Universidade do Minho
Escola de Engenharia

Bruno Afonso Teixeira

Agile and Traditional Project Management: bridge between two worlds to manage IT Projects

Outubro de 2013

Universidade do Minho

Escola de Engenharia

Bruno Afonso Teixeira

Agile and Traditional Project Management: bridge between two worlds to manage IT Projects

Dissertação de Mestrado
Mestrado em Engenharia e Gestão de Sistemas de Informação

Trabalho efectuado sob a orientação do
Professor Doutor Ricardo J. Machado
Departamento de Sistemas de Informação

Outubro de 2013

DECLARAÇÃO

Nome: Bruno Afonso Teixeira

Endereço Eletrónico: bruno.teixeira@gmail.com **Telefone:** +351 919055764

Nº. do Bilhete de Identidade: 11672865

Título da Dissertação de Mestrado: Agile and Traditional Project Management: bridge between two worlds to manage IT Projects

Orientador: Professor Doutor Ricardo J. Machado

Ano de conclusão: 2013

Designação do Mestrado: Mestrado em Engenharia e Gestão de Sistemas de Informação

É AUTORIZADA A REPRODUÇÃO DESTA DISSERTAÇÃO, APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

Universidade do Minho, ____ / ____ / ________

Assinatura: ____

Acknowledgements

This thesis is the result of my life experiences to this day, both as a professional and as a human being. The fervour for the art of Project Management was a by-product of those experiences, from the novelty to the learning, from the experimentation to the personal adaptation… for all that trial and error in such mist, some deserve a special thank you.

To all employers and colleagues, past and current, for all the experienced opportunities, knowledge learned, frustrations overthrown, shared missteps and successes.

To my supervisor Professor Doutor Ricardo J. Machado for not only the support in regard to this document, but especially or all the learning challenges and opportunities granted since acquainted.

To the Project Management community, Agile and Traditional, for sharing knowledge and pragmatic lessons learned to all the ones eager to learned.

To my mother, brother, uncle and grandparents for all the support and understanding in my deranged life quest

To Liliana, for all her love and patience

Abstract

Agile frameworks have won countless followers in the software engineering community during the last decade. Based on a totally new approach regarding already existent frameworks, Agile frameworks are held as more efficient in many levels. However, some say these new practices are nothing but an attempt at corroborating ad-hoc practices, resulting in increased risk in projects and wounding the organization in terms of operational costs, due to an insufficiency of plan or strategy regarding projects.

This challenge takes an even more difficult form when regarding organizations with considerable dimensions, which use Agile frameworks to manage dozens of simultaneous projects that share the same company resources. Using Agile, in its purest form, may negatively affect predictability, risk and project planning, therefore affecting the possibility of achieving an efficient Portfolio Management in the performing organization.

The challenge is, how can we have the work-management oriented Agile benefits at the project Team level, without compromising the equally essential traditional Project Management practices, commonly used to manage the company portfolio and aligning projects with the company's strategic planning.

This dissertation has as its main goal showing that some Agile practices, in this specific case the SCRUM and KANBAN frameworks, have a major benefit at operational level (Micro-Management) and can be interpreted as a complement to some traditional Project Management practices (Macro-Management), instead of being a mere substitute hailed as a silver bullet.

Keywords: SCRUM, KANBAN, PMBOK®, traceability, Macro-Management, Micro-Management

Resumo

As frameworks Agile ganharam inúmeros seguidores na comunidade de engenharia de software durante a última década. Com base numa abordagem totalmente nova em relação a frameworks já existentes, são tidas como mais eficientes a vários níveis. No entanto, alguns dizem que essas novas práticas não passam de uma tentativa de corroborar práticas *ad-hoc*, resultando num aumento de risco nos projectos e lesando a organização em termos de custos operacionais, devido um insuficiente planeamento ou estratégia em relação a projectos.

Este desafio assume uma dimensão superior quando se trata de empresas de grande dimensão que utilizam frameworks Agile, para gerir dezenas de projectos em simultâneo, compartilhando os mesmos recursos. Usar Agile, na sua forma mais pura, pode afectar negativamente a previsibilidade do risco e a capacidade de planeamento de projectos, afectando, portanto, a possibilidade de gerir a carteira de projectos da empresa de forma mais eficiente.

O desafio pode traduzir-se da seguinte forma: como poderemos ter os benefícios práticos da gestão Agile a nível das equipas de projecto, sem comprometer as práticas tradicionais de Gestão de Projectos, igualmente essenciais, comummente usadas para gerir o portfólio da empresa e alinhar os projectos com o plano estratégico.

Esta dissertação tem como principal objectivo mostrar que algumas frameworks Agile, neste caso específico SCRUM e KANBAN, podem trazer grandes benefícios a nível operacional (Micro-Management) e podem ser vistos como um complemento para algumas práticas tradicionais de Gestão de Projectos (Macro-Management), e não apenas como panaceia para quaisquer problemas.

Palavras Chave: SCRUM, KANBAN, PMBOK®, traceability, Macro-Management, Micro-Management

Table of Contents

Index of Figures

Index of Tables

Acronyms Dictionary

ANSI – American National Standards Institute

ASAP – As soon as possible

CTIA – Computer Technology Industry Association

FDD – Feature-Driven Development

INVEST - Independent, Negotiable, Value to the users or customers, Estimable, Small, Testable

IPMA - International Project Management Association

IT – Information Technology

ITTO – Inputs, Tools, Techniques, Outputs

JIT – Just-in-Time

PMBOK® – Project Management Body of knowledge[1]

PMI® – Project Management Institute[2]

PMO – Project Management Office

PRINCE2 – PRojects IN Controlled Environments

ROI – Return On Investment

RUP – IBM Rational Unified Process

USA – United States of America

WBS – Work Breakdown Structure

WI – Work Items

WIP – Work-In-Process

WW2 – Second World War

XP – Extreme Programming

[2,2] “PMI” and “PMBOK” are registered marks of Project Management Institute, Inc

1 Introduction

Software professionals live troubled times. Times of constant change, of daily emerging new technologies, of insane time-to-market dead-lines, of constant strategic market and organization repositioning. Agile brings some simple answers to some of these needs, it delivers lightning-fast results with minimal initial costs.

“A company needs to be adaptable in order to survive in today’s jungle” is a quote from a SCRUM consultant, one that strikes directly in the troubled brain of software professionals worldwide. During the last 10 years, these consultants became professional Agile sellers, and helped to claim the “I came, I saw, I conquered” that characterises Agile nowadays.

Almost manipulated, the poor troubled software professionals adopt Agile for the sake of company future, and focus their attention on the short-term or even day-to-day shorter-term. The importance of “the tomorrow” supplants the importance of long-term strategic planning. Practices like software architectural integration and other commonly known good practices become secondary or even disappear. Companies end up with teams focused only in the Sprint execution, and they forget the other equally important areas of the project. This becomes even more serious when regarding company Project Integration and Portfolio Management.

One fact is undeniable, Agile is clearly the hype trend of present days, and any software professional can agree that Agile brought a different perspective to Software Development and, unfortunately due to its simplicity, it can be easily perverted and bring issues that are now considered as Agile endemic. Indeed, simplicity is the key in Agile, but software professionals commonly use the simplicity concept to cut “boring” regarded practices. Unfortunately, some of these “boring” practices are the core base for Project Management, Software Engineering or Product Integration and Maintenance. A company cannot jeopardize its future by losing sight of those areas when adopting Agile.

1.1 Motivations

Nowadays, software houses struggle with constant change, volatile requirements prioritisation and insane time-to-market deadlines. They have to find ways to balance these factors in the best way possible or they may go out of business.

Agile frameworks brought some new concepts to help put the old waterfall model to rest, a change-oriented and collaborative environment which focuses in on adaptability over predictability and people over process. The essence is simple, incorporate proven Software Engineering techniques without the overhead and restrictions of a traditional development methodology. This is clearly the way software professionals see the issue, but how does the management sees it? Maybe some of the so called overhead is not overhead. Some developer's definition of overhead can be, in fact, the primary top management tool.

Delivering software that serves the customer's business needs is an objective, but maybe it is not the primary one. Maybe the company needs to compromise product quality or customer satisfaction to some acceptable threshold in order to assure other company commitments. Or maybe the market is not ready to pay the additional quality of such a product, choosing instead a less expensive alternative. As Philip Crosby said, "Quality is free", but the company has to plan how much quality should products have and how much should this quality cost in order to keep the product's marketability and sustain the company.

One fact should not be forgotten, in the end of the day we still work in a company, and for the company every cost needs to be accounted for. Our project, Agile or not, is part of a bigger picture.

1.1.1 Agile vs. critical systems

As referred, some Agile principles can be misleading, and can easily lead to an incorrect method, practice or principle adaptation. Both computer games and military-grade weapon control software have very different critical aspects, but nevertheless both can be developed using Agile, even though probably with very different approaches.

Critical software systems need to be implemented with a specific degree of planning, design, architecture, documenting, testing, etc, all in the specific timeframe and in constant alignment. Such projects should not be run in an environment that embraces "responding to change over following a plan" or "working software over comprehensive documentation".

1.1.2 Agile vs. application and product maintenance

Again, Agile principles can be misleading - some companies adapt Agile and agree on stopping documentation of every kind. Again, "working software over comprehensive documentation", right? This should not be taken literally.

Linked to the critical systems issue, imagine a case where some software is to be created with a short 3 months lifespan for some internal operational support, with no real ROI behind it. This is non-critical, and yes, documentation can be reduced due to a short maintenance period and the product being non-critical. This is a clear case of Agile in its best: keeping it simple.

Now let's think back to the critical system, one that is developed, tested and released to production and is vital to the company core business and will be used for decades to come. This product will need constant maintenance (run the business), change to integrate new company solutions (change the business), etc. Some degree of documentation, requirements traceability, architectural design, test cases, etc, will have to be produced during development in order to ensure the future teams will have enough support to conduct maintenance and upgrade the application. This cannot be documented in user stories written in yellow post-its, or be totally supported by the non-documented knowledge of a 6 person Team that can be disbanded at any time.

1.1.3 SCRUM trend

Quality may be free, but Agile is not free… At the present day, we can find 808 SCRUM training courses and at least 5 global events in the SCRUM Alliance site. From official certifications to coaching, it has become a multi-million dollar business for trainers, authors and consultants. We can probably find an event, gathering, seminar, workshop or something similar in our area during the next month. Let's say we attend. We will end up with more questions than answers and with a stack of business cards from a bunch of consultants. The feeling is that no one really wants to help us becoming Agile for free, and this is far from being easy to implement in a traditional software company.

1.1.4 PMI Agile Certification program

Agile trends are so important today that even the Project Management Institute® created a new Agile based certification in their certification program: the PMI-Agile Certified Practitioner (PMI-ACP^{SM}). Quoting from the PMI® site: "Agile is a topic of growing importance in Project Management. PMI® market research shows that Project Management practitioners are embracing Agile principles and practices as a technique for successfully managing projects."
This manifests PMI®'s awareness of the importance and hype of Agile practices. The Software Project Managers community always regarded the PMBOK® as difficult to apply to the software industry, especially in small projects. Does this mean that a new Agile PMBOK® may be created? Only time will tell.

1.1.5 CMMI for Development v1.3

The new CMMI for Development v1.3 model is a good example of Agile hype acknowledgement. The new model, released on November 2010, has dozens of references to Agile practices, approaches and environments. Although the major updates from v1.2 to v1.3 were mostly focused on Maturity Level 4 and Maturity Level 5 Process Areas, Agile important references had to be included in the entire model on different levels. Again, another world-renowned engineering model is accepting the Agile inevitability.

1.2 Challenge (the Problem)

So, what is the challenge here? We know that some Agile practices bring benefits, but they can be misinterpreted and lead to misleading principles on both project and product success and quality. On the other hand, traditional Project Management frameworks can be too heavy for software Project Management, but have the benefit of reminding software professionals that a project or resulting products are more than only coding.
So, the challenge is: where do we draw the line between Agility and Traditional Project Management?

If a company has 100 projects running at the same time, it cannot afford to lose money in an overflow of agility. Yes, we want some of the benefits of agility, but that agility needs to stop went it starts to become a risk to company projects.

So, we have to draw a line, the line that makes us remember that agility is a tool to be used in conjunction with others, because we have a company to run and maintain, and one project cannot be seen as the whole puzzle, but only a small piece in it. The actual puzzle has 100 other more pieces, and all of them have to fit together in the best way possible. In the company's point of view, the best path is the one that delivers on time, on budget and with the quality grade that makes customers pay and come back for more.

Companies are more than the software they produce. Strategic planning, financial cost control and other business management mouthfuls need to be addressed correctly or the company may go out of business. Software professionals tend to forget these so called management details.
Contract Management, Risk Management, Procurement, application lifecycle, maintenance costs are some of the pieces that Agile sees as overhead to the project, but traditional Project Management regards them as constrains to be managed by the Project Manager and the performing organization together.

Because literature and Lessons Learned claim benefit from both applications, the challenge can be summarised in one question: can we use agile frameworks to manager the development team and traditional Project Management frameworks the remaining Project Management activities not addressed by the first ones?

1.3 Objectives

The proposal for this dissertation is to find a bridge between Agile and traditional Project Management concepts. A bridging between two of the most regarded Agile frameworks and one of the most proven Project Management framework: SCRUM, KANBAN and PMBOK®.

The objective is to study the concepts, strengths and weaknesses of both Agile empirical-based approaches and combine them with the detailed defined, proven and world-recognized PMBOK®. This proposal is based on the possibility that boundaries can be established regarding Project Management practices and benefits will come from the application of different frameworks in specific projects or performing organizations.

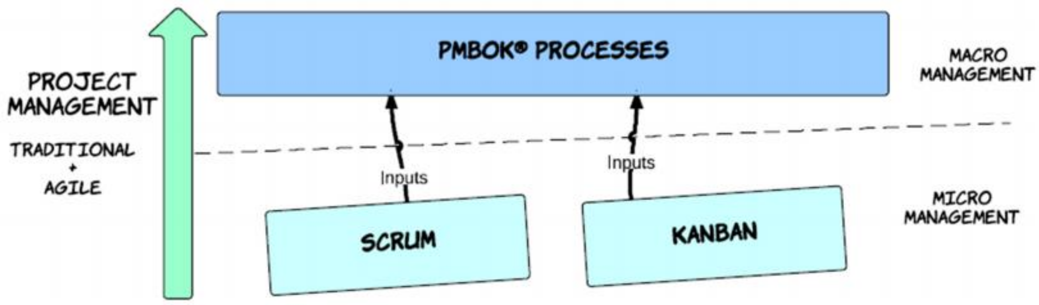


Figure 1 - Micro and Macro-Management proposal

The Micro-Management layer will be addressed by SCRUM and KANBAN best practices and will ensure an Agile-oriented management at Software Development team level. The key is maintaining a self-organized and cross-functional Team with focus on participation and collaboration on the project objectives at that lower level.

