Rapid Software Development

Life Cycle in Small Projects

University of Turku

Department of Future Technologies

Software Development

Master's Thesis

May 2020

Eevert Koskinen

Supervisor: Tuomas Mäkilä

University of Turku

Department of Future Technologies / Faculty of Science and Engineering

EEVERT KOSKINEN:

Rapid Software Development Life Cycle in Small Projects

Master's Thesis, 54 pages.

Software Engineering

May 2020

Small software projects are becoming more usual nowadays. Whether a small project is conducted privately or professionally, the management of the project and its phases is much easier with proper tools and frameworks. The research target of this thesis is to find out a proper life cycle model for small software projects.

This thesis is conducted for Softwarehouse, a professional division of IT services in the University of Turku. The official guide for Scrum framework is adhered in software development but when it comes to formally managing various phases of a software project (planning, design, implementation, testing, reviewing etc.) there is room for improvement. Managing software projects with a proper set of tools and procedures would be beneficial as Softwarehouse works on many projects concurrently.

The intended life cycle model has to be formal and heavy enough so that the benefits of agile project management can be received. However too rigid a model can be too arduous and exhausting to use, which could result in the decrease of Softwarehouse's production volume. Therefore the model has to be light enough to maintain rapid software development and creative atmosphere within the Softwarehouse. This thesis begins by giving outline of existing software development life cycle models and followed by relevant literary exploration. After this the research case is explained in greater detail. These give the foundation and rationale to propose a suitable model. The model is experimented empirically and reviewed by partaking personnel. The results are reviewed and discussed. Finally topics for future research are suggested.

Keywords: Software, Development, Life Cycle, Project Management, Scrum, Agile Methodologies

Contents

1 Introduction

When it comes to successful software development, knowledgeable and competent software project managers and development teams play an integral role as they apply practically verified management practices in various project phases.[1] Nowadays smaller software projects are becoming more common in both the public and private sectors. Whether the intentions are to replicate a similar project with existing intellectual properties or to create something from ground up, the management of a project and its various phases is pragmatic and easier with proper tools and frameworks.

The research target of this thesis is to develop a proper life cycle model for small software projects. It is conducted for Softwarehouse, a division of IT services in the University of Turku. Softwarehouse is comprised of two divisions: implementation and research. They provide software, data management and laboratory services under the name of the University of Turku. Softwarehouse is a relatively new and growing concept that has ambitions to steadily scale up their production capacity by incorporating university students in their daily operations. Currently most of the acquired clients of Softwarehouse come from academic circles, but it has potential to expand their business network in the future to attain clients from the private sector.

Currently Softwarehouse works on many projects concurrently and finishes approximately 40 to 60 projects per year. The sizes of momentarily active projects may vary from small websites to larger mobile applications. Most of the times all the actors in the Softwarehouse are working simultaneously in multiple projects with a "criss-cross" manner. Employees do posses proficiency with multiple software languages and technologies. The official guide for Scrum framework is adhered in software development but when it comes to formally managing various phases of a software project (planning, design, implementation, testing, reviewing etc.) there is room for improvement. Managing software projects with a proper set of tools and procedures by devising a rational life cycle model would be beneficial for Softwarehouse. The intended life cycle model has to be formal and heavy enough so that the benefits of agile project management can be received. However too rigid a model can be too arduous and exhausting to use, which could result in the decrease of Softwarehouse's production

volume. Therefore the model has to be light enough to maintain rapid software development and creative atmosphere within the Softwarehouse. Additionally the intended model needs to be capable of accommodating the diverse growing technical competence that exists in softwarehouse. There may be many active software projects that can be technically quite different from each other. In the end this model has to be able to help managing them all with systematical consistency.

In order to create and design a suitable life cycle model for Softwarehouse, the basic principles of generally recognized software development techniques must be understood. In the following section this thesis provides an extensive literary review on waterfall model, agile methodologies and software development life cycle in general. This section helps to identify benefits and disadvantages of certain software development techniques. In the third section this thesis explores and reviews findings on existing researches that are closely linked to the research goals of this paper. Researches that emphasize on swift production benchmarks and smaller projects without neglecting the fundamental principle of legitimate project management are featured in this section. The fourth section focuses on expanding and explaining the research case more elaborately so that rationale and justification for the intended model may be extracted. Based on the observations of the case and the previous literary sections, the fifth section provides the framework and its details for the suggested model. The sixth section is for explaining the research methods and tools for the actual empirical experiment of the suggested model. The idea is to test this model with multiple simultaneous projects. The seventh section is dedicated for discussing the results of the experiment. This section reviews how well the research goals were achieved and provides suggestions for future research.

Currently Softwarehouse's practices of formal project management are in their infancy. This thesis aims to discover a proper project life cycle model, which helps to add benefits of viable project management in daily operations and to manage the items in the project portfolio with systematical consistency. If these aspects are attained and the model does not harm organizational creativity or lower production volume, one can state that the research goals of this thesis were achieved.

2 Software development in general

This section provides a brief overview on generally known software development techniques. In terms of the research target their suitability in Softwarehouses operations is also evaluated.

2.1 Waterfall

The waterfall model was originally introduced in software engineering environments by Winston W. Royce in 1970. The model is a linear sequential development process, where the progress is systematically flowing through a specific list of phases. The model comprises five different phases: Analysis, design, implementation, testing and maintenance. In order to build a functioning software all the phases must be completed in the right order. Additionally, before proceeding to the next phase the preceding phase must be entirely complete. The core principle is that the output of each phase becomes the input for the next phase. In some instances the model also enables the option to revisit previous phases, if the customer requirements are constantly changing. The figure 1 illustrates all the model phases in the right order.[2][3]

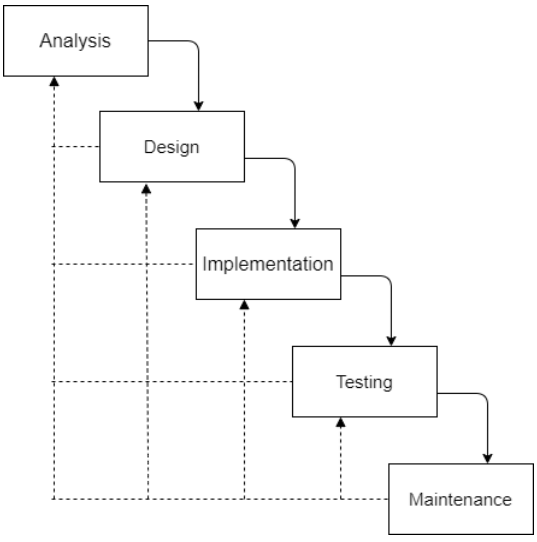


Figure 1: The overall structure of an average waterfall cycle

The purpose of the analysis phase is to extract and identify the core Requirements for the software. The result of this phase is a fully comprehensive

behaviour description for the intended software. Both system and business analysts are deeply involved in this phase. All the functional requirements are determined by defining different use scenarios. According to them, requirements such as purpose, functions, interfaces and database capabilities can be determined. Establishing non-functional requirements is also relevant. They include characteristics such as software constraints and limitations that have a huge influence on design and operations of the software rather than on software behaviour.[2]

During the design phase all the planning and problem solving for the software solution is conducted. Software developers and designers are deeply collaborating to devise plan which includes designs for elements such as algorithms, architectures, database schema and structure, user experience and logical diagrams. All the results achieved in this phase will establish the necessary foundation before the following phase can begin. [2]

The implementation phase represents the part of the project where the actual software engineering is carried out. All the business requirements and technical specifications are fulfilled into a software through the means of programming and deployment. The working and development environments are initialized where the program source code is then authored and compiled. The result of this phase is an executable application.[2]

Testing phase is dedicated for software verification and validation. Software testers check that the software solution meets the business requirements and specifications. During this phase vigorous software debugging is conducted so that various bugs and glitches can be discovered and corrected. Essentially the purpose of testing is to ensure that the results achieved in the implementation phase actually satisfy the goals authored in the analysis phase. [2]

Maintenance phase is the process of modifying already delivered software solution. Actions that are executed in this phase can be about fixing bugs, refining output and increasing performance. Maintenance activities may also include operations, where the software is properly adapted to its intended environment, new user features are accommodated to the software build and reliability of the software is increased.[2]

In practice, software projects have been inclined to encounter various problems and shortcomings due to which there have been significant delays, cost overruns and even utter failures. In most cases, the reason for this is that project managers are not assigning enough personnel or resources for certain activities in the development. Critical phases may be delayed due to insufficient amount of personnel. These delays may result in bottleneck situations, which in turn can lead to a failure in delivering a product in time, within budget or to a specified level of quality. [2][3]

The advantages for waterfall model are the capabilities for departmentalization and control. The schedule, milestones and deadlines for each phase can be easily determined. The purpose and goal of each phase is fairly comprehensible, which makes the model easy to use. The rigidity of the model enables effortless management as the main deliverable and the review process for each phase is very specific. Every phase is implemented one at a time and can not be worked on concurrently with other phases. If the business and technical requirements are very clear from the beginning and the size of project is smaller, the use of waterfall model is well justified.[4]

The main problem with the waterfall model is that estimating the time and cost for each phase is not easy. In some instances If there are multiple problems discovered in the testing phase, it can be challenging to go back and make necessary alterations. If the nature of the project is complex and the requirements have a moderately high risk of changing, the waterfall model is not suitable. [4]

The waterfall model has a huge emphasis on following a specific plan. Every phase in it serves a specific purpose and it is easy to review the success during the project. The goal of this thesis is create a rapid development model for smaller projects. Most of the projects Softwarehouse conducts are smaller in their size. Previously it was established that waterfall model works for smaller projects provided that requirements are clear from the beginning of the project. However Softwarehouse strictly follows Scrum in their daily operations. The agile nature of Scrum framework emphasizes more on learning-oriented approach than plan-oriented approach - there is hardly any rationale or benefits to justify even partial usage of Waterfall.

2.2 Agile methodologies

This subsection provides a literary review on most popular agile frameworks.

2.2.1 Scrum

The Scrum framework is an approach to systems development that is based on defined and black box process management. The performance of Scrum approach in its variants was first witnessed by Takeuchi and Nonaka at Fuji-Xerox. Additionally bigger technology companies such as Canon, Xerox and Hewlett-Packard were among the first to notice its effectiveness. Later on Scrum started to gain a formidable foothold in software companies as they started to notice the benefits Scrum could harness in object oriented techniques and tools. [5]

In practice Scrum is a methodology that can be used in management, enhancement and maintenance of an existing or a prototype system. The core principle is that existing design and source code is constantly being evaluated and reviewed. When Scrum development team plans their product releases, the decisions are made based on the following variables: Customer requirements, time pressure, competition, quality, vision and resource. These variables establish the preliminary plan for software projects. As Scrum is an evolutionary approach by nature these variables are likely to change during the project. Scrum is an enhanced iterative and incremental process that has following phases: Pregame, Game and Postgame. [5]

In the pregame phase all the initial planning for the project is conducted. If the goal is to develop a new system, this phase includes conceptualization and analysis. In the case of an existing system, the analysis is usually limited. Mostly Pregame phase includes activities such as Authoring a product backlog list, definition of project team, estimating costs for the development work and risk assessment. Planning the project architecture is also executed in this phase. The architecture illustrates the designs for how the items in the backlog list are implemented. In some instances there may be more in-depth analysis on backlog items depending on the project complexity. [5]

The game phase represents the actual development phase. It is an iterative cycle of development, which occurs in sprints. A sprint conveys a set of activities that occur in

a pre-defined period of time. The length of the sprint is often assigned by management, and is usually one to four weeks. During this phase the development team performs the following: Develop, wrap, review and adjust. Developing part includes the actual software implementation and testing. However it may include also domain analysis, definition changes for backlog item. The wrap part refers to the part, where the source code with its implemented changes is converted into an executable version. Review part is comprised of activities where the development team meets, presents the work and reviews the progress. This is the time where the team can raise and discuss about various problems. Risks can be addressed and appropriate measures to handle them can be defined. Adjust is the development portion where the information gathered in the review part is processed further into packets. The completion of each sprint is usually followed by a bigger review event. The development team and product management attend this event. The point of this event is to review the implemented changes and evaluate the overall project progress. The changes may be discussed in greater detail and new backlog items may introduced and assigned to development teams. Depending on the project complexity and the desired oversight, this sequence of activities in the game phase can be repeated as many times as it is deemed necessary. [5]

The Postgame phase is dedicated for the project closure. At this point the management team feels that the product meets the defined requirements and is ready for general release by declaring the project as "closed". After this the product is essentially ready for integration, user documentation, training material preparation and other closure related tasks. The figure 2 below illustrates the development flow of this Scrum variant.[5]

The Scrum explained above represents one the earliest versions of Scrum. This model may provide a rough idea on the development flow, but it does not explain the formal roles and events that are very common in modern Scrum frameworks. However, compared to modern versions, the principles of resource flexibility, continuous learning and collaboration do apply in this variant. Nowadays most software companies apply the essence of scrum in software development, but some of the aspects of scrum may have been adjusted to suit their needs better. The Scrum of Scrums, a way to implement cross communication between multiple Scrum teams is briefly explained in the next section.

The official guide of Scrum includes and explains detailed use of sprint events, artifacts and roles. Softwarehouse follows the guide in their operations and will be explained in greater detail later in the research case status section.

