Masarykova univerzita
Fakulta informatiky


# Effective Leadership in Game Industry using Agile Methods

Diplomová práce

**Ján Meravý**

Brno, jar - podzim 2013

## Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Ján Meravý

**Advisor:** doc. RNDr. Tomáš Pitner, Ph.D.

## Keywords

Project management, team management, game development, agile methods, team organization, motivation, effectivity, performance, leadership

## Acknowledgement

I would like to express my gratitude to my parents, my girlfriend, roommates, colleagues and friends for their help and support. I am also very oblidged to my advisor for his time, knowledge, patiece, usefull guides and valuable information. My thanks goes also to every Developer, Tester, Graphic Designer, Director and basicaly anyone from lower support to top management, who had time for my questions, interwiews and sharing expert opinions. Special thanks goes to the whole team of game developing studio CUKETA for their cooperation and guides during my Interim Project.

## Abstract

The aim of this thesis is to give a wider view of management, development and the production of game software. The theoretical part describes Agile development methodologies, their suitability, advantages and pitfalls. In further in my thesis I focus on team support, leadership, communication and overall effectiveness. The last part of this thesis describes the "best practices" and practical experience view in real game development.

# Contents

# 1 Introduction

In this thesis I will explain the agile methodologies in relation to game development, leadership and practical situations. Thesis consists of both theoretical knowledge and practical experience, gathered through meeting sessions and interviews from czech gaming companies and their activities.

This chapter provides brief introduction to game development.

The second chapter shows benefits and difficulties of transition to agile development methods. I focus on smaller development teams and present procedures for streamlining development in an effort to ensure the smoothest transition of the development team from traditional practices to the agile.

Third chapter describes the effective communication habits, improvement of cooperation and team leadership. This includes active listening and communication principles of handling problem solving. I explain how to prevent and resolve conflicts within management and optimize source planning. I show methods how to deal with different people and keep team motivated to improve collaboration on a professional level.

The last chapter consists of best practices, tips and trick during the whole game project lifecycle gained from real experience, sessions and interviews. This chapter advises how to manage game development and overcome problems of decision making in both management and development. Goal is to give wider introduction and informations for game Developers based on real czech and international projects. They should serve as guide, that worked for other game creators and are widely used in many game companies.

## 1.1 Game development

Game development teams (often called as indie development teams)[1] are formed from individual game Developers and their passion for making games. Game titles from the past continue to have strong influence for making new game titles. Teams behind games are considered "crazy" but with potentional that can be seen by Investors. Friendship core and common sense is a fundamental essence, that holds team members together. One person works on multiple things, that project requires. Cooperation with management does not cause much time consuming problems. Self-organizing structure helps indie-development team members to work together with efficiency based on team teoretical knowledge and shared experience. Team is aware of their competitors, technology restraints and targeted audience of players[2]. Wide platform opportunity and the market size gives them easier chance to develop first game product (or prototype) that is used as their flagship and reference. Visual sells. This first moment is crucial to re-

1. Indie development team - Usually small development team up to 5 people working on their own small project(s) with minimal sources.
2. Player - Customer who buys game or it's content. Can be referred as independent Tester(s) or end user(s) means the game playing person (in basis) who is capable of feedback

plenish investment of time and funds for the team. Main problems in the present game industry are strict dates of game product shipment which is often delayed (by multiple reasons from ensuring content sustainability to development maintainability).

## 1.2 Game studios

Decade ago the games were made by enthusiasist who enjoyed making things as best as possible at the costs of personal time. This common sense was key to sucsess. Team expanding was fast and risky, but when the gaming product was successful the company growed. Classical software development practices were not suitable for this kind of game development. Larger teams required extensive human sources and management. For more effective resolution, these teams were divided into departments.
This approach created thousands of top triple-A games[3] (and their content) over the last decade (and many comebacks). Less known is the fact that sources invested into these successful games were tremendous (Valve, Activision, Blizzard, Rovio) (however worthy in the end - because new concept and tittles were not available on franchise [4] market).

Another good example is the game called "Portal" from VALVE. This game was made by nine people with base on Half Life project engine which company developed earlier. This game become very popular by new creative concept of logical spatial thinking that ended up as main product and game of the year 2011 called "Portal 2"[1]. Valve invented new unique strategies within agile development to rehire and maintaining team members. Team members are free of choice to do work they want on the product that suits them best (when the position is available - which mostly is). Another good concept (proposed by Activision) was letting team members choice to do for any project in any company wing for one whole day of a week (overally benefitable over exploiting - the rest week is used as the controlling mechanism).

There is great base of indie Developers, that are focusing on mobile games for tablets, phablets, smartphones and for platforms (and market stores) namely iOS, Android, Blackberry or Windows Phone. These are overflowing with titles, so the quality of gameplay is what maters. (For example rovio and their product "Angry Birds" - the core franchise of this Finnish company now earns most of the income from things around their popular product[2].)

3. Triple-A games (or AAA) - Games developed for major platforms with enormous marketing budgets.
4. Franchise - System and marketing in which one party gives the second side right for direct or indirect use the "game package" as industrial and intellectual property.

# 2 Agile methods for game development

## 2.1 Introducion

This chapter presents brief introduction of most frequently used agile methods that are used in game industry. I will demonstrate their utilization and how to manage them effectively in order of deterministic approach, that ensures the product Quality Assurance (QA)[1]. I will explain and give examples for commonly used agile methods in real situations in relation that affects the team, motivation, stability and maintaining performance, with including best practices and potential pitfalls. In the end of each method I will talk about real experience and recommended solutions for critical situations I gained through objective observing and professional interviews.

## 2.2 Agile development

In further view I will focus on the game development and process of delivering game product through agile approach instead of classical development. Agile methodology cannot be considered as a manual guide that commands how to handle specific life coaching or development situations. In Project Management, the three main components are Time, Features and Quality (Figure 2.1)[3].

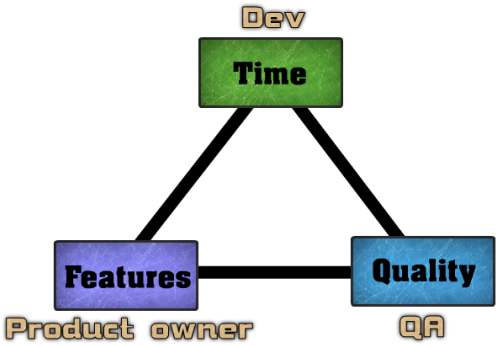



Figure 2.1: Relation of three influencing elements

In order to achieve best results the tact, soft skills and person centered approach is recommended. This thesis should be read actively with Proactive Approach[2] and can be used as an ispiration for self development that helps the way to treat with the team and the customers. Methodology can only recommends what is right, but it's up to

1. Quality Assurance - Activities that test whether the requirements of product are fulfilled on a certain certified level.
2. Proactive Approach - Thinking deeper about issues out of personal subjectiveness, in order to find simplier and more suited solutions.

person to decide each specific situation. Real experience from life and practical situations helps build character and leadership that enhances the personal effectivity. One can also learn from experience of others and this thesis is based on my own personal experience (without embellishments - even fail experience can be valuable), or from top game company managers. I know some of them personally and I believe in their experience, skills and senses for results that can be measured on real released game products. (Blizzard, Valve, Geewa, 4K games, Adobe Flash, Tinysoft, Cuketa, Dark Sheep, Fun2Robots, Keen Software House, Pixel Federation, Mingle Games and many others.)

Agile teams are formed (usually) of generalizing specialists (people who have more technical specialities). Agile team roles are stable but in different sprints[3] there can be changes within team structures (someone can be pulled off to help work on something crucial that is determined/required for project success). In common words the point is that employees have sufficient skills to operate within the team to get job done, when each one can help at position that suits him best. It is not only possible for people to become more agile it's very desirable and necessary.

Agile methods are widely used in gaming industry for their dynamic approach and adaptation for customer's needs[4]. They differ from straight-forward development and aims at completion of product by using specific open-minded and adaptive techniques, rules or habits[5]. This thesis explains positives and negatives of each method with guides that are used in gaming development to make development smoothier and reduce risks. Such problems occur when unexpected requirements appear during the making but their benefits are significant. It is up to decision of whole team if they are aware of this fact and able to estimate if such work is within their capacity to implement, test and add into gameplay. This is the main reason of delays[6]. Gameplay as the result of team effort cannot be fooled (and so do gamers).

### 2.2.1 Team experience

It's a good thing to finish the successful project, but the most important is to look back for failures and learn a lesson from them. Project analyst will do this, but so should Developers, their manners and habits caused them (in case that we exclude management and choices out of the box)[7]. When we got budget for some project it is "easy" to make plans, count with all known possible threats, calculate them, direct development and lead team by the plan. Unexpected issues or special situations might affect the plan in all project phases. With every new project there comes risk, which is lowered by previous team experience. It is a good way to think ahead and plan that something crucial will go wrong for sure. Rather than spend all budget on project that might shine or fail suggested is to revalidate satisfiability of the plan after every milestone[4] (such as alpha version prototype, multiplayer release, etc.). Work out the game features that players

3. Sprint - Limited time lasting event for the whole team that is used to complete selection of tasks. (Will be mentioned later in this chapter at 2.2.4)
4. Milestone - Specialy chosen goal that indicates the completion of some important work, progress state or project phase.

actually like. The game is made for them so their first look on something half finished should be one of team goals to achieve. This way can save time, sources and prevent meaningless development solutions that will be used very rarely (use saved time on more important things). Keep in mind that end user (as the game's future player) want more "meat" than "sauce". Entertained player and his interests are the value, that we want to offer for mass audience. Good team strategy is to build fertile ground for players and stay connected to gaming community.

### 2.2.2 Roles

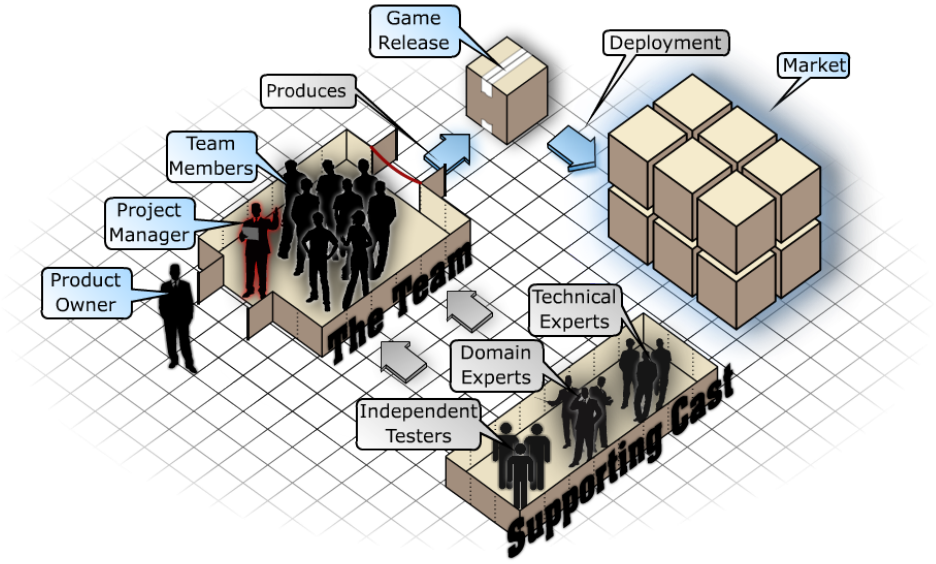



Figure 2.2: Small agile team structure

**Team leader** Act as a Project Leader, that is responsible for obtaining sources for the rest of the team as he is some sort of a their project facilitator (team coach). He helps and supports the team. Sometimes he even defends team from management and distinquish distraction from what belongs to the problem and what is out of the scope[4]. (In scrum methodology this person is called "Scrum Master".)

**Team member** Referred as Developer or programmer. He is responsible for the creation and delivery of product.

**Product owner** Represents the Stakeholders. He owns the product and is accountable for ensuring that team delivers value to the business.

**Optional supporting cast**

**Stakeholder** Practically anyone who supported the project or team members. (Individuals, End users, Vendors, etc.)

**Technical experts** Usually an expert in a particular field of knowledge, hired to provide help and overcome some specific difficult problem,find or make solution that just need to be done on a professional level.

**Domain experts** Sometimes they are required to analyze, guide or explain the details of sponsoring executives or compatibility of project vision. (The Product Owner represents wide range of Stakeholders (not just end users), and it is not possible to be expert at every single kind of domain.)

**Testers and feedback support** Often independent users that works in parallel through the game development lifecycle. They validate gameplay, report bugs or generate feedback report. They act as hidden team support.

### 2.2.3 Size of agile team and scalability

First of all, what is considered as a small agile team? It is a team up to 9 members by theory[8]. Yet in real job it can be few members more, even if they are not primary part of the team, or their job is just some part of task that can be done in parallel. From middle to big agile team is considered to be up to 20 members (or more)[9].

**Small agile team** - Works closely with a Product Owner to build a high quality product on semi-stable incremental basis. In this thesis I focus mostly on these smaller teams (sometimes referred as indie teams).

**Large agile team** - Often known as a team of teams approach. Project strategy involves organizing larger team into a collection of smaller teams by project needs and management architecture. These subteams are responsible for delivering working parts (subsystems) without delays. This strategy is referred as Conway's Law[5]. Larger agile team of teams can contain some special roles of Architecture Owner, System Integrator, Project Management Activities Advisor, Technical Issues Master and(or) someone to deal with requirements, product ownership & law issues. There are several critical issues with work cooperation, user feedback control and project subpart integration. The project is split into wings (departments) for specific tasks (Testing, Gameplay, Core, Management, Specialists, Product Owner, etc.)[10, 11].

5. Conway's Law (1960) - Effective team structure of organizing larger team into subteams around the system architecture. (One of several lean development governance strategies.)