The Macro-Management layer will, in turn, be addressed by the remaining non-Agile areas that cannot be ignored as project constrains: Risk, Procurement, Integration, etc. The focus is the PMBOK® Knowledge Areas since they represent traditional Project Management areas that cannot be left out. Additionally, Macro-Management can also supply a Project Management Office with the information needed to do Program and Portfolio Management. An existing Project Management Office will gather the information reported by the Macro-Management layer and will ensure that the company Program and Portfolio are aligned with the company strategic planning.

The nature of such a proposal is very different from a quantifiable or mathematical proven result. This dissertation is focused on shedding some light in questions already asked by many software and Project Management professionals. The purpose is clearly a research oriented exercise, with a focus on documenting information that was found missing in the present literature. Although Project Management is regarded as science by many professional Project Managers, unfortunately its research results are not easily measured, and sometimes research exercises lead to other future breakthroughs and additional research instead of a final and irrefutable conclusion.

In the end of the working period, some conclusions will be drawn from the research, but it cannot be regarded as proof of concept or an irrefutable Project Management tool or framework. Then again, the result will hopefully serve the purpose of opening paths to additional similar exercises, and may serve the purpose of clarifying the relation between Agile and traditional methods in both project and performing organization levels.

Summarizing, the expected results may be recapitulated in the following major objectives:

- Analyse the SCRUM and KANBAN frameworks, its benefits and weakness regarding Project Management practices, focused on small and contained settings (Micro-Management oriented)
- Identify a set of recommendations for applicability of both KANBAN and SCRUM regarding different projects, scope, teams and/or performing organizations.
- Mapping both Agile frameworks with the PMBOK® framework to trace alignment of processes and best practices that may be found. This will result in an identification of similar areas, input contact points and dependencies between SCRUM/KANBAN (Micro-Management oriented) and PMBOK® processes (Macro-Management oriented).

The final result will be a compilation of the above research information and the conclusions drawn from such an analysis, and outlined would be the foundation principles for creating a mapping for tailoring Macro/Micro Project Management practices. This mapping will help choose the Micro/Macro-Management approach, practices or process to use in real projects, thus encountering a comprehensive way for managing a company portfolio with the benefits of all the frameworks.

1.4 Research Method

In order to draw a plan, every research exercise should follow a Research Methodology, one that attests to scientific integrity and assures the resulting benefits.

Framing the following exercise in a known scientific research approach is not an easy task. The nature of this research is based on analysis of industry contextual data, defined models and personal experience, with a result that cannot be simulated, experimented or subject to a field study during the allowed dissertation timeframe.

Since this theory lacks any clear way to be verified, we cannot take a Design Science Research or a Development Research approach. Both promote some sort of Validation that we simply cannot execute due to the complexity of the Validation process in the amount of time given [HevCha2010].

Having in account the nature of the exercise, we decided to adapt the Grounded Theory, in which theory and models are generated inductively from the analysis of contextual data. Grounded Theory involves the discovery of concepts and hypotheses as theory emerges from data [Villiers2005].

Our adapted Grounded Theory will consist in analysing KANBAN and SCRUM, based on defined models and personal experience, and obtain some shared concepts (categories), which will then be used to do a comparison between both regarding concepts and applicability. This result will then be used to map the PMBOK® processes possible inputs, which will then return possible relationships between the triplet. These input relationships will be represented as diagrams and analysed in a quantitative view.

Said approach is basically a finding of shared concepts and input patterns, first between KANBAN and SCRUM, then between both and the PMBOK® processes, all based on community defined models and personal experience. As a final result we will have not only the quantitative data but also all the analysis that originated it.

Validation of the result will be added to the Future Work propositions.

1.5 Document Structure

This document is divided in 5 chapters:

Chapter 1 "Introduction": the current chapter, basically a presentation for the Motivations and Objectives for this dissertation. Additionally, it also contains some detail regarding the Research Method used and Document Structure.

Chapter 2 "Project Management and Agile Methods": a prologue for the following themes, introducing the History of Project Management, presenting an analysis for Project Success and Project Failure, the Origins of the Agile Manifesto and an introduction to the SCRUM, KANBAN and PMBOK® frameworks. Conclusions are drawn in the end of the chapter.

Chapter 3 "KANBAN vs. SCRUM": the core description of SCRUM and KANBAN. It is detailed to some extent, using the same approach on both, on origins and shared concepts. Additionally, it contains a face-off comparison and some information regarding adaptation, prescriptiveness and field application scenarios. Conclusions are drawn in the end of the chapter.

Chapter 4 "Mapping PMBOK® Processes with Agile Practices": the cornerstone of this exercise. In this chapter, we will explain the Macro/Micro-Management approach and map the concept to the SCRUM, KANBAN and PMBOK® triplet. The result will be Input Tailoring Diagrams and one Input Tailoring Matrix that reflects our findings, ones that will be analysed in the same chapter. Conclusions are drawn in the end of the chapter.

Chapter 5 "Conclusions": summarizes the overall exercise, conclusions and the objective's successful completion, while laying down the ground for possible future work.

2 Project Management and Agile Methods

2.1 Introduction

Project is a concept familiar to mankind for a long, long time, but even so it can be almost invisible if not seen through the correct spectrum. Since the dawn of mankind, projects have been initiated, planned, executed, cancelled, changed, closed, etc, and since then to the present day, from construction to science, using engineering and finance, relaying on technology or craftsmanship on planet Earth or beyond, one fact is undeniable: every project is unique.

With project uniqueness, another challenge is clearly revealed: how should a project be managed?
As ancient as the projects themselves, the way to manage them also evolved. From past eras to present day, Project Management is a continually evolving discipline.

2.2 History of Project Management

Both the Great Pyramid of Giza (2560 BC) and the Burj Khalifa (2009 AD) are man-made projects, but with no doubt using very different management approaches. With almost 5 millennia apart, one built by slaves controlled by the whip of their masters, the other built by outsourced workers controlled by computerized engineering planning, both constructions are proudly standing tall.

What changed in 5000 years?

Despite the fact projects are timeless as is the science of Project Management, the discipline of Project Management as we know it is far too recent. Although Egyptians and other civilizations no doubt used the science of Project Management, the discipline itself is not mentioned in ancient writings. Project Management became a reality in the beginning of the 20th century by the hand of the Frederick Taylor (1856-1915). The inscription on Taylor's tomb in Philadelphia attests to his place in the history of management: "the father of scientific management." Scientific Management is the name of Taylor’s masterpiece, a management theory that analysed and synthesized workflows, introduced resource allocation and work-breakdown principles and is the base of future management concepts. Taylor’s approach is also often referred to as Taylor's Principles or Taylorism [Taylor1911].

Without knowing it, Taylor’s work theories led to the fundamentals of modern management concepts, and as all great artists, his work has inspired others, such as Henri Fayol (1841-1925) and Henry Gantt (1861-1919). Fayol, in 1916, published the book “*Administration industrielle* *et générale*" in which he describes the basic functions of management: planning, organizing, commanding, coordinating and controlling. His work shortened the path to modern Project Management.

Gantt extended Fayol's work, primarily focusing on planning and controlling techniques, and created one of the most famous tools in Project Management: the Gantt Chart. The usage of such a tool is reported since the First World War, when it was used to plan and control the construction of warships [Chiu2010].

The industrial revolution and its quest for productivity improvements brought a huge success to Gantt's most famous tool. His diagrams proved to be such a powerful tool for managers that they remained almost unchanged for nearly a hundred years. In the end of the 20th century some changes were introduced regarding dependencies between tasks, and it became the technique behind the majority of software scheduling tools.

2.2.1 Modern Project Management

First forged during the heat of the industrial revolution, evolving during the demanding production times of WW2, Project Management as we know it only appeared in the middle of the 20th century. The needs had changed and it became obvious that complex projects needed to be managed in a more efficient way. During times of war, meeting the time-to-market, or in this case, the time-to-battlefield, became life-threateningly crucial, and in face of such odds, there came a necessity to improve current management styles, to increase the number of various war machines delivered to the battlefield. Benefits of Project Management became clear to all and good practices spread rapidly to several different industry types. The world was changing rapidly and business leaders were seeking new ways to handle change and growing markets in a more and more competitive world and global economy.

With a growing set of good practices and followed by effective results in the field, Project Management was disseminated by journals, conferences and seminars in the 1970s. Standardization of the Project Management Discipline was the next logical step, in order to ensure a more common usage and global expansion.

Standardization of Project Management gave its first steps during the 1980s with the Project Management Institute® (PMI®) and, later, the Association for Project Management (APM). Both defined a body of knowledge which addressed the Project Management discipline good practices and guidelines (e.g., PMBOK®, APMBOK). However, both approaches were not focused to any particular industry, which made some of the content not applicable to all projects regardless of area. Nevertheless, this was the first step to generate a common base to a standard of generic Project Management [Charvar2003].

2.2.2 Concept of Project and Project Management discipline

There are as at least many project definitions as there are Project Managers. If we talk to project team members we will have the same dilemma. A definition can came in form of a text book quote or just a practical “bunch of stuff that the customer wants done ASAP”.

The PMBOK® defines a project as a “temporary endeavour undertaken to create a unique product, service or result”, and defines Project Management as “the application of knowledge, skills, tools and techniques to project activities to meet the project requirements” [PMBOK®2009].

However, PMBOK® is a mere standard among many others. PRINCE2 suggest another, different, explanation to what a project is, as being “a temporary organization that is created for the purpose of delivering one or more business products according to an agreed business case”, and does likewise with the definition of Project Management, which is defined as “is the planning, delegation, monitoring and control of all aspects of the project, and the motivation of those involved, to achieve the project objectives within the expected performance targets for time, cost, quality, scope, benefits and risk” [MSPP2009].

To further exemplify the multitude of definitions existent for such terms, following is the IPMA Competence Baseline’s definition of project, which is written as “a time and cost constrained operation to realise a set of defined deliverables (the scope to fulfil the project’s objectives) up to quality standards and requirements” [ICA2006].

Definitions apart, as many as there exist, every Standard[3] or framework[4] agree in a few principles, Project Management is the application of the knowledge, skills, tools and techniques to achieve balance on certain project constrains to ensure success, quality and requirement fulfilment. Those constrains are used to evaluate project competing demands, and Project Management is about managing those constrains in the most efficient way possible.

Those project constrains may vary, but one definition, the most common one, is the called *Triple Constrain*

[3] Document that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of optimum degree of order in a given context [PMBOK®2009].

[4] Underlying conceptual structure or abstraction of a system, which facilitates the modelling of solutions to complex problems.

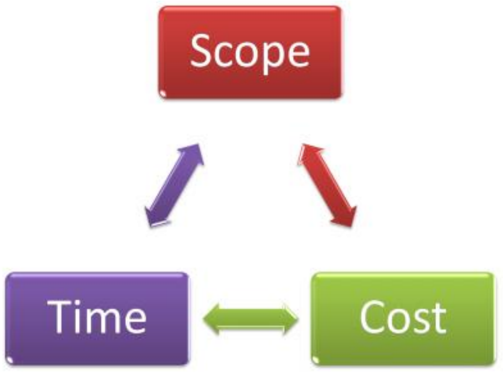


Figure 2 - "Triple Constrain" in its simplest version

Simply put, when scope increases, project duration might have to increase or additional resources might have to be added (cost). Likewise, if the project's execution timeframe is forced to decrease, it's probable either the scope will have to be cut, or additional resources will have to be added.

Dr. Martin Barnes was the first to describe what he called the "Iron Triangle" of time, cost, and output in his course "Time and Money in Contract Control" in 1969, laying the foundations for what has become known as the *Triple Constraint* (time, cost, and scope constraints) [Weaver2007].

Although it is known as *Triple Constraint*, quality is also one axis in this dimension space, making it a 4th dimension in a 3D mapping.

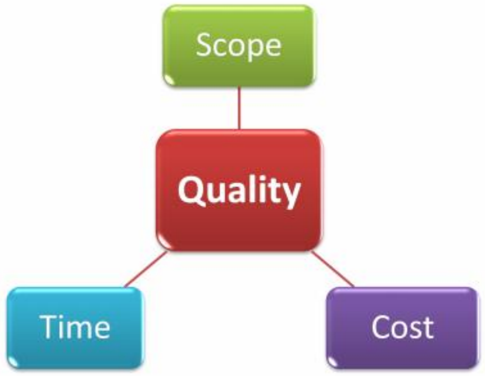


Figure 3 - "Triple Constrain" with Quality as an additional constrain

Nevertheless, other constrains exist in a project, from simple milestones to more critical success factors, and they have direct impact in the 4 constrains above. Due to this, one common and accepted definition for the triple constrains combine not 3 nor 4, but 6 major constrains.

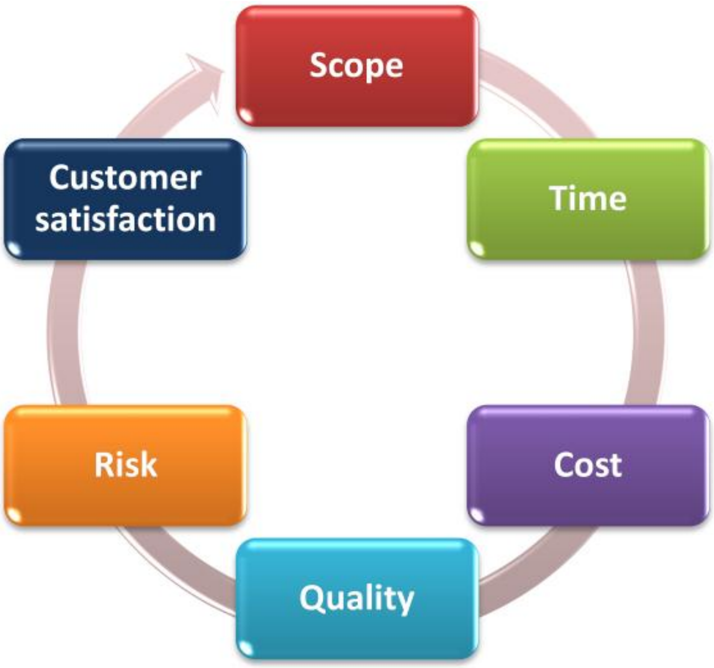


Figure 4 - "Triple Constrain" with six all-dependent constrains

These major constrains are in fact, directly or indirectly, prioritised by the management staff of the company or organization. The job of the Project Manager is to use these priorities to correctly balance constrains and produce a project plan that can be executed successfully. Due to the dependencies between constrains, any minor change in one can impact the others majorly, such a change needs to be correctly analysed, not only in the planning phase, but in the course of project execution as well.

