Towards Agent-based Agile Approach for Game Development Methodology

Rula Al-azawi
Gulf College, Oman
Email:rula@gulfcollegeoman.com

Aladdin Ayesh
DMU University, Leicester, UK
Email:aayesh@dmu.ac.uk

Mohaned Al. Obaidy
Gulf College, Oman
Email:mohaned@gulfcollegeoman.com

***Abstract*—Game development is very complex and the success of the game is based on the game development methods. The purpose of this paper is to investigate on the existing game development methods and provide an upcoming game development method that is based on predictive and adaptive development models. A critical analysis to Agile method which are mostly used in modern game development methods is presented. We identified the weakness of Agile game development and solve it by creating a cooperation with Agent Oriented Software Engineering (AOSE) to introduce a new hybrid methodology named as Agent Agile Game Development Methodology (AAGDM) that combines both predictive and adaptive models.**

I. Introduction

Game creation nowadays is an incredibly complex task, much harder than someone might initially imagine. The increased complexity is combined with the multidisciplinary nature of the process of game development which includes art, sound, gameplay, control systems, artificial intelligence and human factors, among many others. The interact with the traditional software development creates a scenario which also increases this complexity. In this connection we need a methodology for taking into account software engineering expertise in the field of games.

As we know, the gaming industry is very powerful in the entertainment industry, having billions of dollars in profit and creating trillions of hours of fun [1].

Through the process of researching, a number of development models has been used. This paper focused on two archetypical development models, the predictive and the adaptive models [2].

The 'Waterfull' model is influenced by predictive development models while 'Agile' model is influenced by adaptive development models. Both of which are explained further in section 3. Each technique has diverse characteristics and features that differentiate it from other processes. Processed can be classified as either a heavyweight or a lightweight method. The heavyweight method includes traditional methods like waterfall model. In contrast, the lightweight methods are also known as Agile methods [3].

It is important to have formal understanding of game development process, and how we could create a formal game development methodology that will be generic for many game genders. This paper is structured as follows: section 2 presents an overview of the current game development methodology; section 3 explains the archetypical development methodology such as Agile methodology and AOSE; section 4 explains the critical analysis of the problems in current game development methodologies; section 5 presents the new game development methodology AAGDM which solved the problems from section 3; section 6 presents a critical evaluation of the AAGDM; and section 7 presents conclusion and future works.

II. Current Game Development Methodologies

Game development has evolved to have large projects employing hundreds of people and development time measured in years. Unlike most other software application domains, game development presents unique challenges that stem from multiple disciplines which contribute to games. A major issue against the game development industries is that many companies adopt a poor methodology for game creation[4].

There are many methodologies available in traditional systems and software development. Some of these methodologies include Waterfull, Incremental and Spiral. Each of them are structured as a linear or iterative and sometimes hybrid of both and are usually used in game development methodology.

Most of the linear manner methodologies are classified as predictive even if it contains some iteration but it usually follows sequence phases such as waterfull methodology. While prototyping involves breaching the system into small segments. Furthermore, it involves the user in the process.

The Spiral methodology combines the linear and iterative framework. Spiral development breaks the projects into number of cycles, all of which follow a set of increasingly larger steps.

AAGDM is a hybrid between predictive model using AOSE methodology and adaptive model using Agile methodology.

III. Archetypical Development Methodologies

Through the research of game development methodology, we have two archetypical development methodology predictive and adaptive.

This section will answer the following questions: what are predictive and adaptive methodologies, how we could choose between them in game design development process and finally how we could combine components from different variety of game design and integrated into standard game development methodology which needs to be generic and suitable for different game genres.

The majority of methodologies taken and used by game developers can be described as predictive, comprehensively planning as a separate task prior to actual development; or adaptive, using multiple iterations and prototypes to shape a game and its design based on feedback and analysis [2].

In general the predictive models would be preferable when we have clear goal and the customer requirements are clear and

978-1-4799-3351-8/14/$31.00 ©2014 IEEE

complete and the specific structure of the game must withheld at all costs, allowing for a definite vision of the final product to be established long before it takes a playable form as shown in Figure 1.