The Agile Process Maturity Model (APMM)[6] is worth to be mentioned for it's important scaling factor and flexibility in middle sized game projects[12].

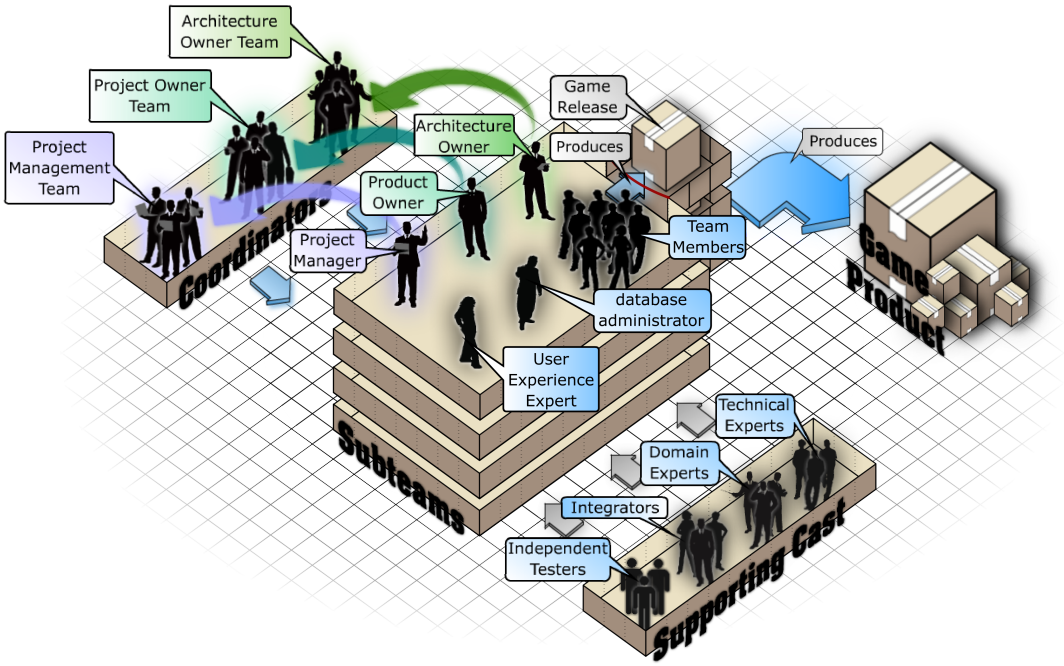



Figure 2.3: Large agile team structure

### 2.2.4 Sprints

Are directed by the coach who suggests tasks and tracks their progress to make sure all team members get tasks by their available time and competencies. Selection of what tasks will be done is marked in the sprint backlog that details the time cost that is determined by the whole team. Team will identify and discuss how much work is likely to be done during the current sprint. (Prioritizing the product backlog and estimation practices.) Suggested is to keep iterations and sprints consistent in duration for better planning and delivering.

### 2.2.5 Artefacts

**Project backlogs** Sprint backlog and product backlog are recommended. They serve similarly but with different purpose. The product backlog is viewed by Product Owner

6. Agile Process Maturity Model (APMM) - Contextual framework (by IBM) that can helps to effectively adopt agile practicles.

as a high point of view of all the work that must be done for the game product. Sprint backlog contains detailed list of all tasks and user stories for the sprint. (In product backlog team estimates user stories with relative unit called story points, but in the sprint backlog team estimates task in hours.)

On more general level - backlog defines the number of tasks completly done (based on their priority) compared to incomplete planned tasks. This two numbers are tracked in time (for example each week or sprint) in a graph. The lower fluctuation graph contains the better is both estimation and measurement.

Tasks can be measured by estimating techniques. From Analogy-based estimation, WBS-based (botton up), Parametric models, Size-based models, Group estimation, Constructive Cost Model (COCOMO), Mechanical or Judgmental combination[13]. The most suited and effective way for agile planning and gaming development itself is Planning poker and Three-Point Estimation (known as PERT). I can recommend these two for their observable, conspicious, transparent, dignified, simple, fast and common use for their uniformity in agile approach[4].

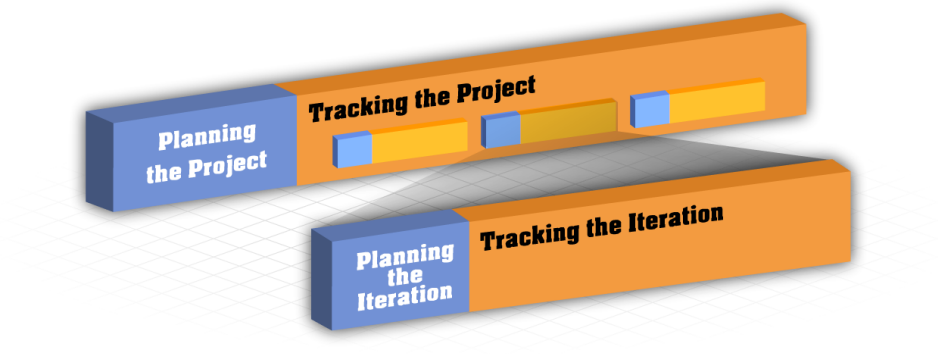



Figure 2.4: Planning and Tracking Project and Iterations

Main purpose of these estimation techniques is to exclude psychological aspect called Social Copyring[7] and discuss the real value of each task from different views. The Planning Poker method can extent meeting time so it should be used with caution and on most crucial or critical tasks that may require variable sources (or team has no previous experience about them).

**Burn down chart** Represents graph of work left in time. Is a prediction of all the work that will be completed. Burn down charts are commonly used in all agile methodologies for projects that contains measurable progress over time. Chart can also be used for measuring performance even though not as most reliable source. (Burn up chart is inverted burn down chart.)

7. Social Copyring - Psychological aspect of adapting to behaviour of others.

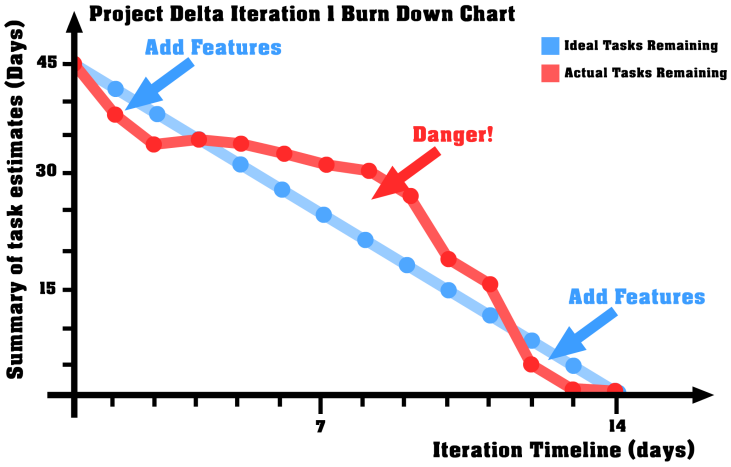



Figure 2.5: Burn down chart

## 2.3 Agile Methods

In this chapter I will introduce the core information (and possible restrictions) for use of each agile methodology[4] in relation to suitability for developing game products. Each subchapter will be enclosed by summary.

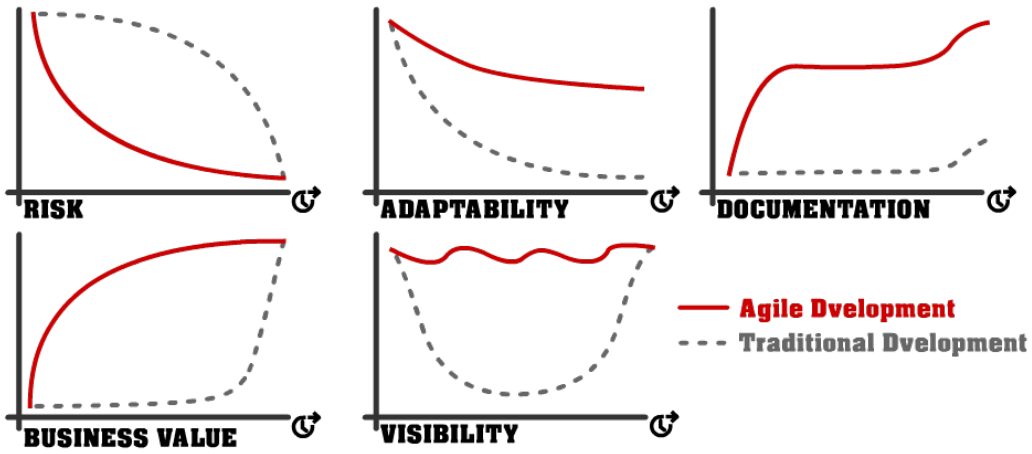



Figure 2.6: Value propositions comparison in relation to traditional development

### 2.3.1 Scrum

Sometimes referred as agile developing framework which is based on iterative and incremental basis. Work process is divided into sprints that represents cycles. These sprints usualy lasts from two to four weeks. During each sprint the selection of tasks is made by their priority, contribution and complexity (or performance) with emphasis to end users. At the end of each sprint there should be working version of game product. The process begins with the Product Owner. His goal is to find game product that will have good market value for customer demands (and is within team limits). The result is the product backlog that is as complete as possible or compartmentalizable into parts. When this is done the prepration of sources and sprints themselves begin. When everything is implemented from product backlog the game is prepared to air on market. Scrum is smoothly handling this by using simple methods. Three kind of artefacts are used in process: product backlog, sprint backlog and burn down chart.

#### Scrum roles

**Product owner** Is liable for business value of the project.

**Scrum Master** Referred as Project Leader who makes sure the team is developing productively and effectively.

**Team member** Develops the game product itself.

There are also some roles that are not listed in scrum process, but they are necessary part of development. I am speaking of Testers, End Users, Investors, Support or Specialists.

Together as a team they are responsible for delivering the product to the market.

#### Meetings

**Regular sprint planning** Is used when there are enough tasks in product backlog. First of all the Scrum Master must decide how long the sprint will be. How long will the sprint lasts and what can be really made in this time. Then team corrects, revises and estimates the time required for each task (for example by aforementioned "planning poker" technique).

**Daily meeting** The sprint progress is notified for transparent visibility of current state of work. Important and crucial things should be discussed which should not take a long time. Recommended meeting time is 15 minutes after beginning of the work. (Often referred as stand up meeting.)

**Sprint review meeting** Regular revision at the end of each sprint to forego future failures. This time is dedicated to analyze what went smoothly, what was wrong, and how to prevent fails for the future sprints. Scrum master is recommended as discussion moderator.

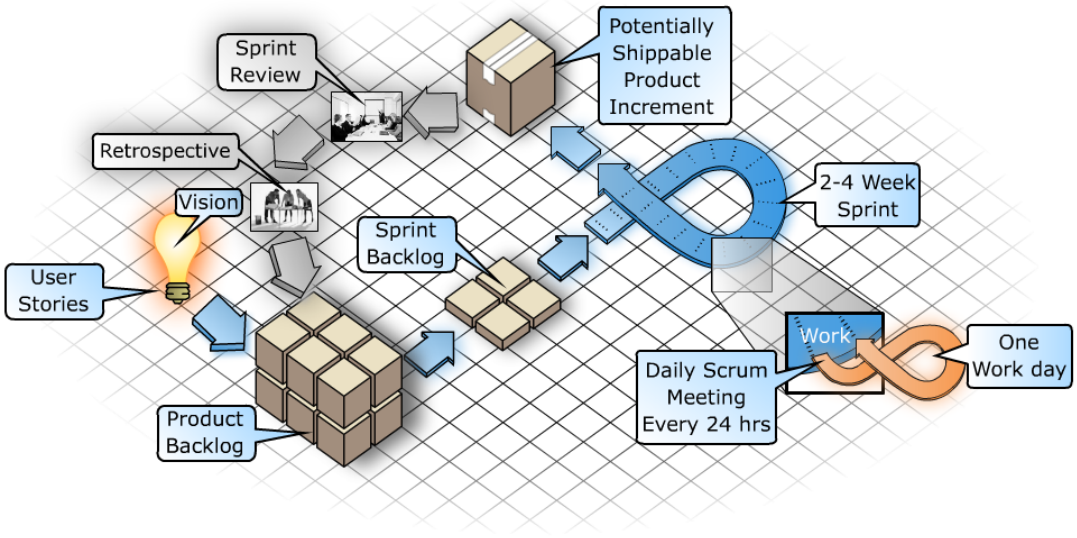



Figure 2.7: Scrum visual lifecycle

**Summary** Scrum focuses more on managing project control and revision than implementation or effectivity principles. Adequacy of followig solutions is unstable and depends heavily on Scrum Master, for he is the most responsible person and critical aspect of this methodology. There is not clearly specified what to do in each situations and it's natively accepted that the team members have at least some experience with agile development, time management, transparent information highway, cooperation and team strategy that also involves the inner culture of team members and their motivation. For very high adaptablity the SCRUM can be used in various situations for it's simple and clear rules that offer stable results in a long period. I recommend this methodology to all sizes of game agile teams at condition, that there are experienced management leaders and skilled Scrum Master with infrastructure support for both indie and senior Developers (good balance is the essence). In some cases SCRUM can be time consuming as the price of excellent documentation artifacts and design quality (with the code inspections). Should the team hesitate over task, the customer feedback is the essence for defining which tasks are the most important.

