility and reduce unnecessary complexity.
- A minimal system is rapidly implemented and deployed. This minimal system will evolve according to project's directions.

Amongst the main benefits of Extreme Programming, the following are worth mentioning [7]:

- Do not force premature specialization of team members. All team members play different roles inside the development team, in a daily basis.
- Analysis and design are conducted throughout the whole project life-cycle, favoring adaptability and rapid response to project environment changes.
- Project infrastructure is built in an iterative way, following project's evolution and meeting its real needs.

Figure 1 presents the main elements of the XP process' life-cycle. User Stories are collected and used in requirements' specification and also in test scenarios definition. An Architectural Spike is conducted to elucidate the relevant solution elements, resulting on a System Metaphor.

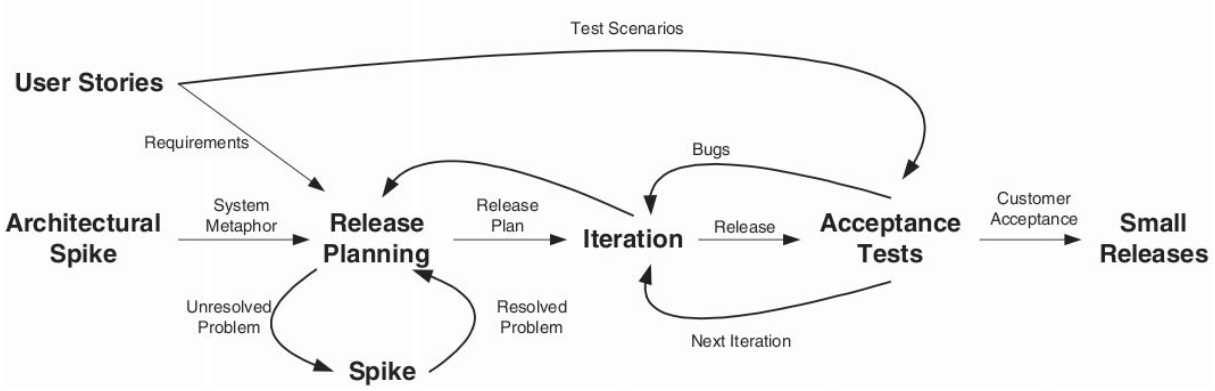

Fig. 1: XP life-cycle [8].

During Release Planning, architectural problems may arise. Each time an architectural problem is detected, a Spike is conducted to solve this problem. The resulting artifact is the Release Plan.

Each iteration targets a subset of functionalities. If a problem is detected during the iteration, Release Planning is carried again, and an updated Release Plan is written. At the end of the iteration, a release is made available and Acceptance Tests are conducted, using the test scenarios defined from the User Stories. If bugs are found during tests, another iteration is lead to fix them. If no bugs were found, the next iteration is started. When customer acceptance is confirmed, a Small Release (release of a working version of the software) is performed.

2.2 Scrum

Scrum is an empirical approach for managing software projects, based on the following principles: adaptability, flexibility, and productivity [9]. In Scrum, projects are divided into *sprints*. A *sprint* is a development iteration, with the typical duration of 30 days. Each *sprint* is associated to a set of tasks, whose priority is rather defined together with the clients. For each task, the remaining time to finish is estimated [10]. Tasks can be relocated, according to project's constraints.

In a nutshell, the Scrum process is composed by a set of rules, procedures and practices, favoring software development [8]. Figure 2 presents the Scrum process' life-cycle.

In the Scrum life-cycle, known requirements are grouped and prioritized in a product backlog [11]. A subset of these requirements, known as the "Sprint Backlog", contains the tasks assigned to a given *sprint*. From the "Sprint Backlog", tasks are elucidated in detail.

During the *sprint* - which is scaled for no more than 30 days - a daily review meeting is conducted. This daily meeting should not last long (15 minutes is a general suggestion), so that all project members can attend it [12]. In the daily meeting, team members are required to briefly answer three questions [13]:

1) What have I done since the last Daily Scrum?
2) What will I do between now and the next Daily Scrum?
3) What obstacles and roadblocks are in my way?

These answers have the objective of providing managers and developers with general information about the *sprint's* progress. Also, efforts can be grouped to help solving common problems, while experience can be shared in a daily basis.