Regarding adaptive models which encourage the change in the

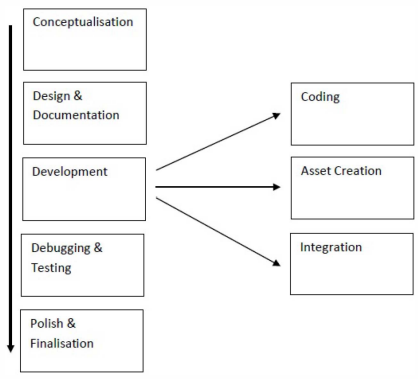

Fig. 1. Predictive development methodology

customer requirements and customers allowed to add new goal or new requirement even in the late stage of games change and thus will not affect the game plan. Furthermore the customer allows to give direct response to its development process and the lessons learned within as shown in Figure 2. We will take

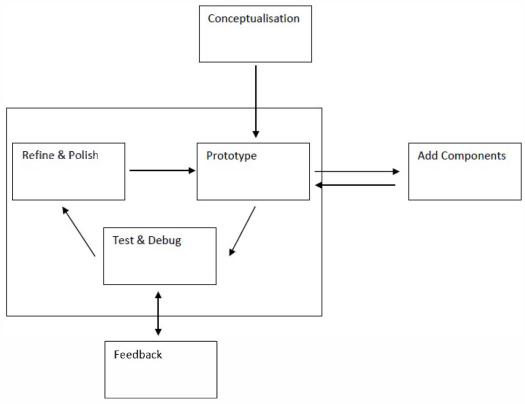

Fig. 2. Adaptive development methodology

AOSE methodology as an example of predictive methodology and Agile as an example of adaptive methodology.

A. AOSE Methodologies

The relationship between games and AOSE is clear given that software agent or intelligent agents are used as virtual players or actors in many computer games and simulations. The development process is very close to the process of game development[5].

There are several methodologies in AOSE. Each one has its own life cycle. However, some of them are precise only analysis and design such as Gaia, while others cover complete life cycle such as Trops, MaSE and Prometheus as shown in Figure 3.

Within the last few years, with the increase in complexity of projects associated with software engineering, many AOSE methodologies have been proposed for development purposes[6]. Nowadays, intelligent agent-based systems are being applied in many domains, including robotics, networking, security, traffic control, games and commerce [7].

The goal when evaluating AOSE methodologies is to discover the most convincing methodology for adaption to game development and incorporation of modifications. Al-Azawi et al [7] focus on comparing different AOSE methodologies from the perspective of the game development domain. The results of their experiment were summarized to select the MaSE as a methodology to be adopted as a game development methodology.

We have selected the MaSE methodology to be adapted for game development methodology for the following reasons:

1) MaSE has a full life cycle. [8].
2) MaSE is influenced by the software engineering root.
3) MaSE is perceived as significant by the agent community [9].
4) MaSE has been selected by [7] as the game development methodology.
5) MaSE has been selected according to many references such as [10] as a methodology for robotics, which is similar to the game area.
6) MaSE has defined the goal at the first stage and each goal has to be associated with its role, which is an important feature of game development.

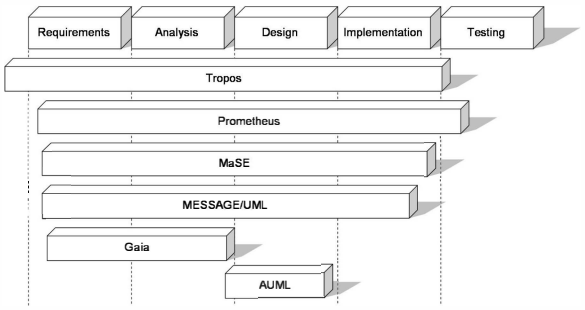

Fig. 3. AOSE life cycle coverage

1) **MaSE Methodology:** MaSE stands for Multi-Agent Systems Engineering. It is a complete life cycle methodology to help the developer work with a multi-agent system from the start to the end. This means that it describes the process which guides a system developer from an initial system specification to system implementation. In each step, related models are created. Models in one step produce outputs which become inputs to the next step that supports traceability of the models across all of the steps. Furthermore there is possibility for free access between components in each phase.[11]. The goal of MaSE is to guide the system developer from the initial system specification to system implementation[10].