### 2.3.2 Feature-Driven Development (FDD)

Emerges from iterative and incremental basis as combination set of techniques and the way they are connected. The feature approach has real value for end customer. The project starts by making the feature list. Each feature is then carefully planned and designed how it will be build. Simplicity is an important factor but not always achievable in real situations (not everything is simple including critical solutions). The measure that comes into process is importance, significance, meaning, weight, value or interest by moment and actual need of end user. These are designed upon understand what needs to be done to warrant feature realization. The features are divided into parts and realization is given to suitable team member (or a subteam) by personal skills and task type (saving precious time). Planning is derivated by full list of features and assignation starts from hardest estimated tasks. Then the first project version is made upon these model semi-parts, solutions and discussed feedback. This proces is iterated through whole set of features until game product is good enough to fullfill plan requirements and team expectations. Planning is critical and should be very durable, because rate of change is significant element[13] (imperfect estimation is the cost of feature adding strategy). The planning strictly based on features can be tricky and costs a lot of unplanned extra time.

**Process** The end user is in the center of team interest strategy. Primary objective is to look upon the project through the end customer eyes. Each task or game part is considered as a feature. Main goal of using FDD is to repetitively deliver working version of game product. This is rated by end user and work is submissioned to this reason. FDD helps to remove the uncertainty from developing process and powers management of critical applications (databases, game core engine, game rules implementation, etc.). Iterations are usually faster. Implemented features should be split into smaller tasks that are simple enough to be completed within given time. Customer representative is part of the development team so he responds flexibly to change and results of his feedback are easier guaranteed [14].

**Six main roles:** Project Manager, Chief Architect, Development Manager, Chief Programmer, Class Owner, Domain Expert

**Support roles:** Release Manager, Deployer, Build Engineer, Tester, Administrator, Toolsmith

**Summary** FDD combines best from both classical techniques and agile methodology (it falls into agile more). It is a good first way of migration to agile development (if the team is not more than 12-15 people). FDD should be used with caution and in relation of how is the team depending on end user feedback (and team abilities to process them). Well managed collaboration is hidden aspect of determining success. Gaining adequate information from gaming comunity is optional but is worth mentioning to include wide set of feedback options.

### 2.3.3 Test-Driven Development (TDD)

Test-driven development helps professional software Developers deliver clean, durable code that works, without delays. It is based on making sustainable tests that rely on the repetition of a very short development cycle. Automated tests are used to detect any unexpected changes in project behaviour. Effective layout of a test case ensures that all crucial actions are finished, visibility of the test case, and smooths the flow of execution (often relies on some principles of extreme programming). TDD can be understood as "tool" that supports to create better written pieces of code and consistent making of a structured documentation test case. Productivity of programmers tend to make more relating code quality. The use of debugger is suppressed at the cost of more code but the direct correlation leads to flexible problem solutions (because of smaller size of project parts). Gameplay is the potentional problem. Automated tests are checking code integrity and solution reliability. They are unsuitable of achieving the level of human perception and sense. Overtesting can become time consuming problem and the level of coverage plus testing detail gained in repeated TDD cycles**[15]** can not always be recreated later (they would require unadequately more sources). The benefits of the approach are significant, and the costs are low by comparison. Gaming is based on mathematics and algorhitms that can be tested with ease (non gameplay related systems are easy to test) But a lot of things are visual in nature, and aren't obvious how to have them properly tested. It is challenge to balance everything within testing limits. For example making of HUD[8] and GUI[9] depends on the feel and effective control of the scene. Some solutions are not definite until team deploys them. They might influence game mechanics, even though game itself pass all tests. Such changes are very common in game development. This continual change is the main reason why it is hard to use pure TDD principle in game making.

**Summary** Previous experience of Single Responsibility Principle (SRP)[10], Dependency Inversion Principle (DIP)[11], and the Open Closed Principle (OCP)[12] are the backbone for serious consider of this development type (with management support during the whole process). Previous experience in other agile methodology is required. Project plan should also count with TDD and have mechanism for sustainable approach (including early and release development phases). Under these conditions I can recommend TDD methodology for middle to long sized agile teams.

---

8. HUD - Head-Up Display used in video gaming to display pieces of information about the player, game objects or goals on the screen such as health bars, score, items or progress.
9. GUI - (Graphical User Interface) Allows to manipulate with the game product on both visual and intellectual level through set of graphical controllers, display objects or in game buttons.
10. Single Responsibility Principle - State that every class should have single responsibility based on principle of cohesion which can be the reason to change.
11. Dependency Inversion Principle - Software parts or modules on high levels should not depend on lower levels and they are both depending on abstractions and details above them.
12. Open Closed Principle - Principle of how software entities (classes, functions, modules, etc.) should be open for extension, but closed for modification.

### 2.3.4 Lean Software Development

Is based upon distinguish everything that does not give added value for the customer in scope. This technique is used in some part of the rest agile methodologies. It represents set of principles for quality, speed and balance. Lean thinking is looking for the solution by approaching to optimalization, reducing waste and improving overall flow of value through a system[11]. Top processes produce parts only if lower processes require them, which helps in minimizing the "work in progress" (they have to pull for them directly). Creating knowledge and eliminating waste helps to build project parts and faster deliver whole product (whcih is optimized as whole).

These principles should not be used as a goal but as decision making guide focusing on improvement of a system on overall basis as much as possible. Opposite from TDD that stands on integrity by inspecting the last phase of creation the lean principle supports establishing integrity during the creation process. It is easier when teams know they have the supporting mechanisms, including dependable version management. Is partialy present in game development from some form to another by type and size of project.

**Summary** In Agile software development, lean has become a common practice to visualize and share project status by posting cards on a wall of the project room or use specific tool for this purpose. Sustaining lean rules, separate and serial processes by type, assigning requirements and task lifecycle is time consuming but the contribution is equally rewarding with better task control and management. For more complex game projects there is a very good and benefitial use for lean principles when the team have will to use them.

### 2.3.5 Dynamic Systems Development Method (DSDM)

Invented to fix some of the weakneses from the Rapid Application Development (RAD) [13] method by providing a framework which takes into account the entire development cycle. DSDM was one of the first agile methods (and widely known method) that defines most of principles used in agile development. DSDM basis are user involvement with iterative and incremental development. The delivery frequency is increased and supported by integrated tests at each phase. The acceptance of delivered products depends directly on fulfilling requirements[16]. It can be used in various situations for managing project subparts, namely exploration, engineering or solution incremental deployment.

**Summary** DSDM is not rare in game development, yet rather hidden. Some of the interviewed czech indie-teams bought third party solutions (engine modules, gameplay

13. Rapid Application Development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping.

tests or tools for testing quality) and poolished them by themselves (For example the cooperation of "Adobe" and game company "CUKETA" poolished new rendering technology "stage3D" in flash by using computer graphics). Generic approach of solution delivery by fixed time and cost with support can be still considerable agile method for especially new teams that works on their own engines.

### 2.3.6 Extreme Programming (XP)

Use of beneficial aspects from traditional software engineering practices that are taken to extreme levels. The early delivery and following improvements of project is based on essentials of communication, simplicity, courage (challenge) and respect. Mainly focuses on engineering processes, explaining the analysis, measure development and test phases with approaches that make a substantial difference to the quality of the end product. XP empowers a confident respond to changing gameplay requirements in all phases of software lifecycle (including the late release). Managers, customers and game Developers are equal partners in a collaborative team. This environment of listening to the real values (goals), revision of design, and testing is best known for it's Pair Programming[14] practices. Self-organizing around the problems supports implementation of simple, yet effective solutions. XP can be higly contributive methodology, when the team has previous experience and is well known of what Product Owner expects. XP is very well suited for small teams and smaller game projects because of it's high success ratio**[17]**. The planning is very dependent on feedback quality, negotiation with Stakeholders and meetings with Product Owner. Preparation is easy and results are simple to measure. Quality assurance is easier to obtain by XP.

**Summary** With experienced practising of teamwork the XP principles are much more effective. This methodology can be widely used for all small agile teams and departments focusing on specific problem solvings. Management has to know how to manage human sources and sustain support to get all main benefits of following XP principles.

14. Pair Programming - Activities involving pair change and labor alteration involving control between jobs on specific tasks.

# 3 Team leadership

## 3.1 Introduction

In this chapter I show most commonly mentioned issues and principles that affect performance and quality indirectly (from view of management). I explain what I have found contained in multiple referenced theory and periodics from multiple views. I explain the solutions in simple words for easier and deeper understanding of team leadership based on readings and conclusions of project management discipline.

### 3.1.1 Quality Assurance

Games are played in realtime with some level of complexity. This makes them hard to compute for all possible options. Tremendous number of combinations brings problems for testing. Some can be made by computer but the Tester (which is in this case player) is required to check them. Not only the development is driven by best features of whole team, but also the final customers needs to be taken into the process. It's because of submarine syndrome[1]. (When team tests the game by playing it so many times they used to disregard some game glitches unaware of knowing how important they can be.) Attention to detail matters. QA activities guarantee that requirements of a game product will be fulfilled. This requires systematic measurent, comparison with standards and monitoring of processes associated with feedback that confers error prevention[18].

### 3.1.2 Qualification & performance

Put your own spin and flair on your job, personalize it. (We will not stick to a diet if we don't like the food.) If something is not working consider modifying it or stop doing it.

### 3.1.3 Cooperation effectiveness

Good example are Testers and Developers. Developers can not do the Testers job on their own work, because they feel responsibility for solutions they had made. It's hard for them to find the bugs in their work because they are thinking in some straight forwarding way to solve tasks and did not made bugs at first place (on purpose). Also reporting mistakes in their own work would undermine themselves. On the other side the Testers are in some way happy to crush (destroy) the game software. Their effort invoves finding as many bugs as possible in each problem resolution. Even though both Developers and Testers are facing in different direction their cooperation is necessary in close relation with solicit to build fine tuned game product[19]. To do so they have to find common language. Objective view is the key to progressive support and the understanding that someone elses work is required to make product better (even thought

1. Submarine syndrome - Attention to detail is absent after many repetitions (of testing or playing the game in this case).

it is agains your work) leads to higher responsibility. Establishing of cooperation takes time and must be planted on good ground with care and support of Project Manager.

### 3.1.4 Meetings management

**Meetings preparation** Personal requirements and topic with containing proposing time, place and time lasting informations. Presence on time and being relaxed for the topic. Have documents ready and prepare required presentation technical base. Make regular breaks and declare how long they will last.

**Effective discussion** Moderator is recommended and he should have skills to know, how will the selected solutions influence things out of the discussed topic. He distributes documents, introduces problem, works with discussion and also summarizes finally selected solution or future sprint steps for everyone present. He should also dampen negative elements and direct the discusion back to the original issue and provide uniformal guide. Team members should focus deeper and with wider range about the problem. This requires presense of different team member roles. Administration of relaxed atmosphere can involve spontaneous conversation stimulations as jokes or crazy solution ideas. Distraction should be managably restrained to point, when present members reclaim new energy to continue in discussed problem. Moderator should also calm down insultations and gauge to discussion transparency. His job is to help team members think spontaneously about the problem from different perspectives in order to achieve more efficient solutions. This can be trained so well that audience comes to conclusions naturally.

**Personality collisions** When game Investor (for example) and company representative firstly meet to agree on contract there is a need to find "common language and understanding". (These principles are also applicable to “customer” that orders the product in the classical development.) Good success rate enhancement trick is to come to Product Owner with set of negotiators of different personalities. After deciding who suits for him (in communication) best will take the role of main talk (yet quite expensive solution).

**Objectively directed person** When the Product Owner is arogant, then objective proofs and facts with cooperation talk are better, than situation when someone entertains too much about game product benefits and advantages so the Product Owner may think team is trying to mess (cheat) with him or offer things he simply doesn’t want[3].

This kind of person is usually assertive, directive, known of what he wants, prepared to check every contract issue and project progress or question anything regullary (and plan or ask things ahead).

**Emotionaly directed person** For emotion like submissive Product Owner (or Investor) there is better to create bond, show compassion and shared value over uniformity, dominance (or that you must be always right).

This kind of person is persuadable, opened, honest, intuitive, little uncertain or shy and trustable, but also most oftenly fully unaware of what he truly wants or gets.

**Balanced direction** Balanced representative is most compatible in meetings. (This comes naturally). Probably the worst combination is when nobody cares on both sides (for example two flegmatics). Serious project discussion goes somehow randomly, but then acts as seriously assigned. Without Project Owner interest, guide and control of the crucial problems comes usually too late in process for the team apply all changes in a given time.

**Meeting closure** At the end moderator should read final meeting summary that the team members agreed on. Everyone should be aware of current situation (even if they get lost at some decisions) and open or close future sprint/strategy plan.

**Last meetings** Revision meetings are good to revise the results, even though in agile development the overall team failure can be known even before final release. (Aside from classical development where the Project Manager is happily shaking customer hand with acceptance protocol on the table while holding the bill in other hand behind his back.)

### 3.1.5 Effective team leadership

Studying leadership at university can be hard. The first thing is to learn how to learn, how to self motivate to get things done. Practice combined with teoretical skills is most self-rewarding. Managing work within the team is challenge and requires both soft skills and knowledge. Manager is the one who takes the whole responsibility. The most important of all is the team support. (It's the core of future business in order to breathe). It's his job to think out of the box and plan ahead while constantly analyze risks. Here comes the opportunity to build level of management that will boost the developing process. Effective leading improve weak points and enhances opportunity. Distribution of responsibilities among employees is necessary to make sure there will be always someone who will be responsible. (In reasonable level of complexity and with back up plan that some team member may be released from the project or inaccessible.)

**Personal effectivity and objective approach** Be fair. There's always some group within the organization that appears to be the pure gold. It might happend that somebody seems to get more attention and popularity than others. Top management can't see into all team interactions but every employee feels and knows decisions made from above. Project Manager must be aware of this.

