Agile Approach in Game Development (April 2020)

Vladimir Solovev

Innopolis University, Russian Federation
MSIT-SE
v.solovev@innopolis.university

Abstract— The current cost of game production reached millions of dollars and the return of investment can make or destroy a game studio, so industry is in high demand for modern and reliable approaches to the game development process.

There are a lot of examples applying Agile methodologies to Game Development, but it still needs to be better adapted to the reality of Game Development and especially to teams, which are always multidisciplinary and includes not only developers, but also different media artists. In this paper, we will discuss the common approaches to Agile in Game development such as Game Scrum, eXtreme Game Development, Game unified process, Triadic game design, Agent based Agile game development and User centered design. We will also discuss the industry processes history of Waterfall approach and related issues. Among all the problems discussed above the implementation of Agile itself is not so easy by itself for the teams, not having experience with it, so Agile itself and the way to learn and implement it will be part of the paper.

Index Terms— Game Development, Agile Methodologies, Scrum, XP, Software Process Improvement (SPI), Software Engineering, Systematic Literature Review (SLR).

1 Introduction

The Game development process nowadays has evolved in a very complex activity. Many different areas professionals are involved such as software engineers, game designers, art and media design professionals and software development management. Hard deadlines for launching the products on time to market creates the needs to develop the strict schedule for development and it often meets the poor time estimation skills of development teams and lack of experience in game development project management. The overall costs of game development also reached millions of dollars and every publisher struggle with a lot of financial risks [GameScrum], related to the success of the final product.

All these issues create the needs of a special game development processes, ensuring the end-product will be developed and released as planned without much delays and defects.

Despite the fact, that a lot of teams now have the proper experience applying the modern software processes like Agile to traditional forms of software, a lot of game development features like for example ‘fun factor’ can ruin the success of whole project. Unfortunately, there are not much open data or literature, describing the prosperous application of Agile methodologies to game development.

The main goal of this paper is to give examples of some proven by industry experience approaches, based on the most popular Agile approaches such as Scrum, Kanban and eXtreme Programming.

This paper is organized in the following manner. Section 2 describes the original Agile approaches for the unprepared readers. Section 3 covers the traditional waterfall approaches in video game development and the reasons the industry is switching to iterative methods. In section 4 we will look at game industry related problems such as lack of management experience and quality, fun factor, creation of game design document, one of the most important game development deliverables, and will provide some of game development postmortem analysis (PMA). Section 5 will cover the related work found in papers and literatures such as Game Scrum, eXtreme Game Development, Game unified process, Triadic game design and Agent based Agile game development. In section 6 we will discuss such topic

as teaching Agile. A lot of universities give a very basic and abstract level of knowledge without any practical experience in applying it to real projects and that appears to be a really big issue for the industry. Finally, the last section will include the conclusion for the whole work and discussion for the future study in the area.

2 AGILE METHODOLOGIES

For better understanding of applying Agile approaches in Game development, we will need to give some brief introduction for the unexperienced readers to Agile itself and its most popular frameworks nowadays, such as Scrum, eXtreme programming (XP) and Kanban.

One of the most reliable documents, describing Agile, is the Agile manifesto, proposed by Beck [Beck, 2001]. According to it the main advantages of Agile is adaptivity, cooperation with Stakeholders and incremental approach. Here are the top and well-known Agile principles:

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

Agile is mostly focused on the product, not the way of development. It gives us the advantages to adapt our development for new changes, focus on features delivery more than documentation, more opportunities to communicate with Stakeholders and elicit more actual requirements.

In [Agile, 2003], Martin defines 12 principles about what Agile truly is:

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 is behavior accordingly.

2.1 eXtreme Programming (XP)

Extreme Programming is one the most popular Agile approach, it was proposed by Beck [Beck, 2009] and builds on five core values [Beck2, 1999]:

- communication: verbal communications between developers and simple design is more important than comprehensive documentation
- courage: courage to throw away bad and redundant code, courage to test and refactor all the code complexity
- simplicity: start with the simplest solution, more complex features can be added later
- feedback: from the team, from the Customers, from the system by performing testing
- respect: respect in team between each other. Never commit the code, that could break some tests.

The eXtreme programming takes its name in the meaning of taking all traditional software development and get it to ‘extreme level’. For example, code review in basics development approach is conducted at some predefined phases, but on XP it is conducted continuously like in pair programming, when one developer writes code and the second one immediately reviews it.