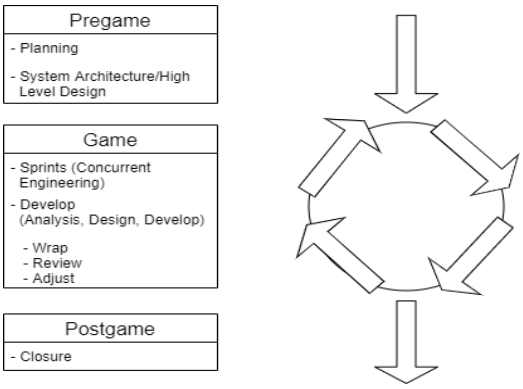


Figure 2: Workflow in Scrum

2.2.2 Kanban

In the 1950s Japanese manufacturing industries introduced Kanban and lean principles. In manufacturing Kanban works as a scheduling system that helps production teams to visualize and manage the flow of the work more efficiently. In software development Kanban was utilized in 2004. Project teams use Kanban to illustrate the workflow, limit the work in progress at each development phase and measure cycle times. Kanban board visualizes all software processes by conveying assigned work among developers, communicating priorities and emphasizing bottlenecks. The core idea in Kanban is to practice lean thinking in development by limiting the overall work in progress, in other words implementing only items that are requested. If executed properly Kanban increases the constant flow of released work items, as developers develop only few items at given times. [6] The figure 3 below shows the structure of an average Kanban board.[7]

Efficient use of Kanban helps team members to understand how their time is being used during development. Usually online Kanban tools provide means to track and gather productivity data, which helps project teams to discover development related impediments. Commonly there are situations where one team member is overworking and another is not doing his share of the work. In this instance the project manager would

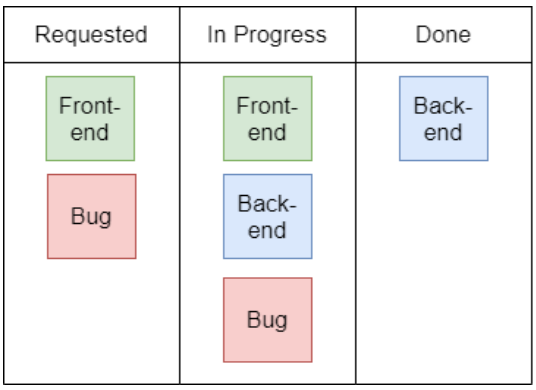


Figure 3: Kanban board

delegate the available members to work on more demanding tasks, which alleviates the pressure from the overall workload the whole team is facing. Essentially Kanban helps to maximize the usage of available resources which enables the possibility to continuously deliver products to clients. The other side of Kanban revolves around the principle of waste reduction. Minimizing excess waste such as over-production, unnecessary motion and defects. In other words the team is not conducting work that is not needed, incorrect or time spent on doing the work with smaller priority.[7]

Most software development experts deem Kanban as an advanced agile tool that necessitates formidable amount of development experience and motivation. Considering the already existing agile working practices and the experienced personnel in SoftwareHouse, the utilization of Kanban could be feasible and viable. In SoftwareHouse the formally agreed practice is the official Scrum. The question whether the proposed workflow in Kanban could co-exist with the chosen scrum framework is not self-evident at this time. However the principle of lean thinking and the reduction of excess waste is worth considering since there multiple small projects worked on simultaneously.

2.3 Software development life cycle (SDLC)

When building any kind of product, it is generally beneficial to devise a model to track and manage the product evolution from its initial conceptualization to its final release - Whatever is the kind or size of the project. A product life cycle model typically has stages to facilitate planning, provisioning, operating and supporting it. The model is

the framework that helps to ensure that the product is able to meet all the requirements throughout its life. Another beneficial aspect of utilizing a proper life cycle model is to ensure that the project organization thinks its operations within a larger framework, which helps to manage all existing projects and overall business goals. The figure 5 below illustrates a typical life cycle model with some of the possible progressions. All the stages have a distinct purpose and contribution to the whole life cycle. Organizations employ stages differently to satisfy their distinct business strategies. [8]

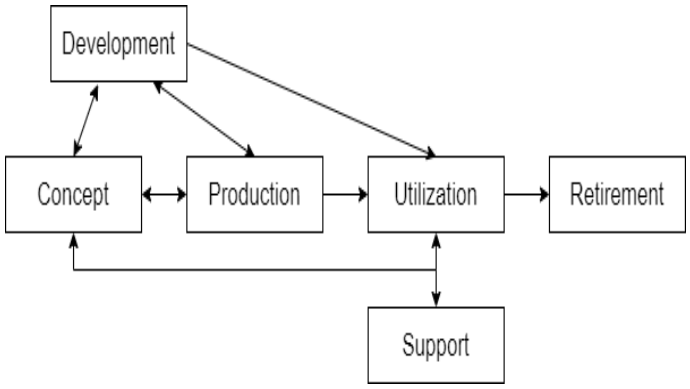


Figure 4: Stages in SDLC

Content management for project life cycle allows a project organization to enable a collaborative environment. Additionally component content management supports the reuse of content objects. In terms of producing high-quality software, documentation is or at least should be an important part of product life cycle. The international standards for managing technical content includes storing elements such as user information, product life cycle information and service management items. Content management activities can be divided into five main modules. The figure 6 below shows the modules and example activities among them.[9]

In project initiation the project organization authors a business case to support the development and implementation of a content management solution. This is done by evaluating certain organizational needs: Analyzing the current state, identifying potential customer benefits and cost reduction opportunities, calculating the cost for needed technologies and personnel and conducting risk assessment for the project. Essentially multiple benefits such as standardized levels of product quality, reduced maintenance and development costs and possibilities to trace cycle reviews and approvals better.[9]

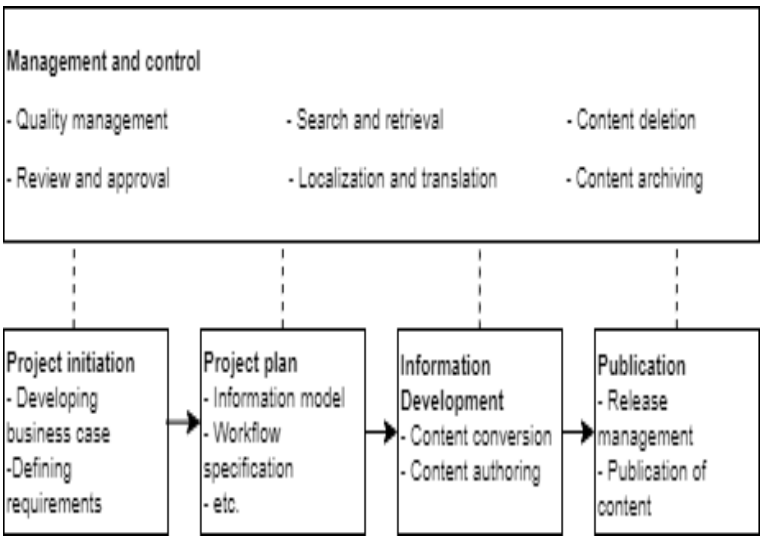


Figure 5: The overall structure of the proposed life cycle model

In terms of efficient project management, A proper software development life cycle can help to deliver a project with reduced development costs, higher product quality and shorter time. Essentially it can help to attain highest level of management and documentation. Everyone in the project organization understands what product should be built, why it should be built and the best way to build it. However multiple setbacks and failure to understand SDLC framework can turn the product implementation into a disaster. Organizational commitment and precise execution are prerequisites to utilize SDLC. [10]

This section has explained in different ways how software development can be done. More or less every development framework has its own distinct features and nuances that defines it. Considering the development flow and the different stages in each of them, they all share lots of similarities with the generally recognized SDLC framework. Most organizations use these frameworks but in most instances they have designed and altered them significantly to suit their needs better. In the case of SoftwareHouse, the intended life cycle model should be able to increase the benefits of project management and progress tracking while sustaining the current production volume. These explained frameworks create a foundation for the proposed model. However like many other Software firms, Softwarehouse has its distinct qualities, which need to be considered when designing and/or adapting a suitable model.

3 Existing research

This section goes deeper into the research goal of this thesis by exploring and reviewing researches with relevant topics. Emphasis is on literature that focuses on rapid production, small project organizations and organizational creativity. As agile methodologies play a huge role in thesis, this section also briefly explains examples of the pragmatic utilization of agile.

3.1 Rapid production

As the competition in the era of e-business is rising, firm's ability to implement and deliver high quality software on time is becoming a relevant metric to consider. In general software development is known to be questionably slow and inefficient due to communication delays and breakdowns. Firms are trying to come up with game-changing ways to produce software to stay ahead of the competition. Typically these new methods experiment various practices such as quality management approaches, automated software design and usage of different development tools. In spite of these new ways firms are still unable to manage their projects with necessary quality.[11]

In large software projects, developers spend most of the time in communicating with or looking for information from other team members. Past studies suggest that commonly developers use less than 30 percent of time in coding during projects. The rest of the time is used in meetings, problem solving and resolving misunderstandings with customer requirements. Additionally communication breakdowns can constitute a significant time sink. The reason for this usually the fact that development related tasks are carried out by different groups. Physical separation and grown distance between team members is proven to increase the probability of diminished communication. Teasly's research suggests that strong team collocation through the concept of War Rooms can mitigate this issue efficiently. In war room, everyone involved in the development such as experts, developers and managers conduct their work in a specific space that is without outside distraction. War room promotes the idea that all the essential resources are centralized and there are no communication breakdowns. The

figure 7 shows an example layout of a "war room".[11]

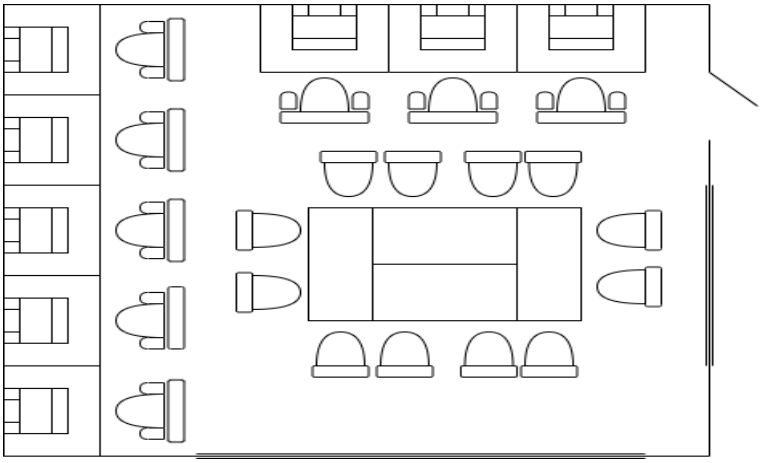


Figure 6: War room

The war room concept was tested with six different projects, which had the development lifespan of 24 months. The company who delivered the projects utilized a waterfall variant called fusion. Additionally the company adopted multiple features from IBM's Rapid application development methodology. Case projects were scoped with timeboxing, which attempts to maintain time and personnel in constant values and emphasizes the importance of implementing the most desired customer features. The test results showed that heavy collocation of development teams increases productivity and interactive continuous communication, which enables overhearing and awareness of other team member's activities.[11]

Strong collocation alone may be a simple way to ensure more efficient communication and help delivering the working application in shorter time.[11] However software firms should also consider adopting appropriate tools to attain rapid production levels. To this date, rapid application development (RAD) methodologies still do not have that high of a reputation. There are a number of ways to implement RAD. Some of them are derived from dynamic systems development method (DSDM), which focuses on issues such as project management, quality assurance and testing within rapid metrics. DSDM has five different aspects: Development process model, set of techniques, documentation, the link between documentation and techniques and the philosophy. [12]

Principles of Joint application design (JAD) seem to appear in different RAD methodologies. The core idea in JAD is that users and developers engage in deep collaboration that spans the entire project life cycle from initial planning to final product release. Arranging time for highly organized and detailed-oriented workshops is a relevant part of JAD. These workshops are often conducted in conjunction with with other analytical methods to improve and enhance systems specification activities. Additionally workshop are expected to be held at clean rooms, where planning and can be conducted without needless distractions and interruptions. All team members must have great communication skills in terms of business. Additionally Users must have knowledge in the application area. Documentation is an important part of JAD - Development teams are expected to author fully documented business and technical requirements early in the project. [12][13]

Time is a prominent feature in RAD. In most instances the key features of the intended system can be implemented in 20 percent of the time required to build the entire system. Due to this the concept of timeboxing has become a standard way to evaluate and specify system priorities. Basically if there is a huge risk that the deadline will be missed, lower priority requirements will be moved to a later time box. These priorities may appear in later timeboxes and development iterations. Timeboxing ensure that the development team is able to deliver the system within a fixed time span before the business environment has changed. Essentially timeboxing helps development teams to maintain focus and chronological control of the project. The figure 8 shows the relation between timeboxes and user reviews.[12][14]

