

PAPER • OPEN ACCESS

Command and Control vs self Management

To cite this article: D Nicula and S S Ghimii 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **514** 012039

View the [article online](#) for updates and enhancements.

Command and Control vs self Management

D Nicula¹ and S S Ghimiși²

¹The Bucharest University of Economic Studies, România

²Constantin Brâncuși University of Târgu Jiu, România

E-mail: nicula.dana@gmail.com

Abstract. It's estimated that in 2017 there were 111 billion lines of new software code generated by developers and most lines were written within a team. Producing software is a creative activity and therefore very different from mass-production lines, because each algorithm tries to solve a different problem. For many years, traditional system development life cycle has been used as the main algorithm for how programmer teams should work to deliver software products. Waterfall methodology is successfully used in simple, unchanging projects but its main drawback is how it handles change. Scrum encourages the creativity of each member of the team, allowing all of them to take on the role they want during the project. Each person can be a developer, a tester, a designer, depending on the responsibilities he takes between two sprints. Considering the diversity of IT projects, it makes sense to understand in what kind of projects we should use the Scrum methodology and where there is a need for more explicit management and control.

1. Introduction

Organizations are trying to innovate rapidly in response to changing market conditions, customer needs, and emerging technologies. Companies live and die on their ability to discover new businesses and create ongoing value for customers. Competitive pressure is increasing, fuelled by rapid changes in technology and society. The business world is moving from treating IT as a utility that improves internal operations to using rapid software and technology-powered innovation cycles as a competitive advantage.

The traditional program and project management models' companies have used for IT are unsuited to rapid innovation cycles. However, they are deeply embedded in the way we manage everything from operations and customer service to budgeting, governance, and strategy, because most companies avoid taking risks and enjoy micromanagement and control. A software process (also known as software methodology) is a set of related activities that leads to the production of the software.

These activities may involve the development of the software from the scratch or modifying an existing system. All companies aim to optimize the process of transforming business ideas and requirements into an application that includes all the necessary functionalities, achieves the business needs and is useful and easy to use by end users.

Selecting a Software Development Life Cycle (SDLC) methodology is a challenging task for many organizations and software engineers. What tends to make it challenging is the fact that few organizations know what criteria to use in selecting a methodology to add value to the organization. Fewer still understand that a methodology might apply to more than one life cycle model. Before considering a framework for selecting a given SDLC methodology, we need to define the different types and illustrate the advantages and disadvantages of those models.



2. Traditional and Agile Development

In this article, we will explore two of the software development models and the advantages and disadvantages of each one in order to give a guidance on when and why to use them. To begin with, the waterfall approach is the earliest one and most widely known method that was used for software development.

The Waterfall Model is a linear sequential flow, in which progress is seen as flowing steadily downwards through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete.

The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. As shown in the picture below, the sequential phases in Waterfall model are requirements analysis, system design, implementation, testing, deployment and maintenance. All possible requirements of the system that should be developed must be captured in the first phase and listed in a requirement specification document.

The requirement specifications from first phase are studied in the second phase and based on them, the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

With inputs from the system design, the system is first developed (implemented) in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, referred to as unit testing. All the units developed in the implementation phase are integrated into a system after the individual testing of each unit. Post integration the entire system is tested for any faults and failures.

Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market. There are some issues which might come up in the client environment. To fix those issues, several patches can be released. Also, to enhance the product some newer versions with added functionalities are released.

Maintenance is done to deliver these changes in the customer environment. All these phases are cascaded to each other in which progress is seen as flowing steadily downwards through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model"(Figure 1).