Finally, at the end of each *sprint*, the new functionalities are demonstrated and tested, looking forward to stakeholders' approval.

2.3 XP and Scrum

Although complementary, XP and Scrum have different application, in different aspects of software development. While Scrum can be considered an agile project management tool, XP is more focused on the development side [14]. Scrum strengths include project's visibility in the market's context, continuous project management and improved collaboration between team members. XP motivation include a simplified requirements management approach and enhanced product quality. Both methodologies are based on iterative and incremental development [15].

Put together, Scrum and XP are valuable approaches, both on management and technical practices [14]. Therefore, in this work, an hybrid process is proposed. This hybrid process can benefit from Scrum management practices (such as "Sprint Backlog" and daily reviews), together with XP engineering practices (product quality, short iterations and test-driven development). This process is presented in detail on Section 4.

3. Virtual Reality Systems Development

The develpment of Virtual Reality Systems (VRS), as well as the development of any software, requires processes and development methods. In the particular case of VRS, methods and processes should be adequate to a rapidly

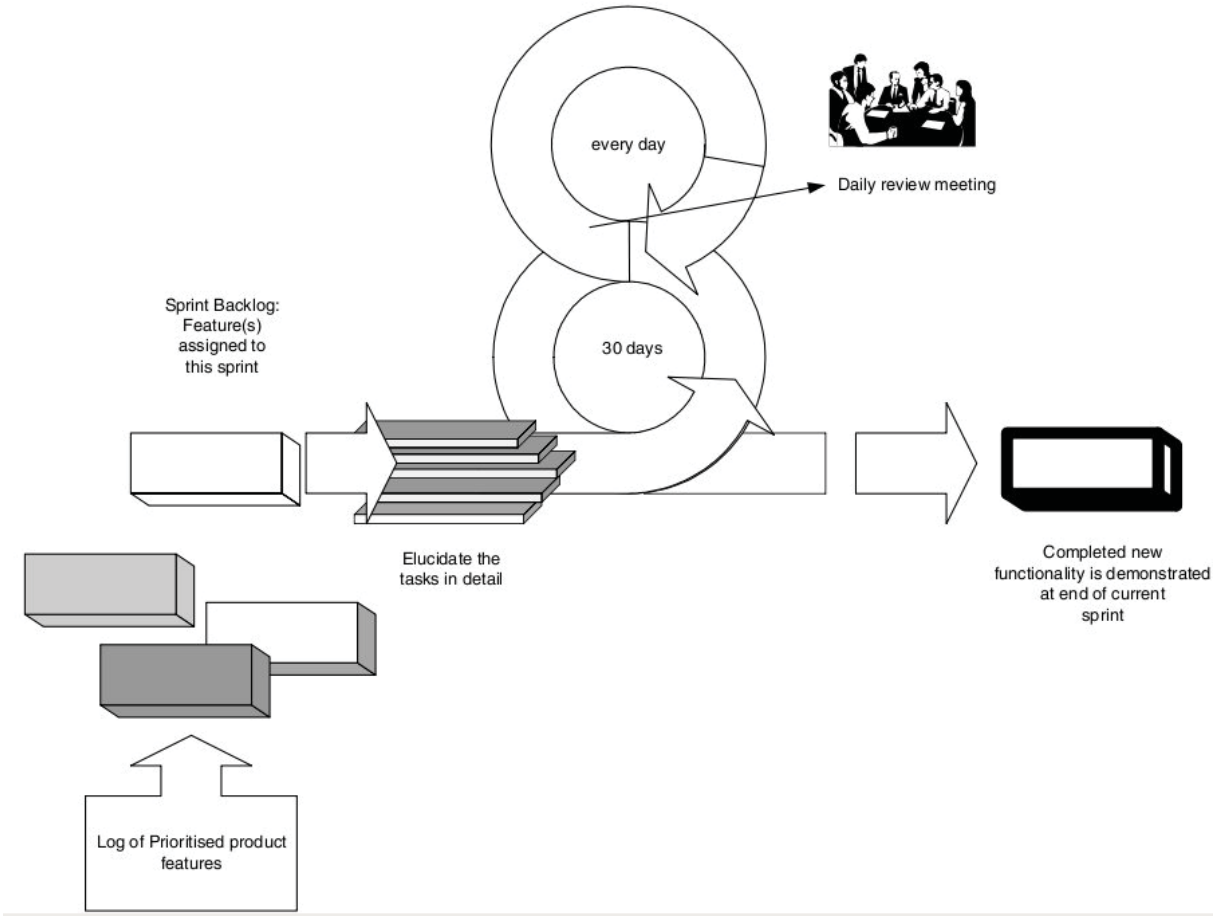

Fig. 2: Scrum life-cycle [8].

changing technological environment and to the particular aspects of user interaction.