Obviously that other constrains exist in every project, but for the sake of this exercise let's consider the 6 dimensions above as the so called *Triple Constraint*.

2.3 *Project Success vs. Project Failure*

2.3.1 Standish group CHAOS Report

When talking about success or failure in projects, one name is always mentioned, the Standish Group. Standish Group is an IT support organization which is specialised in Project Management issues and challenges. Their cumulative research information gathered in the last 15 years, the *CHAOS Report*, helped understanding the success or failure of IT projects, and it has been released biannually since 1994. This 15 year's work represents research made over 70000 completed IT projects, making the *CHAOS Report* the largest study in the IT Project Management industry. Nevertheless, critics of the Standish Group findings are growing and they insist that the results are not as valid as they appear to be.

The *CHAOS Report* identifies software project failures by defining three project categories [CHAOS1994]:

- Resolution Type 1, or **Project Success**. The project is completed on time and on budget, offering all features and functions as initially specified.
- Resolution Type 2, or **Project Challenged**. The project is completed and operational but over budget and over the time estimate, and offers fewer features and functions than originally specified.
- Resolution Type 3, or **Project Impaired**. The project is cancelled at some point during the development cycle.

The *CHAOS Report* is gathering data on projects since 1994 using this simple definition, and the results speak for themselves.

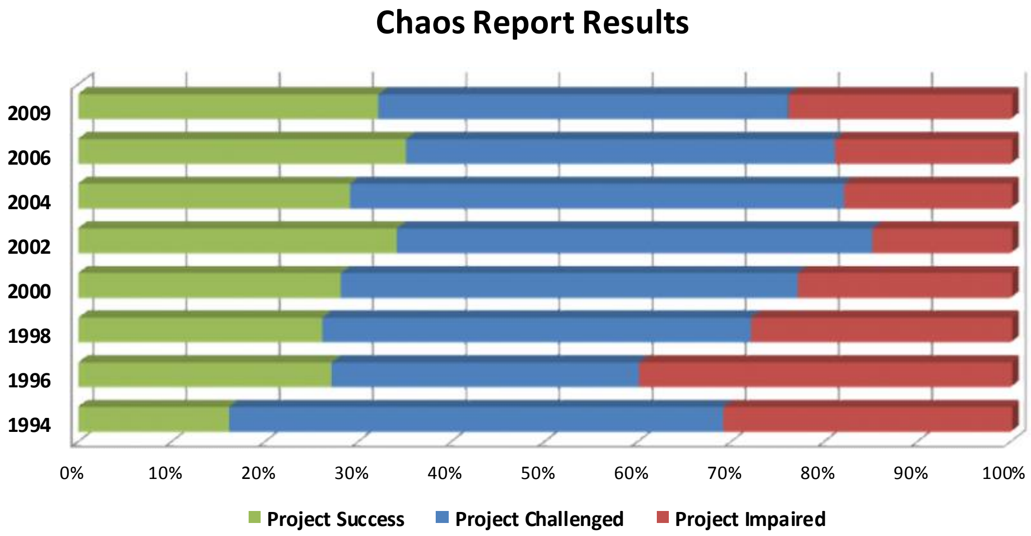


Figure 5 - CHAOS Report Results (1994 to 2009) [EvVer2010]

Although critics insist that these findings don't give a clear picture of the IT industry, one fact is undeniable, the Standish Group is still using the same criteria since 1994, which gives us some degree of assurance on the evolution of success and failure in projects in the IT industry.

Unfortunately the results are not good from an analytical point of view. In 1994 the Standish Group reported a startling success rate of 16%, which evolved with minor setbacks, to a non-spectacular 32% in 2009.

But how do these results bring us closer to the answer of why IT projects fail? To shed some light in the issue, the Standish Group also aggregated the top 10 causes for the three project categories above.

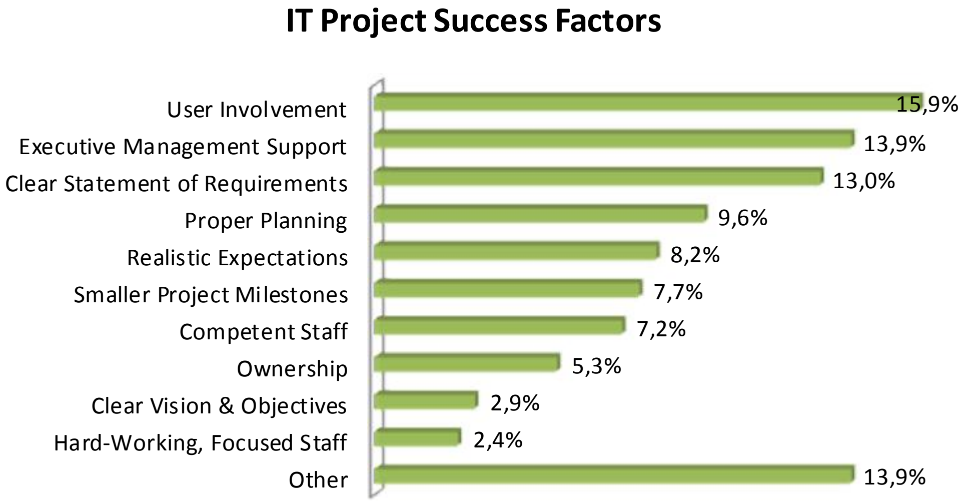


Figure 6 - IT Project Success Factors

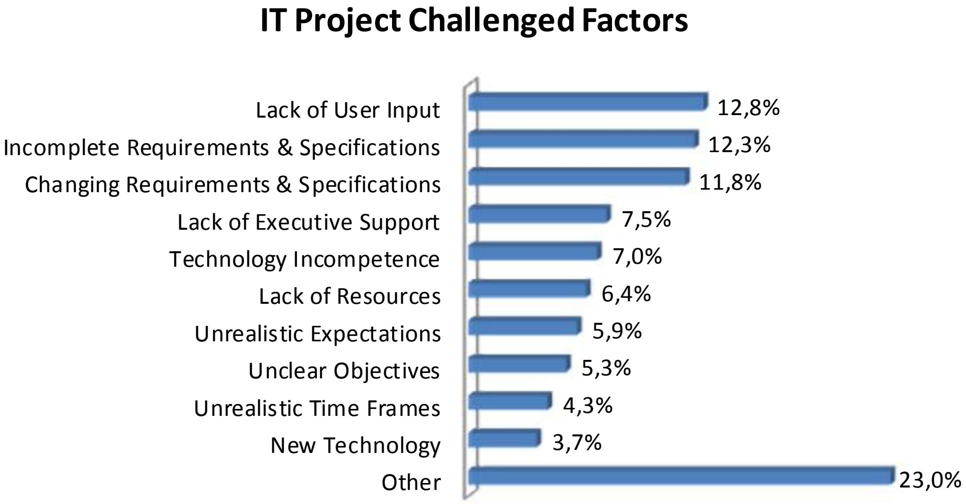


Figure 7 - IT Project Challenged Factors

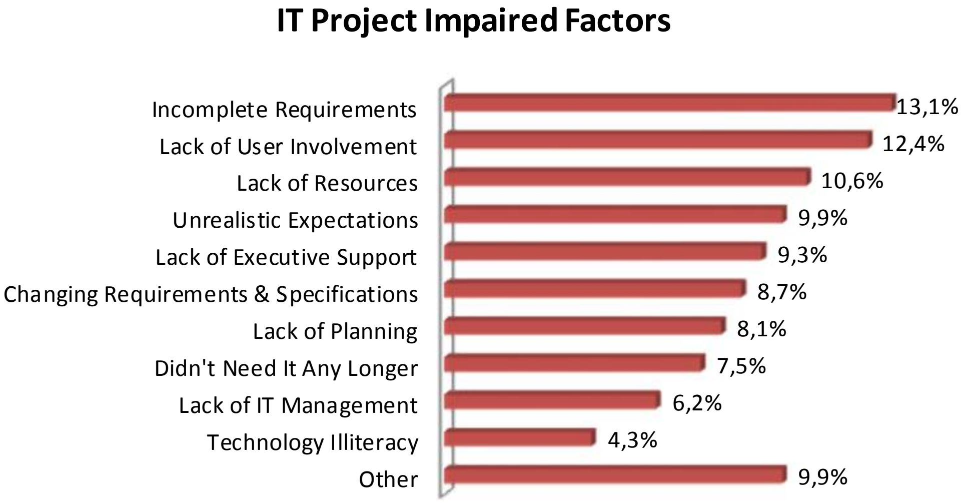


Figure 8 - IT Project Impaired Factors

Although the *CHAOS Report* is seen by the IT industry as a biannually indicator of project success evolution trending, some critics tend to challenge the final results and question the methodology.

Criteria and definitions used in the analysis are far from consensual. From the selection of organizations surveyed, to the questions asked, the list goes on and on. Even the analysis of data is controversial since the method counts project overruns and discard underuns [EvVer2010].

A common joke aroused from all of this. Project Management subject speakers commonly ask IT audiences how many of them agree with the data that shows an "*over 70% failure rate of IT projects*" and everyone agrees. The following question is what the failure rate is in their own companies? Answers diverge, but normally the majority of the audience reports something close to the 10% mark. The speaker then breaks the ice with a comment to his luck for having an audience from such high-performing companies [RLG2005].

Jokes aside, one fact is undeniable, the current IT business shows that the last 15 years had have a lot of successful projects. It would be impossible to achieve the level of innovation, new products and current time-to-market without some serious majority of successful projects.

2.3.2 Reasons for Project Failure

Sometimes, due to a multitude of reasons, some within the project manager's reach, others away from it, projects yield limited results, or just outright fail.

PMI® published in there *PM Network* magazine in July of 2007 [PMNJul2007], a survey called the "Top 9 causes for project failure". The survey was conducted by the Computer Technology Industry Association, USA, and was the result of 1007 responses.

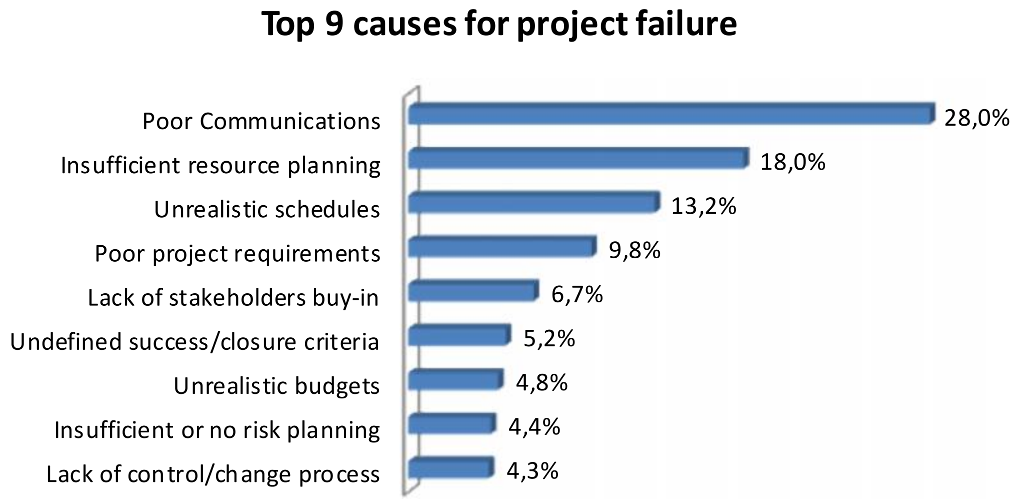


Figure 9 - "Top 9 causes for project failure" Survey results

This survey sheds some light on why projects fail, using the voice of field professionals that deal with IT projects every day.

At first view, communications are the main issue, not only because they are in the top cause, "poor communications", but because they can be linked to several others in the same survey. "Poor project requirements" or "lack of stakeholder buy-in" can be easily linked to incorrect communication to customers and other relevant involved parties. In fact the poor communication issue was so connected to the other 8 causes that some professionals identified the Pareto principle in the results, and attested that in the end, inefficient or poor communications represented in fact 80% of the project failure.

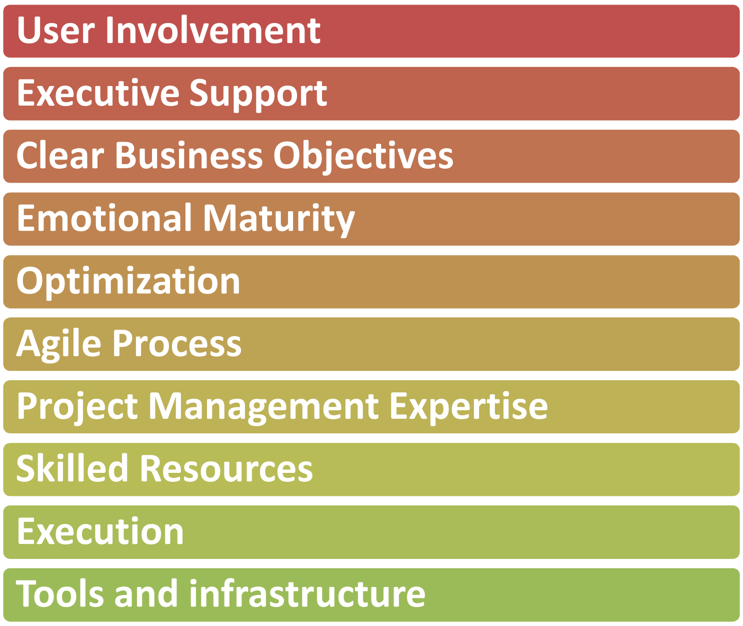
As John Venator, CEO of CTIA at that time, mentioned after the survey was made public, "(…) technical skills alone are no longer enough. Technology workers must have solid communications skills, in order to complete projects".