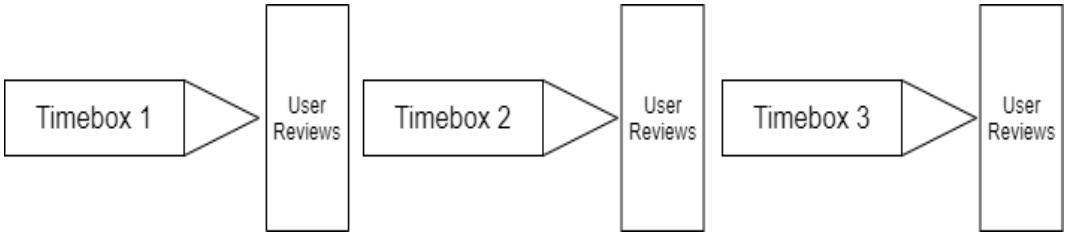


Figure 7: Timeboxing

Incremental prototyping is commonly associated with RAD methods. In prototyping the initial research and development are conducted simultaneously to create a prototype, which may or may not evolve into a final product. The prototype should

include a database, majority of its modules, user interface and information flows for interfacing systems. Prototyping is essentially a process of building a system in an iterative way. Once a prototype is finished, it is tested, reviewed and discussed to determine whether it is actually turned into final products. This cycle of inspection-discussion is usually repeated several times during a RAD project. The figure 9 shows how the prototyping may occur in development.[12][15]

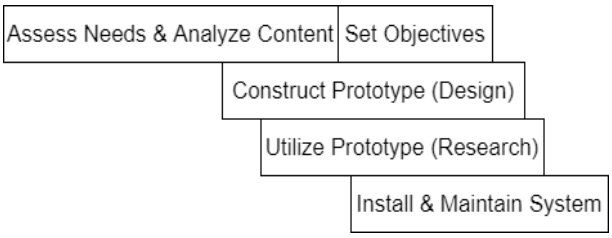


Figure 8: Prototyping

Modern RAD approaches often require support from tools. Even though the goal is to produce software in a rapid manner, software reliability can be achieved with various statistical process-monitoring tools and designed experiments. In some instances effective graphical user interface builders and computer-aided software engineering tools can both expedite and maintain the quality in the development. There are studies that verify the effectiveness of the heavy usage of these kinds of tools.[12] Most of these advanced tools can be quite strenuous to use and require knowledge in data management. Considering the the limited research goal of this thesis and the current state of already used tools in Softwarehouse, A detailed exploration and review of these tools is excluded from this paper. Examples of these tools can be found from these references. [16][17][18]

Many of the principles of DSDM apply in RAD approaches. The highest focus is on delivering products frequently and highlighting the importance for business purpose. In other words delivering a product that has necessary features within required time is the most relevant aspect to consider. Highest level of communication and collaboration between all stakeholders is vigorously adhered and monitored to prevent misunderstandings and development breakdowns. Iterative and incremental development helps to extract and specify the appropriate business solution. The iterative development is usually backed with integrated testing throughout the project life cycle. The suggested life cycle model for DSDM is shown in the figure 10. The first phase represents the

initial feasibility study, where both technical and business requirements are evaluated so that the appropriate RAD approach can be chosen. The second phase is the high level business study. In the third phase, building a functional prototype that demonstrates the biggest requirements is the primary goal. The actual end system design and building of the end product takes place here. Basically the prototype from previous phase is refined further. The project closure and the handover to users is conducted in the final phase. In most instances the success of the project is also evaluated here.[12]

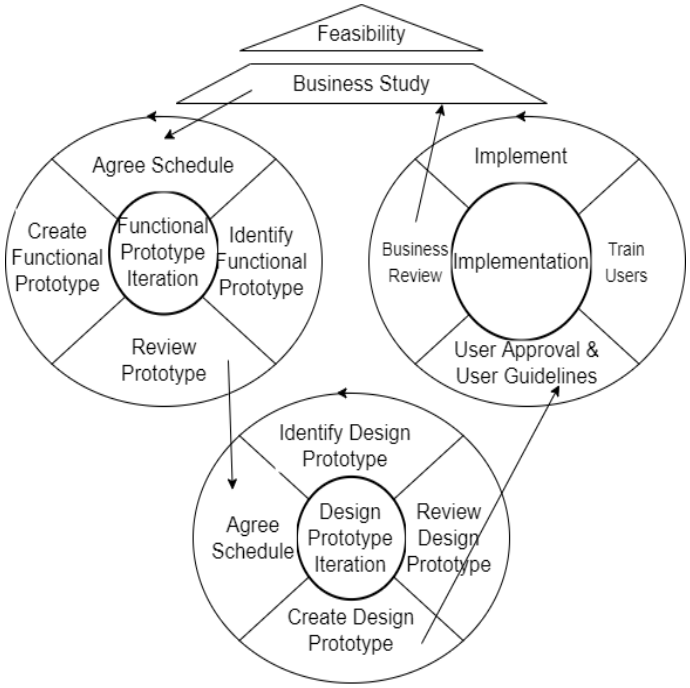


Figure 9: Dynamic systems development method

3.2 Small project organizations

The growth of the software industry has generated many small companies. Much like bigger companies these smaller companies are developing and producing significant software solutions for both internal use and commercial applications. Fayad [19] suggests that smaller-sized organizations need effective and tailored software engineering practices. Metrics such as development speed, mode and size need to be considered when designing and implementing appropriate working methods. It is difficult to estimate cost figures for development. However it can be stated that methods for developing large-scaled contract-based systems do not serve adequately companies

with less than fifty employees. Specialized forms testing and development are prerequisites for effective distribution and feature updates in large companies. Smaller companies tend to have much more specific needs and they are combined with the challenges of frameworks and component integration.

As mentioned before, new software companies are started every day. Some of them fulfill the definition of a software startup, which is a popular way to start developing an idea into a successful business. Emerging and accessible technologies such as cloud platforms and web development tools have made it relatively easy and quick to to get started. The popularity and allure around startups are mainly inspired due to countless success stories such as Facebook and Twitter. In many instances, embracing lean thinking and customer focused development have been suggested as valid ways to operate a startup.[20]

Generally startups have limited resources in terms of personnel and funding. This combined with rapidly changing circumstances and strict schedules can make running a startup difficult. Usually in the beginning a startup is platform for exploration, which means that business models, customers and overall product requirements are virtually non-existent. However it is important to be efficient and systematic. This can be done simultaneously maximizing value gained and minimizing the effort in development. To address all the challenges when finding a product idea worth scaling, Bosch and her partners have suggested the early stage software startup development model (ESSSDM). The model offers an operational process framework that highlights the importance of lean thinking and assists in decision-making. It defines a step-by-step process with clear exit criteria for ideas and guides what practices and techniques to employ in the different stages. In essence the model helps to initialize a product backlog of ideas, maintain and review the backlog items, prioritizing and validating ideas through a funnel and abandon ideas with no proper rationale or prospects. The figure 11 shows the overview of ESSSDM.[20]

Software process assessment models such as Capability maturity model and ISO 9001 are not suitable for small software companies. Studies suggest that smaller companies want to improve product quality and development processes but are having problems with finances, technical domains or organizational restrictions. Sometimes the

company may have difficulties in forming an internal dedicated process improvement group. Financial problems are often encountered when managers are trying to allocate resources for quality groups and consultants. Technical issues may surface when the company attempts to incorporate of modern development that necessitates personnel training. Nunes and Gunha [21] have developed Wisdom method to address the specific needs of smaller software companies. With this method, the development teams are able to build and maintain systems that adhere high quality standards and processes.

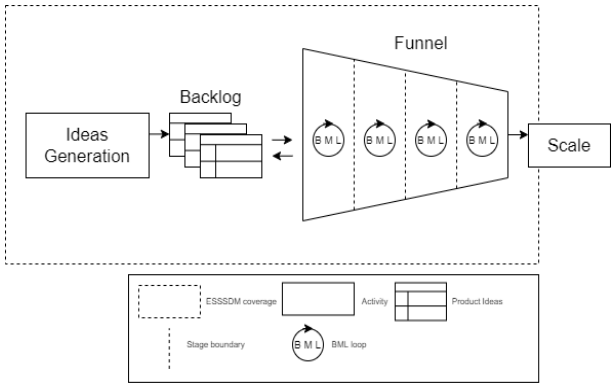


Figure 10: Early stage software startup development model

The core foundation of wisdom was inspired by small companies that were developing software with extremely specific and/or chaotic ways. Having chaotic ways of working does not mean that the company is failing. Despite of chaotic nature, various spiral models and iterative development methods have been proven to be effective. Commonly the principle of "just do it" is associated with chaotic development ways. Wisdom embraces this and promotes desirable aspects such as speed, flexibility and good communication. Wisdom has three central components: process based rapid-prototyping model, a set of conceptual modeling notations and a pragmatic philosophy to manage projects.[21]

The iteration concept is very important in wisdom. The development team states prototype objectives and reviews them at the end of the iteration. This activity provides a sense of completion and control among managers and developers during the project. In the beginning Understanding the teams development pace and assigning appropriate duration for iteration is crucial for success. However later on Wisdom helps increasing the workloads and lengths of iterations. In other words the development team's overall

pace of production will improve and the organization is able to operate multiple product-development life cycles in parallel.[21]

Smaller companies are inclined to benefit from diagrammatic and model-based approaches as they help to specify and document developed software systems. Creating and managing documentation becomes an inherent part of the development process and is not considered to be a supporting activity. In Wisdom different models emerge from participatory sessions and are supported by wrap-up sessions, where the required formal specification is authored. Wisdom does not necessitate usage of direct tool support such as Case tools. Wisdom could benefit from usage of separate modeling tools but their presence in production would in fact hinder the aspects of rapid-development tools. Separate modelling tools do not support process management activities such as task prioritization and progress measurement. With a light-weight approach, wisdom intends combine process management tools with modeling tools, which reduces the need of using additional tools.[21]

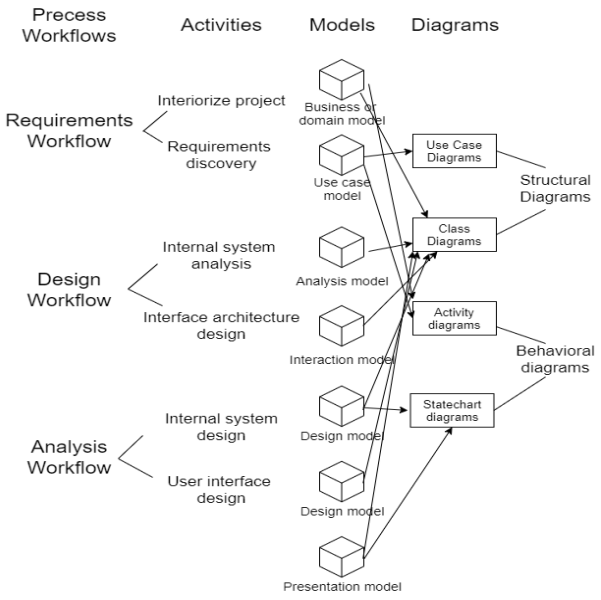


Figure 11: Wisdom workflows

The wisdom notation is a subset of the UML, which is the standard language for visualizing, specifying, constructing and documenting software system's artifacts. UML is comprised of 233 of different basic and diagram model concepts. Figure 12 conveys Wisdom's four major workflows and diagrams. Wisdom is based on seven models and uses four types of diagrams. The models and diagrams are carefully selected

to support the interactive aspects of software systems. Requirements workflow aims for the the development of systems that actually satisfies the client and the end user. The analysis workflow refines and structures the requirements in the requirements model. The design workflow pushes the systems to implementation by refining its shape and architecture.[21]

3.3 Organizational creativity

Richard Woodman, John Sawyer and Ricky Griffin [22] explain in their article that organizational creativity is virtually the creation of a useful new product, service, idea or process in a complex social system. It directly correlates with innovation whether it sparks a new product or improves existing ones. Creativity is represents a complexity of an individual's behavior in a given situation. Contextual and social influences describe various situations and they can either facilitate or hinder individual accomplishments. Factors such as preceding correlations, cognitive style and motivation together combine a function that makes up individual creativity.

Farida Rasulzada and ingrid Dackert [23] convey in their article that psychological well-being of a person is related to organizational creativity and innovation. New challenges and personal growth are often the products of Creativity and they are likely to make individuals feel happier, enthusiastic and optimistic. Various organizational factors such as team climate, available resources and implemented leadership can enhance or reduce creativity. Positive team climate supports innovation, makes individuals feel safe and fosters a vision that motivates work groups. Perceptions of abundant resources alone can make people more engaged and inclined to deliver creativity. Leadership is an important influence as it is well-established that a transformational and charismatic leader invokes more intellectual stimulation in employees.

3.4 Agility at present

Agile practices were originally intended for small collocated teams. Nowadays larger companies utilize scrum by employing multiple teams that are distributed to several

locations. In order to scale Scrum effectively in larger and dispersed software development operations, the use of a method called Scrum-of-Scrums (SoS) has increased greatly. SoS is essentially practice that enables possibilities for inter-team coordination and collaboration. Like with regular Scrum, SoS uses the daily Scrum meeting, except that it deals with teams instead of team members.[24]

Many organizations have benefited from applying lean and agile practices at the team level. The Scaled Agile Framework (SAFe) is a popular knowledge base that has effectively grouped together integrated principles for lean and agile. In practice SAFe extends the agile team by creating a cluster of agile teams. These teams together form an agile release train (ART). In addition to different configurations and implementation roadmap, SAFe highlights the relevance of role based training. To implement SAFe, the target organization must truly understand the underlying lean and agile values. The Agile manifesto influences the agile foundation of SAFe. The four values of the manifesto are illustrated in the list below.[25]

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

List 1: The core values of agile manifesto.

4 Research case status

In this section the research case of this thesis is explained in greater detail to extract the rationale for the intended model. This is done by reviewing current state of working methods, tools and practices in project management.