The role of the process leader who decodes how to implement one or another value goes to the coach or the project leader, who fully decides the application of XP to the current project.

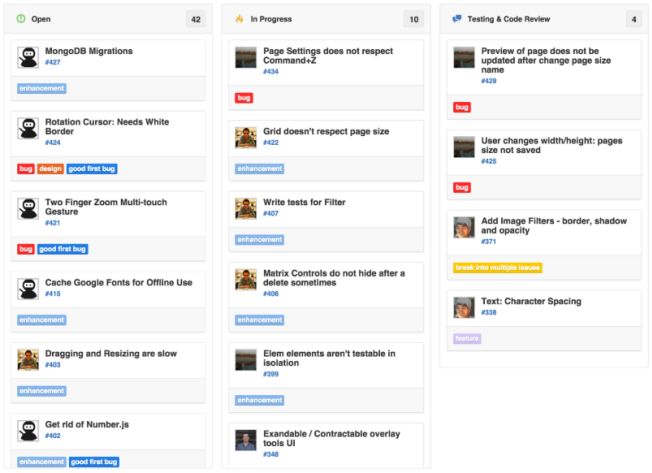
2.2 Scrum

Scrum is all about iterations. At the start of the Scrum cycle, which is called the sprint, we have the list of user stories (backlog), the tasks that either you either the customer is planning to do on current iteration. During the iteration you control the completeness of your user stories splitted in a more common tasks on a burndown chart, and then control the execution of the project.

There are three main roles in the Scrum process – the Product owner which owns the product and control all the user stories and sprint planning, the Scrum master who conduct the Scrum meetings and helping the team with any problems. And the last, but most important is the Team. The Team develop the software by completing the sprint tasks. In the process there is a big importance in trust of the Product owner and the Scrum master to the Team, because Team must be professionals in their area and there must be assurance, they will do their work in time.

2.2 Kanban

Kanban is a Japanese term and the Kanban method was first implemented in Toyota Motor Corporation manufactures processes. Due to the Toyota business in cars manufacturing, the Kanban is a conveyor or a pipeline-based method.



Kanban board example on GitHub

Kanban process utilize the Kanban board that usually consists of the following fields (related to software development in our example) – Open/To Do, in Progress, need code review, need code testing, done. Every task depending on its status travel on the board from left to right. All the tasks for the project are stored in backlog.

One of the keys to success in Game development is to provide the end users the 'Fun factor' and the focus on it must be improved. The best way to do it is the iterative approach and Agile is the best solution for it [GameScrum]. Through Agile iterative process we can get an early feedback on every feature we developed and add to the product, we can also estimate the 'Fun factor' in time related to every new iteration value.

3 Traditional Game Development

There are books and literatures, describing the Game development process. One of the most cited work [Game-Agile] in game community belongs to Bethke [Bethke] and it mostly advocates about Waterfall approach and propose to apply Unified Software Development Process to develop electronic games, which are not iterative procedure.

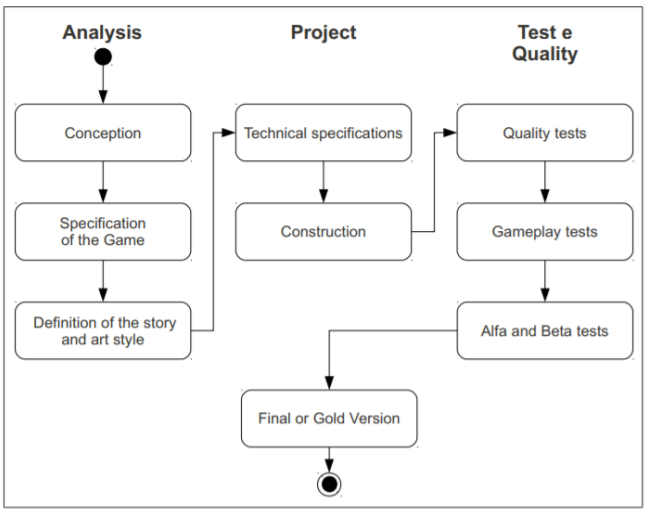
The Waterfall model in game development describes first to define the game rules and then create the Game Design document, one of the most important deliverables in game development software process.

The game rules describe almost everything related to the game: the characters, the interaction between the characters, game balance, game world and environment and others game related things. We can say that the whole game design process is to create all those rules [Game-Agile].