It s a tough situation to be in, but you can definitely take the focus on creating some solid recognition, rewards program or responding personally to suggestions, and performing some other actions that will include involved employees. Just remember what you do for one person or group you should not forget to do (from time to time) something similar for other team members[20]. Project achievements are made by sustainable effort of whole team, no matter how small part was the employee involved in. It's whole team success. From psychology of personality it is recommended to point out effort and personal success in front of whole team while taking personal critisism personally in private talk[21].

You need to understand sometines people choose not to lean. Some people can not (or choose not to) play with the rest of the team. They are most of the times the core of your toxic environment. Even the best coaching and support system makes no difference in their performance and attitude. You got to have a system in place to be able to get rid these people from your organization without ruining social hierarchy.

**Effective problem planning management** Project Manager will face tough decisions every day and constantly looks for possible solutions (that's his job). He will improve processes on the go (by their engagement into the team). Project Leader creates a plan that will be followed and this is never easy. (Because of tremendous possible scenarios in game industry.) The plan should be stable, understandable and divided into milestones that will make sense. In gaming development the plan will probably change multiple times and therefore new misunderstanding and exceptions will follow. Project Manager will constantly check the progress and regulate the product shape. Team cooperation generally gets work done acording to the plan (when their temper and inspiration is well managed), but the product will tremble as the gameplay and design give it it's shape. This process requires effort of the whole team under support and help of Project Manager. The best effective solution is to make space in plan for these dynamic changes that will influence the whole product and even count with delayed delivery and additional testing time in contracts.

## 3.2 Negative behavior of team

In this subchapter I will explain some of the negative aspects that dampen effective cooperation and the work itself. This chapter contains most disturbing practices that management has to face in order to make precaution and prevention. Brief common sense viewed from the role of team leader or manager. Illustrated examples of unfolding these practices are given with possible advice that may lead to their solvings.

### 3.2.1 Intrusion

Every disturbance requires great concentration as the counterpart. Every time an employee has to answer some unrelated question that someone asks it costs his time to

re-adapt into the problem he is solving. (Question is out of his current scope of work.) Constantly annoying other team members with questions can lead to hesitation and resignation to solve the problem until there will be "fresh air" again. This is serious problem with many possible solutions - lock on prioritized tasks, full time coaching, discussion or quote messaging[22]. (New employees should be under supervision of assigned team member that will have time to provide them guide and answers to their questions.)

### 3.2.2 Will to change

It is not always in human (Developer) nature to embrace the change. Team members get used to some specific technology or principle. They can be persuaded or just forced to do it, but always this topic should be discussed with the whole team to identify experience capabilities. New employees might also hesitate too much or act reluctant by new style of approach, but they can surely be coached with support help from the rest of the team.

### 3.2.3 Procastination and excuses

Most people at one time or other have procrastinated, which is normal, but when it becomes obsessive and chronic there could be an underlying issue behind it. Procrastination can result in additional stress, a sense of guilt and crisis, loss of professional productivity for not meeting responsibilities or commitments. Let's just say we have to deal with real procastinator, what can be indications of such behaviour can be seen on Figure 3.1.

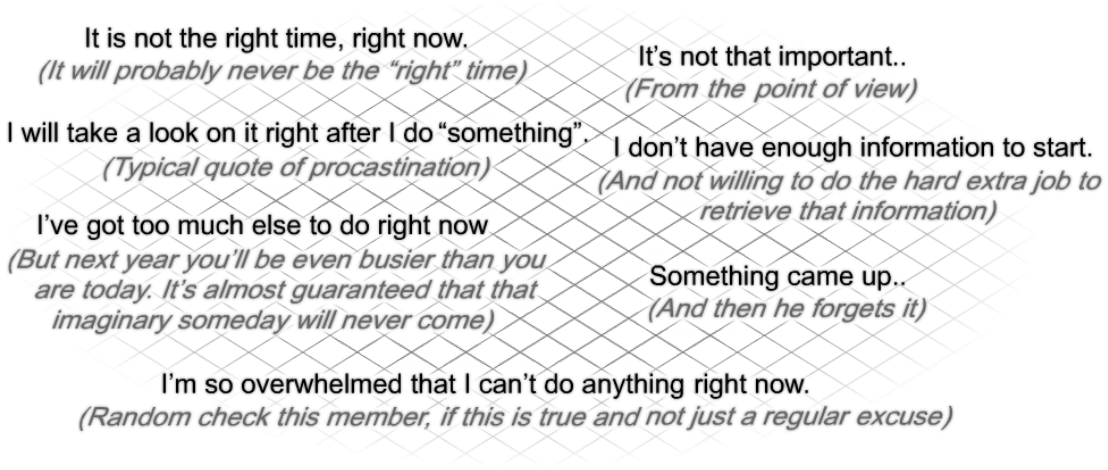



Figure 3.1: Procastination examples

### 3.2.4 Finishing tasks

When something is almost complete and it requires last few final minutes (it can actually take much more than that - even hours or days of work). In this particular situation the main problem is time estimation. For Project Manager small task can look simple, but not for the Developer. You (as Project Manager) simply can not want it precisely, because the Developer is usually not qualified to estimate things precisely with all relations to tasks planning. (This also includes known problem which is thinking inside the box.) Don't force Developer to tell you exact time when it will be done. Ask him for the rought time or extremly worst precise he can deliver (for calculations). Speeding up the process can bring new bugs into the problem - "expensive solving time bugs". Sometimes "When it's done it's done" sentence should be accepted, because excessive planning can stroke productivity.

### 3.2.5 Non-profiled cooperation

Leadership should be taken more as an action than state or position. (Where boss says "Do!" but true leader says "Let's do"[23]. Readable problems and constructive solutions makes good ground to learn new skills and work on solutions as team. The goal is to make problems so readable and give space for their solutions so constructive that everyone wants to work on them. (Cooperation is side-effect of such effort.) Coaching is therefore important part of teaching and training the team and profiling employee skills and competencies. This way they are properly supported with actual information, while achieving specific result of work. On the other side facilitator is an individual who enables team to collaborate more effectively (activities that make tasks for whole team easier).

### 3.2.6 Attention to multiple problems

One solution can bring new problems. These problems are more likely solvable, but they can (and will) spread bugs. Some of the bug fixes will be quick but few last will be tricky and thus the solution will require more testing time. Too many given tasks, simultaneous problem solving or thinking several steps ahead just stroke the mind and when someone is interrupted he usually has to rethink or start over. Solve tasks properly one by one and count with time reserves during their estimation[13].

### 3.2.7 Precautions planning

I will demonstrate this on example. Project leader should manage his time to be able help others with issues that prevents them from doing their job. On the run there appears a serious problem with developing and meanwhile some employee complains, that he can't do his job because something is wrongly configured with his computer. Leader should make precautions to fix this problem, because he is currently working

on critical problem and cannot fix the problem right now. Priority depends on situation and influence of how many people can do their jobs in contrast of problem critical importanity[24]. Precautions can prevent many casual problems related to work, when they are set well. (Someone else dellegation could possibly fix that team member issue with the computer and team leader could still work on critical task.)

### 3.2.8 Exploiting

Artificial systems like school or tests is where people can climb over, or find exploits when they learn how to manipulate with rules up to their edges. This will not work for any long-term activity. In role of Project Manager keep in mind, that there could be always someone who will keep looking for shortcuts of making work easier, even though the risk of postponing the problem elsewhere. The quality of work or extended time spent will later suffer project progress so track issues to their origin and mismanagement prevention should be applyed regularly.

### 3.2.9 Unrealistic expectations

We can not expect perfect solutions, and bulletproof clarifications because they differ from person to person. Explanation can be taken from different perspectives that may lead to collisions, arguing or denial of other principles. Trying to understand the behaviour of others can help building the view of their perceptions. Based on what assumptions and how probable they are right. (For example person who makes tasks done but with bugs, another one effective but at cost of time.) Knowing each person is better for distinquishing and preferable assigning critical task (or allocating sources). This can be benefitial for the whole team. Their efort won't get unnoticed when this is done correctly and sources are invested at their max profit to perfectionalized product[22]. (Project Manager should listen and believe to Project Leader for he knows his team advantages and disadvantages best - team is unlikely to be perfect.)

### 3.2.10 Social copying

Is part of social mirroring phenomenon**[25]** that affects practical process of acquiring new skills. Copying unnecessary actions is more based upon observing actions made by someone else than true learning of how to use objects. This strokes creativity in gameplay. New ideas can achieve game diversity on the market. Another problem lies in personal charm and skills with feigning interest in things that interest others, which is sufficient to make a good impression even though person simply doesn't care. New different way of thinking, visual sense by new unfamiliar perspective is something that players want to experience. They can choose the product that suits their notions above average game products on market, that is why this matters.

### 3.2.11 Same level of understanding

The best practice of knowing that team member understands what he should do is to ask him to repeat the task assignment and talk about procedures he suggests to make. This is an elegant and unoffending way, where employee feels important and aware of what is his goal. (Active listening should be used and possible corrections made to fulfill the task goals.)

### 3.2.12 Inefficient testing and resting time

Developer should take extra time to test functionality and be certain before he provides it. Later it will be hard and expensive to find it and make a fix. Some solutions requires time to make them right. Successful Project Leader is proactively trying to understand each job of each person and make professional relationships separately (with caution and objectively). Knowing the sensitive difference between “slack" and “relax" (not exploited) is required. Sometimes non-violent “check" is not a bad thing to do. Principle of attentin breaks**[26]** can be used to offer mandaroty time to think about new task before starting to develop them (right after finishing previous task).

### 3.2.13 Strict tokenism

Tokens in To-Do lists[2] are very important for tracking the state of progress and self motivation. To-Do lists works individually well, but are not suited for the tasks used by whole team. Advanced task lists based on tokenism might require precious estimations and be properly assigned to individuals, team members or departement, that is able to process them completely.

The strict checklist practice or exaggerated policy of making nothing more than a token effort is undeterministic and should not belong in game development practice of any kind. (Because the suppression of unlisted ideas and dampening opportunity to manifest creativite thinking**[27].**)

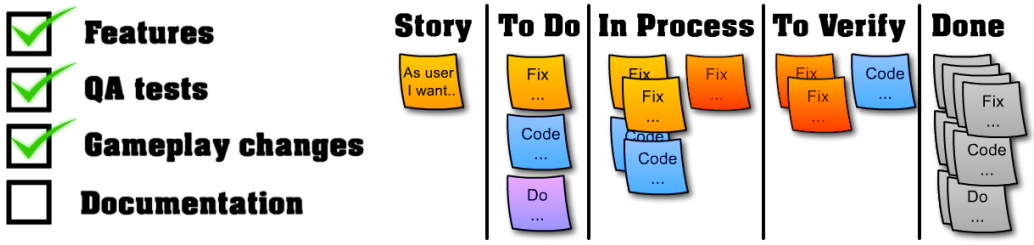



Figure 3.2: Tokens and tasks

2. To-Do list - Set of tasks, improvements or issues that can be tracked and marked with token of completion. (Side by side to other possible token practices as estimation, priority or usefulness.)

## 3.3 Positive benefits from team behaviour

In this subchapter I will focus on the advantages of team cooperation and give advice to strenghten the team through relationship consisted motivation, communication practices and deeper view to the act of effective decision making.

Most people want to learn and are teachable. Even the ones that use odd strategies over recommended usual. So often people are not given encouragement or feedback of any kind. People get bored, feel unchallenged and are not supported in any way. It could be any employee. Without recognition support of work might happen, that team members just get discouraged and give up[20]. With good manners the person can become very benefitial for the whole project. Good manager should give chance to everyone, even crazy or stupidly simple ideas can help to avoid problems in the future and simplify the journey to achieve goal. (Spark of enforcing new ideas have much to offer.)

Motivation does not work same way for everyone. Sometimes we try the quick fixes to improve near environment but that doesn't work. Or we "shock and awe" by bringing in some specialist or radically changing the working proces and procedures. But then we stop pushing it. We need to change the overall habits and team culture starting with our environment which is always a lot of work. (Blaming is pushing water uphill and will result only in foam at the surface.)

### 3.3.1 Motivation

There is positive and negative motivation which can be conditioned and unconditioned (spontaneous). Treat well, but with care, patience and understanding. For long term periods the cooperation with good motivation is more suitable (and required) for the job[8]. This differs from person to person. Being open minded and listen can help understand team member and based on type of his character decide how to treat him to gain his attention for doing work right. Show facts to assertive person or acknowledgement for melancholic person. Use the best of unforced psychology of character[20]. Sugar (give extra limited time bonuses or create motivation) and slash (push effort or lower bonus funds) can suit well and varies from team to team, therefore finding a good balance is recommended[28]. We simply want to feel better, be cheered and feel that someone trust in us. Mastering our feelings can lead to better experience and that is why the team motivation is so important (Motivation is in this case the feel of being part of a team, where you can stand for your solutions and support others).

### 3.3.2 Win - Win strategy

Approach where every agreement, task, job and cooperation is equally rewarding for both sides. When one side profits severely more than other it is not win-win strategy it is act of exploiting the partner[29]. (It is like having golden goose that lays golden eggs, you can either kill the goose and get many golden eggs at once, but if you feed her it be

mostly rewarding for both sides.) This should be used not only in pacts with Investors or partners, but mainly inside company environment. Leader should give employees reasons to stay in the company. Presentation of company growth, information meeting or any other team every effort counts and will be remembered in team. Sometimes a little is enough to motivate all members of the team for career growth. Create program to let them know management did not forget about their future and they will work hard for the company to achieve it (if setted up beneficial for both sides, exploiting can be noticed in time).

### 3.3.3 Managing assignment of tasks

Modeling task interactions and task representations are just one side of the business task management systems[30]. Personal capacity consisting of passion, collaboration, event plan or solution exploration (research) is basis for making more suitable tasks solutions and code optimized performance. Everyone has different set of skills and weaknesses. These affect will to work on tasks that are challenging, creative or out of routine. Team leader should do best from least and that kind of effective manners will be noticed by team members. Sort the tasks with reason and give opportunity to the whole team to work on them by their abilities and capacity. Nobody wants to do only boring tasks, and free will of having option helps even if tasks are not interesting at all.