4.1 Overview

Software House is a division of IT professionals and researchers from various fields of science. It is comprised of two sections: implementation and research. They provide software, data management and laboratory services for academic researchers and research groups. Softwarehouse's business network does not possess typical characteristics and is essentially a complex. It includes its own project groups and other actors like customers, researchers, other IT-teams of University (server and middleware) and sometimes external IT-suppliers.

Softwarehouse works on many projects simultaneously and completes approximately 40 to 60 projects per year. Smallest software projects are usually simple websites that have a lifespan of one to two weeks. On occasion Softwarehouse also has larger well funded projects, which can last several years. An example of this kind of project could be implementing a complex analytic platform that supports a large research undertaking. In terms of project portfolio management, Softwarehouse has always at least 5 active projects, which are managed and implemented by 8 developers. There is only one software tester with whom Software House collaborates. Due to this, the development iterations and project release dates need to be scheduled and planned according to software tester's timetable. At the time of this thesis there was no one appointed as the head of Software. Managing the project portfolio of Softwarehouse was included in this role. To ensure formal accountability of managed projects, a head of project management was appointed as a temporary solution.

Softwarehouse has been working with Scrum for a long time and follows strongly its guidelines in every project. In certain distinct instances Softwarehouse has also used Kanban, if there is a case, where a finished project requires maintenance or

needs to be developed further. In spite of using certified software development frameworks, practices in formal and higher level project management are still in their infancy. There is no systematic way how smaller projects are initiated, managed and finished. Based on unconventional project management and personnel dynamics, one could state that Softwarehouse's operation style is a bit chaotic by nature.

4.2 Scrum

Scrum is a framework for developing, delivering and sustaining complex products. Schwaber and Sutherland [26] have explained the definition of scrum in this guide. The following subsections explain how Softwarehouse operates with the official Scrum methodology.

4.2.1 Scrum team

The Scrum team is comprised of a Product owner, the development team and a Scrum master. Teams are self-organizing and cross-functional. They determine how to best accomplish their work and have all the needed competencies to complete the work without having to rely and answer to others, who are not part of the team. The team structure is designed to Optimize flexibility, creativity and productivity. Teams implement and deliver products incrementally to ensure that the customers have access to best possible version of the product at all times.[26]

The product owner's job is to maximize the value of the product. Product owner manages the product backlog by ordering the backlog items, ensuring the development team truly understands backlog items and optimizing the value of the work the development team performs. The product owner is one person who is accountable for the work mentioned above.[26]

The development team consist of competent individuals responsible for carrying out the work and delivering a functional version of a product after each sprint. Development teams are empowered to manage their own work. In Softwarehouse the size of the development team often varies depending on the size of the project. All the individuals in the team may have specialized skills and areas of focus, but accountability

for the work belongs the entire team.[26]

The Scrum Master is responsible for making sure that the entire Scrum team follows rules and constitution of Scrum as defined in the official guide. This is done by promoting and helping everyone understand the theory, practices, rules and values in Scrum. Depending the on level of team's experience, Scrum master has lots of various activities that serve the specific needs of product owner, development team and the firm organization. For example Scrum master ensures the Scrum events are organized and held whenever they are needed, helps product owner find the best ways to manage product backlog and removes impediments to the development team's progress. Softwarehouse aims to have a dedicated scrum master for every active project, but in smaller projects the scrum master role can be deemed unnecessary.[26]

4.2.2 Events

Scrum framework is known for its prescribed events, which are designed to create regularity to the workflow and minimize general misunderstandings. All the events have a specific duration. However apart from sprints, all the events may end whenever the purpose the event is achieved. This ensures that the appropriate amount of time is spent without allowing needless production delays. These events are designed to give a formal opportunity to inspect work and enable transparency.[26]

The heart of Scrum is a sprint, which acts as a general container for all the other events. The length of the sprint is always fixed and can not be modified once the sprint has started. The duration of a sprint can be one month or less. In softwarehouse the sprint duration is two weeks by default in every project. Once a sprint ends, a new sprint starts immediately. Each sprint may be considered a project and are used to accomplish something.[26]

Sprint planning is the event where all the work for upcoming sprint is reviewed and decided. Generally sprint planning answers, what can be done in this sprint and how the chosen work will be done. The entire scrum team participates in this events, so that everyone understands the work of the sprint. Sprint goal is defined during planning. For example it can be set of backlog items that need to be implemented during the sprint.[26]

The daily Scrum is a 15-minute event held at every day of the sprint. During this event, the development team plans the work for the upcoming 24 hours. It is basically a tool to inspect the work since the last daily Scrum and forecast the upcoming work. In Softwarehouse, the daily scrum is structured around three questions: What did I do yesterday, What will I do today and Do I have any problems with my current work. The daily scrum essentially helps to improve communication and to review work progress towards the sprint goal.[26]

The sprint review is held at the end of the sprint to inspect the finished product and adapt the product backlog. Sprint review a mean for all the project stakeholders to collaborate about the word conducted during the sprint. Depending on the project the sprint review duration can vary from 1-hour to four-hours. In softwarehouse, the sprint review includes at least the following elements: The product owner explains what backlog items were completed and what were not completed, The development team demonstrates the completed work and entire team reviews the project timeline and budget. Additionally the entire team discusses what do next in the project.[26]

The sprint retrospective is an event that is used to inspect the team and its performance. In softwarehouse, the retrospective is used to review the sprint performance in terms of people, processes and tools. The main goal is to identify what went well and potential improvements. Additionally a plan to implement improvements may be devised.[26]

4.2.3 Artifacts

Scrum's artifacts represents the work or value to provide transparency for inspection and adaptation. The product backlog is a list of every item that is known to be needed in the product. It is essentially the primary source of product requirements. The sprint backlog is the set of product backlog items that are selected for the sprint. In softwarehouse, this is often combined with a plan to deliver the working product of the sprint and to realize the sprint goal. Scrum relies on transparency. The decision to optimize value and control risk are made based on the current state of the artifacts. The definition of done is a term used to describe whenever a product backlog item or the working product of the sprint is

completed.[26]

4.3 Workflow with Scrum

In order to implement Scrum properly, all the necessary events must be held and all the textbook artifacts must be managed. The figure 12 below illustrates the approximate life cycle sketch for most of the projects in Softwarehouse. The cycle begins from assigning a project from product portfolio. After this all from Sprint planning to Review and Retrospective are kept according to the official Scrum guide. Software testing usually occurs after a sprint to ensure that the deliverable reaches a certain level of quality.

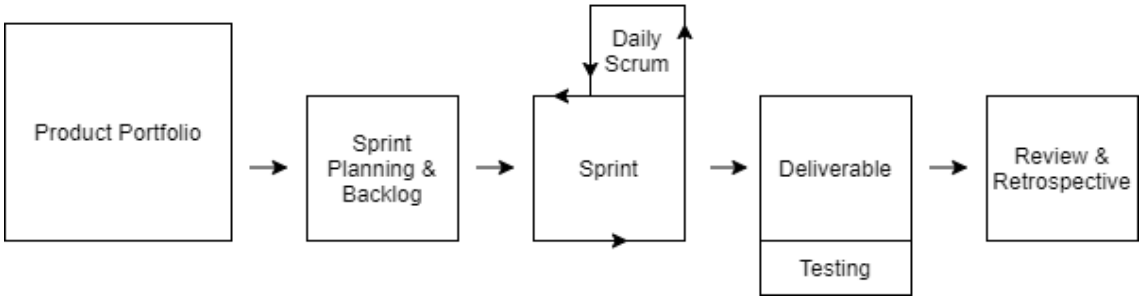


Figure 12: Scrum Workflow

4.4 Tools

Softwarehouse uses Atlassian Jira project management software to manage and schedule their workload in this project. For Softwarehouse's needs, Atlassian gives extensive means to drive a project with Scrum. All the central activities such as managing the backlog, starting a sprint and reviewing progress through comprehensible burndown charts can done with Softwarehouse's Jira license.[27] Softwarehouse uses also slack for project communication and messaging. All the active projects have their own specific slack channels to keep conversations focused and teams more aligned.[28] In terms of electronic tools, Softwarehouse is currently not particularly inclined to add any more tools to their development practices. However they are looking for feasible ways to improve their software testing practices - Incorporating a light-weight testing tool could

be valid way to improve testing.

4.5 Project management

Software House gently follows a discovery-oriented project management style which lets the organization work independently under the parent organization. It has a separate internal structure and processes to create and develop products with uncertain outcomes.[29]

4.5.1 Resource and personnel management

Project management requires versatile personnel management skills. Project team consists of different kind of people which in many cases represent different cultures and work from different parts of the world. Successful teamwork requires strict rules, communication, stimulation and constructive feedback. A skillful project manager knows how to exploit individual skills for the benefit of the group. [30]. Software House's personnel management is open and trustful which is typical for scrum management. Responsibilities are divided between team members and each of the members had their own responsibility areas. The part of Software House that conducts the research, evaluates and investigates projects feasibility and then adds them to the project order database. This database is constantly monitored so that the right amount personnel from the implementation section can be assigned to projects as early as possible. However it is worth noting that most of the projects in Softwarehouse do not have a dedicated project manager. The overall accountability of every project falls solely on the Head of softwarehouse, who admits that there is too much work all the time.

4.5.2 Risk and uncertainty management

In software development, risk can be defined as a potential problem that may delay or even hinder the development progress of a project. Risk management means risk containment and mitigation. This can be done by identifying and planning. Additionally risks can also be classified and prioritized. If a risk likely hood and its potential impact is high, the responsible team may devise a separate plan to avoid and mitigate the risk.[31]

Risk management in Softwarehouse is mostly based on work amount estimates that are determined by evaluating the level of challenge within the project and the needed technical know-how to finish the project. As most of the projects are small and can be executed swiftly and fluently, it is often deemed unnecessary to conduct vigorous risk management. If a project has complex or uncertain elements, they are identified early on. Steady and open communication is Softwarehouse's primary strategy in risk prevention and mitigation.

4.5.3 Project scope and estimation management

In software development it is often hard to answer the question: How much work will it take to deliver a new product? Cost estimation is inherently difficult as it is difficult for humans to predict absolute outcomes.[32] The visibility of milestones is a critical feature in highly singular projects.In cases where the project is quite vulnerable for delay and even failure having visible milestones is especially important.[33] Currently scoping and estimation is closely linked to existing risk management. The level technical challenges and uncertainties serve a basis for story point estimation. Additionally during sprint planning, the outcomes of previous sprints are used as a baseline to evaluate the effort needed to complete tasks. Discussion is mostly the primary and only mean to assign the amounts of points to each task. Softwarehouse could use common agile estimation techniques that utilize visual aids - this could make planning activities more sensible and entertaining.

5 Suggested model

This section describes the framework and its details for the suggested model. This is done by explaining design considerations followed by model descriptions and analysis.

5.1 Design considerations

There were significant factors and issues to consider when designing the model. Firstly the main purpose of the model was to increase formal project management in Softwarehouse's daily operations and to enable consistent management of Softwarehouse's small projects. Secondly, the suggested model could not be too strenuous to use - Maintaining Softwarehouse's production volume, organizational creativity and team specific autonomy were important aspects to take into account.

In terms of lower level details, the model was supposed to be built around the official Scrum framework, due to which the model needed to incorporate key scrum events such as sprint planning and sprint review. The quality management in Softwarehouse is determined by Scrum's Definition of Done. This means that all the implemented features are properly tested before marking them as truly done. Currently the considerable factor is that there is only one dedicated software tester with whom the work has to be planned and scheduled. The head of softwarehouse's project manager asked, if the suggested model could aside from traditional j-unit tests, incorporate more testing in development iterations. Additionally ways to improve initial resource planning and schedule estimation were also requested.

5.2 Model descriptions

With the design considerations in mind, the suggested model is conveyed in figure 13. It shows chronologically all the significant phases and steps from initiation to project closure. Like the original Scrum framework introduced in the second section of this thesis, this model groups its activities and steps in three phases: planning, development and wrapping. It also indicates all the relevant roles attached to every step. Overall the whole figure represents a more detailed, embellished and nuanced model compared to

the previously presented life cycle model in section four. Like with the waterfall model, each stage and internal step has a specific purpose. The model takes into account all the necessary scrum events and artifacts. Continuous iterative nature is accommodated by including four different development steps. Compared to existing work flows in Softwarehouse, the usage of this model introduces three significant additions to daily operations. These are highlighted with the color of red in the figure. Firstly after initiation comes the Kick-Off and planning step. Secondly there is added monkey testing with the box m. testing. Thirdly there is increased project management with the box risk management and cost estimation. Additionally the arrows around development steps symbolize a newly formatted style for development.