B. Agile Methodology

Agile methodology is based on implementation over documentation with customer collaboration and has the ability to solve problem and change with agility.

As use of Agile development grew, a number of different methodologies surfaced. Some were derived from Agile, others were systems that had been in use but never fully defined or applied to software development. One such method was Scrum [12].

The main characteristics of Agile methodologies are: customer cooperation, simplicity, individual, interaction, adaptiveness and being incremental. These characteristics are important to understand an approach to game development based on an Agile methodology. [13].
The Agile methodology as mentioned earlier is an iterative and incremental approach and it achieves the quality and productivity through iterations. Each iterations of sprint phase includes a software development team working through a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing as shown in Figure 4 which was adapted from [14] and [15].

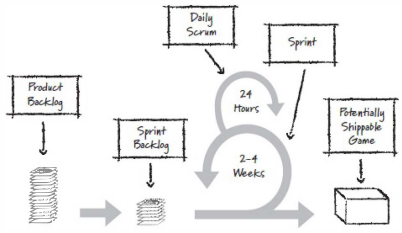

Fig. 4. Agile methodology diagram

The Agile phase approach diagram which is used by Keith [16] as shown in Figure 5 shows that Agile methodology is based on iterations that could start new iteration before completing the previous iteration. The Agile methodology

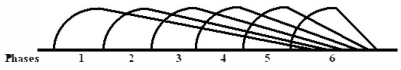

Fig. 5. Agile phases approach

has been discussed the use and application through iterative development framework named Scurm [16]. The Scurm game development method is an Agile process which manages game development using iterative and incremental approaches which are the life of the game project. It works in game development methodology by breaking down the process of creating game into series of tasks named ”sprints”. To facilitate the work with sprints, the game developers break up the game into groups of related tasks or features that must be written in the product backlog. As mentioned in Figure 5, every two to four weeks at the end of sprint phase, the whole teams met to discuss the current state of games to improve version of game to the stakeholders, and to select new tasks from backlog.
According to Keith [16], the Agile game development with Scurm could be labeled as iterative and adaptive model.

IV. Critical Analysis of Current Game Development Methodologies

A vast majority of the problems facing the game industry and development are deep seated in the very production methodology that is employed. Teams of approximately 100 people are still using methodologies developed for a time when ten people were considered a bloated team [12].
Games and software engineering have important things they can learn from one another and mostly they share the same methodology and same problems. The game contains a confluence of interesting properties, such as emergence, real time interaction and challenge components that create a new field of study [17]. The software engineering has much to help the game industry to solve problems. The unique aspect of game which is not available in traditional software development is the requirement for game to be ‘fun‘ which has no metric to apply; it is purely subjective.
There are specific features of game development that have been found to prevent the success of great games. The major problems that arise are in the areas of project management. The use of methodology focuses on game development and takes into account the project management concept to help avoid management problems.
After a survey of the current game development methodology problems, we would highlight the main problems found in the literature:

- **Schedule problems:** According to Flynt et al [18], a key reason for a project being delivered behind schedule is that no target was established. Likewise, problems may occur when a deadline estimate does not include the time needed for communication, lacks documentation or emergent requirements that may alter the system architecture and thereby cause serious problems. Furthermore, delay can be caused by a multidisciplinary approach. Since it is essential to include input from different teams, delays may occur. A task involves a series of risks that imply underestimates, causing cumulative schedule delays. Flynt et al [18] report that developers recurrently fail in their estimates due to lack of historical data to assist them in determining a realistic time frame to carry out a task [19].

- **Crunch Time problems:** In the game industry, crunch time is a term usually used for the period of work when overload may happen; usually it happens in final weeks before the validation phase or deadline for project delivery.
  In this period of time, the developer may work in excess of 12 hours a day and take from 6 to 7 days to complete unfinished tasks. In the game industry, crunch time is a fact of life [19].