### 3.3.4 Do not reinvent the wheel

We can build upon establishment of what previous generations of Project Managers have learned. Even if you don't have a mentor or are unfamiliar with your situation you can always stop looking for excusions and do some research. Do what other successful Project Managers do. There are many support groups, books, periodicals, sessions, events or semminars to find. (Good start can be "How can leaders achieve high employee engagement"[31] and "The Agile Project Manager"[24]). You have more of a support group, than you think, if you listen to people and whole team (some can offer much more, and gladly share it on ask).

### 3.3.5 Teamwork

The vision is clear to create systems for success[32]. There is a need of a good firm culture and work balance to let employees be effective as a team. True proffesional friendship is long term win-win strategy. The weak elements are recognized, but that does not mean they are no usefull at all. Keep looking and find the suited way of how to involve everyone in the process, because there is almost zero probability that everything will be perfectly suited, supported and sourced just for the job. Teamwork is result of trust, honor and support of management.

## 3.4 Communication practices

This subchapter will explain the way how we precieve information, learn, think, make decisions and understand the opinions of others in order of productive cooperation. I explain possible ways, how to learn from every experience and make personal enhancemet practices for discussions, meetings, negotiations, problem solvings, communication advancing or criticism facing.

### 3.4.1 Active Listening

Active listening is act of communication understanding. It is not just listening to the words, but also thinking about them. It can be trained and mastered, as long as we consciously aware the fact of allowing changes of our view to the opinions of others (Initial rejection[3]). Being fair and critical to our own ideas is very difficult, but most personally rewarding. We can bring into discussion a lot and take even more, when we learn to listen actively. There are many tools and techniques that will help us, from person centered communication to soft skills. The main issue leads us to psychology. It is founded largely on the use of psychological knowledge about human actions[33]. (For example everyone unconsciously searches for friends like them who act (and think) in a similar way, this is coming out of the psychology of personality[34].)

### 3.4.2 Communication experience

There are many definitions, opinions and views about communication itself. Some information from the discussions has been surely familiar to us in some way. Thinking about them, we sort our own knowledge or take a new interesting points.

It's like building skyscraper from "stones" of our knowledge. Solid foundations give us chance to make a "higher tower". The more sophisticately we can reshape them, the better we are able to arrange them, so that they fit into each-other more smoothly. Each of us will gain a different amount of these "stones" from one specific conversation, that may fit into our knowledge "building". And every single act, expression, gesture and words, to some extent affect experience of others. In meeting or any discussion we do compare what we have learned and heard so far. We think about what we hear in that real time. And also we assume what will come next or think steps ahead of that. We decide whether it is precious and interesting for us (or not). Feelings during actual moment with active listening with will be unique (never happend again) and lasts longer**[27]**. (But we will remember how we "felt" about the discusion, because only in the future we are able to truly measure the value of memories from the past moment.)

3. Initial rejection - (The first part of the personal defiance law) "Well, it's not true at all.", "Under certain circumstances it could be true", "I've always maintained that this is so right!".

### 3.4.3 Body language

Represents body gestures, intonation or facial expression with possible relation to personal distance area[4]. Warm welcome, shaking of hands or friendly gesture manners can lighten future communication or reveal hidden feedback. Practice body talk at the mirror. Give it half an hour and compare results. You will teach yourself how to act at certain situations more appropriately. Training in this discipline comes naturally. (It can improve quality of information that you want to show in gestures.) Strongly focusing on acting or excessive gestures can be noticed and you miss the whole purpose of doing them. Body language should feel unforced and natural.

Some countries have specific conscious gestures (usually related to certain continent or time of history) that represents certain action. There are also widely used signals for expressing enhanced feeling for situation. Knowing the possible unconscious reactions in discussion can indicate many emotions and acts. (attention, nervousness boredom, shyness, trust, vigorousness, critical evaluating expressions, uncertainty, lieing, superiority, taking or giving control, thinking or making a decision). They can be tracked personally, but are not always that reliable (and can be cheated). Keep in mind that there is information hidden within words in every discussion.

**Influencing the discussion** Is the higher level of Active Listening which differs in using all of our senses for following and comprehending the personal activity. Sentences used in talk steers the conversation towards unconscious direction that contains the subtly colored opinions of the speaker. Well trained manipulators can notice lies and deep character decision making. Whether consciously or unconsciously, positive or negative, the person who leads a conversation is kind of manipulative for convincing others. Like when we go to a meeting with an important person. We dress professionally and act nice, especially if we want something from that person. Dresscode is therefore important and varies by situation, firm rules and occasions. In meetings keep a mind of the bussiness goal. Female members of the team should wear decent proffesional clothes and not attract unnecessary sexual attention. This helps everyone to focus about the discussed topic more than being distracted by the look of a person.

Over time we realize that the actions of others and our own are based on similar behavior. We may also find more precious information in details and words aside of conversation, such as body language. The effort to understand the problems enable us customize it with significant influence of the relation to discussed person and the problem view from his perspective[35].

4. Personal Distance Area - May differs from person to person and represents the level of personal unenforceable acceptance or “access" towards person - intimate zone(under 45cm), personal zone(45cm-1.5m), social zone(1.5m-3m), public zone (3m or more)

### 3.4.4 Mental Strategies (MS)

Set of decisions that lead us towards a certain goal. MS are sometimes referred as a habits of thinking or active learning srategies[36]. MS represent absolutely any decision(s) we do. It is plainly sayed a whole process, that we have to think through and consider, untill we came up to decision. (This process happends within Decision Moment and uncosciously.)

(Example of situation:) The goal is to reach some finish place from start location within some city infrastructure. Each of us can get there by different ways. (”Straight” - ideal and perfect. ”Zigzag” - any solution close to ideal. ”Around” - out of the box thinking.) We use different sequence of direction changes (decisions) that lead us towards our finish place goal. We constantly try to improve our strategies. (The effective enhancement of “zigzag” solution can be for example taking a “bus” for partial way near begin and end locations that can save traveling time.)

The main difference between habit and MS is uniformity and applicable on higher level of thinking abstraction[37, 38]. All habits are MS but not all MS are habits. Habits are easier to understand (more like overall personal rules), because they do not directly explain problem solvings (or interraction) how we are recieve and process information in decisions**[39]**. (Habits rather permit the inference, but MS highlights the awareness of decision making steps.)

**MS Matching** Noticing similar experience in some situation or similar behaviour (attention focusing of brain neural association) gives opportunity to revise and rethink the possibly different reaction of person (in same situation with different experience) can be very informationly self-containing. (Like living the similar situation again with different decision and observing of possible results.)

**MS Learning collisions** The first basic understanding of the perception of narrative impersonal from our opinions. It means new opinions can conflict with our own. Think about them from the different perspective. The more the person observes the sooner he will understand not only how it formed, but also why. Everyone learns new things differently, what we like, what we are good at, and what seems good in this teaching is a form of coincidence. The understanding we have gained can lead to refution[38] of other opinions. (First step of understanding the problem is knowing of it’s existence.)

**MS Defense against change** Our mind naturaly rejects ideas that would affect our current way of thinking too unduly. This psychological defensive system works best against collisions in our MS. We can usually conspiously decide whather we have will to allow the change or defend and reject it. The deeper strategy fits into other strategies that build our personality the harder it is to change it.

**MS Rapid change** Any new experience that collides with our behaviour can cause mental strategy rapid change (catharsis, neutral or hurting).

### 3.4.5 Decision moment

All of our considerations, knowledge and experience helps us to make certain decision within moment of a second. This moment can cause Avalanche Effect[5]. By noticing it we can extend this moment and even find out what was the main cause which directed result decision in that certain way. We better understand based on what pillars we make decisions in that instant moment. This can be trained. (We can also notice the decision moments of others, but we don't know how "close" or "little" could made their decision ended different.

The first and fastest decision is more based on our mental strategies than be misleaded by our thinking "reason". Decisions based on first impression and reflex can be more right**[21]** than when we consider them in the second thought. Worst thing in decision moment is not to come to any conclusion be constantly hesitating. (Deciding to not to decide can also be an option.)

### 3.4.6 Perception glasses of our experience

We see the world through a kind of filter glasses with a color filter which represents our experience. We color that filter through which we are looking as we learn. This leads to denial of some things that are in collision with what we know.

Each of us will gain something different from the same situation. We have to understand that even very different strategies have reason from certain point of view. Observing the reality is rooted deeply in our mental strategies. It is therefore important to realize and investigate not only what we see, but also how we see it through these glasses of our perception. Learn from every single experience. We can learn more from active listening (and precieving all sense information), when we correct (or throw off) our color glass filters to see clearly without prejudices. The way we understand and perceive reality helps us create mental strategies and form our habits and behaviour[40].

Active understanding and even comprehension of the world through eyes of someone is very hard, reconciliation is reachable but this can be manipulatively glitched. Every word, gesture or act of any kind is manipulation to some point. Simply because when the person hears similar thoughts, he won't notice any manipulation. Reconciliation is usualy learned as a result of catharsis. Catharsis is plainly sayed "the peacefull" solution of win-win agreement (Usually leads to bond and true friendship in work, family, or with friends). Religion exploits this but our actions can be considered also on interaction level as Self-Fulfilling Prophecies [6] and Pygmalion effect[41].

---

5. Avalanche Effect - Small "snowflake" idea(experience,feeling, etc.) can be responsible for coming of an events that may shape and affect us much more later.
6. Self-Fulfilling Prophecies - Is the karma-like habit reaction or expectation that something will happen because we think that way.

### 3.4.7 Effective habits

Seven[21] habits and eight special habit [42] of effectiveness described by Stephen R. Covey in shortened core idea form:

**1. Proactive approach** Rather than react to specific situation as the circumstances victim keep looking for alternative solutions. (I could approach this differently instead of reacting with "that is the way I am".)

**2. Begin with the end in mind** Imagine how will you look or remember the current situation from the future. Do things in what you believe, you like and the way that worths your time.

**3. Putting first things first** Make a schedule of your main goals and base your time management on that. Prioritize important and critical tasks over features.

**4. Think win/win** Is the belief that there is a solution that is profitable for both sides (Symbiotic cooperation). Most people have a win/lose mentality. If the other loses, we win. But this usually not lasts for longer term period.

**5. Seek first to understand, then to be understood** We often have our opinions ready even that we do not understand the person or the situation. To learn to listen is therefore a crucial habit for effective communication. We will not only listen in order to understand, but also to reply more precisely.

**6. Synergize** Synergy is the added value of the whole in relation to the separate parts. Synergy requires trust and openness. We should keep looking for the qualities and potential in others, so we can create bond and make cooperation.

**7. Sharpen our habits** Constant effort to improve, enhance, maintain and renew out mental, social, emotional and knowledge capabilities. It is vital to keep nourishing and developing our habits.

**8. Find your own voice** Personalize your work in an effective order. Known what you want and became the change that fulfill your dreams. Help others but don't forget about yourself, else you just end up completing the dreams of others. This habit is not ment just for individuals, but also for organisations.

**Summary:** In order to achieve better decision effectivity through our habits we need to listen and act with open mind (§1,2,3 - Independence). So we can start influencing

reality outside more effectively (§4,5,6 - Interdependence), and with our continous improvement (§7 - Effectivity enhacements). And most of all developing new habits, that suits us best, through our skills and experience (§8 - Comprehending).

### 3.4.8 Person centered approach

**A growing openness to experience** Stop applying perceptual defense and have no need for subception[7].

**An increasingly existential lifestyle** Living each moment fully and enjoy each single experience. Be open-minded to what is going on. This results in happiness, excitement, tolerance, adaptability, spontaneity and trusting the others.

**Increasing trust** Trust in own judgement and ability to choose appropriate behavior (not only expected ones) for each moment. Do not rely on social norms, have wider and more reasonal choices by basic senses.

**Creativity** Learn how to think creatively and how to adapt to circumstances and how to conform.

**Reliability and constructiveness** Individual is open to all needs and maintain balance between them. He can be trusted and act constructively.

**A full rich life** Life is described as fully functioning individual which is exciting and suggests to experience intensely joy remind of pain and face problems without fears with courage.

**Propositions of thinking** All individuals exist in continually changing world of experience of which they are center. This experience is Phenomenal Field[8]. We actualise, maintain and enhance the experiencing process. Behaviour is described as goal directed attempt to satisfy needs in the preceived field. Experience is organized in some relation to self. Under certain conditions, involving absence of threat, the experiences which are inconsistent with it, are examined. Structure is then self revised to assimilate and include such change. Psychological adjustements of behavior are on symbolic levels which involve experiencing some established life rituals. Consistency is adopted from real experience.

7. Subception - Perception (or reaction) of absorbing information without conscious awareness.
8. Phenomenal Field - Experienced reality for the individual (is similar to perceptual field).

### 3.4.9 Uncospious problem solving

Someone needs to walk, keep clicking on pen, turn eyes in different direction (directly letting know that he is thinking and not paying attention). But this kind of thinking is conscious[43].

Subconscious or uncospious mind solving is the fact that we can figure out solutions in time without consciously thinking about them. We use around 10% of our neural capacity in cospious thinking**[39]**, but the rest (most is consumed in associations and collecting the relevant information) is uncontrollable part of our thinking. (When solutions are interesting this process is enhanced.) Trust in your brain, that it will uncospiously try to solve problems which bothers you. Sometimes it at least defragments information on the way, which may be catalyzator for it's solving. This can be used in variable things in both development and social management.