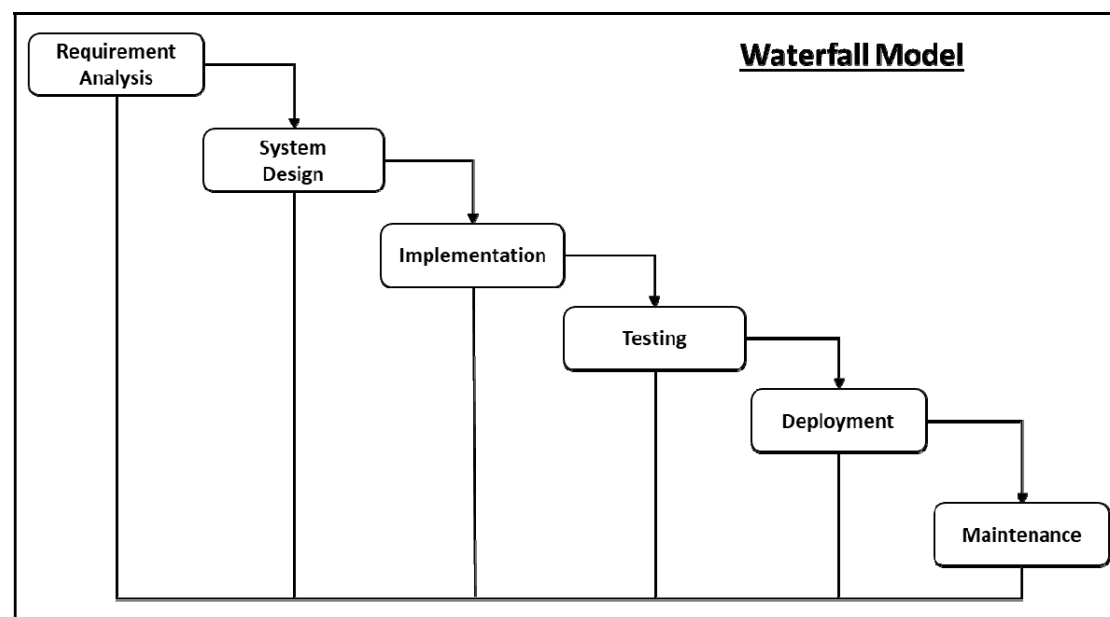


Figure 1. Waterfall Model

Since Waterfall has been widely used for software development in the last decades, we would like to emphasize that there are some situations where the use of this model might still be the most appropriate one.

For example, in short projects with stable product definition, where ample human resources with required expertise are available and where requirements are very well documented, clear and fixed, waterfall still turns out to be the appropriate methodology to use.

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage. While the tools of project management, such as plans, budgets, and schedules, are necessary to control the complexity of IT initiatives, no team member was ever inspired by a Gantt chart. When projects travel beyond the realm of the known and require innovative ideas and deliverables, planning becomes less relevant and inspirational and enabling leadership becomes a key success factor.

In the past few years, the diversity and complexity of the projects has greatly increased, and a large implementation time is no longer serving in too many business situations. If development takes too long, the final solution may be outdated. Over the past 10 years, the outsourcing of software development services has been the norm. Contracts between developers and beneficiaries contained very tough clauses on how to intervene with changes to the initial requirements. Changing requirements has become a challenge both in terms of associated costs and sometimes in prolonged delivery times. For this reason, the current trend of large organizations is to give up outsourcing of project development and to hire their own developers.

Certainly, hiring developers and maintaining previous workflows could only solve one of the above-mentioned issues, namely to reduce the costs of changing requirements. The delivery term of the application can be the same regardless of whether working with a team of internal programmers or a partner firm. To reduce project risks, agile development methodologies have been adopted.

A good software engineer should have enough knowledge on how to choose the SDLC model based on the project context and the business requirements. According to Mike Cohn [1] the most suited projects for the agile process are the ones with challenging deadlines, a high degree of complexity, and a high degree of novelty for the team. If it's something the team has done before over and over then they don't need to use an agile approach. On average 65% of requirements change during software development causing waterfall projects to have an 14% worldwide success rate during the past decade. (Jim Johnson, Standish Group, 2011).

The agile methodology is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds.

These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing and acceptance testing. At the end of the iteration, a working product is displayed to the customer and important stakeholders. Agile is based on the adaptive software development methods, whereas the traditional SDLC models like the waterfall model is based on a predictive approach.

Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle. Predictive methods entirely depend on the requirement analysis and planning done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization. Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed.

There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future. Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development

environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location [2].

Scrum is currently one of the most studied and used agile methods, because of both its novelty and the assumption that this method is able to improve project productivity. The method uses an empirical process based on flexibility, adaptability and productivity, different than the rigid and very focused on planning way of developing with waterfall. Scrum puts an emphasis on communication and collaboration, functioning software, team self-organization, and the flexibility to adapt to emerging business realities.