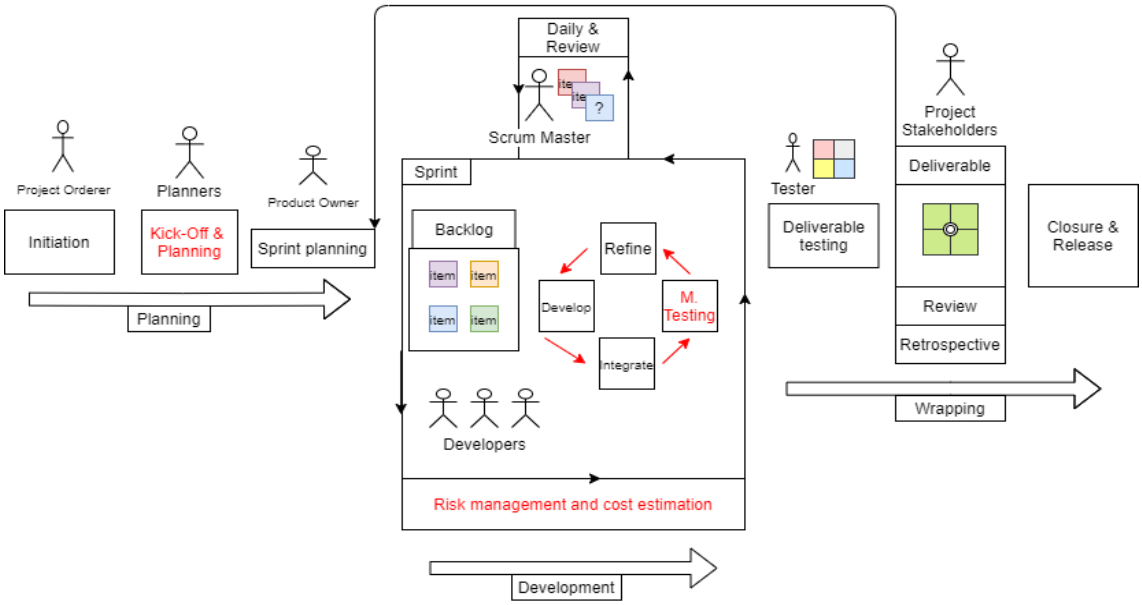


Figure 13: The overall structure of the proposed life cycle model

5.2.1 Planning phase

The first step in the planning phase is the initiation. This where the head of project management orders a project and assigns a group of individuals for the project planning. The Kick-off and planning step follows next. This is where the assigned individuals meet for the first time to discuss about the project. They talk about the general aspects of the project such as project difficulty and used technologies. Additionally the product owner

is assigned and the initial product backlog is authored.

Aside from agreeing of all the details, conducting resource planning and management is also a relevant part of this phase. As Scrum sprints have a fixed length, the principle of time boxing is already used. However this model introduces the importance of resource boxing. This means that the planning team will choose the right amount of personnel for the actual development. In order to do this, the team will estimate and choose the right amount of story points for every item in the backlog. This can be done with lots of different ways such as Planning Poker and T-shirt Sizes. Based on the total amount of story points for the sprint, the planning team will choose the right amount personnel, which which will form the scrum team.[34][35]

Moving on to sprint planning, the scrum team chooses all the items for the sprint and plans how they are implemented. Depending on the estimated amount story points in the previous phase the team will choose an appropriate schedule estimation and review technique for the upcoming work. In standard projects with enough diversity and complexity, a traditional burn down chart is a valid way to estimate the work progress. However in very small projects, a burn down chart may be too cumbersome and heavy to use. As previously verified, Softwarehouse has on occasion used Kanban to conduct projects with smaller scopes. For these instances, the team might consider using a cumulative flow diagram.[36] Compared to traditional Burn down chart, a Cumulative flow diagram has better interoperability with Kanban technique and is a lighter way to review cycle times, throughputs and work in progress. After agreeing on the schedule estimation technique, the team will continue with the normal agenda of sprint planning.

5.2.2 Development phase

The development phase is the portion where the sprint cycle and the actual development takes place. The primary actors of this phase are the development team and the Scrum master. The sprint backlog is the primary source of all given tasks and the work progress is reviewed in daily scrum events - Scrum master makes sure that is properly done. The tendency to develop, integrate and refine is quite typical a chain of activities in software production. As an added element, the purpose of M. Testing is to increase testing in

development. Conducting J-unit testing is already a standard procedure in feature integration, but M. testing means adding monkey testing in development. Considering that the intended product is not large, a sub variant of monkey testing called smart monkey is used. This means that in addition to developing, the developers will on occasion conduct smaller scale testing with the mentality of knowing the purpose and functionality of the system. Essentially the developer navigates through the system and gives valid inputs to perform random tasks. This is done in the interest of finding bugs and errors without predefining any specific test or user cases. The durations of monkey testing sessions should be kept as short as possible.

The box below the development sprints, indicates all the added risk management and cost estimation for the entire project. The development team has the complete autonomy to choose how this is actually done. There can be small separate meetings. However since there formally no project specific managers in Softwarehouse, every individual in the development team should practice this whenever it is possible. Risk management should have the following process flow: Identification, planning and monitoring. This applies to risks that could affect either the current sprint or future sprints. Additionally cost estimation practices are added to determine whether the development team could alter the sprint backlog either by eliminating current tasks or adding new tasks during the sprint. Altering the size of the sprint backlog should not be done unless everyone including developers, Scrum master and the head of project management accepts it. In the interest of keeping this model as light as possible, the documentation can practiced whenever deemed necessary.

5.2.3 Wrapping phase

In the final phase of the suggested model, the project is seen through the finalizing work. The testing step is the final measure for product related quality assurance. This is done by the tester. Provided that every implemented task is truly done, there will be no need to revisit them again in upcoming sprints. If everything goes according to plan, all the implemented tasks and features together comprise a fully operational deliverable. Naturally if there are significant problems and very limited time, these problems will be discussed in the sprint review to determine appropriate plan of action. All the relevant

stakeholders including product owner, Scrum master and the development team by minimum should partake in the sprint review. Sprint retrospective should be kept right after the review. Depending on the discovered findings and authorized decisions, the project can proceed either to closure or sprint planning after which another development iteration begins.

5.3 Model analysis

The quick glance of the suggested model would imply that the model shows exactly how modern day scrum currently works. In the first planning comes the added concept of resource boxing, which is done with the aid cost estimation games. This is overseen by the selected planning committee, who will based on their findings, assign the right amount people for the project and form the actual development team. Moreover in the sprint planning comes the alternative schedule estimation tool, which is allegedly in smaller projects a better option compared to standard burn down chart. These concepts will ensure a higher probability that the project development is started with the right amount of resources and proper work schedules.

This model introduces smart monkey testing in development iterations. As the team has the complete autonomy to plan and conduct their work, the team can choose appropriate times to conduct a small monkey testing sessions. In consideration to smaller scale operations of Softwarehouse, Smart monkey testing provides added quality control that is easier and cheaper to execute compared to traditional testing. Theoretically Monkey testing could limit future work and minimize software tester's work. However due to its random nature, it could also be improper use of developer's time, which is why developers should aim to keep monkey testing sessions as short as possible. Monkey testing could also be automated with dedicated techniques and tools, but there is currently very limited access to proper tools in the open market.

Scrum framework itself acts as an effective way to oversee project management. According to Scrum's values, the development team has an utter autonomy to choose the overall development style for planning, deciding and conducting all the scheduled work as long as scheduled events are held. Daily scrum helps to review the overall progress

and inform if there are production related impediments arriving. However the practices formal project management is often neglected or entirely omitted in agile methodologies. Added risk management can help prevent production delays. Increased cost estimation helps to evaluate the use of resources. For example if a certain project turns out to be more demanding, the head of project management can transfer more personnel to more arduous projects and remove personnel from projects that require lesser amount of work.

In terms of project portfolio management. There are no significant additions to enable a systematic way to examine projects as singular units. As there are many active projects worked on simultaneously, there should be a more coherent way to review their current technical weight in the sprints and financial outcomes. Currently the head of project management is solely accountable for the results of the projects. The proposed model does encourage developers to think about risks and estimations more carefully, which should enable a greater visibility of the overall situation. This should give basis to make informed decisions in terms economical viability and project profitability.

All things considered the model does have a clear structure and every aspect of it is explained adequately. The model describes the life cycle of a single project from project initiation to closure. The process of project sales and project delivery to a client is omitted from this research, as the primary purpose and research outcome is to extract the point where production efficiency and project management are perfectly balanced. However this model can extended to consider the sales side more carefully in future research.

6 Research overview

This is the section that conveys the details of the empirical research for the suggested model. This is done by explaining the research setting, cases and measures.

6.1 Setting

The research setting takes place in the premises of Softwarehouse's daily operations. The duration of the research period is one month, during which the model is tested on two consecutive scrum sprints. The developers will be given instructions on how to use the model in life cycle parts where the biggest additions and changes occur. In terms of honoring the agile nature, the development team is trusted to follow the instructions within absolute autonomy and the best of their abilities. The head of project management will spectate the model processes and workflows during the experiment to see how well they are followed. This instruction document can be found in the appendices section of this paper. Before the experiment begins, a life cycle model tutoring session will be organized for the developers if there is need for one. Finally it will be mentioned that the head of project management will take no special consideration when ordering and prioritizing projects. The intended cadre of stakeholders for this experiment is comprised of 8 developers, one software tester and the head project management. The changes this model brings to daily operations, affect mostly the developers. The primary goal is to evaluate how well the model works for all the projects that are active during the research period. All the projects and teams that are active during this period, are briefly listed and explained below.

6.2 Cases and teams

During the experiment, four different products were developed as a joint effort of IT-services and Department of Future Technologies. The estimated team size for each of them was expected to between one and three individuals.

6.2.1 Innovation platform

This is a platform designed to handle project order requests and monitoring said projects. It designed for people interested in understanding and driving transformations. It brings together researchers and companies to co-create and share knowledge. In addition to conducting basic and applied research, this platform is an interface for expert services.

6.2.2 UGIS-system

This live system manages the studies of doctoral candidates at the University of Turku. It is a portal that assists in training processes and contains information regarding the doctoral training of each doctoral candidate. The Ugis portal instructs and maintains the progress of a doctoral degree. Candidates can use this system to check status their status and instructions. Additionally the portal can be accessed by thesis supervisors with active university credentials.

6.2.3 Material database

This database assists and manages metadata of various research material collections. It enables the possibility to repeat and evaluate research experiment results by reviewing original academic source material. It makes sure that existing source materials are stored appropriately and enhances the usability of them. By exhibiting consistently the metadata of various materials, this database ensures that every item in the database has a clear description.

6.2.4 Lataamo

This system is virtually infrastructure that enables possibility to pool together all the official documents produced during the legal processes of individual bills. This covers all the documents from the preliminary preparation phase to enactment and allows nontraditional form of research in human sciences. It contains tools for distant reading and analysis of vast materials. The system is capable of data collection, integration and knowledge visualisation.

6.3 Measures

As the purpose of this model is to increase practices in project management and by minimum maintain the current production volume, the key research metrics are divided into three parts: Personal experiences, persistence of production volume and quality of project management. Based on combined results of them, the success of this research can be determined. To extract and evaluate the results, all developers will be given a questionnaire after the research period is finished. The survey is comprised of three sections: Individual experience, Project Management and Finale. The core style of questions is shown in the list below. The actual questionnaire can be found in the appendices section. Additionally the head of project management will be interviewed personally with similar questions.

1. Did the model increase the production volume in Softwarehouse?
2. How did the model come accross on the instruction sheet before the research started?
3. On the individual level, how did the model affect performance?
4. On the group level, how did the model affect the performance?
5. Was the concept of having an initial planning committee cumbersome and straining?
6. Was the added monkey testing cumbersome and straining?
7. was the added project management cumbersome and straining?
8. How would you rate the quality of Resource and personnel management?
9. how would you rate the quality of Risk and Uncertainty management?
10. how would you rate the quality of Scope and estimation management?
11. Does the model help managing projects systematically?
12. Would use this model in the future?
13. Could the model be improved?

14. Any additional comments or suggestions?

List 2: The style of questions

6.3.1 Individual experiences

There is considerable amount of forces affecting the individual experiences. In the beginning the information on total amount active scrum teams and team structure is basically unknown. The suggested model introduces new activities both in group and in individual level. Additionally the nature and type of active projects may affect the individual experiences. Like the model itself this questionnaire was supposed to be as light as possible to fill out. It would take approximately 5-minutes to answer every multi-choice questions. These questions gave every developer a decent and effortless opportunity to review and reflect their opinions about the models core activities and its effects on overall performance - both individual and group. This questionnaire mostly inquired developers' opinions on how strenuous, rigid and cumbersome the model and its activities felt after the experiment. However it also gave them the chance to reflect their predictions about the model before the experiment started. Additionally in the final section every developer also had the chance provide detailed feedback about the model.

6.3.2 Productivity

The chosen productivity measure usually depends on the domain of software projects and the programming languages used. Lines of code is a typical measure of output. However this metric is valid only when each comparable project is developed with the same language.[11] For all intents and purposes, all the active projects in the research period can be regarded as heterogeneous. As such the measure of a standard story point used to evaluate the needed effort for any task, is an adequate way to review progress in every active project. The total amount of completed story points after each sprint is the primary metric to determine and evaluate the total output and productivity. Production volume was an important metric to consider when this model was designed. The very first question in the form asks directly whether the model increased or decreased the production volume during the experiment.

6.3.3 Project management

The quality of added project management was the third relevant aspect to evaluate. The research case section identified four important areas of project management in software projects: Resource management, risk management, Project estimation and portfolio management. This questionnaire had a dedicated to section and questions to evaluate the overall quality of each to them. The scoring of each area is between one and five. At the end of experiment the aggregated average values directly correlate whether they have a strong or weak performance.

6.4 Research analysis

The research outcomes and results are mostly based on human experiences and perceptions. Individual attitudes towards the model and the quality of project management are effortlessly extracted through a well defined and structured online survey form. The answers of the survey and an interview with the head of project should serve as an adequate method to draw conclusions and lay ground work for potential future research. Additionally the total amount of completed story points is a robust metric to determine the production volume during this experiment.

7 Results and Discussion

This is the section where the results of the empirical experiment are presented. They give basis for deeper analysis and discussion. Additionally topics for future research are briefly explored here.