Tori *et al.* present a development process that aggregates prototyping with iterative and evolutionary software development [5]. This process is based on Software Engineering models, adapted to the particularities of Virtual Reality Systems. The proposed process is composed of 5 stages, executed in each iteration: Requirement Analysis, Design, Implementation, Evaluation and Deployment. These stages are graphically represented in Figure 3.

Another development approach, also suitable for use on Virtual Reality Systems development is presented by Kim [16]. At first sight, this approach can be seen as an extension of the classic spiral model, adapted to Virtual Reality Systems characteristics, such as interaction models and scene modeling. Figure 4 presents the main elements of the proposed process.

Although both processes addressed in this section present strong influences from the structured approach, some VRS features are closely related to agile practices. Among them, one can highlight:

- The evolutionary nature of Virtual Reality Systems.
- The need for models that represent, iteratively, form, function and behavior of Virtual Reality Systems components.
- The better acceptance of systems which are developed with active participation of stakeholders, due to the constant need for evaluation and feedback.
- The need for exhaustive tests, aiming at reducing interaction problems between users and Virtual Reality Systems.

Based on these observations, the application of agile methods in Virtual Reality Systems development is discussed in Section 4.

4. Agile Development of Virtual Reality Systems

No software development process can guarantee, by itself, any improvement in productivity and reliability [17]. However, some characteristics are common in successful processes [18]:

- Iterative development: complex projects, with many modules, are more likely to face integration issues. An

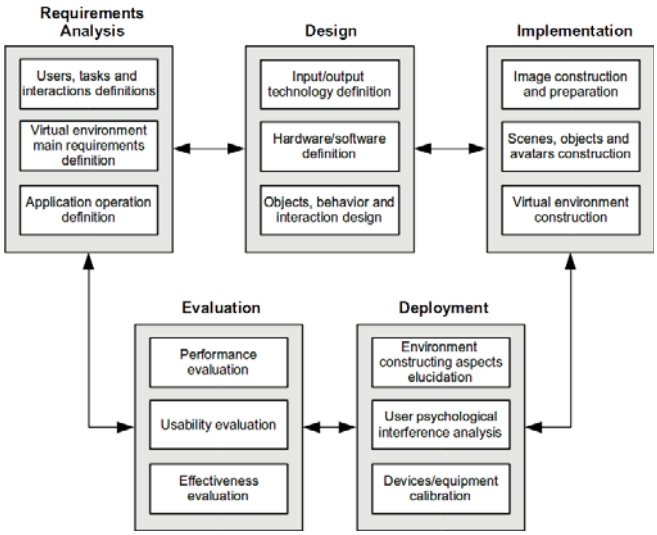

Fig. 3: VRS development process. Adapted from [5].

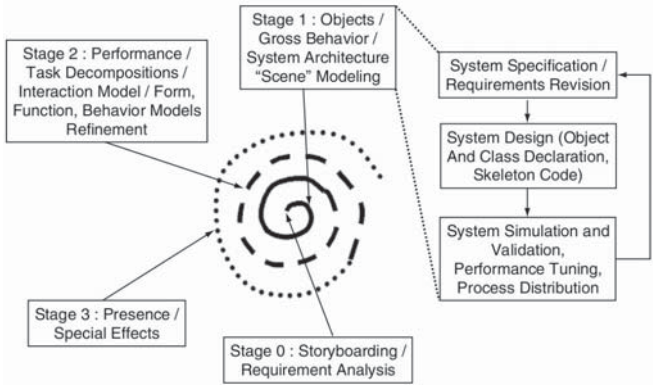

Fig. 4: VRS development process [16].

adequate iteration planning can reduce these integration issues and favor development process management.