Last meeting of the day can focus on algorhitmic assignment solving. The task is given and discussed only the rules. Next day in the morning the Developers (meet again) and tends to find solutions more accurate, creative and competitive.) Adaptation in time is responsible and possible to learn practically anything, if we keep on trying (even complex ideas). This is silent reason what of long-period employees have so extended hardly measurable experience value. for the company by their suited personal experience.

Inteligence can be developed. Inteligence Quocient[9] means in fact nothing more than a number, that simply can't content all personal qualities and aspects. But the true improvement is measurable. (If you keep up to learn more than you forget in time.)

### 3.4.10 Uncertain truths

Consider that personal truth is not always correct, let's think of it as semi-correct (or partially incorrect) by the point of view. Some things can be very incomprehensible to Project Managers while simple for Developers, the difference lies in perception. Accept something new as a partial truth is possible (in case we are at least slightly critical), but it's very hard to admit a mistake, even if we do not know what in our behavior caused it. We learn, that some of our opinions are most certainly wrong due to something we do not understand (or we do not have a way to confirm it or disprove). Over time we can understand even complex ideas when we acquire the necessary knowledge and experience. This affects how we approach to the negotiations with team members and accepting their opinios or excuses.

### 3.4.11 Association thinking

We think in associations. The more of them, the easier we will remember because the neural connections weaken in time. When someone chewing a gum of certain flavour

9. Inteligence Quocient - Is metric used for measuring person inteligence in relation to the rest of the population.

while learning, then he will remember more when on the test chewes the same flavour[40]. These associations are made within all of our senses. Specific thoughts are causing what we most commonly do - thinking and remembering what we learned to figure out the answer for forgotten test question (in this example). Knowing how this works and noticing it allows to remember more and in longer sequences (which can be used to track project issues). Bringing more relevant information to the discussion makes it more clear and asking right questions based on remembering more project issues save time of a whole team.

### 3.4.12 Attention hardening

Ammount of information that human mind can process (and absorb) is limited (differs from person to person). On difficult problems or longer meetings our brain also looses attention. Brain simply cannot keep all neural connections and context indefinitely. Breaks helps overally, relevant jokes helps even better (must be felt as they are unprepared), expression of opinions can be containing and inspiring. (Active listening leads to Mental Strategies Matching.) Spark works best, but is somehow unpredictable (It makes a long lasted bridge to skip multiple steps and learn extremly fast by unconspicous prethink[10]). Know where your absorb limit is, (when you cross it - And don't forget to tell others that you need a break.)

Misunderstanding drowns us and we have to go back on some most stable rock of what we know but we are lost for a moment in what we cannot learn because we have to swim to safe ground - something that we can step upon and start over.

### 3.4.13 Knowledge influence

When we learn something new (in lecture for example) we might remember 90% of it in an hour, like 50% in couple of days and in matter of months or years we might not remember exact words or the identic idea, but there will be something that lefts inside of us, an experience that build our "perception glasses". Something that will become some part of our decisionings "mental strategies" and even though that we will not remember exactly what happened in that lecture it will be part of us, it will be something based on what we will maybe decide differently (that if we were not present on that lecture) and this is something that form us. We are becoming part of our experience that we live, learn change or trying to erase.

10. Unconspicous prethink - When we like something, we think in ideas related to close range. (When we know enough we simply can walk upon "rocks in the swamp"(rocks represents piece of knowledge) to go somewhere where we already have throven "rock", that way we can stand upon advanced place and go even far to unknown and most explorable part of "swamp".)

### 3.4.14 Conclusions summary

The understanding of how we think is crucial to make better and more effective decisions. Observing world through glasses of our experience helps us actively listen and precieving more information from different view. Sharpening our senses, not only to spoken words, but also to the body language gestures helps us understand and notice the true opinions. We build our habits to achieve higher effectivity in our mental strategies. Know what the decision moment is and how we can extend it gives us time to revise more aspects than we would do naturally. When this becomes habit, we can think much deeper about the problem and even include new different perspectives. Mental strategies uwareness shows us what sequence of work needs to be done to achieve our chosen goals, and how can we enhance this process or learn it over. The more we think of something the more it becomes part of us. Knowledge without right habits doesn't support proactive MS to form both creative and objective thinking considerations. That is the reason of how are our decisions made. Only then we can understand and realize how to alter them in the terms of efficiency. Feelings affect our thinking mood and point of view. We should be on guard of too critical thoughts, for they are standing in way of full detachment from our collisions in thinking. Thrusting in yourself helps to reduce stress levels and uncospious thinking (problem solving) will sort what is important for your later decisions. Nonetheless the unremitting effort to improve our leadership is the most valuable lesson we can learn on every decisions we made (piece by piece).

# 4 Best Practices

## 4.1 Introduction

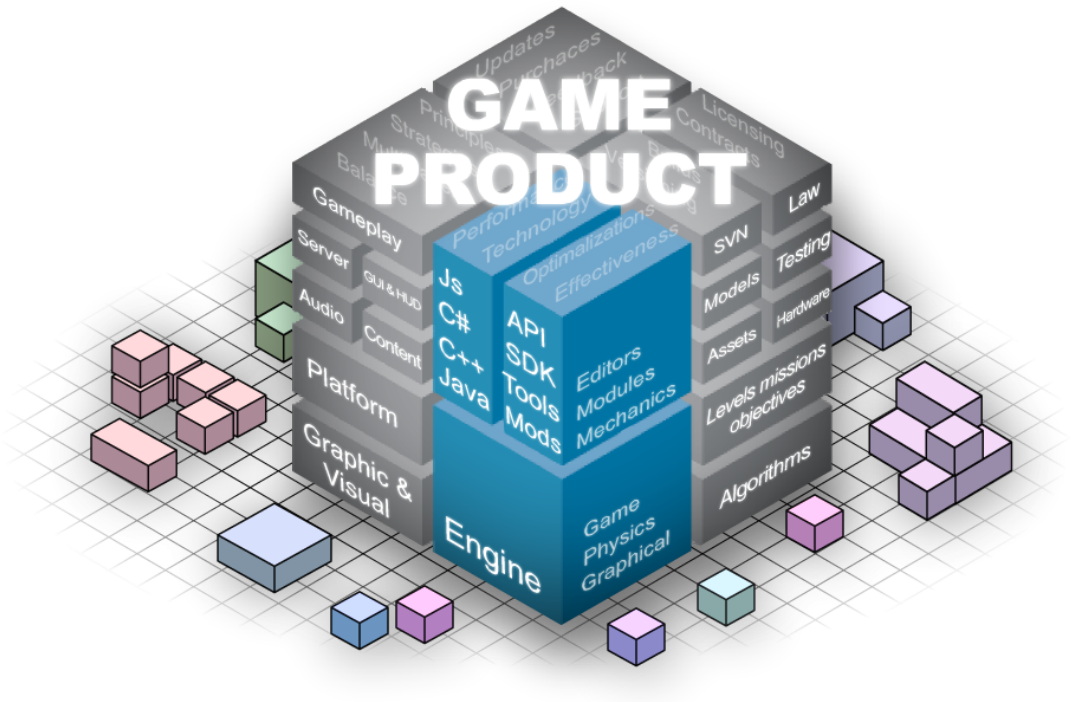



Figure 4.1: Game product consisting of many little solution parts

**Game Product** Many game makers (both development and management) tends to forget of how many small things can game project consists. (Figure 4.1) The team should be able to contain and process them. Better familiarization of team experience leads to better planning. Each team has different set of skills and weaknesses. Management can greatly help teams by gathering contacts to people who have faced same problems and found their solutions. In Czech Republic there are just few developing companies that they are well known of each other. (Know other game developers from your vicinity.)

**Best practice** Technique or method that gains better results over other traditional techniques. Best practice can be referred as a process of developing or following the way of doing things that other companies use. Inovation and improvement of practice or it's adaptation can lead to a it's replacement by more sophisticated practice that gives better results. Maintaining quality can be noticed in management standards[18]. Organizations balance their unique abilities of the team and adopts certain practices,

that suit them best. When practice is used as a standard way of doing things with better measurable results[24], it becomes best practice for that certain purpose. They have strategic meaning and because of their formality and application are diverse between many disciplines.

**Personal opinions of experts** Underline I present brief opinion of experts in game development that I interviewed. They agreed to share personal advices of usefull practices they experienced during development process. Like cherry on a cake I wanted to give real reference from gaming representatives of various leading czech and international companies in actual project positions, situations and proffesional experience. Also serving to proove the value of previously presented information I have described from theory. (That I am not the only one who thinks this way.)

## 4.2 Project organization phase

### 4.2.1 Idea and vision

Should be made by all main founders of the project. The idea should be simple enough to explain to targeted casual customer. The vision should be even more sophisticaced summary of the team belief.

Licencing and rights with deployment (on platform,in location, with support, and marketing) should be mentioned here.

Project Idea should also contain the core principle or game mechanic, that gameplay will be build upon.

Make a deeper research on the web or use contacts in game development companies helps save time and money. (there are many located here in czech, mostly in Brno and Prague). Also consider delays and funds spended on marketing, because even e-mailing hundreds of e-mails to redactors won't help that much and game community is hard to obtain (for a new project) without showing them at least something more of an idea.

Bussines plan should be well prepared for possible future Investors and check posibility of allowing new inverstor(s) to join on the go. Also good idea can receive external funds from kickstarter websites, social or community support and crowdfunding. There are also possibilities to publish project on specific markets and make contract with Microsoft, Google, Apple, Blackberry or video game platforms as Xbox, Playstation or Ouya.

### 4.2.2 Project start

We get really excited about having opportunity to start something new. Whether it is reading new book, learning something inovative, or even invinting new game concept. But as days goes by the enthusiasm fades and fades until it just abadons our minds and motivation dies. The hard thing here is deciding to want something, and do proper

actions (or set of events in order) to make it happend. It is the opposite of "I don't think I want to invest the time necessary to complete chosen goal". Motivation from results is claimed only after task is done and not during work of reaching it. Many of us have great ideas (and dreams) that are just not reachable if we don't start small. In software development and certainly game development this is serious reason why indie Developers abadon their projects, plus the gaining of new knowledge (from and during these unfinished prototype projects).

**Management of time and setting realistic goals** Most team members effectivity suffer from time mismanagement or undeveloped motivation plans of doing their work (inability of planning whole project from the beginning).

**Simple motivation planning trick:** Let's say you tell yourself: "Today I will do SDK documentation reading for two hours." Ask yourself as if you were arguing with yourself: "Is this really something, that I want to do?" I force myself because I have to do it. "Honestly?" (Notice how self-critical talk from yourself helps you test whether your goal is achievable) Ok let's just say now, that you lower expectations to one hour of reading. Is your answer yes or maybe? How about thirty minutes. How about fifteen minutes? Ok fifteen minutes of reading? I can do that, totally and for sure. Start making goals that you know you will complete. Rather than big fails concentrate on small wins, that cheers your motivation and enables you to complete more very small tasks.

### 4.2.3 Pre-production

Scrum basis is excellent (and recommended) start for creating a reliable game production plan. Let Product Owners (Stakeholders) set and discuss the game features they want to be developed and together prioritize them. Work out schedules, budgets and time estimation. Define sprints, meetings, resources, features, milestones, goals, roles and assign competencies (prevent arguing). Make requirements for each sprint, process workflow and write tasks into the backlog. Provide stable management for accurate problem handling and specify firm culture rules. Hire team members required for the job, fill gaps and obtain required sources. Make collaborative plan how will Level Designers and Gameplay Designers work together with Artificial Intelligence Developers and define their close cooperation channels. Make technology prototype (for multi-

*Personal opinion of expert:*

*Showing investor that you have not only spent your time for the project, but also your own money is the best way to capture his attention of how serious you are about it. Be aware to reveal disadvantageous contracts offered by investors (be able to reject and be rejected).*

*Vladislav Spevák, Mingle Games*

player game, new concepts or performance analysis). In case of bigger project consider distributed teams. Let feedback flow into the team swiftly and constantly (no need to wait for alpha, beta or milestone to know what features are working). Direct remote working members and make effort to locate team at one place (for feedback swiftness and prevention of bottlenecks). In case of very small team choose good partners or work alone as long as possible.

**Try open source SDK**[1] Gaming community gladly gives advice, tools, mods or show partialy finished game prototypes. While deciding which engine to use or chosing the platform direction and targeted audience there is a good thing to do a little research and engine checks. This helps to manage what is important and what we need to make game better thatn competitors. In “Valve” this strategy was used multiple times to test attractiveness of game scores. “Valve" also allow players to download “Source Software Development Kit" so they can advance the game, creating mods and further content, that can be shared among them. This gives extended credit to game, replay with new ideas and communication with the community helps to develop better tools and engine itself. This should be considered in case of making own engine or use tuned and more common one that will save sources, time and money.

**Proceses** Avoid branching assets. Use safe processes (or define safe methods) to branch game versions (if they are required) to avoid collisions. Consider using external level editing tool and saving levels in XML instead of in scenes. Because it makes loading much faster to re-setup each scene. It will be easier to merge scenes and keep track of data across levels. Consider writing generic custom inspector code. (Inspector components should be defined as a class-type rather than field-type level.)

**Organisation** Do organize all objects and entities carefully with system that suits best for purpose of solution. (For example - folder objects hierarchicaly for more logical access.) Separate interface from game logic. Make the game runnable at all times. Use

1. SDK - Set of software libraries and tools for creation, extension or usage of certain software solution.

*Personal opinion of expert:*

*Keep solutions deterministic to the rules of proffesional agile software development and personalize them in order to achieve more effective results according to unified plan. The less variability in the game the team can be better adapted to the needs of players and induce enhanced experience in less time. Gaming culture that constantly evolves is essence of success, because sometimes doing something inovative or exact oposite of what everyone else is doing can be the key to outrun trends (and very good prevention of routine).*

*Michal Gabriel, Creative Director, Tinysoft*

extensions to work with components that share an interface or an network interface. Avoid using different idioms to do the same thing. Maintain your own time to notice organization bottlenecks and do report them. Divide specialisation configuration for different behaviours.