7.1 Results

First of all it is formally worth noting that there were substantial issues with data extraction. This issue was properly discussed during the interview with the head of project management. The timing for this research was less than optimal. This research was initiated during the time when employees were returning from their summer vacations - There was a minor acclimation process going on during this research. Additionally it was hard to include proper projects to this experiment. Agile frameworks were hardly utilized as most of work was conducted individually. These notions were followed with a direct statement that company of this size may not able adopt centralized model such as this. However it was clearly pointed out that the elements in the model can be applied well in daily operations.

In the end there were only a few replies in the original questionnaire. There was an attempt to gather additional data. This was done by altering the overall tone of the questionnaire. The updated questions inquired information with a hypothetical style from those who could not partake in the experiment. Their opinions on how does the model come across on paper were asked in this modified questionnaire. To clearly state the total amount of gathered data, there were two replies in the original questionnaire, one reply in the modified questionnaire and the interview with the head of project management. The sparsity of replies does affect the quality of this research. However when combining all the answers in all the questionnaires and the interview, some clear conclusions can be made. For the very least, it can be clearly stated that the model did not lower or increase the production volume in softwarehouse - one of the key stipulations of this research can deemed as verified.

In terms of the individual experiences and organizational creativity, there was

minor variance between answers. It can be stated that the model did not have any noticeable effect on group and individual performance. The concept of having an initial planning committee to conduct scoping and planning before the sprint starts was not deemed cumbersome or straining. To some extent The concept was actually appreciated and it was pointed out that initial design and planning has been somewhat neglected in the past. The added monkey testing was deemed beneficial and well-received. However the added project management was partially deemed cumbersome and straining. One of the answers in the free comment section stated that the responsibility of formal project management should not be included in the developer's role. According to the head of project management, this model does not harm organizational creativity.

The part of the questionnaire where the assessment of project management took place yielded in relatively high appraisal. Every identified area in the project management received excellent score by average. According the to the combined results, The weight of resource and personnel management was slightly lightened. Risk management was acknowledged as a useful practice. The quality of quantity of it is predicated on the size of the project. Project scoping and design was already executed well. However this model formally recognizes it and makes the process fractionally more fluent. The head of project management highlighted the fact that as important as initial planning is, the key in agile frameworks is the occurring learning throughout the project.

At the end of questionnaire, the given answers would indicate that the personnel of Softwarehouse are mostly willing to use to the model in future projects. As with the answers on the question whether the model could be improved would suggest that the model is good as it is - perhaps minor improvements could be made. When analyzing the answers in the free comment section further, it was repeatedly pointed out that the core essence of the model resembles and reflects many of core practices that were already utilized before the experiment took place. One of the comments even directly claimed: "I do think that the model is good as it describes our practices well." According to the head of project management this model can be very much utilized in the future projects. However it was pointed out that projects in Softwarehouse varies very much in type and size - treating every project case-by-case and applying the practices of the model accordingly is more important than forcibly adhering a centralized model in every case.

7.2 Discussion

In the following subsections the impact of the model on organizational creativity and project management is discussed in greater detail.

7.2.1 Organizational creativity

James Moultrie [37] points out in his article that the term creativity is mostly used to describe processes and outputs instead of inherent traits of individuals. The organizational creativity can be detected in the output from its employees. Elements such as humour, risk taking, freedom and idea support can collectively describe creative capacity in an organization. Additionally the article conveys that resources, management practices and motivation are factors that interact with each other and influence the level of innovation in an organization. Amabile's componential theory of Organizational creativity is based on this notion. The Figure 14 below illustrates this theory

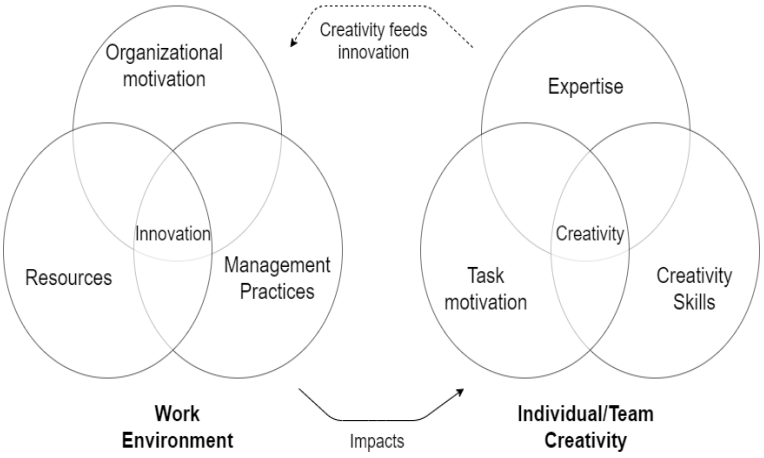


Figure 14: Componential theory

The suggested model used in this experiment was designed based on preliminary observations made during a brief employment in Softwarehouse. Additionally inputs from the leader and the head of project management were used to make the necessary formal adjustments into the model. It can be stated that softwarehouse fosters and supports humour, autonomy and idea creation in its daily operations. Creativity can be detected in individuals, team dynamics and the core environment. The fact that Softwarehouse operates under an academic institution also

supports the pursuit of innovation. The personnel are more inclined for exploration and research and the resources granted by University of Turku support this kind of dynamic. Management practices do not share a lot similarities with those adhered in the private sector. This is understandable as too strict processes may hinder the creative flow among academic workers.

Apart from minor deviations in the attitudes towards formal project management, the model and its practices did not strain the creative environment in SoftwareHouse. The model implores to follow certain repetitive pattern, which could lower the motivation on individual and task level. Those who responded in the survey stated that they appreciated the added practices. These practices were designed to add more insight in planning and give means to understand the state of current situation with product and short term goals. This would help to prevent unplanned disasters and delays production. Generally everyone wants to avoid unplanned work and finish their work in time. Moreover it is valid to state that even creative workers who prefer autonomy can appreciate more formal and vigorous practices if they are in the interest of increasing operational awareness.

7.2.2 Risk, resource, estimation and portfolio management

When working with agile methodologies, risks are usually managed differently. Due to careful planning and open communication there are greater prospects for preemptive risk management - dealing with smaller impediments before they turn into considerable risks. The quality of risk management did receive high scores however the idea that developer should allocate separate time for identifying potential risks may seem a bit needless and unfruitful activity. Extensive risk management may be of use when the software project is scheduled to last years and there are large amounts of resources at play. Identifying potential risks and assigning a level of severity is sort of useful. But discussing about minor about minor impediments in daily Scrum might be more effective for Softwarehouse.

There is a considerable chance that resource and estimation management was significantly enhanced with Kick-Off and planning step in the proposed model. Nearly

all the respondents identified it as pleasant way to implement estimation and the resource and estimation management did receive high scores as well. Swift and robust planning has always been a part of agile methodologies. However like one of the respondents said, it often dismissed altogether in sprint planning. This model for the very least identifies the importance of it and has assigned an additional step for it.

Finally this thesis can reflect the quality portfolio management. At this point it has to said that despite of high score in this category there is no way to review the effectiveness of this aspect with this experiment alone. To determine the systematical management of multiple project with a centralized model, the experiment duration would have had to be much longer so that actual long-term effects could have been identified. It is good that Softwarehouse now has a officially documented model that represents their practices. However with further consideration, a separate longer experiment with different measures would be a prerequisite to truly review systematical portfolio management.

7.2.3 Formal project management

Lung-chun Liu and Ellis Horowitz [38] state that there are certain essential features that must be followed in software related project management. To list a few, The underlying management framework or model should describe that software development is a design process, which is inherently evolutionary by nature. Working in parallel enables multiple individuals to work on same the project simultaneously. If an activity fails the framework should have the means to indicate the activities and resources that are affected. The minimum set of features include: Specific objective, budget, consumption of other resources and a time-frame. Essentially the article emphasizes the fact that Project management is a mixture people, resources, systems and techniques required to carry out the project successfully.

In Softwarehouse the combination of people, processes and resources is not ordinary compared to regular Software firms. The utilization of formal project management is still significantly small. This may be due to the fact that Softwarehouse has not worked on a project that was large enough to benefit from formal project management. Nowadays with agile development methodologies the idea is to reduce the needless activities and focus on what is important, which is delivering actual value in

smaller increments. This becomes especially important in smaller development operations with fewer resources and smaller time-frames. Considering this notion, formal project management in a context such as Softwarehouse may be actually deemed as a wasteful activity.

Avi Yaeli and Alexander Kofman [39] from IBM suggest that several skills are needed to conduct software development. In addition to technical practices such as programming and testing, Managerial skills such as agile coaching and stakeholder coordination play an important part. To this date there is no consensus on the distribution of roles among development teams and various management styles to different role schemes.

In the questionnaire one of the developers pointed out that project management should not be a responsibility of the developers. However when working with multiple smaller projects in parallel, it may be challenging to delegate a holistic managerial responsibility to a single person. In terms of Software development governance at Softwarehouse, it is difficult to say whether the responsibility of project management should be a functional role or a group role. The quality of project management did receive a high score in the questionnaire. However in the future, the experimented model could be revised to adopt a custom functional role, which would allocate more focus on managerial and coordination activities.

As the usage of agile methodologies increases, the emphasis on individual and group autonomy becomes more topical. Development teams are relatively free to choose their own pace and practices. A cohesive team who collaborate based on trust can be hindered by developers who prefer to work on their own. Trust is predicated on the fact that team members believe in the competence and integrity of their colleagues.[40] Softwarehouse is a relatively small and tight organization where everyone knows each other. It is clear that the personnel organize and conduct their work with high autonomy and trust towards one another. Clearly stated goals and well-established practices forged in autonomy may be the only crucial synergy needed to deliver working software in a timely manner in Softwarehouse - needless formality of high level project management could be justifiably dismissed altogether from daily operations.

7.2.4 Model compliance with agile foundation

Nowadays having well-established agile workflows in an organization can be seen a huge advantage. Aside from increased quality of work and reduced probabilities for production delays, being agile also makes the working environment better for current employees and more attractive to potential job applicants. However this only happens if core agile values are truly understood and followed. In the third section, this thesis briefly introduced the agile manifesto and its core values. If an organization claims to be working agile ways, it really needs to able verify this.

The proposed model operates with the assumption that agile values are followed in its workflows. During the preliminary data gathering and the post experiment phase, it was vehemently pointed out that deleting and adding items to a sprint backlog during a sprint goes against the nature of agile estimation - items for upcoming sprint should be locked down based on the production velocity of previous sprints. The agile manifesto claims that responding to change is crucial over following a plan. However it is worth pointing out that the official Scrum guide does not suggest that developers or virtually any stakeholder has the authority to add more work for during a sprint let alone alter the size of the development team. This notion again raises a point whether formal project management practices can be mixed agile methodologies.

The model does recognize the importance of working software over comprehensive documentation - as practicing documentation was instructed to be kept as light as possible. However the question whether the model itself forces the organization think more about processes and tools over individuals, makes the exploration and conceptualization of a centralized model for agile methodologies seem more than controversial. Perhaps the proposed model should be more like a tailored set of guidelines that are considered in the case of each individual project.

7.3 The deployment of the suggested model

The survey results indicated that the model represents many of the practices that were already in use before the experiment took place. Additionally when most of the respondents conveyed their willingness to use the model in the future, the adoption and

standardization of the model and its practices as a key reference was nearly effortless. Essentially it can be stated that one of the key outcomes this thesis is a well-documented model that is supported by the head of project management and some of the core personnel in Softwarehouse. In the future the model will be used as a key consideration and guideline, when new projects are ordered and their approximated life cycle is planned. Like with any new set of practices and a culture that supports them, there are usually individuals who are willing to adopt new ways and individuals who are hesitant at first. A well-made set of instructions can be helpful for both existing and new employees, but those who have understood the value of the model will be the ones who truly provide assistance by tutoring and setting an example - it is the best way to drive change towards new frameworks and cultures.

7.4 Future research

In the interview with the head of project management it was discussed that the model could perhaps assist more in high-level decision making - which customer's needs are most important at the moment. The following topics and articles go through well-known practices in assessing customer's needs and delivering value consistently.

7.4.1 Customer prioritization

It is widely claimed that organizations should prioritize between their customers and allocate resources accordingly. This means that certain customers receive different and preferential treatment in certain situations. In practice, some organizations utilize a customer tier system, where customers are evaluated based on their importance and potential sales volumes. Implementing differentiated use of marketing instruments for different customer tiers is crucial as it would increase probability that marketing efforts are directed towards the right customer relations. Homburg, droll and Totzek [41] have proposed framework that assesses the effects in customer relations based on two factors: relationship characteristics and performance outcomes. The overall estimation of customer portfolio is based on evaluating the causality between these factors. There are three important characteristics for a customer relationship: satisfaction, loyalty and the share of wallet. The respective average of each of these is determined.

The idea of customer prioritization is often challenged. The fact that lower level customer can be dissatisfied due to potential neglect, which could result in weakened reputation as dissatisfied customers are more inclined to spread negative word.[41] In the case of Softwarehouse, the customer acquisition and management strategies are very different as most of customer portfolio is comprised of other academic institutions. The reputation among academic institutions is very important aspect of softwarehouse. Which is why the usage of a model, where decisions are made based on financial profitability, would likely yield in undesirable outcomes. Additionally a model like this does not scale to a smaller organization like Softwarehouse as it does not utilize vigorous marketing strategies. This is likely due to the fact that the organization mostly revolves around smaller day-to-day development operations. When evaluating the quality of development, measuring customer satisfaction is a key aspect in agile methodologies. Softwarehouse could benefit from a satisfaction based portfolio management system. Like with the model mentioned, customers could be placed in tiers however their scoring would be based more on satisfaction and loyalty - financial performance and factors would not be considered as much.