- Process' continuous evaluation: even requirement-oriented development processes cannot make software projects totally immune to changes on development teams and on user requirements. The evaluation (and consequent adaptation) of the development process has a major importance throughout the project's life-cycle.
- Best practices: improvements associated to the use of development best practices [19], [20] and also design patterns [21] should be considered and discussed in software projects.

When adapting an existing process to a given context, the suggested approach is to customize the existing process, iteratively testing and refining this customization, in accordance with each project's characteristics [22]. During this customization, some principles might be observed [23]:

1) Larger teams require robust processes and methods.
2) Carried-over process complexity represent additional costs.
3) Critical applications require highly-detailed methods.
4) Clients' feedback and team communication reduce the need for intermediate documentation.
5) As the number of legal issues involved in a project increase, methods' level of detail should also increase.

From the presented literature review, this work's objective was defined: to propose a process model for the agile development of Virtual Reality systems. The proposed model - detailed in Section 5 - consists of a hybrid model, gathering elements from both XP and Scrum, adapted to the context of Virtual Reality systems development.

5. Results

In this section, a development process for Virtual Reality systems is proposed. The presented process is composed by 8 main activities: User stories / storyboards definition, architectural spike, interactivity requirements elucidation, iteration planning, spike, development, integration tests and client tests. Development is executed iteratively, and feedback received in past iterations is used to help planning the next ones.

By reviewing the state of the art of VRS development methods, some key features of these systems were defined:

- The evolutionary nature of VRS.
- Iterative building of high-fidelity models.
- The need for clients' feedback.
- The need for interaction and usability tests.
- The need for system modularization.

A VRS development process should keep these features in focus during the entire project life-cycle, in each of the activities presented above. In Figure 5, a graphical representation of the flow of activities in the proposed process is displayed. In the following sections, each of these activities is detailed.

5.1 User stories/storyboards

An user story is a brief description of a system functionality, from the user point of view. User stories are very helpful on requirement analysis because they provide developers with users' real expectations about the system.

When developing high-complexity graphical systems - such as VRS - text based user stories can be limited to detail users' needs. To overcome this, the proposed process suggests the use of storyboards, used to complement user stories. Storyboards are graphical sketches, elaborated by clients (or with their supervision), whose objective is to help developers on performing an accurate requirement analysis.

5.2 Interactivity requirements' analysis

Interactivity is the central aspect of many Virtual Reality systems, having a major role in these systems' usability. Thus, the detailed analysis and definition of interactivity