**Direct the team** The main values from agile manifest are that "individuals and interactions are more important than processes and tools"**[4]**. This concludes that the working software is over documentation and collaboration with customer is over contract negotiation. Responding to change means more than following a strict plan. Personalize these into development process and make sure everyone knows what is the team strategy.

**Project compiling time** The size of project grows in process and compilation times can be a significant problem, this is the indication that planning failed with estimated distribution of sources (extended visual or sound content, high detailed polycount models, database loadings, speed of internet connection or access to extended content).

**Version management** Using effective Version Management[2] means less concern about trying something out. It gives team the freedom to focus on creating code and assets to explore new ideas without having to worry about losing any valuable work.

## 4.3 Development phase

### 4.3.1 Project development

**Prototype** Gather user insight and make first test version to test the technology or concept drawbacks. Plan multiple scenarios of design and investigate possible known bottlenecks. Discuss with team members or other game Developers technical restraints and risks. Big game studios create and test several prototypes in mods of other games (or simulatations). Developing own prototype contains best observed principles and game elements. Work out game prototype at the earliest possible stage and iterate it to revenue from a very early stage of game (Prototype strategy of creative building concept of game "Minecraft").

2. Version Management - Changes will not erase the original version, simple roll back can be made if required.

*Personal opinion of expert:*

*Android multiplatform and overall support helps to ease developer tasks and fasten development more than other platforms, which makes it most suitable for new and small (indie-like) agile teams.*

*Filip Hanáček, Developer Relations - Google*

**Gameplay** Main goals of gameplay is to set balance between game aspects and in-game elements. Give player space to manipulate with them and create strategy of play. Fun, creative and challenging factors represented within level design and style of the game retain playability. Have player entretained through the playing requires constantly showing new possibilities, content, principles, threats and rewards. Players want to be guided through singleplayer campaign and be praised for their job by video or audio effects.

**Sprints** Sprint and task management can take more time than planned. Different set of human sources and feature opportunities make selection uneasy. Estimations are time consuming and department meetings are in human source collision (multiple technical roles of one person). Do not divide sprints or make changes in planned interations. Milestones should not change. Every change or unexpected situation should be told to Project Leader or Project Manager, right when is it unfolded. Mismanaged tasks, supression of interconnected problems, understatement or depreciation of tests resuts in bad estimations and unrealistic project planning.

**Engine development** Involves a lot of math along with the complications with development. Code require time to master and building own engine should be always made by top eperienced professional Developers of team crew. (This happends somehow naturaly, but noticing absence of someone important is a must in an early phase.)

**Motivation** Constantly support team motivation. When you feel like you have set up and prepared everything to make them productive and you have nothing more to do, think ahead and plan. At least make efforts to know team better and fullfill their future needs. Notice every slight advantage inside new game concept and find the best way of using it (for example give free space to talk in team meeting). Direct and push the progress according to feedback and feeling of entertainment.

**Documentation** Document your setup, but do not push it or make it as priority (layer management, tags, GUI depths, idiom preferences, object structure, animations, scene setup, etc.). Follow a documented naming convention and folder structure are the basics.

*Personal opinion of expert:*

*Small agile team communication with the game community is essential to help publish regular updates and react on players feedback. Continual estimation of progress and team support with prioritizing tasks helps to catch up deadlines with time reserves to provide featured quality with less bugs.*

*Marek Rosa, CEO & Founder, Keen SW House, Project Space Engineers*

**Online and multiplayer development** Coding client and server side depends on application domain. Network and database issues are common in online game development and there are many widely published server solutions. Good thing is to provide client-server architecture for specific (or targeted) platform rather than support multiplatform sevice systems. Keep the implementation simple, consider storing data in XML format and create stable unified services. Having server side solution is good prevention of cheating.

**Physics** In computed physics where animation is combined with physics consider use of ragdol with soft-body physics and poolish them to game expectations. Particle systems can consume a lot of performance and should be managed on level of detail. Meta balls, fluid simulations, or particle flowing systems are hard to make right. Tuning of visual alternatives can achieve better looking results with significantly lowered performance costs. Keep static objects and use static and dynamic batching to ease process of collision detection. Using physics within game environment has to be properly tested and make sure player can control objects as intentioned. Using cloth meshes are very performance expensive and should be used with caution or in pre-scripted situations.

**Visual effects** Make sure to visually reward certain player actions and in-game events (playing satisfaction). Use them adequate to game design and game style. There are various animation and composing techniques along computer generated imagery and particle effects or screen alterations. Using shaders, depth of view, blur, color defocus, lens or sketches can improve overall feeling from the game. Visual is remebered and sells.

**Sounds** Use uncompressed sound or soundtrack assets, to prevent processing them in real time (costs a lot of performance). Use 3d sounds when adequate. Sounds should feel appropriate to the situation or reaction to a player actions.

**Keep the transparency** Make goals specific and trackable. Focus on bug reports, making fixes and subject sources. Every team or sub-team generally should have it's own task board to provide a way to make visible progress daily.

*Personal opinion of expert:*

*Make gameplay with concept alternatives, sketched up mindmaps and managed plan of all development phases at one place. Work out and focus of what your team can do offer best, rather than what is missing. Often is easier to redevelop things than longly discuss them. Throw away what is unimportant and most important don't give up.*

*Jan Mrověc, Gameplay Designer, Project: Age Of defenders*

**Integration of social aspects** Communication within the game (chat), providing connection with facebook, google account or sending twitter information about player progress is requested by players to let their friends know of their current game achievements or status. This can significantly help to viraly spread the game and challenge players to invest more of their time playing. (Flash platform released web games of Geewa and Cuketa during years 2011-2013 notified significant increase in purchase of in-game content by sale statistics because of good social integration.)

**Project meetings** When you feel some information on meeting is important feel free to repeat it to spread that feel. Insulting and constant arguing doesn't help anything. After members chill down, they end up facing at the same problems (of which they are paid to solve). In order to do that they just need to keep discussing. Prevention of resigning, future problems and misunderstanding is the essence of a good communication. Identify situations that in each scenario will result in failure, anti-motivation or insulting. Stop actual discussion, think different and figure out new direction all together. When you are not prepared to hear critism do not ask right away, gather more information about the problem, work it out, or give it some time until you will be able to discuss it even with risk that it might be a failure. (Prevention of meaningless and undirectioned discussions.)

### 4.3.2 Game design

**Creative thinking tips** Collect information about the problem, before making important decisions. Simple solutions are usually the right ones. Learn how to explain objectively and give easy story examples.

**Think oppositely** Think what we do know, what is defined and why. Make simple list of it. Partialy or completly change the headed idea direction.

**Think differently** Make unique unlinear change or event, that was not previously mentioned.

**Think ahead** Make "mind maps" and then time plan according to mentioning listed ideas with certain continual escalation.

*Personal opinion of expert:*

*Making multiplayer game requires good cooperation where the whole team has to adapt and improve game principles on the run, according to player expectations, in contrast with development possibilities and chosen technology (and network limitations).*

*Tomáš Mizerák, Server Specialist, Geewa Brno Office*

**Scalability** Get things together at the same scale right from the beginning. Strict by concept and check game product parts redundancy. Make main game aspects in the same style as planned concept.

**Game structure** Scale transition efforts by schedule that require collaboration among disciplines. After each sprint make sure the core mechanics are still fun and still related to gameplay.

**Information workflow** Working group conversation channel is a good thing, but sometimes requires supression of misleading directions towards unrelated topics.

**Tags** Strict taging is comfortable, but slows the progress. Tabs, forms, configs, tools or modules are recommended (when they deal with crucial elements in specific phase of game making).

**Graphic** Use test art, polygonial grids, skyboxes, normal maps, lightning, colours and gardients for shader testing and better visual enhancements. Work with prefabs objects for specialisation and do not specialise instances. Try to establish links between instances automatically. (Many algorithms are simplified if models have the same mesh facing direction.)

### 4.3.3 Testing and profiling quality

**Achieve quality** There are many tools that can be used for project enhancement, statistical information gathering, tracking, monitoring, diagnosis, security issues, platform compatibility, support resolution, portability, validation, player performance monitoring and can positively improve quality of the game product.

**Profile difficulty** Keep gameplay deterministic and stable. Ability to respond and submit changes according to many testing sessions fasten the level design and difficulty production. Profile patency of playing through the game levels and checkpoints.

Personal opinion of expert:

*Tablet is necessary for proffesional visual and containing design. Details for game graphic from pixelart through texturing and animation requires precision with suitable style and skill. Graphical background overview in games history and unique skills are very usefull and wanted as well.*

*Prokop Smetana, Dreadlocks, Lead Artist, Project Dex*

**Test management** Don't test too early or too late. Enforce integrated testing or continuous integration of the source code produced for the game. Test only when there is a reason to do it. Do not always concern the actual situation because it might not be suitable. (Different testing articles (fixing bugs) are before basics of gameplay and after.) In later game prototype phase debug with FPS counter, graphical logger, profiler to debug physics, animation, and Artificial Inteligence (AI). Make fast way to process screen shots for gameplay and shortcuts for printing the state of world (player position and state, list of important artefacts,objects or enemies, etc.). Maintain a scene with all gameplay elements. Create constants for debug shortcut keys, and keep them consistent in one place.

**Reaction to negative comments** There is generally better to result issues positively or do not react at all. Deleting posts, blaming the person or banning accounts is very inappropriate action and such situations should be notified but avoided. (Exception is frequent issue occurrence, but answering should always be made politely and on proffesional level.)

**Personal issues** Working long enough on a game project can be hard. Team members might be frustrated or whole team do not see results or side rewards of their work. This is why Project Manager job (with Project Leader) is to hold the team, help them, motivate them, know when to push and when to be their guide. Treat with the team members as equal because confidence and trust will not go unnoticed and cohere company together. When it comes to personal issues, do never directly punish someone for some unintended failure in his work, mention it overally. (Rather reveal why it happened and create new motivation.) Sometimes even not interacting can be better than reconcilitiate quarreling team members.

### 4.3.4 Poolishing

**Task terms** Best inspiration and motivation for having things done are strict terms**[20].** This may results in earlier game fixes which translate into better gameplay, budget savings, improved production and possiblility of adding new features. Transparent view of the reached milestones(through tasks) is better than presentating documentation.

*Personal opinion of expert:*

*Building solid software testing operative strategy will efficiently reduce risk, push sprint releases and in the end fasten software deployment to the market. Integration tests, functional tests, code inspection or deep analysis can create a quality assurance. Proving quality of given product features to the stakeholders is important.*

*Jaroslav Ráček, Senior Consultant, IBA CZ*

Playable game at the end of each sprint make it easier to let feedback show defects and bugs.

Team level of task certainity depends on overall progress made on specific project, better explained tasks fasten the comprehension of problems which leads to better solutions (also time spent on fixing the way till team gets there). This is the point of success, where team effectivity is high because of fact that project is almost finished (this can be applied on distribution of tasks).

**Performance** Visual and logical results of accomplished game should feel stable and without lags (by used resources). The rate of processing work, gameplay utilization, input response time, information compression data processing or network transmission affect game requirenments for hardware on specific targeted platform.

**Graphical optimizations** affect how efficiently the game is at displaying visuals or rendering the graphics.

**Procesor optimizations** affect logical calculations, artificial intelligence, gameplay elements, running engine, event or trigger tracking.

**Memory optimizations** are required because of platform or mobile limitations (allocation, leaks or underspiked garbage collecting).

There are many techniques to reduce perifery data transfer latency or how to use best of throughput limitations. Network (and communication overhead) or database optimisations are a must for online and multiplayer game services. Game might also run slowly on machines with older hardware and outdated drivers (this is important in mobile game development and should to be considered from the beginning).

## 4.4 Production phase

### 4.4.1 Project release

**Advertisment and marketing** Write about game release to game community redactors, bloggers, social networks or marketing representatives. Ideally meet them in person, or invite them to team office and spread information about the upcoming game. Let game community know release date and schedule reviews and publications according to this date. Targeted marketplace evolves and adapts on new titles, which may affect date of release or properly selected platform to start with. Have prepared game intro and practical training guides(brief video gameplay introduction) on game website (youtube can do the trick). Be aware when marketing cross the line and become agressive against customers. Generally plan multiple marketing channels and use the best

of the most influent ones. It is unrecommended to agree of making support to devices you can not really test (mobile development).

**Sale strategy** There are many strategies of selling the game and there is no general solution. It greatly depends from product, targeted audience and company brand. Most common practice is to release game product payed or free (with or without microtransactions or in-game advertisments). Game can be also released in two versions - demo(lite and free) and full(paid). There is also financing alternative of pre-release public or private donations, Investor and Stakeholders investments, crowdfunding websites or kickstarters(regional).

**Founding strategies to mention:**
**"Pay to play"** - Player needs to buy the game (or in-game time) to be able to play.
**"Free to play"** - Players are able to play the game without paying.
**"Pay to win"** - Player simply buy the game features for making it more fun or easier.
**"Free to win"** - All players are able to play the game equally (donation strategy).

**Microtransactions** In-game money bought by real money for extended value and features inside the game. Do not rush it. In current situation of 2013 the profit of microtransactions is a most profitable strategy on mobile platforms (average players buy their in-game content features). There are some ideologies of soft and hard in-game currency. Goal is to let player know that he can either play the game to earn(or reach) most of the stuff or pay to get it instantly. In time player will compare his own time spent to the game with real money value. When the gameplay with in-game currency are in balance with good microtransaction strategy the profit is assumptioned to become most.