- **Scope and feature creep problems:** Feature creep is a term used in the game industry when a new functionality is added during the development phase to increase project scope and change schedule time [19]. Any new functionality should be evaluated carefully. Any unmanaged feature creep can lead to increased error, possible defects and increased chances of failure. However, some feature creep is unavoidable, since it adds fun to the game [4].
  The biggest reason for game project imperfection is failure to accurately establish project scope.
  Risk management helps the project manager to understand the changes to a plan and the potential costs in time and money.
  Project scope will never be a true reflection of the required effort, due to the iterative and exploratory nature of game development; however, it can be an effective guide when predicting success, such as when

discussing milestones, time lines, and budget [4].

- **Technology problems:** All games are technology dependent. Technological components generate risks for game projects that can require greater effort and a high investment of time. According to Gershenfeld et al [20],technology risks are generally high when a team works on a new platform because of two risks. The first risk is that the developer has not worked with the technology before. The second risk is that frequently the related hardware contains problems.
- **Documentation problems:** Lack of documentation is a common source of additional problems. The documentation can be valuable in reducing feature creep. Having a finite amount of documentation is useful when game developers work on difficult projects, as this helps to obtain a good estimate for project scope and schedule. Usually, GDD generates a lot of uncertainty around a games goal and solution requirements [21].
- **Collaboration and Team Management problems:** One of the main problems when creating games is the communication between teams. The teams in games include people with distinct profiles, such as developers, plastic artists, musicians, scriptwriters and designers. Different teams need to collaborate and explain their work and instructions to others.
- **Training problems:** One of the biggest problems in Agile game development especially and generally in game development methodology is new employee training.
- **Linear process problems:** Game development is not a linear process [18]. Iteration is the life of game development. Game developers use Waterfall methodology with enhancements, by adding iteration to the methodology.

Petrillo et [19] present in Figure 6 the histogram of occurrence of problems in decreasing sequences.
From the previous study, we can observe that most traditional software problems are the same as game development problems. In the following section, we propose game development methodology that resolves most of the previous problems.

V. Agent-Agile Game Development Methodology (AAGDM)

AAGDM methodology has attempted to provide an adaptive and predictive development lifecycle. Sometimes a combination of those models maybe more suitable [15].
Agile methodology is usually used to deal with dynamic changes in requirement specification by the customer, customer involvement in the development phases. For the flexibility in adding new requirements even before game release which does not add extreme cost to the project, Agile game development methodology will be adapted to suggested game development methodology as adaptive model.
AOSE provides such intelligence through agents. Agent may perform the tasks individually. In complex and distributed system, Agents can be used to monitor the interaction among components and to interact as human interaction. The MaSE used in the Sprint phase is at the core of the AAGDM. Each

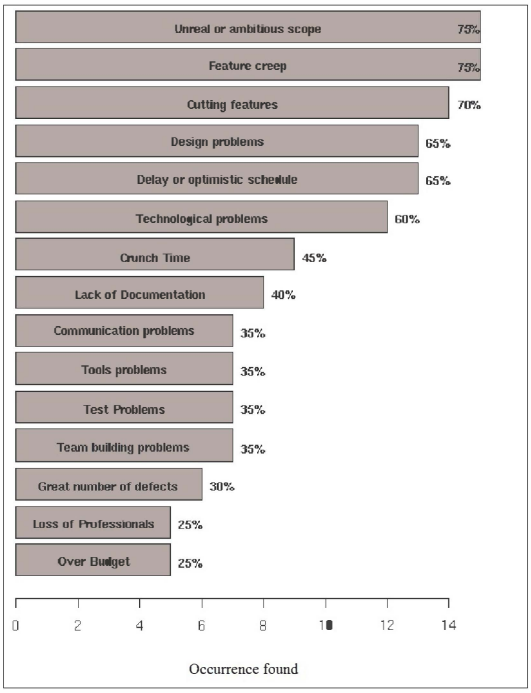

Fig. 6. Occurrence of problems in current Game Development Methodology

iteration includes analysis, design, implementation, testing and evaluation of MaSE as shown in Figure 7.
The reason to adapt MaSE is that it is the core of Agile, because in complex systems and distributed systems such as games, it is difficult to trace a single point of control, since the objects are distributed [15].
Figure 7 illustrates the suggested game development methodology which we name as Agile-Agent Game Development Methodology (AAGDM).