The game Design document (or the Project document) is the formal and most important document, which includes all the game rules, created by the team in informal form. All the game mechanics, what player can do in the created game world and environment. The 'Fun factor' must also be included in Game Design document, what actions will lead to the game user satisfaction, what challenges you need to provide to your game end customer to make him feel fun from the product. Also, it should include the story, all the characters, all locations, items, game balance, design of the world and the characters. In other words, it could be called an architecture design of the whole game project.

There is also a document called Proposal Document and it is more related to business side of the game development. It analyzes such topics as development schedule, budget to fulfill the project, the needed team and its skills (it is also called a team profile), the platform to release the game and technologies for development and a high-level description of the game for management or investors.

Almost every game until 2005 years used a Waterfall based process [Rucker] and you can see example of it on the figure below.



Waterfall process in Game Development

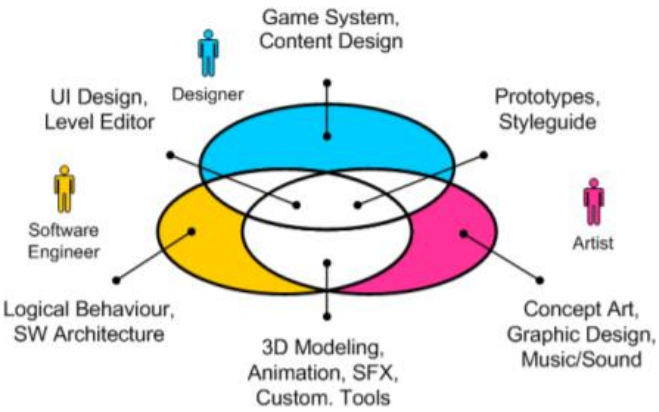
As we already mentioned modern game development struggles with a very high game cost and changing requirements, and it makes it very important to know as soon as possible will investment return or not. In traditional game development utilizing the waterfall model all these knowledges can be found only on the last stages of development, when all the budget is spent, and the development schedule reached its end [GameScrum]. The risk is too much on that stage to make any changes to the

product.

The Agile process let you focus on what is important, on 'fun factor', on each iteration. And it helps you to avoid the risks of failure with the problems, described above. It still has its problems, but the industry is moving to Agile [Game-Agile]. According to the National industry survey in Austria [FlexibleProcess] the new game development software trend is a flexible process.

4 Problems in Game Development

The modern games are very complex software systems, requiring top quality real time 3d graphics, artificial intelligence, a lot of audio content including game music and voice acting, and highly reliable software architecture for cross platforming applications. All of it requires to connect in one project many of different multidisciplinary specialists. And creating such environment for creativity often leads to conflicts and misunderstanding between the artist and developers [Petrillo].



Multidisciplinary activity in Game Development

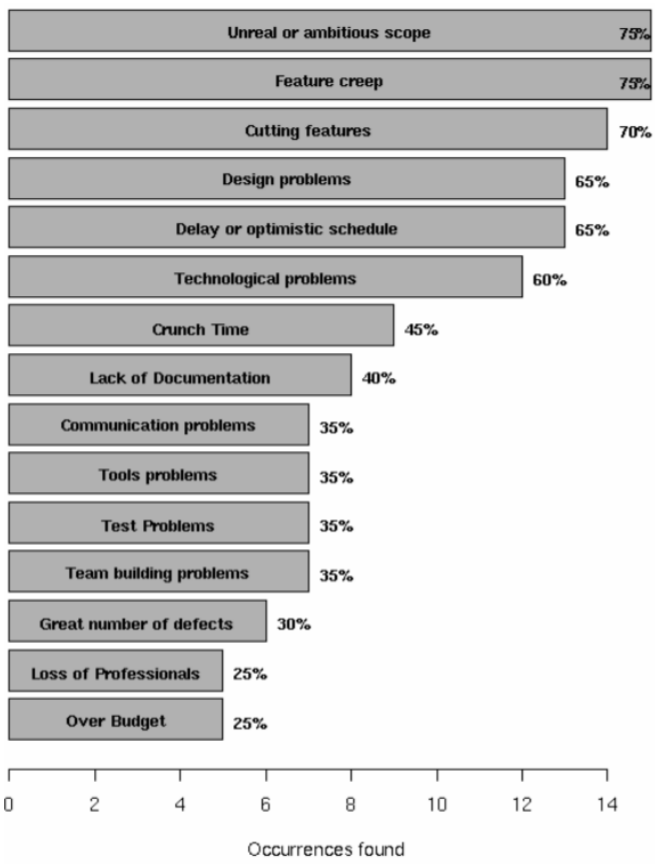
Collaboration within those teams are the major challenges in Game Development [FlexibleProcess]. Get a common view on the product, successfully exchange the ideas and methods and to apply cross discipline defect detection all are very complex tasks. A lot of different software tools are used to control those processes and support integration between developers, designers and management. As an example of such tools we can provide Crytek Sandbox preview for live preview of achieved results, Avid AlienBrain to demonstrate 2d/3d assets, Epic Unreal Engine Blueprints for visual scripting and more [FlexibleProcess].