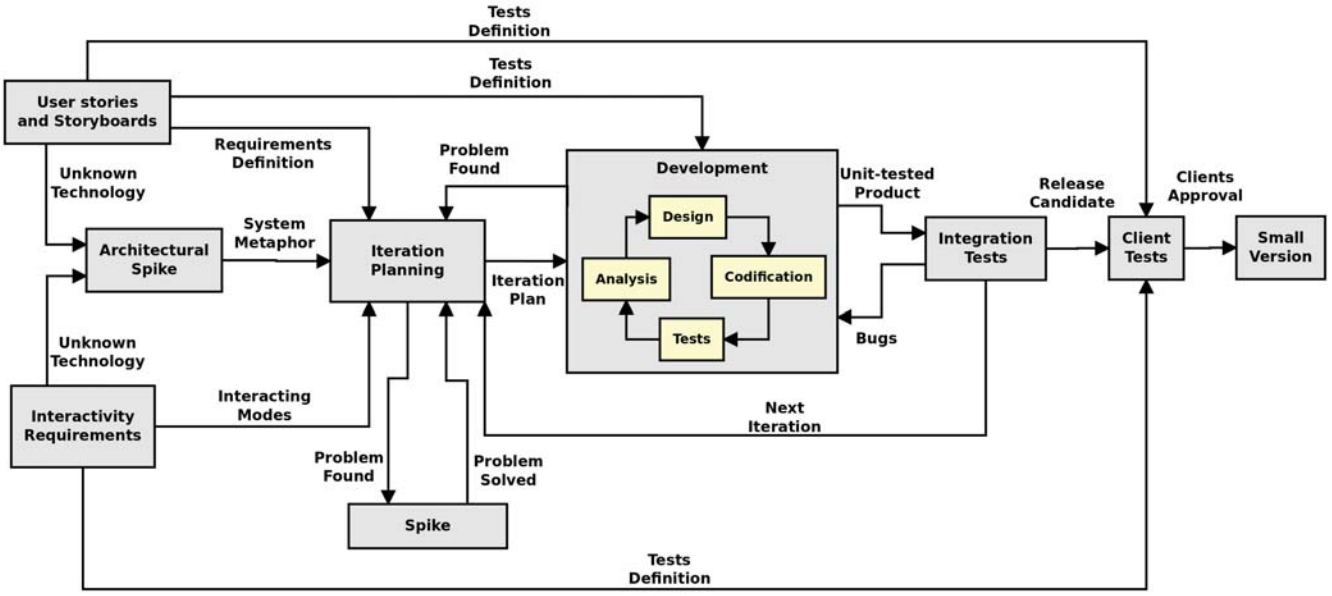

Fig. 5: Agile development process for VRS.

requirements has a prominent position ind the development process. In this activity, test applications - implementing the desired interaction methods - can be used. These applications should help developers and clients in the evaluation and viability analysis of the required interaction methods.

5.3 Architectural spike

In the architectural spike, a viability analysis of the new requirements is conducted. The requirements are extracted from user stories, storyboards and interactivity requirements' analysis. This activity has the goal of reducing risks related to unknown technology (for example, third-party libraries). The architectural spike results in the metaphor definition. This metaphor will be used by the development team to represent the subset of requirements in focus at the current iteration.

In the architectural spike, the resources available for requirements' implementation are investigated. This activity is very important in VRS development, since it suggests and encourages experimentation. Together with the "Interactivity requirements analysis", this activity is strongly related to technological advances in Virtual Reality.

5.4 Iteration planning

Iteration planning takes place at the beginning of each iteration. The resulting artifact - the iteration plan - addresses a subset of requirements, elucidated from the user stories, storyboards and interactivity requirements analysis. During iteration planning, each time a problem is detected, a spike is conducted, in order to investigate and propose possible solutions.

It's very important to highlight the adaptive behavior of iteration planning. Ideally, iteration planning should be flexible enough to embrace changes on application requirements and solutions to the problems found inside the iteration.

5.5 Spike

A spike is a small development cycle, whose main objective is to provide developers with possible solutions to a given problem. Inside the spike, test applications (or prototypes) can be built, to help developers on testing and discussing proposed solutions. If possible, clients' feedback can be used to guide the development team on the right direction, according to users' needs.

5.6 Development

Once the iteration plan is defined, development takes place. Development is composed by 4 main tasks, adapted from the consolidated Rational Unified Process [24]: analysis, design, codification and tests.

Analysis and design share the common goal of structuring the implementation of the iteration plan's requirements. A set of tests - proposed by the clients - is used to guide the development team on the implementation of the most important requirements, from the clients' point of view. Then, developers are requested to propose their own tests. With clients' and developers' tests defined, the codification task begins. Development activity is finished when the system successfully executes the proposed tests.

The evolutionary nature of the development activity should be highlighted. In the beginning, the system is composed by simplified models, that represent the main elements of the VRS. As the system evolves, these models are refined,

resulting in components whose shape and behavior are each time closer to the represented elements.

5.7 Integration tests

In each iteration, integration tests are conducted right after the development activity. The modifications performed in the current iteration will be integrated to the main system only after successfully passing these tests. If any problem is found during integration tests, development activity is restarted. Developers will then propose and test possible corrections to the problems found.

When the new version passes the integration tests, next iteration planning takes place. New requirements will be selected, according to the clients' defined priority. When a significant number of modifications is integrated to the main stream, a working version, called "release candidate", is produced and submitted to tests by the clients.

5.8 Client's tests

In this activity, clients are requested to perform functional, usability and interaction tests on the release candidate. If any problem is detected, or if any improvement is perceived by the client, user stories and interactivity requirements can be redefined. When clients' approval is obtained, a small version - which successfully implements a subset of proposed requirements - is delivered.

Specifically for the case of VRS development, interactivity tests play a major role in the overall development process. Therefore, interaction tests should be exhaustively executed by developers and clients, in order to avoid a significant drop in system's usability and efficiency, caused by poor interactivity.

6. Conclusions and future work

Agile software development processes and practices can be adapted to the development of Virtual Reality systems. In particular, the iterative nature, the embrace of requirement's changes and the importance given to tests are some of the characteristics that favors their application in VRS development, since these same characteristics are shared by many VRS projects.