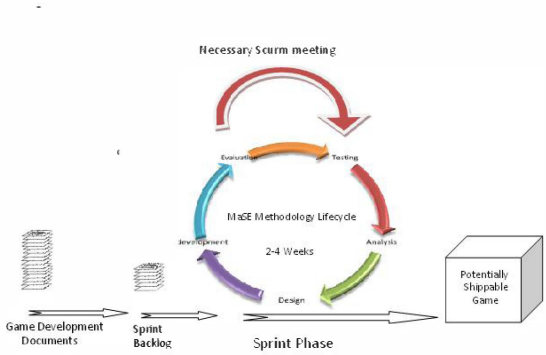

Fig. 7. Agent-Agile game development methodology diagram

VI. Critical Analysis of AAGDM

The creation of GDD is an important step in preproduction phase in the game, being responsible for guiding the projects scope and effect to development and testing phases. There is no standard way to build GDD, but it must have a comprehensive description of the game in all its aspects and describes the

objects and characters in the game. This affects how their interaction, role and behavior in the game. The GDD will change many times and will add extra requirements but we should evaluate the risks of changes if the deadlines can still be met. Later, GDD will be translated to a Product Backlog in the production phase. For small games, it may be optional which translates the requirements directly as a Product Backlog. This may save time from the team as they go faster to the production, but may also increase risks of feature creep or it may not become an entertaining game.

If GDD is designed carefully, the project manager can plan the iteration of sprint backlog so that the game is playable at the end of each iteration. This has several benefits. For one, testers can check the game for errors in a playable state which mimics what the end-user would encounter. Having a playable game as early as possible helps the team to see the potential of the end product, and it could be beneficial in game publication before final release.

It is a need to think to get a better approach. Keith [16] suggest that at the end of sprint, we could start new sprint even if there is still work that are still under development. The goal is to achieve a continuous flow in the content of creation as shown in Figure 5 which is the core concept in AAGDM.

The second step in AAGDM and the same with Agile methodology is Sprint backlog. The GDD should be transferred to Sprint backlog.

At each iteration in the game life cycle, the most important backlog should be started first and then divided into smaller pieces.

The last step which is the core of AAGDM is Sprint phase. This phase deals with MaSE to cover the sub-phases of AAGDM instead of dealing with standard waterfull life cycle that has been used in Agile game development methodology.

The purpose of working this way is to show customers the value of a feature every two to four weeks, show how it improves sprint-by-sprint and at the same time acquire documentation that will be useful in the evaluation or creation of the new game version.

When AAGDM uses Agile concepts, we improved the quality and efficiency of large and complex games projects. Furthermore, it strengthens the communication between the developer and the end user.

The management is important in game industry. Poor management can negatively affect the best of teams. While the complexity in game and number of teams increases good communication in a company is necessary for success. Agile methodology usually depends on daily Scurm meeting to get good communication, but in many times there is no need to discuss daily because it is only a waste of time for the teams. AAGDM is an iterative methodology that focuses on delivery features. The AAGDM has the ability to start dealing with new features before completing a current feature. In this case, game development duration will be reduced because there is time wasted on waiting.

In the planning part, the customer and developer usually cooperate to select new features. Then new features add to the sprint backlog to discuss if the features have highest priority.

The effort of using only AOSE will mostly be expended on the preparation of the documentation, as shown in Figure 8 [22]. AAGDM reduces documentation by creating Game development documents and dividing these into sprint. AAGDM prefer software development over documentation. The game documentation is important and required in the analysis and design phases because we need those details to maintain games or to create new versions of the game.