According to survey of games postmortems [Petrillo] another very big problem relates with project management. Many problems in game development can significantly raise the costs of development and create a big delay for the final product, and project management often just don't know what to do with it [Petrillo]. Mismanagement of the schedule, final product quality and available budget to reach the desired goals are the all well known related to project management issues. There is also a term in Game development called crunch time (period before the deadline, when all processes are overloading and turning into uncontrolled chaos with a lot of stress and teams demotivation).

Another really big issue is the unrealistic customers expectations. Customer nowadays are often oversaturated and waiting AAA quality product from any game on market and the games often are very far from such high standards [Kortmann].

So how we can define good or bad practices in game development, not to repeat the mistakes of others? One of the approaches is the Postmortem Analysis (PMA), a technique from the Empirical Software Engineering. PMA providing us the knowledges form failures and success of different projects, giving us opportunity to apply the good practices and locate the bad ones. PMA summarize the project development experience and it is a quite useful technique, but unfortunately it is an inside of the game studios and we don't have much of it in open access [Game-Agile]. But still a lot of them can be found on Gamedev (http://www.gamedev.net).

Authors of [Petrillo2] analyzed 20 video games PMA and calculated the appearance of top game development problems. Here is the table:



Problems occurrence in game development

We also need to mention the Feature creep when a new feature added to the project during development, destroying all the planning and schedule. Unmanaged feature creep could lead to lots of defects and errors and increase the risk of project failure.
Next is the crunch time. In game industry it is the period of work overload, often in the end of the project development or before any of the deadlines. Developers could work 12 hours a day and it leads to a lot of stress and errors. Unfortunately, in game industry it is the fact of life.

So as the summary we can say that not meeting the cost, schedule, unexpected technical complexity, poor project management, unrealistic customers expectations, unclear and poor requirements and game design documents are the top problems in video game development industry [Kortmann].

5 Related work

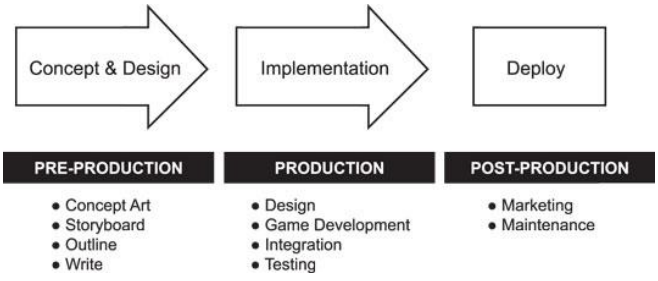
In this chapter we will describe all the related work which was found in literature and other papers and describes the Agile methodologies implementation to Game Development. There are different approaches with different pros and cons and we will do our review about every of it. The found approaches are the following:

- Game Scrum
- eXtreme Game Development
- Game unified process
- Triadic game design
- Agent based Agile game development

5.1 Game Scrum

Game Scrum is the combination of Scrum and XP, adapting all the best practices from both approaches and have its focus on people with little or no experience in Game Development.

As we already discussed before project management related issues are often the most harmful to success of the final product. The Game Scrum process add iterations development and Scrum meetings to the team daily work. Every iteration in Scrum, called the sprint, consist of some basic steps to develop the iteration value. But the game development is not always a straightforward process. To be applicable to Game Development reality Game Scrum process is separated in three main parts: Pre-production, Production and Post-production.