Following the result of this survey announcement, Bill Dow and Bruce Taylor decided to write the "Project Management Communications Bible" which was, at that time, the first book focused on Project Management communication. The authors saw that this survey gave a clear view in a new and yet unexplored area of Project Management. The book addresses the usage of communication tools, techniques, tip, etc to be used in the 9 Knowledge Areas defined in the PMBOK® [WDBT2008].

2.3.3 Reasons for Project Success

The archrival of failure is success, and this success is the result that every Project Manager, organization or project Team sets itself to achieve.

As seen above, the Standish Group results set a very dark shadow over project success rate. Since 1994 the 16% of success rate evolved to 32% in 2009. Additionally to the project results shown in the previous sub-chapter, the *CHAOS Report* also details a number of Success Factors that are correlated with the projects tripartite results reported. That being said, the Standish Group also sets the *CHAOS Success Factors to IT projects*, them being (ordered by descending importance):

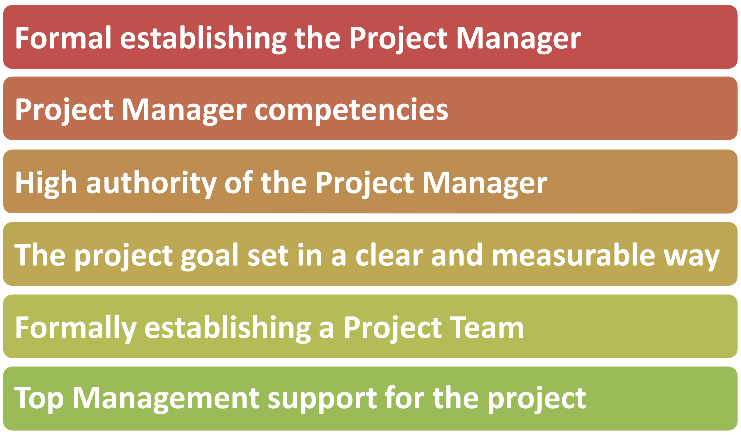


- User Involvement
- Executive Support
- Clear Business Objectives
- Emotional Maturity
- Optimization
- Agile Process
- Project Management Expertise
- Skilled Resources
- Execution
- Tools and infrastructure

Figure 10 - CHAOS Success Factors to IT projects [SumCHAOS2009]

Again, critics of the *CHAOS Report* try to engage in different ways to challenge these results. One of those criticisms is about a 2005 survey to 82 project managers that were asked to rank the factors that influence project success or failure. This survey was presented in the 2005 PMI® Global Congress Proceedings [Spalek2005].

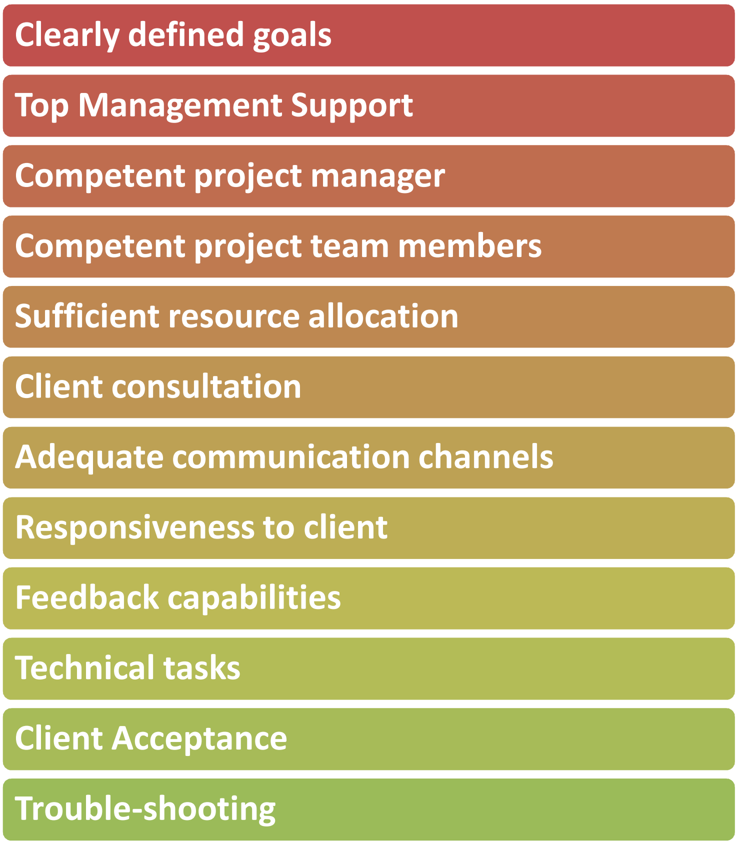
The result of this survey showed that among all factors, the most important factors which substantially influence project success are (ordered by descending importance):



- Formal establishing the Project Manager
- Project Manager competencies
- High authority of the Project Manager
- The project goal set in a clear and measurable way
- Formally establishing a Project Team
- Top Management support for the project

Figure 11 - Critical Success Factors in Project Management [Spalek2005]

A 1996 publication in the Project Management Journal insisted on the same issue, factors that influence project success. Using 50 companies as the survey universe, 13 statements were presented and a rating was asked to each one on them. The result being (ordered by descending importance):



Clearly defined goals

Top Management Support

Competent project manager

Competent project team members

Sufficient resource allocation

Client consultation

Adequate communication channels

Responsiveness to client

Feedback capabilities

Technical tasks

Client Acceptance

Trouble-shooting

Figure 12 - Ranking of system implementation success factors [JKB1996]

2.3.4 Success and Failure Factors analysed

Merging the results of the studies and surveys some conclusions can be drawn. The different results can be grouped in 4 groups: "Communication & Collaboration", "Skills & Knowledge", "Stakeholder Involvement", "Organization Support" (4 different colours below). These 4 groups represent, in general terms, the root causes of the Success and Failure in IT Projects.

Success Factors		
[JKB1996]	**[Spalek2005]**	**CHAOS Success Factors**
Clearly defined goals	Formally establishing the Project Manager	User Involvement
Top Management Support	Project Manager competencies	Executive Support
Competent project manager	High authority of the Project Manager	Clear Business Objectives
Competent project team members	The project goal set in a clear and measurable way	Emotional Maturity
Sufficient resource allocation	Formally establishing a Project Team	Optimization
Client consultation	Top Management support for the project	Agile Process
Adequate communication channels		Project Management Expertise
Responsiveness to client		Skilled Resources
Feedback capabilities		Execution
Technical tasks		Tools and infrastructure
Client Acceptance		
Trouble-shooting		

Table 1 - Success factors combined from three sources mentioned

🟦 Communication & Collaboration 🟨 Skills & Knowledge

🟪 Stakeholder Involvement 🟩 Organization Support

Failure Factors		
Top 9 causes for project failure	**Project Impaired Factors**	**Project Challenged Factors**
Poor Communications	Incomplete Requirements	Lack of User Input
Insufficient resource planning	Lack of User Involvement	Incomplete Requirements & Specifications
Unrealistic schedules	Lack of Resources	Changing Requirements & Specifications
Poor project requirements	Unrealistic Expectations	Lack of Executive Support
Lack of stakeholders buy	Lack of Executive Support	Technology Incompetence
Undefined success/closure criteria	Changing Requirements & Specifications	Lack of Resources
Unrealistic budgets	Lack of Planning	Unrealistic Expectations
Insufficient or no risk planning	Didn't Need It Any Longer	Unclear Objectives
Lack of control/change process	Lack of IT Management	Unrealistic Time Frames
	Technology Illiteracy	New Technology

Table 2 - Failure factors combined from three sources mentioned

Clearly “Stakeholder Involvement” is one of the primary influences on success by being the top cause of success in most quoted surveys and a strong failure factor if the project lacks that involvement. It’s clear that this factor affects the project strongly and should be one of the top priorities.

The lack of "Communication & Collaboration" is the top reason for failure, but when it comes to success is merely regarded as important, but not too highly. This can mean “Communication & Collaboration” is truly an issue that causes failure but it’s usually regarded as invisible until it is actually missing. This leads us to conclude that successful project teams communicate effortlessly between themselves in a way so natural they do not recognize it directly.
Nevertheless, the top success factors can be understood as being part of to “Communication & Collaboration” as well. "Stakeholder Involvement", "Organization Support" are clearly linked to communication issues. PMI® regards communication as being 80% of Project Management work, and the PMBOK® is clear in stating the importance of communication by emphasizing the importance of creating a communication plan that involves all project stakeholders regardless of their importance. The same standard also refers the importance of clear communication channels and types of “noise” that have to be dealt with which may impact those same channels.

The “Organization Support” is pointed to be the second top reason for success and the lack of such support it also visible of being a big issue failure-wise.

The apparent unimportance of "Skills & Knowledge" brings a dilemma. It is regarded as being the less important factor regarding project failure, but it comes as second when stating the factors with are needed to insure project success.

2.3.5 Project Management Success challenge

So, what is the challenge regarding project success and failure… How can we assure success factors and avoid failure ones?

The key is a correct balance between the mitigation of probability and impact of failure factors, and the enhancement and exploitation of success factors.

Many companies address this balance with the definition of an internal Project Management Methodology, one that is adapted to industry, market, company requirements and culture.

Theoretically speaking, and assuming the Project Management Methodology was developed to be efficient upon creation, the usage of such processes would ensure that common failure factors are avoided and success factors are enhanced. Ensuring this behaviour should be the main focus of the methodology created, such creation is normally based on existing frameworks, standards or models which define the best practices of Project Management.

The reality is, the most used Project Management frameworks appeared 30 years ago, and their worldwide usage is well known. Even though these frameworks are not IT specific, management factors remain roughly the same. Therefore, such success and failure results as the ones presented before should not exist.

Around the year 2000 a group of individuals acknowledged these factors and began a movement that changed the course of IT Project Management. Their goal was to prove that in the year 2000, Project Management Standards and Frameworks were not appropriate to manage IT projects as they set to simplify the way to manage them. They called it the *Agile Manifesto,* and it will be reviewed in the next sub-chapter.

2.4 Agile and Traditional Project Management

Waterfall based Project Management was the common base for managing projects in the last half of century. Forged in the construction and manufacturing industries, it has been adapted to the software industry with less than stellar results. The *CHAOS Report* results are a statement of the efficiency deficit of such usage.

Through the course of this chapter the genesis of Agile will be presented, the Agile Manifesto, plus two of the most used Agile frameworks and the most known Project Management framework will briefly introduced.

2.4.1 Agile Manifesto

In February 2001 in Utah (USA), 17 people meet for a 3 days gathering, each with different IT backgrounds and similar visions to how Software Development should be addressed. Present were representatives from SCRUM, Extreme Programming (XP), Feature-Driven Development (FDD), etc, all with a common objective: find a suitable alternative to the document-based, process-enslaved, rigorous Software Development processes and standards [JH2002].

17 gurus talked for 3 days and a common ground was established. This common ground was simple and accepted by all 17 as the basic principles for Software Development. Even though the concept as not new, previously known as “lightweight methods”, after the gathering a new name was found: Agile.

The result of the gathering was a so called document that would change the way to approach Software Development from that day on, and its name was *Agile Manifesto*.

The *Agile Manifesto* reads, as follows:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more [AM2001].

Simplifying the complexity of planning, focusing on customer value, and enhancing Team participation and collaboration, these are all simple truths believed by all signatories. These principles govern all the techniques and rules in the different Agile methods, which all strive to make Software Development more flexible and overall more successful [TSUH2009].

Additionally to the simple values stated above, 12 principles were also agreed upon:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the Team reflects on how to become more effect.

In the next sub-chapters two Agile frameworks that were created in light of those principles will be reviewed, and some of the values and principles of Agile will be analysed from project and/or performing organization point of view.

2.4.2 SCRUM, an Agile framework

“SCRUM is an Agile framework for completing complex projects” is the simple way that the SCRUM Alliance found to introduce their concept. Many professionals and even Project Management authors refer to SCRUM as an Agile-based Project Management framework, even though the SCRUM Alliance does not.

Based on an incremental and iterative life cycle, SCRUM offers a simple and effective way to managing projects. Such simplicity results from a drastic reduction of some of the common areas of traditional Project Management.

The name comes from the sport rugby, where the teams are cross-functional, high-performing and in constant communication with each other. The game itself is played advancing in the field by running sprints with constant passes to the rear and side players. Passing the ball to the rear or the side in order to find better ways to advance with a sprint is a clear analogy to the idea of incremental and iterative cycle in Software Development.

SCRUM brings a new perspective on dealing with project change: the project needs to embrace change, because it’s inevitable, over trying to seek change predictability. When using SCRUM, changes are accepted throughout the project, and adjustments are made accordantly to better accommodate change. The project team is aware of this fact and does its best to exploit this to the project’s advantage.

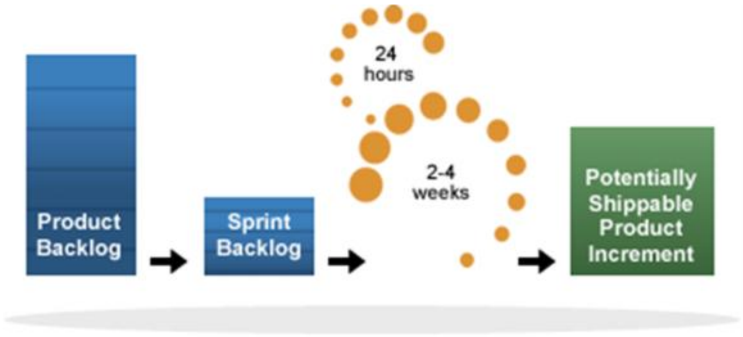


Figure 13 - SCRUM life cycle [SA2011]

SCRUM is divided in three roles, four ceremonies and three artefacts, all centred around small 2 to 4 weeks incremental interactions called Sprints, in which small functional product slices are produced and released to the customer. Although, this may seem to be a very simple way to describe a development framework, the truth is that it is just as simple as this, at least in basic terms.