It sets high-frequency management activities in order to track real-time problems during a project development process. In traditional management, there is a heavy command-and-control mentality with clearly defined roles and responsibilities for every member of the team, whereas in many agile methodologies, especially Scrum, there is no real leader. Basic collaboration mechanisms are required in order to make it work. The environment must be set up so that people have an incentive to help one another in order to optimize the whole, rather than optimizing their individual niche at the expense of the larger community. Failure to do this will prevent individuals and companies achieving the full benefits of Scrum [3].

Seen as a process, Scrum (Figure 2) takes a project and breaks it down into regular cycles referred to as Sprints. Sprints generally last between 1 and 2 weeks and involve functional teams of 5 to 9 people. Each sprint encompasses some segment of work that the overall team has agreed to accomplish. The teams that work together on each sprint are generally cross-functional and self-managing. They take a certain task that has been established in the “product backlog”, similar to a list of goals for the product, and take it to a deliverable phase.

A crucial aspect of Scrum is that each sprint ends in some sort of deliverable, whether that is a formulated idea, a useful simulation, or all the way to a fully designed component. This enables iterative feedback loops in the design process and keeps progress from being made in the wrong direction.

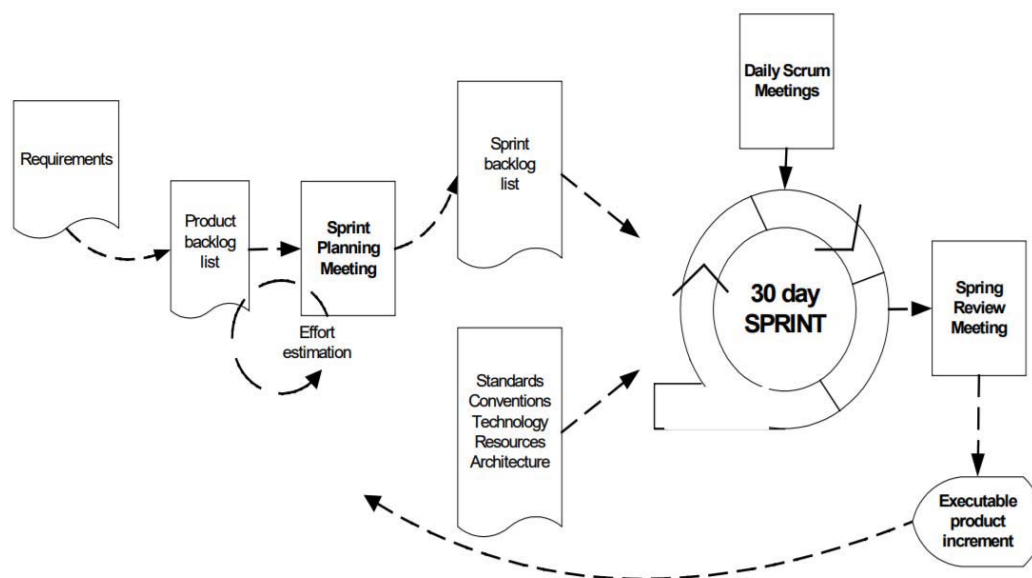


Figure 2. Scrum Model

Inside each Sprint there is a mini waterfall project since each executable product increment needs minimal requirements, design, implementation and testing. What makes Scrum different than any other methodology is the combination between the principles on which all agile methodologies are

based, the limited number of roles within the team and all the ceremonies that a scrum master makes sure the team applies rigorously.

There is a document called Manifesto for Agile Software Development where twelve principles are listed. One of them says that “The best architectures, requirements, and designs emerge from self-organizing teams” [4].

Therefore, Scrum teams have the freedom to choose how to accomplish their work, rather than being directed by other stakeholders or managers. Members are empowered by the organization to structure and manage their own work. If they have the knowledge to do so, each person can be a developer, a tester, a designer, depending on the responsibilities he takes between two sprints. No one tells the development team how to turn Product Backlog into increments of potentially releasable functionality. Self-managing teams offer potential advantages over traditionally managed teams because they bring decision-making authority to the level of operational problems and increase the speed and accuracy of problem solving (Moe et al. 2009) [5].

A risk factor not to be neglected when applying the scrum methodology is related to staff fluctuations. Given that the requirements are poorly documented, changes within the team may cause the project to be affected.

Also, for the good progress of the project, it is important for team members to work well together and to respect each other. There can be a conflict between the traditional, tool-and-process focused style of project managers and the agile approach. Many project managers have difficulty stepping away from the classical techniques, such as scope, budget, and schedule planning, and migrating into a world of changing requirements, evolving expectations, and self-directed teams.