Game Scrum development phases

5.1.1 Game Scrum Pre-production phase

This is the first phase and it can be called the start of the game design process. On this stage the main factor of product success- the ‘fun factor’ must be discovered. The team also need to describe all the game related objects like the world, the characters, game process, the story line, all the game balance and mechanics. As stated in [Kanode] – “great pre-production reduces the need to find that elusive element of ‘fun’ during the production stage and allows the team to focus on implementing the game, rather than experimenting with it”.

Preproduction fully defines the Production phase which will realize the created game design and all the concept and models. Brainstorm and different informal discussions are used to create all the needed deliverables.

One of the most useful method is to create the prototype and check on it if the ‘fun factor’ will work as planned. It should contain the most basic features of the future game. Usually such prototype is created using the visual programming tools in some game engines like Unreal or Unity and not expect any of complex code creation. After the needs of the prototype are reached, it should be discarded for the future development and new product should be developed from scratch.

The most important deliverable of the Pre-production phase is the Game design document. In game development world it is the same as Requirements document in Software engineering or the Software Architecture document. Game design document defines all the future project scope. Bad document can lead to a feature creep and ruin the project management in case of the deadlines and milestones.

Nowadays no any standard exists for Game design document. It should just describe the future game in any manner the team planning to perform it. Due to the Agile approach it should not be immutable, any necessary changes during the future development could be added. But the management need to take in mind all the risks of such changes and avoid the future creep.

At the end of the Pre-production phase the Game design document is used to create the Product backlog, Scrum process set of the product features, including all the user stories and tasks for future development.

5.1.2 Game Scrum Production phase

At Production phase the Team should already have the fully described Game design document and have a clear vision what they need to create. Here the game design fully moves to Product backlog.

Every iteration the backlog is splitted to more smaller tasks and the team need to implement it the during the sprint,

that usually goes for one or two weeks. Almost always in Game Development the team consist not only from developers, but also with the different type of 3d, design or sound artists. It should be taken into account, because often the artist pipelines differ from the software development ways of doing it [Game-Scrum]. Many different artists could work on one task and it is hard to complete it during one iteration. For example, the 3d artist is preparing the 3d arts and models, only after it is done then the animation needs to be created. In [Keith] the authors propose the Kanban method for all related to artists tasks. Kanban approach opens the opportunities not to push the artist in the hard time boundaries, unlike Scrum requiring all the tasks be perform during the allocated sprint.

Keith also proposed to add some XP techniques like full time testing, so in fact Game Scrum includes all the Agile best practices, not only Scrum, and that makes it a good method applicable to Game Development and teams with the artist included.

5.1.3 Game Scrum Post-production phase

After the production and all the development is finished, the game testing takes its place. Game testers must fully test the product in case of defects allocation and 'fun factor' testing.

It is a good moment to create a game development postmortem, which we described in Problems in Game Development section. Such document could be very useful for the future projects, it should describe all the strength and weakness of the current project and summarize the development experience. It is a team artifact helping to improve all the processes and allocate the weak spots of current ones. It should contain historical data for better estimation and justify any changes in future process. [Milliaho] describes all the advantages of Postmortem analysis for the Game development. Also, we will mention here that a lot of Postmortem analysis could be found on Gamasutra's website (http://www.gamasutra.com/) and Gamedev (http://www.gamedev.net) websites.

Although the methods proposed by Game-Scrum do not deliver something brand new methods to the game development, its innovation is to combine several suggestions with the same goal. This enables us to create a systematic method to create a game, having not only practical references from those working in the area, but also the support of researchers confronting experiences with the knowledge available in literature [Game-Scrum].

5.2 eXtreme Game Development

eXtreme Game Development method [Demachy], also known as XGD, is an adaptation of Agile eXtreme Programming for game development. According to Demachy, its focus is on how to habituate XP to game design and engenderment of multimedia content and how to automatically test categorical elements of the game, like the "fun factor". Some adaptations from XP include to integrate multimedia content in perpetual integration; make tests for multimedia content; consider the team and use UML to describe the elements of game design and interaction and communication between game designers, programmers and even artists.
The methodology elucidates some adaptations from XP to XGD, but does not address, however, how to work with multidisciplinary teams. It only recommends the vision of the team holistically and suggests paired work for artists such as pair programming is utilized for developers in order to accomplish this. Withal, it does not discuss the results of applying the methodology in the projects mentioned.

eXtreme Game development is most suitable for small projects like mobile games, that not utilize a lot of artists and it can be very helpful in such scenarios.