Architectural spikes are considered a major improvement in the VRS development process, since they allow developers to conduct experiments in a constantly changing technology environment. The correct elucidation and definition of interactivity requirements has a strong effect on system's resulting usability and efficacy. Finally, given the subjectivity of some VRS concepts - such as systems' interactivity quality - stakeholders' participation in the development process leads to the production of improved quality systems, and results in well satisfied clients.

A quantitative evaluation of the presented process' application in a case study is an interesting proposal for future works. Also, the extension of the proposed process to other domains - such as Augmented Reality or mobile application development - is a valuable subject for future research.

Acknowledgments

This research is supported by FAPEMIG (Minas Gerais State Agency) to which the authors are deeply grateful, as well as to CAPES/Brazilian Ministry of Education & Culture and to the National Counsel of Technological and Scientific Development (CNPq).

References

[1] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, C. Jim, and H. K. A., *Object-oriented analysis and design with applications*, 3rd ed. Westford: John Wiley & Sons, 2007.

[2] K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. J. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," Available: http://www.agilemanifesto.org, 2001, accessed March 25, 2009.

[3] E. G. d. Costa-Filho, R. Penteado, J. C. A. Silva, and R. T. V. Braga, "Padrões e métodos ágeis: agilidade no processo de desenvolvimento de software [Agile patterns and methods: agility on software development process]," *5th Latin American Conference on Pattern Language of Programming*, vol. 5, pp. 156–169, 2005.

[4] G. J. Kim, K. C. Kang, H. Kim, and L. Jiyoung, "Software engineering of virtual worlds," *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pp. 131–138, 1998.

[5] R. Tori, C. Kirner, and R. Siscoutto, Eds., *Fundamentos e tecnologia de realidade virtual e aumentada [Virtual and augmented reality fundamentals and technology]*. Porto Alegre: SBC, 2006.

[6] A. Cockburn, *Agile software development*. Boston: Addison-Wesley, 2002.

[7] K. Beck, *Extreme programming explained: embrace change*, 2nd ed. Addison-Wesley Professional, 2004.

[8] J. Hunt, *Agile software construction*. London: Springer, 2006.

[9] M. A. Khan, A. Parveen, and M. Sadiq, "A method for the selection of software development life cycle models using analytic hierarchy process," in *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on*. IEEE, 2014, pp. 534–540.

[10] V. Subramaniam and A. Hunt, *Practices of an agile developer*. Dallas: Pragmatic Bookshelf, 2006.

[11] G. Kumar and P. K. Bhatia, "Comparative analysis of software engineering models from traditional to modern methodologies," in *Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on*. IEEE, 2014, pp. 189–196.

[12] M. Cohn, *Succeeding with Agile - Software development using Scrum*. Boston: Pearson, 2010.

[13] A. Stellman and J. Greene, *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. "O'Reilly Media, Inc.", 2014.

[14] M. Cohn, "Scrum & xp: Better together," Available: https://www.scrumalliance.org/community/spotlight/mike-cohn/april-2014/scrum-xp-better-together, 2014, accessed March 10, 2015.

[15] K. Waters, "Extreme programming versus scrum," Available: http://www.allaboutagile.com/extreme-programming-versus-scrum, 2008, accessed March 10, 2015.

[16] G. J. Kim, *Designing virtual reality systems: the structured approach*. London: Springer, 2005.

[17] F. Brooks, "No silver bullet: essence and accidents of software engineering," *IEEE computer*, vol. 20, no. 4, pp. 10–19, 1987.

[18] D. Pilone and R. Miles, *Head first software development*. Sebastopol: O'Reilly Media, 2008.

[19] B. W. Kernighan and R. Pike, *The practice of programming*. Reading: Addison-Wesley, 1999.

[20] A. Oram and G. Wilson, Eds., *Beautiful code*. Sebastopol: O'Reilly Media, 2007.

[21] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[22] J. Shore and S. Warden, *The art of agile development*. "O'Reilly Media, Inc.", 2007.

[23] A. Cockburn, *Agile software development: the cooperative game (agile software development series)*. Boston: Addison-Wesley Professional, 2006.

[24] P. Kruchten, *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.