Starting by project roles, they are simply divided in three roles:

The **Product Owner**, normally the customer or a representative, is responsible for the business value of the project, and defines and prioritizes the Product Backlog, a wish list of possible project requirements.

The **Scrum Master**, the representative of management in the project, has the responsibility to ensure that the Team is functional and productive, by being a facilitator and clearing project roadblocks, therefore ensuring the Team is not distracted or deviated.

Finally the **Team**, a small, 5 to 7 elements, cross-functional group of individuals which self-organizes to get the work done, implementing the product backlog aligned with the Product Owner's expectations.

Continuing with the framework description, the SCRUM Alliance defines what it calls as ceremonies, which is a less formal name to give to meetings:

The **Sprint Planning** where the Team meets with the Product Owner to choose a set of work to deliver during a sprint.

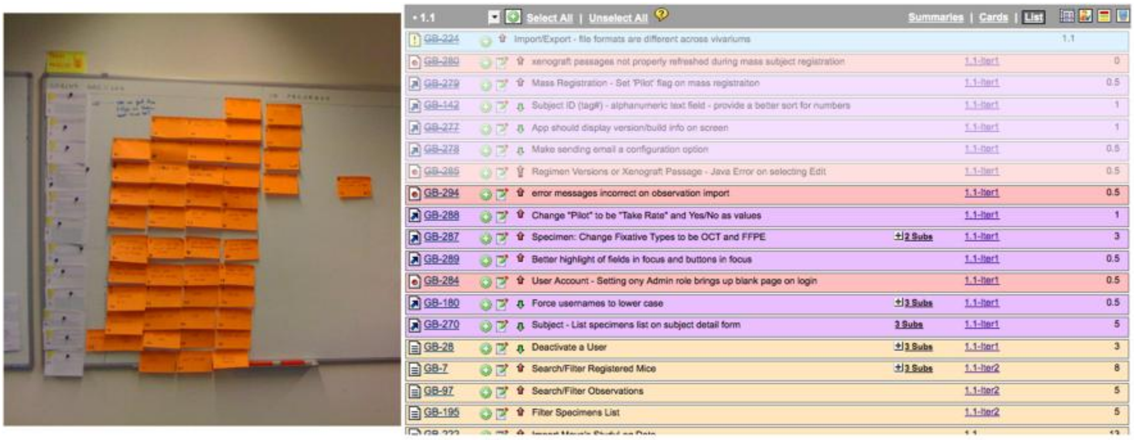
The **Daily Scrum**, a daily 15 minute meeting where the Team meets to share struggles and progress on the current Sprint, a fast Team roundabout of what did you do the day before, what you will do the current day and what roadblocks you have encountered.

Sprint Reviews, in the end of the Sprint, is where the Team demonstrates to the Product Owner what was completed during the Sprint. It is basically an inspection of the work produced during the Sprint. This is commonly called the Sprint Demo or Sprint Demonstration.

The **Sprint Retrospective**, normally done after the Sprint Review, is where the Team analyses the current Sprint execution and looks for ways to improve the process in the next sprint.

And concluding with the description of the SCRUM framework, defined are three major artefacts, which can be physical documents or take any other form:

The **Product Backlog**, created by the Product Owner, is a prioritized list of desired project and product outcomes and features. It's normally written in the form of user stories, a small piece of useful functionality that can be defined in one sentence. Stories normally represent functionality that is useful to the end user.



GB-224	Import/Export - file formats are different across vivariums		1.1	
GB-280	xenograft passages not properly refreshed during mass subject registration		1.1-Iter1	0
GB-279	Mass Registration - Set 'Pilot' flag on mass registraiton		1.1-Iter1	0.5
GB-142	Subject ID (tag#) - alphanumeric text field - provide a better sort for numbers		1.1-Iter1	1
GB-277	App should display version/build info on screen		1.1-Iter1	1
GB-278	Make sending email a configuration option		1.1-Iter1	0.5
GB-285	Regimen Versions or Xenograft Passage - Java Error on selecting Edit		1.1-Iter1	0.5
GB-294	error messages incorrect on observation import		1.1-Iter1	0.5
GB-288	Change "Pilot" to be "Take Rate" and Yes/No as values		1.1-Iter1	1
GB-287	Specimen: Change Fixative Types to be OCT and FFPE	+2 Subs	1.1-Iter1	3
GB-289	Better highlight of fields in focus and buttons in focus		1.1-Iter1	0.5
GB-284	User Account - Setting ony Admin role brings up blank page on login		1.1-Iter1	0.5
GB-180	Force usernames to lower case	+3 Subs	1.1-Iter1	0.5
GB-270	Subject - List specimens list on subject detail form	3 Subs	1.1-Iter1	5
GB-28	Deactivate a User	+3 Subs	1.1-Iter1	3
GB-7	Search/Filter Registered Mice	+3 Subs	1.1-Iter2	8
GB-97	Search/Filter Observations		1.1-Iter2	5
GB-195	Filter Specimens List		1.1-Iter2	5

Figure 14 - Two examples of a Product Backlog

The **Sprint Backlog**, a set of work from the Product Backlog that the Team agrees to complete in a Sprint. It is normally broken into smaller, more team-manageable tasks.

And finally the **Burndown Chart**, a simple at-a-glance evolution chart, showing what work remains in the Sprint.

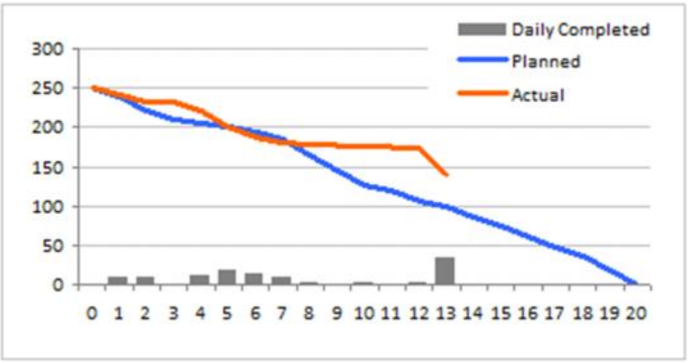


Figure 15 - Burndown chat example

From the SCRUM Alliance’s view, the SCRUM choreography runs ordered as below:

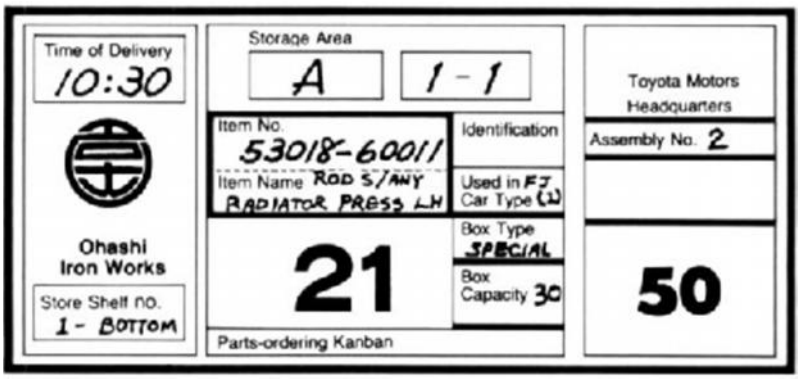
1. A Product Owner creates the Product Backlog.
2. During Sprint Planning, the Team pulls a small chunk from the top of the Product Backlog, creating the Sprint Backlog, and decides how to implement those pieces.
3. The Team commits itself to a Sprint to complete its Sprint Backlog, but meets each day to assess its progress (Daily Scrum).
4. Along the way, the Scrum Master keeps the Team focused on its goal.
5. At the end of the Sprint, the work should be potentially shippable, as in ready to hand to a customer.
6. The Sprint ends with a Sprint Review and Sprint Retrospective.
7. As the next Sprint begins, the Team chooses another chunk of the Product Backlog and begins working again.

The cycle repeats until enough items in the Product Backlog have been completed, the budget is depleted, or a deadline arrives. Which of these milestones marks the end of the work is entirely specific to the project. No matter which impetus stops work, SCRUM ensures that the most valuable work has been completed when the project ends [SA2011].

Some specific execution practices were not described above, since this is only an overview of the framework.

2.4.3 KANBAN, the manufacturing simplicity

Another present Agile trend is KANBAN. KANBAN is a Japanese word which literally translates as ‘‘signboard’’ or “billboard” and was forged in the Toyota’s manufacturing control challenges during the 1950s. This was the response to the need for demand scheduling control and work in progress levelling in the manufacturing shop-floor. Quoting Taichi Ohno, mentor of Toyota production System, “the two pillars of the Toyota production system are just-in-time and automation with a human touch, or *autonomation*. The tool used to operate the system is KANBAN.”



Time of Delivery 10:30

Ohashi Iron Works

Store Shelf no. 1- BOTTOM

Storage Area A 1-1

Item No. 53018-60011

Item Name ROD S/ANY RADIATOR PRESS LH

Identification

Used in FJ Car Type (2)

Box Type SPECIAL

Box Capacity 30

21

Parts-ordering Kanban

Toyota Motors Headquarters

Assembly No. 2

50

Figure 16 - KANBAN Ticket

Just-in-Time (JIT) means that execution (sequential manufacturing process steps) should only be done when a necessity is clear (demand for the product or sub-product being produced). This principle is the opposite of the Just-in-Case one, which would have the production stockpiling products waiting for future demand, even if that demand never comes. JIT is focuses on ensuring the most-non-interrupt flow possible, which makes sure that we only produce what we really need right now, no more no less.

Autonomation is key to ensure a focused and immediate response to production blocking points. When an abnormality is detected in a manufacturing process, the operator is warned and the production line is stopped until the problem is solved. This good practice focus attention on the defect that just happened, ensuring that the root cause is addressed immediately avoiding future recurrences and making sure that the next manufacturing process step will not be affected by any defected input. The objective is to continuously improve towards.

Control of both JIT and *Autonomation* was assured by a visual board, where work queue, manufacturing process inputs and outputs are tracked accordingly to the board steps.

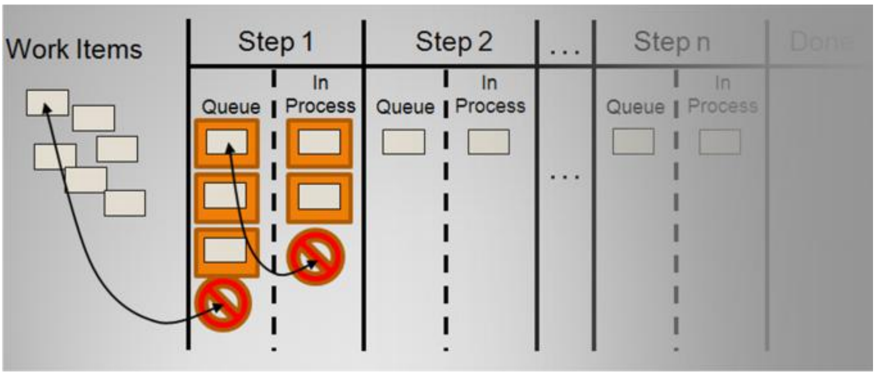


Figure 17 - KANBAN Board

In simple terms, KANBAN is a schedule work system, governed by the Just-in-Time principle, which manage demand "pulling" by knowing the demand-forecast to receive such a "pull". The principle is so simple that Taichi Ohno called it a "tool".

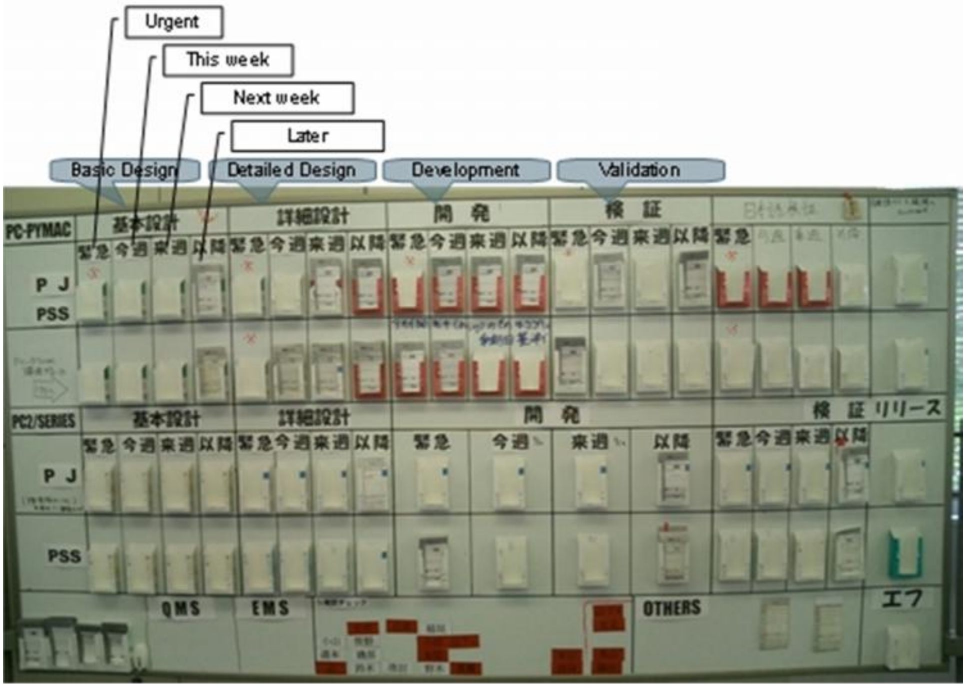


Figure 18 - Manufacturing KANBAN Board

Based on a "pull" JIT manufacturing process, relays on the principle that the current manufacturing process should only produce (output) the amount of parts needed as input in the following manufacturing process operation. The main objective is to level the inputs and output (materials) of the process manufacturing workflow, to ensure we are focusing the operation capacity in the correct process to avoid waste resources in non-urgent activities and bottlenecks in manufacturing route [Cimo2013]. This choreography is organized in a signboard which represents the manufacturing processes as phases in the board, each with their production capacity and stock limit.

Other KANBAN main goal is to precisely manage Work-In-Process (WIP) in each manufacturing process (step). Because current process should not produce no more no less than the next process needs, each process product production should be levelled to the correct amount. Applying this concept to all manufacturing process will reveal possible equipment our man-power bottle necks, which should be address to insure continuous improvement in the manufacturing cycle.