7.4.2 Measuring value and satisfaction

In terms of product life cycle, prioritization is a relevant aspect to consider. With prioritization, key activities can be extracted, boundaries of work can be identified and differentiating between wants and needs is easier. There are many techniques that help in prioritization such as Kano Methodology, MoSCoW technique and Kj methodology. These techniques deliver good results in long term planning. However lean prioritization is better, when applying a quick approach and a daily basis decision making. Lean prioritization offers the power of value and effort matrix. The figure 15 below this paragraph illustrates this.[42]

The value and effort matrix provides a simplified way to evaluate how much value the feature brings to the product and actual the effort needed to complete the task. It is self-evident that the development team should prioritize on tasks that are produce high value and require minimal amount of effort. However it is not guaranteed that implementing high value tasks results in high customer satisfaction. In addition to smart

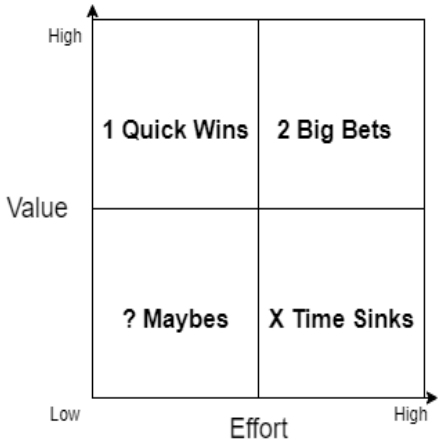


Figure 15: Value effort matrix

and lean prioritization, there should also be an effortless way to measure customer satisfaction and engagement.

In a competitive market, companies are inclined to focus their efforts on the demands of their customers. Advertisement is the primary way to conduct corporate communication towards customers. This occurs with various promotions and public relation efforts. However the flow of information and feedback from the customer is more than often quite limited. Collecting actual feedback is usually implemented with voluntary surveys, which enable companies to inquire specific topics of interest and customers to communicate their opinions about the supplying company. Daniel Scheiner and Jon Crosnick [43] mention in their article that the best question to measure customer satisfaction is: "How likely is it that you would recommend [brand or company x] to a friend or a colleague?". The responses to this question are then aggregated into a single metric, the net promoter score (NPS). In addition to NPS, there are other popularly recognized metrics such as customer satisfaction score (CSAT) and customer effort score (CES). These can be extracted with fluent social media monitoring and various electronic surveys such as In-App, post service and email questionnaires.[44]

Creating good and robust surveys that would suit the needs of Softwarehouse, is not a simple thing to implement. Prioritization and development decisions are made on daily basis and these decision usually reflect the outcomes of the current and the next sprint. Reaching out to the customers to assess their needs regularly, has to be effective and as effortless as possible to the customer. Authoring survey that gives more

understanding about customer's satisfaction could prove to be beneficial. In the best case scenario the survey and feedback collection would be automated to reduce manual and technical overhead. Implementing continuous feedback into Softwarehouses operations could be the next step.

7.4.3 DevOps

Nowadays with emerging agile methodologies and rapid development techniques, comes the new core concepts such as continuous integration and delivery. Self-organizing teams and easy infrastructure management enables DevOps practices, where developers and systems administrators work together to deliver value systematically and with faster velocity. Frequently measured feedback comes into aid as well-configured systems are able to detect internal systems errors. External feedback is obtained when engineers conduct robust code reviews and business users test the deployed code to assess the usability.[45]

Analyzing DevOps and its practices further goes well beyond the scope of this research. However it is worth pointing out that Softwarehouse could benefit from DevOps as concepts such as continuous delivery and continuous feedback are included in it. Larger companies such as Facebook, Amazon and Netflix utilize DevOps practices in their daily operations. Due to its efficiency, they are able to deliver new features and software updates even on a daily basis to their customers. This happens with practically non-existent system down-time and reduced technical overhead. Delivering high value items rapidly and consistently creates a strong basis to increase existing customer satisfaction and to attract new customers. Currently there are many tools available to implement DevOps practices with different levels of licensing. Due to this even smaller organizations are able to adopt DevOps processes.

8 Summary

The first section began by explaining how smaller software projects are becoming more common nowadays and why adhering a proper life cycle management is beneficial for all kinds of software projects. Through this notion it was presented that creating a customized life cycle model for Softwarehouse would be the core deliverable of this research. After this the section explained briefly the background and the current situation of softwarehouse. Despite of the strong usage of official Scrum framework, Softwarehouse was short of formal and systematic project management. Additionally primary considerations for the suggested model and the justified outline for this thesis were explained.

In the second section of this thesis, the basics of traditional waterfall model were explained to elaborate how the first official software development life cycle model was practiced. After this typical agile methodologies were briefly explained. These included the original variant of Scrum framework and Kanban. Additionally the traditional SDLC was explained. The idea of this section was to explore core thoughts such as lean development, redundancy of certain core practices and systematic workflow with well-defined stages.

The third section explored existing research in topics such as rapid production, small project organizations and organizational creativity. It was established that developers tend to use less than 30 percent of time in actual coding during development. The rest of time of is used in meetings, problem solving and clearing misunderstandings. Enabling efficient communication is one of the key elements to establish rapid production velocity. Generally recognized rapid development methodologies such as rapid application development, joint application development and dynamic systems development method were introduced. All of these frameworks speak for the consistent usage of various robust techniques such as time boxing, incremental development and adoption of advanced statistic and graphic tools. Smaller project organizations have much more specific needs, which is why the adoption of general frameworks and components is not self-evident. When dealing with reduced resources, the core idea is to focus on lean thinking and iterative development. Organizational creativity can be

regarded as product in a complex social system and it is directly linked individual well-being. Finally the section explained two examples on how larger companies scale agile practices in their operations while adhering agile manifesto.

The fourth section elaborated the research case and its context further. Softwarehouse is a small division of IT professionals and researchers, that provide software development, data management and laboratory services. Softwarehouse works on many projects simultaneously and completes approximately 40-60 projects per year. Often Softwarehouse has at least five active projects, which are managed and implemented by 8 developers. Softwarehouse mainly uses the official Scrum process and is capable of using Kanban on occasion. Jira Atlassian is used for project portfolio management and Scrum backlog management. Slack is the primary messaging and communication tool. In terms of project management, Softwarehouse followed a discovery-oriented management style. Resource and personnel management happens by monitoring the project database and adequately assigning the right amount of employees to carry out the project. Risk management is based on work amount estimates, which depend on the level of technical challenge. The importance of project scoping and estimation was recognized, but lacked a standardized and robust way.

The fifth section began with explaining the design considerations for the proposed model. Preserving organizational creativity and production volume where two key conditions to keep in mind. The model was graphically presented and all the stages and steps were explained in great detail. Additionally the model was analyzed. It shared many similarities with a typical scrum framework however introduced and emphasized the importance of project scoping, risk and resource management.

The Sixth section explained the details of the experiment used to determine the effectiveness of the proposed model. The experiment setting and the actual software project cases were explained. The primary measures for the experiment were the production volume, individual experiences and the quality of project management. These were to be measured with a light weight online survey and an interview with the head of project management. Additionally the quality of the experiment was assessed.

The seventh section was meant for the results of the experiment and deeper

discussion of research outcomes. Initially it was explained that were considerable challenges with data gathering which lowered the quality of the results. However by adding another modified questionnaire and combining results, some clear conclusions could be made. It was clear that model did not lower the production volume in Softwarehouse. There were minor impacts on organizational creativity and project management.

It was stated that organizational creativity can be detected through humour, risk taking and idea support. Management practices and motivation interact with each other and influence the level of innovation in an organization. Apart from added the project management the suggested model did not harm organizational creativity. The model implores to follow a certain repetitive pattern, which could lower the motivation. However in general the model was well received. This could have been due to fact that he model increases insight through systematic planning, which could reduce unplanned disasters. Everyone appreciates that work is finished in time - The cost of added project management is worth the gain of enhanced operational awareness.

Even though project management did receive high scores every identified category, it was pointed out that separate risk management is not so beneficial in smaller software operations. Results in systematic portfolio management were deemed inconclusive due to insufficiently executed experiment. However the quality of scoping and estimation could be verified to be successful.

In software development, project management is essentially a union of people, resources and processes that is required to carry out the work in a project. Efficient management requires at least a specific objective, budget, time-frame and consumption of other resources. The quality of project management did receive relatively high scores in every category in the experiment questionnaire. However it was prominently raised that project management should not be the responsibility of a single developer. The model could be revised to adopt a functional role in project management, which would alleviate the weight project management as a group activity. Using the model to manage every project consistently and systemically was challenged by the head of project of management. According to him, every project should be treated case-by-case and the suggested practices should be applied accordingly. Additionally it was explained that

Formal project management is mostly in beneficial in larger projects. Softwarehouse mostly has smaller projects, which lead to conclusion, that Softwarehouse might not actually benefit from having a centralized template model. Furthermore when working with agile methodologies, the focus is on learning as opposed to following a plan - formal project management could be deemed as a wasteful activity. When working with smaller projects with a non-traditional organization, clearly stated goals and reliably self-organizing teams may be the only synergy needed to deliver working software within the best possible capacity.

The model was reviewed in terms compliance with core agile values. Due to various reasons, attaining the full range of benefits of agile is only possible by truly understanding agile foundation. The notions of agile manifesto were reflected against the proposed model. In a few crucial ways, the model and its processes did not adhere agile values, which lead to conclusion that centralized life cycle models may be overly controversial in the context agile methodologies.

The model deployment to Softwarehouses daily operations was explained by reiterating the positive survey results and how they promote the willingness to use the model in the future. Additionally aside from a well-documented model and instructions, the model is supported by the upper management and some of the core personnel, who will drive the change of using the model in future projects by referencing the model as a key consideration and providing assistance to employees, who are unfamiliar with the model.

This thesis ended with exploration of topics that were suggested by the head of project management. Essentially the ideal model could be equipped with better means to conduct high-level decision making. To determine which customer's needs that deserve more attention at certain times. Customer's satisfaction and loyalty were recognized as the only relevant metrics for Softwarehouse, when evaluating or attempting to prioritize between current customers. Simple value estimation of user features and stories can be done with low maintenance techniques. However with well implemented forms, collecting customer feedback through metrics such as NPS and CSAT was identified as a robust way to measure customer satisfaction and engagement. As the benefits of implementing automated ways to collect feedback was raised, this thesis finally briefly

explained the concept of DevOps, which in addition to continuous feedback includes lots of other useful concepts and techniques that could really increase customer satisfaction and even attract more business.

References

[1] Robert T. Futrell, Linda I. Shafer, and Donald F. Shafer. *Quality Software Project Management*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[2] Youssef Bassil. A simulation model for the waterfall software development life cycle. *CoRR*, abs/1205.6904, 2012.

[3] P. Trivedi and A. Sharma. A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation. In *2013 2nd International Conference on Information Management in the Knowledge Economy*, pages 188–194, Dec 2013.

[4] TOOLSQA. Waterfall model, 2016. Last accessed 10 April 2020.

[5] Ken Schwaber. Scrum development process. In Jeff Sutherland, Cory Casanave, Joaquin Miller, Philip Patel, and Glenn Hollowell, editors, *Business Object Design and Implementation*, pages 117–134, London, 1997. Springer London.

[6] M. O. Ahmad, J. Markkula, and M. Oivo. Kanban in software development: A systematic literature review. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, pages 9–16, Sep. 2013.

[7] Planview LeanKit. Maximize your time, improve efficiency with the kanban system, 2019. Last accessed 10 April 2020.

[8] Ieee guide–adoption of iso/iec tr 24748-1:2010 systems and software engineering–life cycle management–part 1: Guide for life cycle management. *IEEE Std 24748-1-2011*, pages 1–96, June 2011.

[9] Iso/iec/ieee international standard for systems and software engineering – content management for product life-cycle, user, and service management documentation. *ISO/IEC/IEEE 26531:2015 (E)*, pages 1–60, May 2015.

[10] Stackify. What is sdlc? understand the software development life cycle, 2017. Last accessed 10 April 2020.

[11] S. D. Teasley, L. A. Covi, M. S. Krishnan, and J. S. Olson. Rapid software development through team collocation. *IEEE Transactions on Software Engineering*, 28(7):671–683, July 2002.

[12] P Beynon-Davies, C Carne, H Mackay, and D Tudhope. Rapid application development (rad): an empirical review. *European Journal of Information Systems*, 8(3):211–223, 1999.

[13] E.J Davidson. Joint application design (jad) in practice. *Journal of Systems and Software*, 45(3):215 – 223, 1999.

[14] D Tudhope, P Beynon-Davies, H Mackay, and R Slack. Time and representational devices in rapid application development. *Interacting with Computers*, 13(4):447 – 466, 2001.

[15] Steven D. Tripp and Barbara Bichelmeyer. Rapid prototyping: An alternative instructional design strategy. *Educational Technology Research and Development*, 38(1):31–44, Mar 1990.

[16] W. Q. Meeker and M. Hamada. Statistical tools for the rapid development and evaluation of high-reliability products. *IEEE Transactions on Reliability*, 44(2):187–198, June 1995.