5.3 Game unified process

Game unified process (GUP) is the combination of the eXtreme Programming and Rational Unified Process (RUP). In many ways it could be considered as a waterfall-based method, but XP applies the opportunities to make changes and minify the amount of documentation. It lightweight all the RUP heavy process, making it more Agile [Stewart]. So, it is a hybrid of waterfall and flexible XP techniques.

The XP implementation in Game unified process methodologies are logically like RUP, yet with some key contrasts. XP isn't documentation-driven in the customary sense. It is intended for short cycle improvement and activities that are built in moderately little advancement gatherings. XP is intuitive driven and recommends that through the procedure of pair programming and exceptional little gathering connection, the program and results itself become the documentation. A key differentiator for XP is its accentuation on testing right off the code completion in the improvement cycle. It suggests that experiments, test outfits, and test models ought to be made before the application code is created. This order powers the improvement group to consider the final product of the advancement before improvement starts.

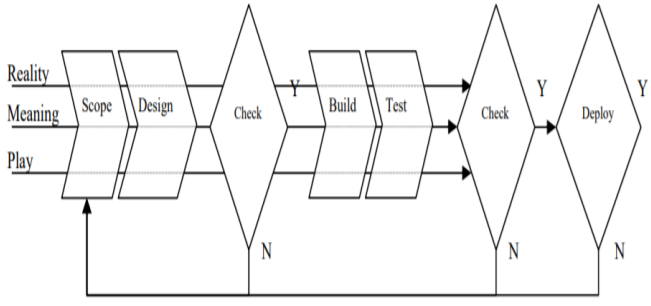
XP is a scrum type improvement cycle that encourages fast advancement by disposing of formal documentation stages; it substitutes, from the beginning of improvement, unit tests for documentation. The undertaking moves along the cycle and keeps on including unit tests as the item turns out to be progressively advanced. With each form, these unit tests are executed to ensure that the item is working as indicated by the comprehension of what the item ought to do. XP improvement is intended to be quick. A half year seems, by all accounts, to be the normal term of the advancement cycle. It likewise accepts an exceptionally close working relationship for the entirety of the gatherings engaged with the achievement of the item. On the off

chance that any gathering isn't all around spoke to all through the cycle the advancement could endure.

From a management viewpoint GUP is a good option. If GUP implemented correctly, product quality improves, high management have a clear vision on the process and results, there is less documentation, but it is still traceable for any project deliverable.

5.4 Triadic game design

Triadic game design (TGD) is another approach to Game development. It tries to formalize the game design with software development professional practices. Like all the others approaches it focused on the multidisciplinary nature of game development combining it with the Agile model [Kortmann]. It utilizes the triad of the following: Reality, Meaning and Play. Reality connects the game environment to real world. It describes the game scope in terms of game models and variables. Meaning defines some meaning of the game, should it teach something or reflect something from the player. Play is the game design, it defines the game mechanics, characters, actors, challenges and the fun factor. This part is often made by game designers and media specialists.



TGD model

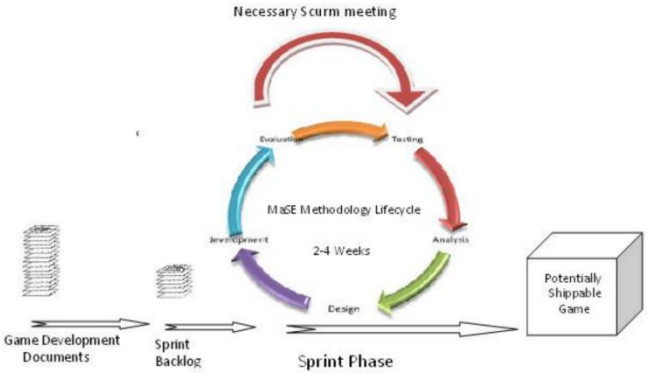
TGD force the team develop the game in along this triad. Game designers need to balance among all of them which is not a simple task. TGD is used for complex project where the final product requirements are not fully elicited at the project start. Once again eXtreme Software development model come into place. Designers, managers and customers can apply changes during the project implementation. The iterations are short, and they need to reach project goals as fast as possible.

TGD suggest a parallel and iterative development process, not setting designers in hard time limits. Designers always need to work at reality, meaning and play factors. Full product backlog or game scope is not required at the start of the project, making it very flexible. TGD uses the 'Customer on side' approach, building most of the game deliverables with collaboration with the game Stakeholders.