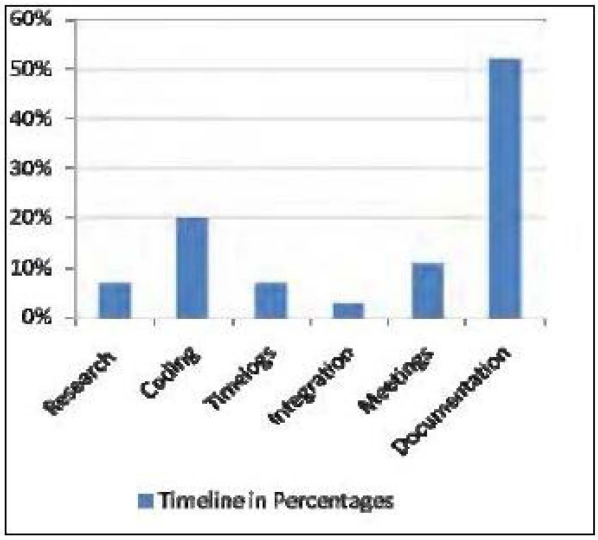

Fig. 8. Duration for AOSE

In AASDM, we suggested the meeting is not necessary in daily basis to save time. It should only be done when important issues arise from the multidisciplinary teams such as artists, musicians, developer and clients. Furthermore, the group may have sub-group such as AI team or a textures team because AAGDM requires the creation of functional unit combinations of specialties. An example would be a unit composed of two programmers, a texture artist, and an animator. Combining groups enhances communication across disciplines. Bringing the diverse groups together enhance understanding and communication between teams [4].

Regarding project scope and feature creep, we have many situations in the game industry that show the many features discovered during game development. These features can transfer into success in a game.

AAGDM is not a linear process, it is an iterative process. Thus, if an interesting feature is discovered, it must be analyzed in terms of its risk and, if viable, it should be added to the project schedule [19].

In Figure 9, we noticed that the cost of change in traditional software increased in terms of late project deliver. In Agile delays can increase also, but these would normally be towards the end of the project [23], where it becomes necessary to pursue a better approach.

Keith [16] suggests that at the end of sprint, there is still work that may be under development. The goal is to achieve a continuous flow in the content of creation as shown in Figure 5 which is a core concept in AAGDM.

VII. Conclusion and Future Work

In this paper, we dealt with current trends of software engineering methodologies and we found that sometimes a combination of those models may be more suitable.

AAGDM combines agile methodology that meets the dynamic requirements of the customer with AOSE which is

Fig. 9. cost of change over time

a rapidly development area of research designed to support development of complex and distributed system in open and dynamic environments with the use of intelligent component. Game development methodology worked better when we used iterative methodology because it allowed to have the features ready and to discover and work the fun of the games easier.
Our AAGDM solved the most of the previous problems in game development by considering being suitable for searcher and professional in the industry.
Future work in this line of research includes evaluating the performance of AAGDM. Overall, predictive models would be preferable when there is a pre-defined customer expectation or specific structure. The game must withhold at all costs, allowing for a definite vision of the final product to be established long before it takes a playable form. Adaptive models encourage change and thus they do not usually allow for all aspects of a game to be planned in unison, seeking to allow a games final project to be a direct response to its development process and the lessons learnt within [2].
Ideally, the type of hybrid development methodology approach which we already defined in AAGDM is recommended for use by independent game developers. This possesses a mix of characteristics that would sit somewhere between those of a predictive or adaptive approach to be generic methodology useful for small or large game projects.
Since there are few academic studies on game development generally and on Agile and AOSE methodologies, this work opens up perspectives for future research.

REFERENCES

[1] F. Petrillo and M. Pimenta, "Is agility out there?: agile practices in game development," *SIGDOC '10 Proceedings of the 28th ACM International Conference on Design of Communication*, pp. 9–15, 2010.

[2] L. Hunt, "Predictive and adaptive game development a practical application of development models to the independent video game industry," Master's thesis, School of Communications and Arts, 2011.

[3] C. Raghaw, V. Sharma, and R. Singh, "An Experimental based Study on Challenges of Game Development with Scrum using Agile," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 9, pp. 255–261, 2012.

[4] C. Kanode and H. Haddad, "Software Engineering Challenges in Game Development," in *Information Technology New Generations 2009 ITNG 09 Sixth International Conference on*. IEEE Computer Society, 2009, pp. 260–265.