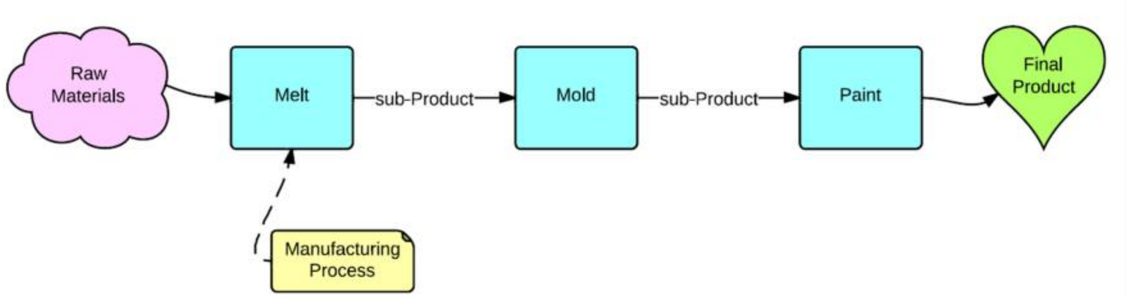


Figure 19 - Manufacturing line

Let's use the simple example above, a simple 3 process manufacturing line, where raw material becomes final product after be is transformed in the 3 steps showed. Theoretically, if the Paint process can only handle transformation for a max of 5 sub-products at a given time, then the Mold process should not produce more than the Paint can "pull". Obviously that this concept may be adapted with inclusion of sub-product queues before the Paint step, but the simple WIP levelling idea remains as a good practice to follow, this will minimise inventory between the steps, therefore reducing manufacturing costs [Kenji2008]. KANBAN is based on the simple principle of having to agree on a limit to your WIP, and "pull" new work from the queue only when something is finish.

Everything comes to completion by adding the Toyota's KANBAN six rules [Lu1989]:

- Do not send defective products to the subsequent process
- The subsequent process comes to withdraw only what is needed
- Produce only the exact quantity withdrawn by the subsequent process
- Equalize production (Level the production)
- KANBAN is a means to fine tuning
- Stabilize and rationalize the process

Customer-wise, customer being the manufacturing shop-floor, customer value comes from a continuous, predicted and non-interrupted flow of small units, in opposite of delivering big batches of product from time to time, which cannot be accepted or review by the customer in a timely fashion (in this cases, consumed by the next manufacturing process).

Regardless simple or complex approach, examples of KANBAN based processes or only KANBAN boards can be found all around us, from the schedule boards used in hospital to the pull-ticket line in your grocery shop. The question is, can complex IT projects or teams see benefit of such a simple approach?

2.4.4 PMBOK®, the World best known Project Management Standard

The "A guide to the Project Management Body of Knowledge" is the current standard for Project Management in the USA (it's a ratified ANSI standard, ANSI standing for Approved American National Standard).
PMBOK® is a Project Management framework which merges several Project Management concepts into a unit, a standard, enabling reduction of complex Project Management concepts to something easily conceptualized and managed. It is what a large number of practicing project managers agree to be generally good practice in Project Management.

It contains the generally recognized good practices in the area while reflecting Project Management's continually evolving knowledge. First developed as a white paper in the 1983 PMI Ethics, Standards and Accreditation Report, the PMBOK® Guide has become the foremost global standard for the practice of Project Management [PMI®2011].

The standard is revised every 4 years and it's currently published worldwide in 11 languages, currently in its fifth edition, which was published in January 2013.

The PMBOK® framework is largely intended for managing single projects of any size, and can be used in different application areas and industries, from construction to engineering, if combined with the specific practices needed in each intended area. It defines Project Management as the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements [PMBOK®2009].

To ensure such objectives, the PMBOK® defines 47 processes which are mapped in 5 Process Groups and aggregated in 10 Knowledge Areas.

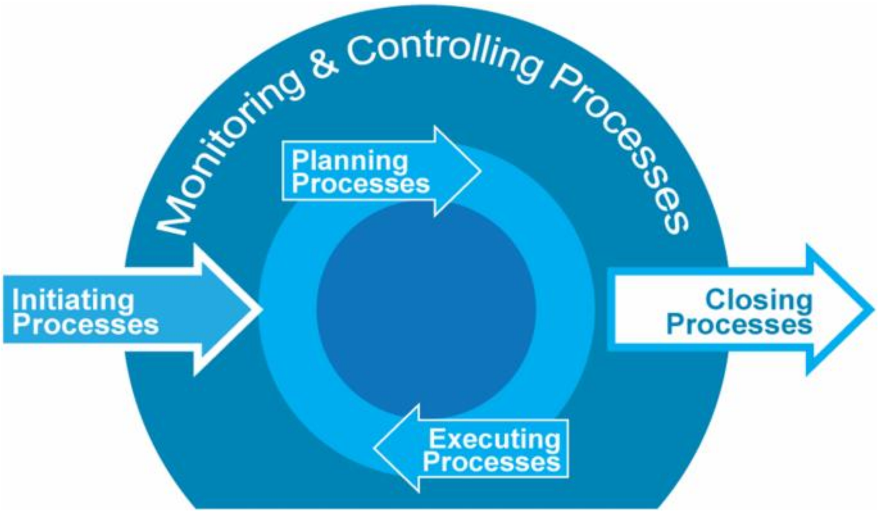


Figure 20 - 5 Process Groups [PMBOK®2013]

Those 5 Process Groups may suggest a waterfall project lifecycle, but they cannot be seen as such. Far from one-time events, they have a repeatable nature and have subsequent activities during the course of the project or project phases.

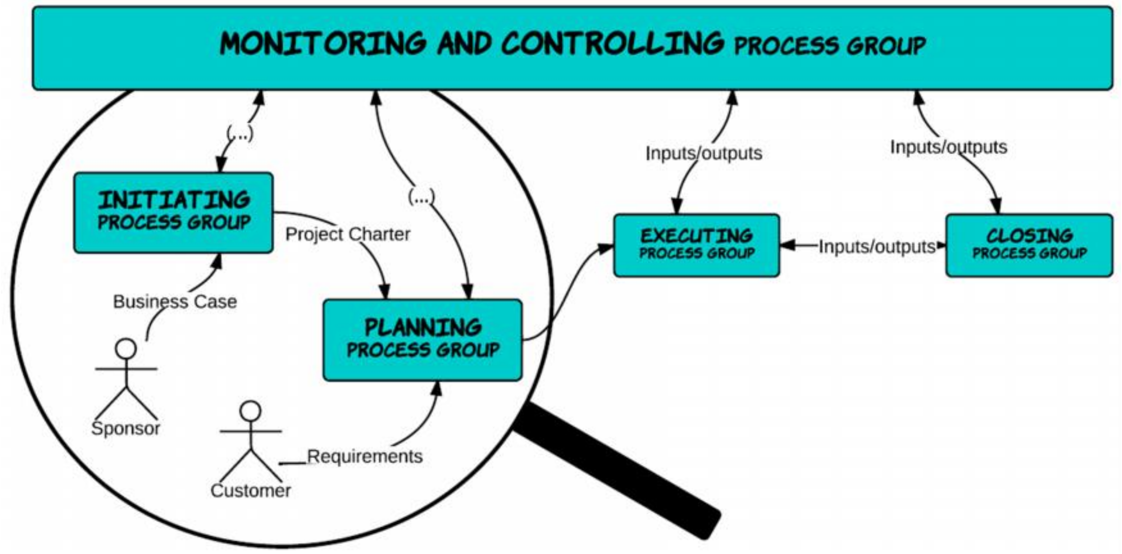


Figure 21 - Process Interactions [PMBOK®2013]

As shown above, the Process Groups produce outputs that are inputs of other Process Groups in constant integration and overlapping with each other during the project execution. If the project has several phases in its execution, the Process Groups may repeat themselves intra and extra-phase interactions (through the overall project) [PMBOK®2009].

The 47 processes can also be gathered in 10 Knowledge Areas, which are commonly accepted into the Project Management discipline and can be applied to all projects.

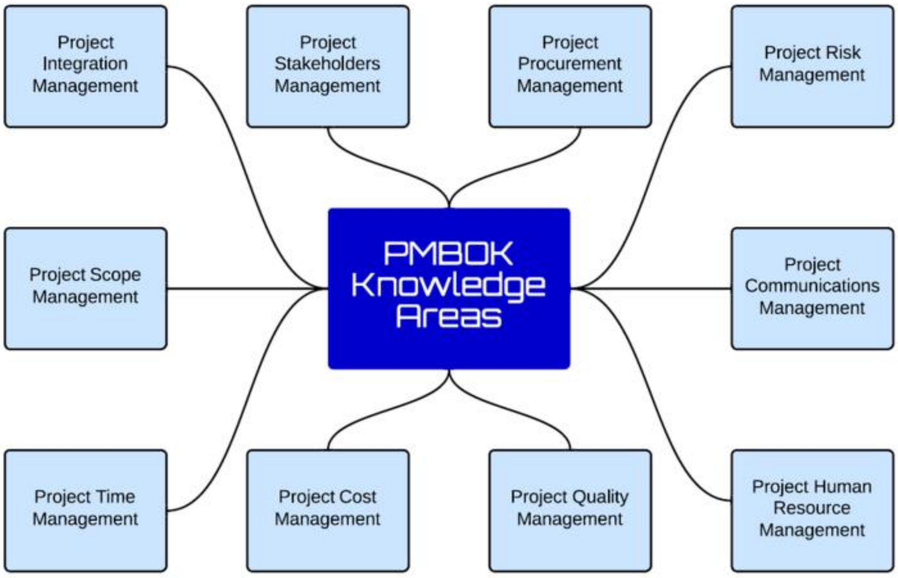

Project Integration Management
Project Stakeholders Management
Project Procurement Management
Project Risk Management
Project Scope Management
PMBOK Knowledge Areas
Project Communications Management
Project Time Management
Project Cost Management
Project Quality Management
Project Human Resource Management

Figure 22 - PMBOK® Knowledge Areas [PMBOK®2013]

The concept sustains that the Process Groups gather a number of processes which have dependencies between them and are easily mapped to a project cycle or project phase of any size and in any industry. This concept brings a different overview on how to select the needed processes to a project, based on relations between Project Management Process Groups and Knowledge Areas.

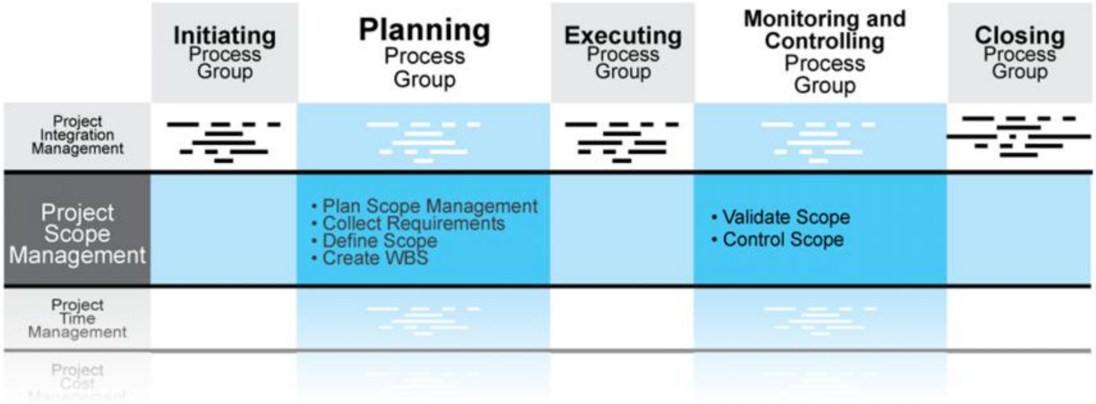


Figure 23 - Project Management Process Groups and Knowledge Areas Mapping [PMBOK®2013]

Each Knowledge Area is a group of several individual processes that are input and output interdependent within the other Knowledge Area processes, and extra dependent from processes from other Knowledge Areas. Each process is composed of inputs, outputs and sets of Tools & Techniques.

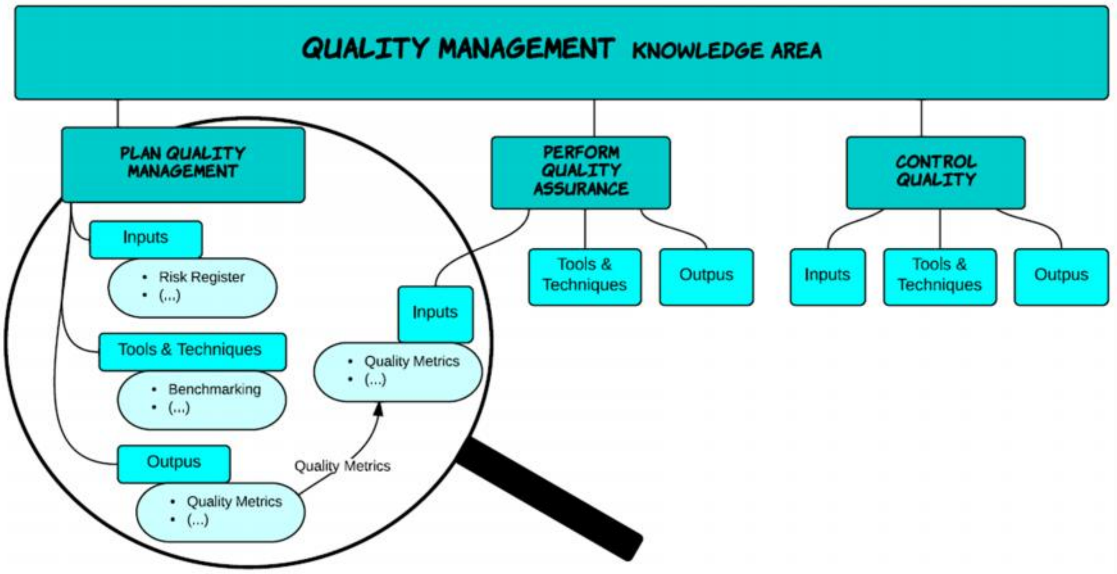


Figure 24 - Project Quality Management Overview [PMBOK®2013]

Above is an example from the three processes that aggregated compose the Project Quality Management Knowledge Area, each one with their respective inputs, outputs and sets of Tools & Techniques. Inputs came from processes from other process areas, and the outputs are inputs for both remaining Quality Management Knowledge Area processes and others from other Knowledge Areas.