[17] M. R. Lyu and A. Nikora. Casre: a computer-aided software reliability estimation tool. In *[1992] Proceedings of the Fifth International Workshop on Computer-Aided Software Engineering*, pages 264–275, July 1992.

[18] Anton Kokalj. Computer graphics and graphical user interfaces as tools in simulations of matter at the atomic scale. *Computational Materials Science*, 28(2):155 – 168, 2003. Proceedings of the Symposium on Software Development for Process and Materials Design.

[19] M. E. Fayad, M. Laitinen, and R. P. Ward. Software engineering in the small. *Communications of the acm*, 43(3), March 2000.

[20] Jan Bosch, Helena Holmström Olsson, Jens Björk, and Jens Ljungblad. The early stage software startup development model: A framework for operationalizing lean

principles in software startups. In Brian Fitzgerald, Kieran Conboy, Ken Power, Ricardo Valerdi, Lorraine Morgan, and Klaas-Jan Stol, editors, *Lean Enterprise Software and Systems*, pages 1–15, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[21] N. J. Nunes and J. F. Cunha. Wisdom: A software engineering method for small software development companies. *IEEE Software*, 17(5):113–119, Sep. 2000.

[22] Richard W. Woodman, John E. Sawyer, and Ricky W. Griffin. Toward a theory of organizational creativity. *Academy of Management Review*, 18(2):293–321, 1993.

[23] Farida Rasulzada and Ingrid Dackert. Organizational creativity and innovation in relation to psychological well-being and organizational factors. *Creativity Research Journal*, 21(2-3):191–198, 2009.

[24] Maria Paasivaara, Casper Lassenius, and Ville T. Heikkilä. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '12, page 235–238, New York, NY, USA, 2012. Association for Computing Machinery.

[25] James Halprin. Safe: Four key things you should know, 2018. Last accessed 10 April 2020.

[26] k. Schwaber and J. Sutherland. The scrum guide. the definitive guide to scrum: The rules of the game ™.

[27] LAIRE MAYNARD. Learn scrum with jira software, 2019. Last accessed 10 April 2020.

[28] M. Mankins and E. Garton. Software development teams and slack: A handbook, 2017. Last accessed 10 April 2020.

[29] Karlos Artto, Jaakko Kujala, Perttu Dietrich, and Miia Martinsuo. What is project strategy? *International Journal of Project Management*, 26(1):4 – 12, 2008. European Academy of Management (EURAM 2007) Conference.

[30] Satu Rekonen and Tua Björklund. Adapting to the changing needs of managing innovative projects. *European Journal of Innovation Management*, 19:111–132, 01 2016.

[31] Cast. Risk management in software development and software engineering projects, 209. Last accessed 10 April 2020.

[32] Paul Barnes. Software costs estimation in agile, 2020. Last accessed 10 April 2020.

[33] Richard Whitley. Project-based firms: new organizational form or variations on a theme? *Industrial and Corporate Change*, 15(1):77–99, 02 2006.

[34] Meisterplan. Agile principles as a guide for resource management, 209. Last accessed 10 April 2020.

[35] ROBBRECHT VAN AMERONGEN. 7 agile estimation techniques – beyond planning poker, 2009. Last accessed 10 April 2020.

[36] Kanbanize. Cumulative flow diagram for best process stability, 2009. Last accessed 10 April 2020.

[37] James Moultrie and Alasdair Young. Exploratory study of organizational creativity in creative organizations. *Creativity and Innovation Management*, 18(4):299–314, 2009.

[38] Lung-chun Liu and E. Horowitz. A formal model for software project management. *IEEE Transactions on Software Engineering*, 15(10):1280–1293, Oct 1989.

[39] Yael Dubinsky, Avi Yaeli, and Alexander Kofman. Effective management of roles and responsibilities: Driving accountability in software development teams. *IBM Journal of Research and Development*, 54:4:1 – 4:11, 05 2010.

[40] O. McHugh, K. Conboy, and M. Lang. Agile practices: The impact on trust in software project teams. *IEEE Software*, 29(3):71–76, May 2012.

[41] Christian Homburg, Mathias Droll, and Dirk Totzek. Customer prioritization: Does it pay off, and how should it be implemented? *Journal of Marketing*, 72(5):110–130, 2008.

[42] Pavel Kukhnavets. Value/effort matrix: Lean prioritization for product managers, 2018. Last accessed 10 April 2020.

[43] Daniel B. Schneider and Jon A. Krosnick. Measuring customer satisfaction and loyalty: Improving the 'net-promoter' score. 2008.

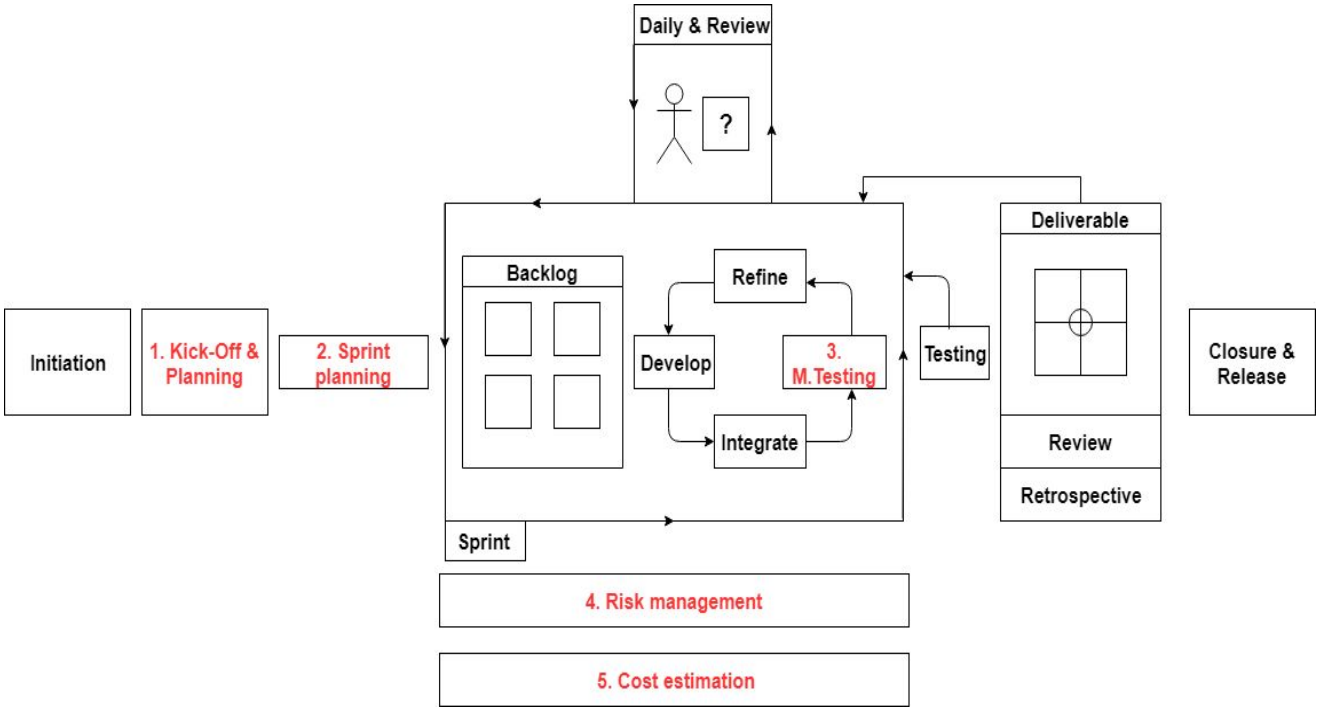
[44] Userlike Pascal. 6 proven methods for measuring customer satisfaction, 2016. Last accessed 10 April 2020.

[45] Oliver Krancher, Pascal Luther, and Marc Jost. Key affordances of platform-as-a-service: Self-organization and continuous feedback. *Journal of Management Information Systems*, 35(3):776–812, 2018.

Appendices

Instructions for developers

The diagram below illustrates the proposed software development life cycle model for small projects. Everything from the initiation to closure is shown here. However the significant changes and additions in development team's daily operations only appear in boxes 1 - 5.



1.Kick-off & planning

The head of project management orders a project a projects and assigns a group individuals for the project planning. This initial planning committee will meet to discuss about the general aspects of the project (Difficulty, used technologies etc). Product owner is assigned and the initial product backlog is authored.

Resource planning is an important part of this phase. As such the initial planning committee will choose the right amount of personnel, which will form the final/actual scrum team. In order to do this, the team will estimate and choose the right amount of story points for every item in the backlog. This can be done with lots of different agile effort estimation ways such as Planning Poker and T-shirt Sizes. Based on the total amount of story points for the project, the initial planning committee will choose the right amount of personnel.

2.Sprint planning

In sprint planning, the scrum team chooses all the items for the sprint and plans how they are implemented. Depending on the estimated amount story points in the previous phase the team will choose an *appropriate schedule estimation and review technique* for the upcoming work. In standard projects with enough diversity and complexity, a traditional burn down chart is a valid way to estimate the work progress. However in very small projects, a burn down chart may be too cumbersome and heavy to Use. For these instances, the team might consider using a cumulative flow diagram. However it is worth noting that this model is designed for Scrum projects with enough diversity. Cumulative flow diagram works well in projects where Kanban is used. It is just an option.

3.M. Testing

This box represents an added bonus to software testing practices and quality assurance. Conducting Automated J-unit testing is already a standard procedure in feature integration, but M. testing means adding monkey testing in development. Considering that the intended product is not large, a sub variant of monkey testing called smart monkey is used. This means that in addition to developing, the developers will on occasion conduct smaller scale testing with the mentality of knowing the purpose and functionality of the system. Essentially the developer navigates through the system and gives valid inputs to perform random tasks. This is done in the interest of finding bugs and errors without predefining any specific test or user cases. The durations of monkey testing sessions should be kept as short as possible.(5-10min) Developers can have these sessions whenever they deem it possible.

4.Risk management and 5. Cost estimation

The boxes below the development sprints represent all the added risk management and cost estimation for the entire project. The development team has the complete autonomy to choose how this is actually done. There can be small separate meetings. However since there formally no project specific managers in Softwarehouse, every individual in the development team should practice this whenever it is possible.
Risk management should have the following process flow: Identification, planning and monitoring. This applies to risks that could affect either the current sprint or future sprints.

Additionally cost estimation practices are added to determine whether the development team could alter the sprint backlog either by eliminating current tasks or adding new tasks during the sprint. Depending on the needed effort for a project, assigned developers may even be “hotswapped” between projects during sprints.

(It is worth noting that deleting and adding features during a sprint is not entirely in compliance with scrum methodology, as sprint planning and locking upcoming tasks should be bone based previous burndown charts. However sometimes altering the sprint backlog and development team during a sprint can be the right thing to do in terms of *resource management*.)

In the interest of keeping this model as light as possible, the documentation for risk managment and cost estimation is optional.

Rapid software development life cycle model in small projects - A survey

A survey to assess the developer experience, project management and the efficiency of the proposed life cycle model.

The following survey is divided into three sections: Individual Experience, Project Management and Finale. The survey is comprised of entirely multiple choice questions. There is also free space for comments in the finale section.

1. **First of all a question about the overall production volume in SOHO. Based on your own estimations, could the model...**
   Mark only one oval.
   - A. Increase the production volume?
   - B. Decrease the production volume?
   - C. have no noticeable effect on this?

Individual and group Experience

How could model affect the social construct and organizational creativity in SOHO. (Option 3 means neutral)

2. **How does the model come across on the instruction sheet?**
   Mark only one oval.

	1	2	3	4	5	
Very light						Very rigid

3. **On the individual level, could the model...**
   Mark only one oval.
   - lower the overall performance?
   - increase the overall perfomance?
   - have no noticeable effect on this?

4. **On the group level, could the model...**
   Mark only one oval.
   - lower the overall performance?
   - increase the overall performance?
   - have no noticeable effect on this?

5. **The initiation. Kick-off & planning. Is the concept of having an initial planning committee cumbersome and straining?**
   Mark only one oval.

   - Yes
   - No
   - Unable to determine.

6. **Is the added monkey testing cumbersome and straining?**
   Mark only one oval.

   - Yes
   - No
   - Unable to determine.

7. **Is the added project management cumbersome and straining?**
   Mark only one oval.

   - Yes
   - No
   - Unable to determine.

Project Management

Models ability to adopt and enhance practices in formal project management. (Option 3 means neutral)

8. **Resource and personnel management**
   Mark only one oval.

	1	2	3	4	5	
Weak						Strong

9. **Risk and Uncertainty management**
   Mark only one oval.

	1	2	3	4	5	
Weak						Strong

10. **Project Scope and estimation management**
    Mark only one oval.

	1	2	3	4	5	
Weak						Strong

11. **Management of project portfolio. Could the model help to manage projects systematically?**
    Mark only one oval.

	1	2	3	4	5	
Weak						Strong

Finale

Final questions and free space to give feedback in greater detail.

12. **If the head of project management in SOHO called a vote to decide, whether to use this model in future sprints. Would you vote...**

Mark only one oval.

- ◯ Yes?
- ◯ No?

13. **In terms of making alterations to this model. Could the model be improved?**

Mark only one oval.

- ◯ The model could be improved significantly.
- ◯ The model could be improved slightly.
- ◯ The model is good as it is.
- ◯ No comment.
- ◯ The model's core structure and foundation needs to be designed from ground up.

14. **Free space to provide comments. Suggestions, Strengths, weaknesses etc.**