Many games Postmortem analysis telling the same things- not meeting the cost, schedule or the customers expectations. The TGD approach propose the process without strict initial requirements with customers onside and iterative approach again applicable for multiciliary teams.

5.5 Agent based Agile game development

Agent based Agile game development (AAGDM) have in its core Agent Oriented Software Development approach (AOSE). It utilizes the intelligent agent-based approach and predictive models combined with the Agile methodologies.



Agent based development model

Like in other approaches that we already mentioned the Game Design Document DGD is the start of the process. GDD is the main description of the future product. GDD could change many times during the process, obtaining new and new requirements. After GDD is created it is turned into a Product Backlog. According to Design document management can plan the tasks for current iterations and create the milestones for the project. On each iteration testers are carefully checking the new developed features for defects and the fun factor requirements. One the important part of the process to have a playable prototype on each iteration for the testing. At each 2-4 weeks sprint the Team need to develop and demonstrate the management or the customers some working part of the end solution. Using agent-based development (AAGDM) model is improving the quality and efficiency of developing large and complex game products [Rula].

Management have a big importance in game development industry. Poor or low experienced management have a very high risks to fail the project. AAGDM focus on deliver the product increments and value on each iteration and the game development schedule are more predictable and manageable. Customers and developers usually cooperate during the planning processes discussing the needed features for the final product. Like in XP AAGDM delivers less documentation than legacy software development approaches, in fact only the Game design document is needed.

AAGDM don't consider daily meeting as important part of the process not to waste time of the developers and artist. The meetings should be conducted only on most important project days like sprint planning or Game Design document changes.
Another AAGDM approach is mini subgrouping – divide the teams in little groups of one or two developers and for example 3d animator and texture artist. Those mini groups are assigned with their own tasks and are responsible to complete it.

As an Agile process AAGDM is not linear, but iterative. So, if some new interesting idea or a feature appears during the process, it should be discussed and in good scenario for it added to the GDD. The cost of changes in Agile approach are often much less like in waterfall like processes [Rula].

Like many others approaches we discussed in this section, AAGDM process needs to be proved by the time and industry experience and this work opens a lot of opportunities for future researches [Rula].

6 Teaching Agile

All the Agile methodologies and their different implementations are very popular nowadays. According to the latest industry surveys many companies adopt it to their development processes [Kropp]. Swiss Agile Study confirms, that Agile based methodologies bring much better outcomes than a plan based [SAS]. But the same surveys show that there is a very high level and abstract understanding of how to apply Agile methodologies correctly. There is a lack of Agile professional in team leads and software management. So not only Agile by itself, but its correct implementation will deliver the highest result possible.

How Agile could improve the following aspects:

Aspect	Much worse	Worse	Un-changed	Improved	Significantly improved	Don't know
Ability to manage changing priorities	1%	0%	9%	45%	44%	1%
Development process	0%	2%	17%	58%	22%	1%
Time to market	1%	2%	19%	53%	23%	2%
Alignment between IT and business	0%	1%	25%	46%	23%	6%
Project visibility	0%	2%	25%	39%	28%	6%
Team morale	0%	4%	25%	42%	24%	5%
Requirements management	0%	2%	29%	51%	13%	5%
Productivity	0%	2%	33%	47%	15%	4%
Risk management	0%	5%	32%	42%	17%	4%
Software quality	0%	2%	45%	35%	16%	2%
Software maintainability	0%	7%	55%	23%	12%	3%
Development cost	1%	12%	52%	22%	7%	6%
Engineering discipline	0%	4%	42%	42%	9%	4%

Agile influence on development aspects

SAS shows that there are very few software engineers with the required skills to apply Agile. In most of the universities all over the world there is only a quick and abstract introduction to Agile methodologies without any practice or workshops to really learn someone to implement it. The first adopters of Agile were very skilled and experienced software development professionals, but nowadays teams are often consisting of junior and middle level developers without deep understanding of Agile methods and philosophy and they are far less effective than teams truly implementing all the Agile have to offer. In [Kropp] there is a deep analysis of this situation and how to solve such issues by applying special Agile trainings and educational programs.