The group of processes to be used in a given project need to be selected based on the specifications of each project, its objectives and requirements and the project performing organizational culture.
The standard does not advise a specific minimum set of processes to manage a project. Instead, it insists that the process selection needs to be done by the Project Management Team of each project. The PMBOK® perspective is that every project is unique, therefore it needs to be managed accordingly to that unity and singleness.

2.4.5 Agile Methods vs. Traditional Methods

Agile based frameworks application proves to be a valuable asset in several constrains that are not easily resolved with more traditional methods. On the other hand, those methods also manage parts that Agile disregards as fruitless.

Let's take the Agile Manifesto values and try to see how they hold up through a company management perspective.

*“**Customer collaboration** over contract negotiation”*

> Is an excellent example of a benefit that Agile brings to a project. Having the customer more close to the project team ensures early feedback and creates a communication line to ensure that changes will come earlier in the project, therefore easier and less expensive to accommodate.
>
> But, assuming that we are using a contract to establish cost boundaries, contract negotiation needs to occur as well and it may set limits to the customer changes. Unlimited resources are a myth and all project resources need to be accountable to a specific level or they will be generating too much work which maybe is not what we can bill the customer for.

*“**Responding to change** over following a plan”*

> Linked with the value above, responding to change is maybe the most important Agile principle. From a customer perspective this is an excellent principle and it should not be questionable at all.
>
> From the company perspective it can be a disaster regarding cost overruns and Portfolio Management. An unfocused customer may bring change in a day-to-day basis whichever good job the Team is doing. Additionally, if the company has 50 Agile projects running, some kind of plan needs to be drawn and followed, at least to a certain level, to ensure the correct usage of finite resources. Project Portfolio Management needs to ensure resource levelling and cost control throughout the projects and programs of the performing organization.

*“**Working software** over comprehensive documentation”*

> A similar statement was said by many software professionals since software begun. It's normal for programmers to enjoy coding and hate documenting or testing it. “It's boring” seems to be the following phrase. Some say that Agile was made from software engineers to software engineers, ensuring that pains and “boring stuff” would be left out to be forgotten.

Let's check the company perspective: Imagine a big company that develops internal software, which it also has to maintain until it's not used any more. The actual cost of the software is the cost of the development plus the cost of maintenance during future use. It is common, especially in big companies, that the development of new products is made by different teams than the ones that do maintenance. Ones are "changing the business" and the others are "running the business". In this case a common ground needs to be found in order to pass the knowledge from the development team to the maintenance team. Several techniques can be used to ensure this rite of passage, but almost all techniques require some kind of documentation being produced and delivered.

*"**Individuals and interactions** over processes and tools"*

This statement attests to the potential of individuals and their constant communication over complicated and restraining process and tools. Simplicity and team-work are the keys for success. Agile advocates that teams should be self-organized, cross-functional and capable of ingenious and original solutions. Then they will be able to use simple ways to manage the activities needed to get the job done.

Big companies tend to forget the value of the individual, usually seen as mere resources to bundle up in team silos. Traditional processes tend to enforce the same way of work to all the teams in order to standardise activities. This is proved to wear down ingenious spirits and affects motivation.

Agile cannot be analysed by only one perspective. As all frameworks, models or methodologies, it has its strengths and weaknesses, and this balance should be accounted for when choosing to go *Agile*. Every strength and weakness comes with a price for the company, and Agility is no exception.

2.5 Conclusions

Let’s talk about Project Management. After more than a decade of usage, Agile frameworks proved to bring valuable assets in several constrains that are not easily addressed with the more traditional Project Management methods. On the other hand, traditional Project Management methods also manage areas that Agile tends to forget or consider as overhead.

Currently, SCRUM is on top of the Software Project Management trends. Its simple vision delivers fast results with low investment and the majority of the project teams enjoy working in such a distinct way. It brings an astonishing approach regarding traditional frameworks like PMBOK®, and unlike the latter, it is much more aligned with the IT present necessities.

KANBAN, with its Lean root preceding by far other Agile approaches, is still generally not accepted in the IT community, and even less so when regarding Project Management practices. Currently, the SCRUM hype shades this simplest model, forsaking it to usage by only more mature teams.

PMBOK® on the other hand is being perfected for almost 30 years and it’s the result of the combined knowledge of Project Management professionals from several areas. It may not be as IT specific as SCRUM, but what it lacks in industry specifics it makes up for in globally accepted Project Management practices.

Software Project Managers claim Agile frameworks have benefits and advocate that a perfect methodology, which all agree is unreachable, should be a combination of the more IT aligned Agile frameworks with a more Project Management oriented framework like PMBOK®.

3 KANBAN vs. SCRUM

3.1 Introduction

“Change is upon us… Be ready for it”. When hearing this claim from a well-known author in an Agile Conference, it got me wondering. Yes, several things changed regarding IT Project Management in the last decade, but is it for the best? Are we sure of that?

Ten years have passed since the Agile Manifesto has been signed. It was not only a milestone in Software Project Management but also a wake-up call to professionals and organizations. Agility was not really a new concept, but until that sit-down it never been the coolest kid on the block. Lean Management can attest to that fact.
A sea of frameworks, standards, methodologies and models have come forward since that day, and the IT universe sees itself in a kind of cross roads, with the destination not really clear.

But really, in the end, what we really want? Successful projects with mind-blowing quality products? Agile teams that respond well to adaptation, scope change and customer requirement storms? Yes, that and much more.
Let’s see this Agile trend from the company perspective. How much does it cost to be Agile? Can we have full committed and centralized allocated teams? Can we have non-rotated team members? Can we allocate all the best resources to one specific project? Should we have such proximity with the paying customer?

Every company see the benefits of Agile but, regardless of the model or framework used, the benefits can cost too much. Seeing Agility as a mitigation plan to modern day IT risk, we will certainly have a backlash somewhere in the middle point. Every mitigation plan comes with either a secondary or a residual risk. The issue comes down to a point of view: one benefit to the development team can be a financial risk to the upper management. “There is no such thing as too much agility” is a bold claim used by consultants and trainers, but the company “feels” this financially.

As said before, both SCRUM and KANBAN derived from Agile concepts. In the course this chapter we will detail both, established some middle ground foundations, review the pros and cons and come to some conclusion regarding the applicability of both in the IT industry.

3.2 Detailing SCRUM

3.2.1 Origins

SCRUM foundations derived from Lean Management, but Lean is so vast that some other starting foundation must exist. Some exemplify the “The New New Product Development Game” by Takeuchi and Nonaka, published in the Harvard Business Review in 1986, as the starting point of all the fuss. Its subtitle sheds light to the origins of SCRUM’s name, “Stop running the relay race and take up rugby” [Takeuchi1986].

The article relates the use of six managing characteristics to achieve a fast and flexible process for new product development, to break with the old sequential approach. Those characteristics being:

- Built-in instability
- Self-organizing project teams
- Overlapping development phases
- "Multilearning"
- Subtle control
- Organizational transfer of learning

Takeuchi and Nonaka laid the carpet for a new development process, based on sequential vs. overlapping development phases, self-organized empirical teams with seldom higher-management control and continuous improving practices derived from shared lessons learned.

In 1993 Jeff Sutherland created SCRUM based on Takeuchi and Nonaka’s article, combining an iterative and incremental time-box-based approach with a simple empirical management process. In 1995, Ken Schwaber published the first official paper on SCRUM. Sutherland and Schwaber, both signatories of the Agile Manifesto, are commonly regarded as the fathers of SCRUM as we know it, and have written several articles and books, and lectured SCRUM worldwide [Rubin2012].
Recently, in July 2013, both have revised and signed the “SCRUM Guide”, a free, simple 17 page guide in the attempted to again lay the foundations for a simple framework.

Introduced in the previous chapter, the SCRUM framework is one of the Agile-based mostly used models in the IT industry. SCRUM is described by some as a way to manage complex projects [SA2011], today its main use is management of Software Development projects. While SCRUM might have been developed with the IT industry in mind, various reports exemplify a far wider scope. Plan decomposed requirements, small incremental interactions and frequent customer communication are benefits for all projects regardless of product or industry.

However, since no company, university or organization owns SCRUM, there is no clear definition for it. It’s a no man’s land, ruled by authors and consultants, both with their books, sites, blogs, lectures, training and even globally accepted certifications.
Described in the past as a Software Development process or a Project Management methodology [MBB2011] or simply a methodology [SMY2013], today we can came to an agreement: SCRUM is a Framework [SA2011].

Why a Framework? SCRUM lays a set of values, principles and practices that outline a model based on true and tried approaches that are accepted by the industry and professionals, but in the end are not mandatory and subject to adaptation [Rubin2012]. This provides a foundation in which to build your own SCRUM. Customization can make your SCRUM more or less similar to the mainstream definition. This is commonly known as SCRUM-But as in “we do SCRUM, but we don’t have a Product Owner in the Team nor we do Sprint reviews… they take too much time”.

Why not a Process? In order to be a Process it would have to be clearly defined with scope application, detailed activities with inputs/outputs, roles and responsibilities, some degree of activity definition and a repeatable capability. Although SCRUM checks some of the above marks, its lack of a clear scope definition defies the definition of a process.

Why not a Methodology? In order to be a methodology it would have to have detailed procedures and rules for all the activity described in itself. This couldn’t be farther from the truth, as we know SCRUM aims to be a not exhaustive, not prescriptive, no rules barred set of common accepted Agile good practices [Charvat2003].

Why not a Project Management Methodology? Again, it’s not a methodology because of the reasons above. It’s also impossible to call it a Project Management Framework because the Project Management Discipline has far more globally accepted Knowledge Areas not covered by the common SCRUM definition. Nevertheless, we can say that some of the SCRUM principles and practices are linked to the Project Management Knowledge Areas, especially regarding Communication, Scope definition and prioritization, Schedule, etc. This will be further detailed ahead.

Quoting the revised SCRUM Guide: “SCRUM is a framework for developing and sustaining complex products. (…) consists of roles, events, artefacts, and the rules that bind them together. (…) is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. (…) is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known. SCRUM employs an iterative, incremental approach to optimize predictability and control risk” [SGuide2013].

3.2.2 Lifecycle and activity flow

SCRUM lifecycle is rather simple and well known, it's based on small time-boxed incremental development cycles called Sprints and it's supported by a sequential meeting choreography.

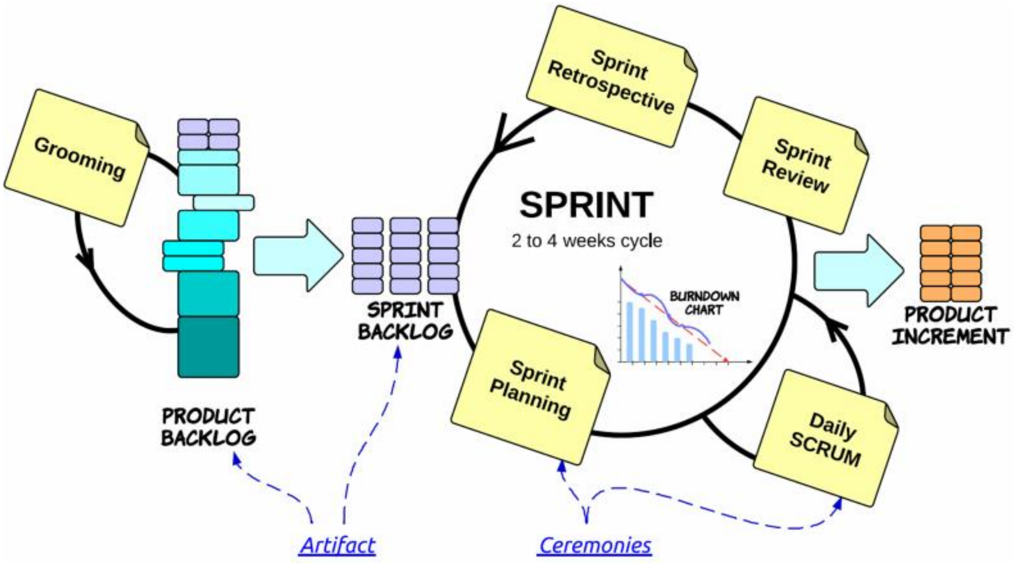


Figure 25 - SCRUM Lifecycle

Those meetings support the Sprint planning, execution and closing. Each meeting, also known as ceremony, has rules and principles applied to both Team and artefacts supporting it. These will be described later in some detail.

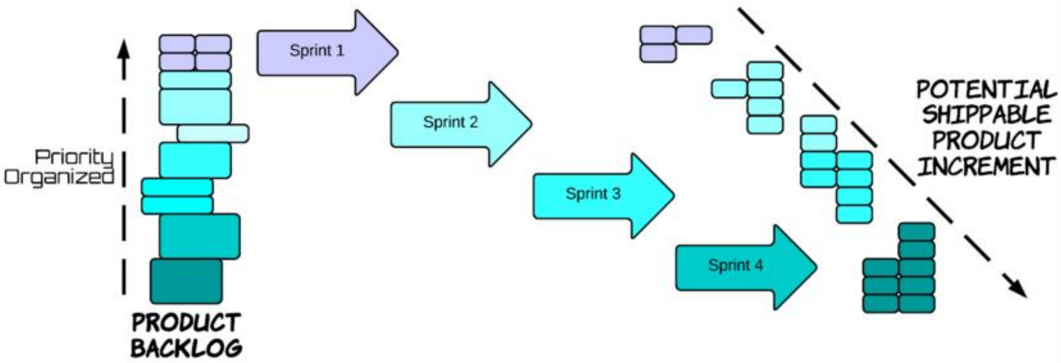


Figure 26 - SCRUM incremental delivery approach

Simply put, each sprint will end with an incremental set of product features being presented to the customer. The final product will be the result of the integration of all features.