**Finding good player base for beta testing** Make some research and investigation (start with asking product managers). Search through the charts of games played on certain platforms (or engines) to locate good ground with real valuable information. Make limited testing access. Send limited invitations. Test what is important and let team to decide together. Testers have sharp eyes while playing games in their early stages but wide audience keeps in mind and critically evaluates every aspect. Collect

*Personal opinion of expert:*

*Game sells by first released day so it is crucial to gain attention from gaming community. This is why marketing, social aspects, reviews, and advertisment have meaning in order to increase game sales. Financing such projects on specific market can also be made by crowdfunding or by kickstarter funding platform, if the project have clear plan of how to stand on chosen market.*

*Tomáš Jukl, Czech Startups, Regional Manager*

both positive and negative feedback of kind, relevant or not, is the essence of developing the game players want to play and choose to spend their time and money.

**First impression** Lowering expectations can be used as strategy to lead into more happiness about the game. For exampe intro of game so epic (several levels above the game) will result in players leaving the game because their expectations were not fulfilled. But the bad thing is that some of these players will write the critical comments and bad ratings - which lower new players (possible paying customers) flowing into the game.

**Manage delays** Main issues of not delivering the games are delays, which are results of mismanaged time plan. Team should be aware of time and not add more features (bite more) than they can actually process (chew). Project Manager is the one who defines where the limits are and consultate them with Project Leader (with supervise and control). Estimation is necessary for planning, but in smaller projects it is usually underestimated by whole team. (Estimations require knowing the team well and wisely plan time reserves.)

**Repay effort** Show that the company can be generous for team members that created the product. They supported game from the beginning and might help it in the future, because they understand it deeply. Teambuilding is win-win strategy and good option to do aside from extra salary or deal out free time. Some Europe gaming companies supports revenue investments bonuses to team members. As the reward for believing in company they work for. (Namely Ubisoft studios, EA UK studios or Crytek)

### 4.4.2 Maintenance and support

**Processing fixes** After release, agile method enforces faster patch development delivery (usually couple of days or weeks). Avoid rapid changes in gameplay that can be risky and expensive. Make production culture sustainable in longer term plan.

**Maintenance** Address issues when they arise and set feedback flow straight to team department that can process them without disturbance of others. Provide ongoing sup-

*Personal opinion of expert:*

*Give team a working mechanism with sources, space for inovation, your friendship, and be always there to direct them. Contact with other developer teams and bring knowledge enhancements for the team. Know the difference between "doing things right" and "doing right things" is value that distinquishes management and true leadership.*

*Tomáš Botek, Team Leader, Cuketa*

port for the game inovations (keep it alive and popular). Team can operate on their own in general, but will surely encounter new situations and political decisions for which they need an expert advice.

**Mobile market practices** Some mobile markets, (namely iOS and Android) have policy set to keeping applications up to date. This results should results in regular release of updated game version else game market ranking will fall down (can be exploited by "empty" updates just for the market to keep game app in top lists). This can be measured and tracked in sale statistics and represents real issue among other popularity gaining practices.

**Team and project condition** Managers can do simple things that can have a great impact on team mood which means "swim one extra pool length". It is constant effort of finding motivational programs that the team can apreciate and inspire them to work.

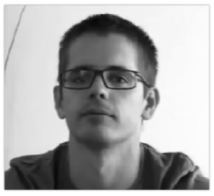
Management should make creative attempts to provide more positive motivational climate on a regular basis. Constantly gather actual information from the team. Make workplace open and add space for fun and relax. Share information about organization and current situation with the team. Involve team members with decisions that will affect them and if you are not sure which solution is right always, give team members opportunity to speak or even decide. Reward the team for their performance. Appreciate the work made by every single member of the team and ease critical working conditions.

## 4.5 Best rest practices

**Planning process of saving time and sources** Save time at programming editor. Save hundred dollars on assets. Save thousand euros at building certain solution (by using advice of companion). All these can make together remarkable savings that you may use much more efficient on creating new content (for example), that can make the difference between casual average rating and top A or five star market ratings.

**Find the most efficient way to develop** Even usage of public community solution, exploiting existing editor tools for configuring or generating things is tremendously

*Personal opinion of expert:*



*Surround yourself with the best people you can find. Treat them as equal, and with right support help them. Trust and results will develop through the working process. Regular meeting with team and informational presentations of progress is good way showing results of cooperation and supportive company culture.*

*Filip Kuna, Project Manager, Zoznam.sk*

cheaper compared to doing everything on your own.

**Fight at places where you can win** Invest some time to research, explore market and critical paths, find and compare competitors and most importantly plan the process to success.

**Do what other successful game companies do** Right knowledge reduce risks. Keep looking for skilled individuals or partner companies and if possible try achieve cooperation or at least share experience.

**Be benefitial for your partners** Contacts will be worth in longer period of time. They can in fact multiply sales by many levels. You don't know which contacts will be benefitial, but be fair and generous (as a company) and in time you surely notice results of such behaviour.

**Be consistent and straight for both the team and customers** Don't lie. It is the best impression when team companions notice this, and you no longer have to track what exactly you said to someone. Good reputation in game community brings loyal players, that makes most of the feedback. (ARMA II - 10% of users generated 50% of comments during a year of alpha to beta phase of development.)

**Fight for your team.** Don't complain, find solutions. Don't be shy to ask for advice to a fellow managers. Be always there for everyone to hear individual or company issues and consider their complaints.

*Personal opinion of expert:*

*Keep core idea simple and easy to play. Development of inovative game concepts may be tricky because game mechanics tend to get more complicated than entertaining. It is often useful to recycle assets and generate stuff by code. Knowing your technology well helps too.*

*Matouš Ježek, Designer, Trickster Arts*

# 5 Conclusion

The goal of this thesis was to get acquinted with agile methods and their use in game development. I studied people-related aspects, communication management, leadership with connection of leading a game development team. I explained player involvement into game process and ways of his involvement for testing quality, content and gameplay. I described brief resource management, assigning tasks and time management of the team with given strategies of more effective managing these processes in a role of Project Manager. I studied, compared and noted differences between agile approaches and methods with relation to game development of small agile teams. From third chapter I proposed set of recommendations and best practices used in real projects of top czech leading game development studios, by their knowledge and support. I explained how to handle management, communication and cooperation with their favor and assistance. In chapter Best Practices I noted brief personal expert opinions under the line, collected through interviews, calls, mails, developer sessions and meetings with representatives of game development companies in Prague, Ostrava, Brno or Bratislava.

**Ideas for thesis extension** In a given thesis size I noted what was most accurately referred and the extension of this thesis may concern the preparation of the communication management system with process competence approach to management culture of the agile team.

There are fields of platform development that I was unable to focus deeper. Yet in order to give wider introduction to game development, it is claimed by many people who read this thesis to be very good start, especialy for new Project Managers, Project Leaders, Cooperators, and many more including Developers (or any members of indie team) interested in agile development.

**Author's note:** The making of games in the terms of software should be made with passion, sensation, feelings and most of all with love to the product. I have been collecting knowledge of making games over many years. I did my best to gather most valuable and precious information in comparison to thesis size and content as for role of Project Manager and team Leadership in game development. Empower team members and learn to be part of the team. Give credit, added value and helping hand. Make friends but act fair, objectively, uninsulting, with open mind. Support to the team, for you share the victory and the failure together. When in doubts stick with common reason and work out each difficulty one by one at a time. I learned that respect in communication, active listening, understanding the problems, their solutions and even comprehending the view of others has it's meaning, that everyone can learn, and the best way to do it is to keep trying.

# Bibliography


[1] Meagan VanBurkleo. Portal 2. *[[Game Informer]]*, pages 50 – 62, 2010. 2

[2] Maura Bouça. Angry Birds, Uncommitted Players. *Proceedings of DiGRA Nordic 2012 Conference: Local and Global – Games in Culture and Society*, 2012. 2

[3] A Godoy and EF Barbosa. Game-Scrum: An Approach to Agile Game Development. *IX SBGames (November)*, 2010. 3, 17

[4] Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9:28–35, 2001. 4, 5, 8, 9, 39

[5] A Godoy and EF Barbosa. Game-Scrum: An Approach to Agile Game Development. *IX SBGames November*, 2010. 4

[6] S W Ambler. The agile end game: Catch and release in software development. *Dr Dobbs Journal*, 32:67–69, 2007. 4

[7] R Morien and P Wongthongtham. Supporting agility in software development projects - defining a project ontology. *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies*, pages 229–234, 2008. 4

[8] Orla Mchugh, Kieran Conboy, and Michael Lang. Using Agile Practices to Influence Motivation within IT Project Teams. *Scandinavian Journal of Information Systems Special Issue on IT Project Management*, 23:85–110, 2011. 6, 24

[9] Scott Ambler. Agile Software Development at Scale, 2008. 6

[10] Scott W Ambler. Scaling scrum. *Dr Dobbs Journal*, 33:52–54, 2008. 6

[11] David F Rico. Lean and Agile Project Management: For Large Programs and Projects. *Lean Enterprise Software and Systems*, 65 LNBIP:37–43, 2010. 6, 14

[12] Sean Cohan and Hillel Glazer. An Agile Development Team's Quest for CMMI® Maturity Level 5. In *2009 Agile Conference*, volume 2, pages 201–206, 2009. 7

[13] M. Nasir. A Survey of Software Estimation Techniques and Project Planning Practices. *Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)*, 2006. 8, 12, 21

[14] V T Sarinho and A L Apolinário. A generative programming approach for game development. In *SBGAMES2009 8th Brazilian Symposium on Games and Digital Entertainment*, pages 83–92, 2009. 12

[15] Steven Fraser, Kent Beck, Bil Caputo, Tim Mackinnon, James Newkirk, and Charlie Poole. Test Driven Development. *Test*, 06:459–462, 2003. 13

[16] Jennifer Stapleton. *DSDM, Dynamic Systems Development Method: The Method in Practice*. 1997. 14

[17] Linda Sherrell, K Bhagavathy, and Natasha Velaga. Experiences with Extreme Programming. *Journal of Computers in Mathematics and Science Teaching*, 38152, 2010. 15

[18] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*, volume 37. 2004. 16, 35

[19] J Rodney Turner and Ralf Müller. Communication and Co-operation on Projects Between the Project Owner As Principal and the Project Manager as Agent. *European Management Journal*, 22:327–336, 2004. 16

[20] Dorothy Ann Brenner. Achieving a Successful Project by Motivating the Project Team. *Cost Engineering*, 49:16–21, 2007. 19, 24, 44

[21] Covey Co Franklin. New Book Features The 7 Habits of Highly Effective People Author, Stephen Covey: THE WISDOM AND TEACHINGS OF STEPHEN R. COVEY. *Business Wire English*, 2012. 19, 29, 30

[22] Vibhu Saujanya Sharma and Vikrant Kaulgud. Studying team evolution during software testing. In *Proceedings - International Conference on Software Engineering*, pages 72–75, 2011. 20, 22

[23] Douglas Roby. Teacher leadership skills: An analysis of communication apprehension. *Education*, 129:608–615, 2009. 21

[24] Mike Cottmeyer. The Agile Project Manager, 2009. 22, 25, 36

[25] Kylie A. Peppler and Maria Solomou. Building creativity: collaborative learning and creativity in social media environments, 2011. 22

[26] Charles Margerison. Team competencies. *Team Performance Management*, 7:117–122, 2001. 23

[27] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, and J. Still. The impact of agile practices on communication in software development, 2008. 23, 26

[28] A L LeDuc. Motivation of programmers. *ACM SIGMIS Database*, 11:4–12, 1980. 24

[29] Stephen Covey. The winning principle. *Incentive*, 170:18, 1996. 24

[30] Todor Stoitsev and Stefan Scheidl. A Method for Modeling Interactions on Task Representations in Business Task Management Systems. In *Engineering Interactive Systems*, volume 5247, pages 84–97. 2008. 25

[31] Jessica Xu and Helena Cooper Thomas. How can leaders achieve high employee engagement?, 2011. 25

[32] Anonymous. What makes a self-managed team work? *Supervisory Management*, 39:8, 1994. 25

[33] Ajir Chaturvedi, Ashwani Kr Yadav, and Sumit Bajpai. Communicative Approach to Soft & Hard Skills. *vsrdjournalscom*, 1:1–6, 2011. 26

[34] Joseph Peterson. Imitation and mental adjustment. *The Journal of Abnormal Psychology and Social Psychology*, 17:1–15, 1922. 26

[35] Sharlett Gillard. Soft Skills and Technical Expertise of Effective Project Managers. *IInforming Science and Information Technology*, 6:723–729, 2009. 27

[36] Stacy E Walker. Active learning strategies to promote critical thinking. *Journal of athletic training*, 38:263–267, 2003. 28

[37] Jeffrey Voas and Linda Wilbanks. Information and Quality Assurance: An Unsolved, Perpetual Problem for Past and Future Generations. *It Professional*, 10:10–13, 2008. 28

[38] Delini M. Fernando. *Existential theory and solution-focused strategies: Integration and application*, volume 29. 2007. 28

[39] Richard Mayer E. *Thinking, Problem Solving, Cognition*. 1991. 28, 32

[40] B K Griepentrog and P J Fleming. Shared mental models and team performance: Are you thinking what weÂre thinking, 2003. 29, 33

[41] Paul Loftus. The Pygmalion effect. *Canadian Manager*, 30:1–4, 1995. 29

[42] R Covey. Stephen Covey talks about the 8th habit: Effective is no longer enough. *IEEE Engineering Management Review*, 33:26–26, 2005. 30

[43] Samuel Greiff, Daniel V Holt, and Joachim Funke. Perspectives on problem solving in educational assessment: Analytical, interactive, and collaborative problem solving. *Journal of Problem Solving*, 5:71–91, 2013. 32