One of the success stories for the implementation of Scrum, comes from Spotify. They have grown from 30 to 250 employees three years after adopting the agile methodology, reporting a huge increase in employee satisfaction. According to Statista [6], their revenue also increased from 1 billion euros in 2013 to 4 billion euros in 2017. The engineering team from Spotify took advantage of the flexibility and experimentation that Agile encourages and decided to rewrite some rules that apparently prevented them from operating more effectively. They have created their own agile methodology, which was strongly influenced by the values of their company. One of the first changes they implemented was in the terminology used.

The terms Scrum Master and Teams were replaced with Agile Coach and Squads, which consisted of no more than eight people who were given full autonomy to operate as a cross-functional and self-organising unit, having the ability to decide what to build, how to build it and how to work together while building it.

People at Spotify tried to create a culture where employees mutually respect themselves and share credit for their achievements so that each person can feel like a valued member of the team.

Finally, and maybe the most important change of all, is the fact that Spotify encourages finding ways to ensure fast failure recovery rather than simply avoid failing at all. Some squads even use a fail wall where they publicly share their failures and whenever there is a failure, the squad conducts retrospectives which allows it to capture any learnings from the experience. Besides removing fear, experimentation also promotes a data-driven approach to taking decisions, rather than having decisions taken on the basis of personal opinion, ego or authority.

In the authors' opinion, Scrum application brings a large number of benefits to employees, managing to get them to leave the paradigm of executants and to become idea generators. It most likely encourages people to learn new things and allows them to have less aversion to risk since failure is seen as a learning experience. If the business and technical requirements received from hierarchical superiors are incomplete or even erroneous, the agile workflow allows the team to challenge the proposed solutions to a greater extent and even to create alternative ones. When applying Waterfall, the large number of requirements received simultaneously is overwhelming and the focus of the development team can be to write the code rather than to consider end user experience. A smaller number of requirements can be examined more carefully compared to hundreds of requirements that are simultaneously sent on the way to development.

The evolution from traditional project management to agile techniques requires an evolution for the project management as well, from authoritarian, top-down management to collaborative, influence-based leadership. The ability to migrate from a world of tightly-bounded scope and schedules to a world of change, vagueness, and uncertainty is the key distinguishing factor for successful agile projects and project leaders.

Every agile expert agrees that agile projects are not for novices. Process has its place in bringing order from the chaos of complex IT projects but creating the right atmosphere and staffing it with the right contributors is the foundation of a high-achieving agile team [7].

3. Conclusion

This article is only a starting point for our study. Anecdotal evidence suggests that agile methods are effective and suitable for many situations and environments. However, at present very few empirically validated studies can be found to support these claims. The existing evidence consists mainly of success stories from practicing professionals. Although these provide valuable information about practical applications, empirical studies are urgently needed for evaluating the effectiveness and the possibilities of using agile software development methods, especially in very large projects.

The goal of this type of research is to enable software professionals, projects and organizations to choose and to apply the right method at the right time and to take into consideration human motivation as well.

4. References

- [1] Cohn M, 2011 Deciding what kind of projects are most suited for agile *Mountain goat software* <https://www.mountaingoatsoftware.com/blog/deciding-what-kind-of-projects-are-most-suited-for-agile>.
- [2] Dingsøyr T, Moe N B, Fægri, T E and Seim E A 2018 Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation *Empirical Software Engineering* **23** pp 490-520 <https://doi.org/10.1007/s10664-017-9524-2>
- [3] James M, Walter L 2010 *Scrum Reference Card*, <http://scrummethodology.com/>
- [4] <http://agilemanifesto.org/>
- [5] Moe N B, Dingsøyr T, Dybå T 2009 Overcoming barriers to self-Management in Software Teams. *IEEE Softw* **26** pp 20–26
- [6] Felix Richter 2018 - Spotify's Losses Widen as Royalty Costs Pile Up <https://www.statista.com/chart/4894/spotify-revenue-vs-costs/>
- [7] Freedman R 2009 Agile teams focus on the people rather than the process *Tech decision market* <https://www.techrepublic.com/blog/tech-decision-maker/agile-teams-focus-on-the-people-rather-than-the-process/>