We found this topic very important, because even now when Agile reached its high popularity and proved itself as a new and very effective and reliable methodology for every software development including game development, it's full potential is not yet reached. Understanding the high-level Agile ideas is not enough for success and if one wants to fully get its benefits, there are a lot of skills and years of experience needed to implement it the right way [Rula].

7 Conclusion and Future Work

Video Game development is a unique challenge to the software development industry. And applying Agile principles can help to solve many of those challenges. But Agile itself should not be implemented blindly but should be leaded by a highly professional project managers to get the most of it. Game development companies should invest time and money to find the proven methods and find skilled managers to lower the risks of their projects.

Different Agile game development approaches described in this article could be applied to game Pre-production phase to create the Game design documents and then turn it into Product backlog. Qualified project managers need to apply requirement engineering practices to fully describe the future product. Agile methods allow the Game design document to be open to changes during the whole development and iteration testing could prevent the losing of the game 'fun factor', which is so important for success. Agile planning also helps to prevent the feature creep, the process of uncontrolled creation of new features and requirements, and ruining all of the project schedule and planning.

Large amount of different content needs to be created for modern game product and many multidisciplinary specialists are working together in one team during the game development process. Special Agile game development approaches that we discovered helps us to control the work pipelines of such teams.

As a summary of what we need for success we can emphasize the following: applying best software engineering practices like requirement engineering and

risk management to create the high quality Game design document for the project scope, create the deeper relations between development team and the end customers or investors, invest in project management training in scope of Agile methodologies, create the best possible process for multidisciplinary teams, monitor the current processes providing the development postmortem analysis, find and apply tools for fast and easy prototype development and iterations results demonstration.

All the proposed methodologies still don't solve all the possible problems. There may still be a feature creep, end user expectations might not be reached, technological issues could appear, conflicts between artists and developers could ruin the schedule and many more other problems. Future work on this problem includes the study of more related literature, contacts and interviews with most successful game development companies, more research on open game projects postmortem analysis and implementing the own mobile game using one of the described approaches.

REFERENCES

[Agile] Agile Software Development Martin, Robert C. Agile Software Development: Principles, Patterns, and Practices. Prentice Hall, 2003.

[Beck] Beck, K., Agile manifesto.
http://www.agilemanifesto.org. 2009

[Beck2] K. Beck. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, Reading, MA, 1st edition, 1999

[GameScrum] Andre Godoy, Ellen F. Barbosa - Game-Scrum: An Approach to Agile Game Development

[Bethke] E. Bethke. Game Development and Production. Wordware Publishing, Plano, 2003

[Game-Agile] Fabio Petrillo, Marcelo Pimenta - Is Agility out there? Agile Practices in Game Development

[Rucker] R. Rucker. Software Engineering and Computer Games. Addison Wesley, December 2002.

[FlexibleProcess] Juergen Musil , Angelika Schweda , Dietmar Winkler , and Stefan Bif - Improving Video Game Development: Facilitating Heterogeneous Team Collaboration Through Flexible Software Processes

[Petrillo] PETRILLO AND DIETRICH, C.
2008. Houston, we have a problem...: a survey of actual problems in computer games development

[Petrillo2] F. Petrillo, M. Pimenta, F. Trindade, and C. Dietrich. What went wrong? a survey of problems in game development. ACM Computer in Entertainment, CIE: 7(1), 2009

[Kortmann] Rens Kortmann, Casper Harteveld - Agile game development: lessons learned from software engineering

[Kanode] KANODE, C. M., AND HADDAD, H. M. 2009. Software engineering challenges in game development. In ITNG, IEEE Computer
Society, S. Latifi, Ed., 260–265

[Keith] KEITH, C., 2010. Kanban for video game development.

[Myllyaho] MYLLYAHO, M., SALO, O., KA¨ ARI ¨ AINEN ¨ , J., HYYSALO, J.,
AND KOSKELA, J., 2004. A review of small and large postmortem analysis methods.

[Demachy] DEMACHY, T., 2003. Extreme game development: Right on time,
every time.

[Stewart] Jason R. Stewart & Arvin Agah, Teaching a software engineering course on developing video games: a Unified Process versus Extreme Programming

[Rula] Rula Al-azawi, Aladdin Ayesh - Towards Agent-based Agile Approach for Game Development Methodology

[Kropp] Martin Kropp, Andreas Meier - Teaching Agile Software Development at University Level: Values, Management, and Craftsmanship

[SAS] Martin Kropp, Andreas Meier, Swiss Agile Study