[5] Gomez-Rodriguez, A. and Gonzalez-Moreno, J.C. and Ramos-Valcarcel, D. and Vazquez-Lopez, L, "Modeling serious games using AOSE methodologies," in *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2011, pp. 53–58. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6121630

[6] O. Akbari, "A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance," *Journal of Computer Engineering Research*, vol. 1, pp. 14–28, 2010. [Online]. Available: http://www.academicjournals.org/JCER/PDF/Pdf2010/April/Akbari.pdf

[7] R. Al-Azawi, A. Ayesh, I. Kenny, and K. AL-Masruria, "Towards an aose: Game development methodology," in *Distributed Computing and Artificial Intelligence, 10th International Conference. Advances in Intelligent and Soft-Computing series of Springer*, May 2013.

[8] J. Sudeikat and L. Braubach, "Evaluation of agent oriented software methodologies examination of the gap between modeling and platform," in *Agent-Oriented Software Engineering, P. Giorgini, J. P. M uller, and J. Odell, Eds. Lecture Notes in Computer Science*, vol. 3382. Berlin, Germany: Springer Verlag, 2005, pp. 126–141.

[9] Dam, K and Winikoff , M, "Comparing agent-oriented methodologies," in *Proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems (AAMAS).*, vol. 3030. Melbourn, Australia.: Lecture Notes in Computer Science; Springer Berlin / Heidelberg, 2004, pp. 78–93.

[10] S. DeLoach, E. T. Matson, and Y. Li., "Applying agent oriented software engineering to cooperative robotics," in *the The 15th International FLAIRS Conference (FLAIRS 2002)*, Pensacola, Florida,, May 2002, pp. 391–396.

[11] S. DeLoach, "Multiagent systems engineering of organization-based multiagent systems," in *SELMAS 05: Proceedings of The 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*. New York, NY, USA: ACM., July 2005, pp. 1–7.

[12] R. McGuire, "Paper burns: Game design with agile methodologies," *Gamasutra: The Art and Business of Making Games.*, pp. 1–7, 2006.

[13] A. Godoy and E. Barbosa, "Game-Scrum: An Approach to Agile Game Development," in *Proceedings of SBGames 2010 Computing*, I. S. F. SC, Ed., November 8th-10th 2010, pp. 292–295.

[14] H. Takeuchi and I. Nonaka, "The new product development game," in *Harvard Business Review*, January- February 1986, pp. 137–146.

[15] M.Nachamai, M.Senthil, and V.Tapaska, "Enacted software development process based on agile and agent," *International Journal of Engineering Science and Technology (IJEST)*, vol. 3, no. 11, pp. 8019–8029, November 2011.

[16] C. Keith, *Agile game development with Scrum*. Addison-Wesley Signature Series, 2010.

[17] C. Lewis and J. Whitehead, "The whats and the whys of games and software engineering," in *Proceedings of the 1st International Workshop on Games and Software Engineering, GAS '11*. New York, New York, USA: ACM Press, May 2011, pp. 1–4.

[18] J. Flynt and O. Salem, *Software Engineering for Game Developers*, illustrated, Ed. Course Technology Ptr, November 2005.

[19] F. Petrillo and M. Pimenta, "Houston, we have a problem...: a survey of actual problems in computer games development," in *In SAC08: Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 707–711.

[20] M. L. A. Gershenfeld and C. Barajas., *Game plan: the insiders guide to breaking in and succeeding in the computer and video game business.* St. Martins Griffin Press,New York, 2003.

[21] R. Kortmann and C. Harteveld, "Agile game development: lessons learned from software engineering," in *Proceedings of the 40th conference of the international simulation and gaming association*, 2009.

[22] M.Nachamai, M.Senthil, and V.Tapaska, "Enacted software development process based on agile and agent," *International Journal of Engineering Science and Technology (IJEST)*, vol. 3, no. 11, pp. 8019–8029, November 2011.

[23] V. Szalvay, "An Introduction to Agile Software Development," *Danube Technologies, Inc., Bellevue, WA*, no. November, 2004.