3.2.3 Roles, Ceremonies and Artefacts

3.2.3.1 Roles

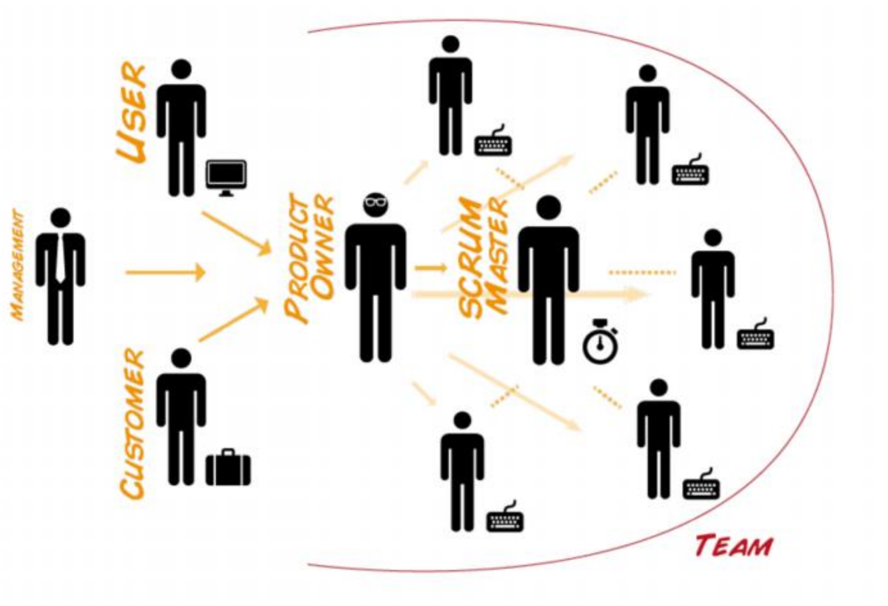


Figure 27 - Team landscape

Product Owner

The Product Owner's responsibility is with the final product. His function is to maximize value of the work being done by the Team, therefore maximizing the value of the final solution being developed.

He has the sole responsibility of deciding what to build and when to build it. Basically he manages the features wish list, the priority and value of each one, detailing and ordering the items in the Product Backlog to best achieve goals and missions [SGuide2013].

Even if not all activities are directly done by him, Product Backlog visibility, all-team understanding, feature detailing and validation are his sole responsibility.

The Product Owner may be the actual customer, but in the case that he is not, he is the customer representative within the Team.

Scrum Master

The Scrum Master is the responsible for adding or removing elements that may affect Team productiveness [Schwaber2011]. He is a problem solver, an issue fixer and he assures that SCRUM agreed-upon theory, practices, and rules need to be followed at all times [SGuide2013]. Basically, he oversees the process ensuring continuous improvement, and removes impediments to ensure project/product success [Keith2010]

The Scrum Master leads or maintains order in SCRUM Meetings, helps the Product Owner in finding effective Product Backlog management techniques and coaches the Team to self-organization [RisJan2000].

This role is normally mistaken for team lead or the Project Manager. The Scrum Master does not lead the team, instead he insures SCRUM adherence and continuous improvement, the Team will alone organize itself. At most, the Scrum Master may serve as a tiebreaker in specific occasions.

Also, the Scrum Master does not have legitimate authority to manage Scope, Cost, or Risk. Projectized Management may not even apply in some teams.

Team

The Team, sometimes referred as Development Team, is composed by all the members who build or develop the Product. Since the Team is built of an unchanging set of elements, they need to be internally cross-functional and holders of the needed skills to create the expected product. These characteristics will increase resource levelling[5]and facilitate teamwork.

Because the Product Owner and Scrum Master are not managers in the traditional sense, they need to allow self-organizing and work in internal consensus. This means that the Team is responsible for Sprint Planning, built product inspection, the Sprint Retrospective, and the continuous improvement of SCRUM practices.

In the course of this document, we will refer the "Team" as the Development Team with Product Owner and Scrum Master.

3.2.3.2 Ceremonies

Sprint Planning

Each Sprint starts with a Sprint Planning meeting. As the name suggests, this is the place where the Sprint is planned, meaning the Team will discuss and agree on a set of features that will be collaboratively implemented in the Sprint. This set is only a part of the overall features list presented in the Product Backlog, it's imperative to select the most high priority features or/and the one that bring more value to the customer as shippable and independent features [Sadd2012].

The meeting choreography starts with the Product Owner presenting the most valuable or higher priority User Stories and clarifying User Stories detail, the Team selects how many of those can be fitted in the Sprint and the Scrum Master maintains order and SCRUM good practices focus. This accurate selection is the result of two assumptions: the sprint time-box size being always the same and the historical average of how much work can be delivered in that time-box period based on past Sprints [Rubin2012]. User Story estimation and Sprint Velocity will be explained later on.

[5] Resource levelling - project management technique, consists in the analysis of unbalanced use of Project resources over time and finding ways to deal with over-allocations and/or other conflicts.

The meeting adjourns when a consensus is achieved on what will be the list of User Stories to implement, and that will be the Sprint Goal. This agreed upon consensus ensures the Team commitment for that Sprint [SGuide2013].

Daily Scrum

The Daily Scrum is a small and simple way to review Sprint status in a daily basis. It's a 15 minute meeting used by all Team members to synchronize the past and current day activities, assuring that all team members are informed of all being done at all times.
It normally takes place in front of the Task Board with each team member answering 3 simple questions:

- What did I do yesterday that helped the Team meet the Sprint Goal?
- What will I do today to help the Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Team from meeting the Sprint Goal [SGuide2013].

With this simple meeting choreography everyone is aware of the work being done in between Daily Scrums, reinsures Sprint commitment, issues surface and actions are agreed upon. Its main objective is to provide a safe haven to communicate daily status and re-focus Team collaboration on Sprint Goal.

Adaptations exist but it normally takes place in the early morning with all members standing-up to avoid deviation for the focused time-frame. Because the meeting objective is so clear and time is of the essence, the Scrum Master is responsible of making sure that all non-Daily Scrum subjects are dealt with in another meeting.

Sprint Review

The Sprint Review takes place at the end of the Sprint, in which the Team will present the developed feature to the Product Owner. It's basically a product demonstration, even if the product isn't complete, its goal is to demonstrate the individual working features implemented in that particular Sprint. Due to the nature of the meeting, this is also known as Sprint Demonstration or even just Sprint Demo [Sadd2012].

Feedback and Team collaboration are the main goal, as the Product Owner acts as inspector to the work developed, accepting the implementation done and/or providing feedback on the demonstrated features.

Commonly, the Team demonstrates the User Story in the Sprint Review, ending up with a list of revised User Stories, some accepted as "Done" and others needing additional work [SGuide2013]. Changes will be included in the Product Backlog and planned for the next sprint based on prioritization.

Sprint Retrospective

The Sprint Retrospective is a Team self-examination procedure that takes place after the Sprint Review. After demonstrating the Product, the Team should reflect on the SCRUM practices used and identify compliments and future Sprint improvements. It's a safe haven outside the overwhelming daily activities. Here the Team can be focused and share ideas on process and practices future improvements. Discussion may include not only SCRUM related practices but also other factors that may be used or influence the product development, like tools or stakeholder communication [Rubin2012].

Because it's an opportunity to celebrate success and identify improvements, this meeting is based on some simple, ice breaking questions that need to be answered by everyone in the Team. Definitions may vary, but the foundations are the same [Sadd2012]:

- "What things should we stop doing?"
- "What should we start doing?"
- "What is working well that we should continue to do?"

It's normal to give 3 bullet-type answers to each question, followed by a roundup of all answers. The Sprint Retrospective is for the Team only, but some say that external elements can be invited if value is identified [Sadd2012].
Again, the Scrum Master maintains order and act as a facilitator to positive discussion. He also takes notes on the answers given by everyone.

After all discussions, all elements agree on the most priority/value improvements. These take form as action items, which will be implemented and will have their status revised on the next Sprint Retrospective. This normally takes form as a plan containing the agreed Action Items. Again, this is an all-element collaboration exercise [SGuide2013].

The focus of the Sprint Retrospective is to allow SCRUM self-examination, identifying strengths and opportunities to continuous improvement in future Sprints.

Grooming

The value of the Product Backlog is linked to its detail level, organization, prioritization and size, so to have sufficient User Stories to feed the next Sprints. Because the Product Backlog is a living artefact, these commitments have to be frequent reviewed and are known as Grooming activities. These activities, focused on the Product Backlog Items[6], are divided in 3 groups: creating and refining Product Backlog Items, estimating Product Backlog Items, and prioritizing Product Backlog Items [Rubin2012].

[6] Product Backlog Item - entry in the Product backlog which represents a feature, defect, or technical work that brings valuable to the customer

Extensive Grooming activities may occur in the project kick-off, but they should be maintained in constantly throughout the Sprint cycle [Ko2013].

It's a collaborative activity that should be done by everyone on the Team in order to ensure a global understanding of the needed requirements. The key is a continuous progressively refinement of the Product Backlog, but mostly focusing on the most high priority Product Backlog Items. The customer value drives the refinement process, therefore the Product Owner is the responsible party.

Due to the nature of these exchanges, it's normal to invite external stakeholders in order to clarify requirements. End-Users, Solution Architects and others are a common presence in Grooming activities.

Grooming activities are focused on the Product Backlog and not in the Sprint cycle as the other choreography meetings. These may not even have a specific meeting in Sprint cycle, nevertheless they should be recurrent as needed. As a general rule the Team should allocate up to 10% of each Sprint to grooming activities [Rubin2012].

3.2.3.3 Artefacts

Product Backlog

The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any additional changes to be made to the product [SGuide2013]. Commonly known as the customer wish list.

The biggest and most important artefact is nothing more than a top-down list of the requirements, but some details are important:

- Content is always prioritized, the Team implements requirements based on customer priority (most-important are the first-done)
- It's a living artefact, in constant update due to new features, changes to existing features, defect, or technical improvements
- Its content, availability, and ordering are the Product Owner responsibility

As the concept is simple, with the introduction of specific SCRUM management applications, it's also normal to see some other information linked to the Product Backlog Items. Size, effort, complexity estimations are the most common, but Risk, sequencing and external dependencies are also possible [Rubin2012].

Sprint

SCRUM is an iterative and incremental approach to product development, resulting in work being performed in time-box iterations called Sprints. Each Sprint is planned, executed and completed, always with the main goal of delivering tangible value to the customer at the end of each Sprint [Rubin2012]. The Team commits itself to deliver the Sprint Goal.

Each Sprint follows the same meeting choreography and a new Sprint starts immediately after the conclusion of the previous Sprint. During the Sprint, scope changes are not welcome but can occur if imperative, always with the Team agreement [SGuide2013].

The community advocates a Sprint's maximum duration as no more than one month. Minimum duration is unclear, but return work value can be compromised, resulting in low productivity if the Sprint is too short, due to time-consuming choreography meetings. Two to four weeks is the golden rule, imperatively having always the same length in order to create consistent Sprint velocity.

Sprint Backlog

The Sprint Backlog is the set of Product Backlog Items selected for one given Sprint. The items on this list are selected by the Team and represent what they commit to deliver as the Sprint Goal. This selection is made based on previous estimations, the history of past performance and sometimes sheer gut feeling.

The detail contained in such a list is the sufficient one to ensure that everyone in the Team can follow progress in the Daily Scrum. It represents the team's commitment on what to deliver at the end of the Sprint.

Burndown Chart

The Burndown Chart is basically the always visible Sprint tracking mechanism, a simple chart in which Sprint progress is measured against the sprint commitments. This helps the Team to track daily progress until sprint completion. Units used can vary but it's always some unit that is universally understandable by the Team. Normally this chart also gives some sort of completion forecast calculated in line with the progress of the work being done by the Team. Scope unit, decomposition and estimations will be detailed below in Techniques and Concepts.

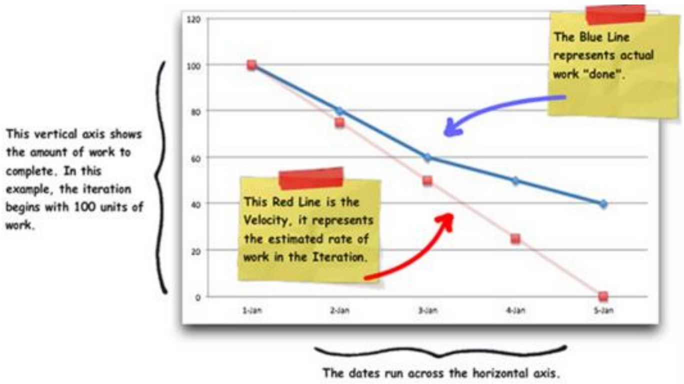


Figure 28 - Burndown Chart [Ikompass2013]

Because the units used are understandable and acceptable by the team, it's normal to print a Burndown Chart and pin it in the Task Board to be reviewed on the Daily Scrum. It's a simplistic progress and forecast of the Sprint commitments [Sadd2012].

Task Board

Also known by other names, the Task Board is one of the Lean, or even KANBAN, remnants in SCRUM. It's based on the notion that visual, always-updated task status are of easy and fast understanding. In this case, the units also may vary dimension-wise, but the common approach is that unit should be as small as needed to the level of being assigned to one Team member. The concept is that all Team members update the status of their unit.

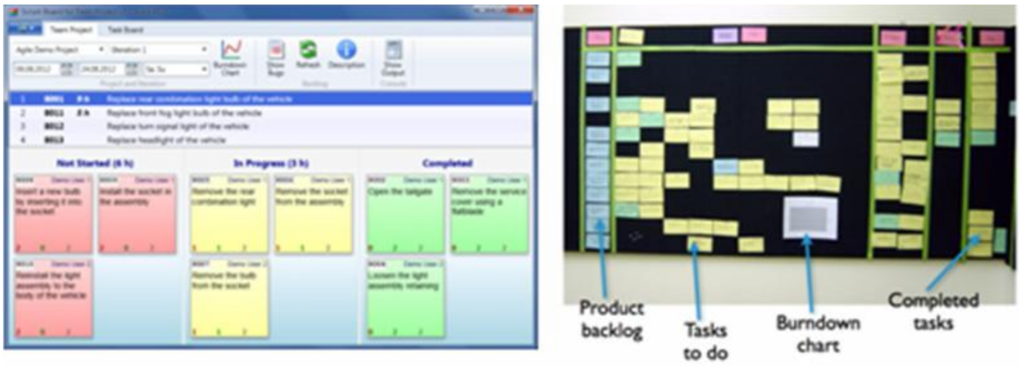


Figure 29 - Manual and Electronic Task Board [TBoard2013]

In the most common approach, a Task Board displays the goal, Burndown Chart, and tasks for a Sprint [Keith2010]. The more simple model has 3 columns, representing planned tasks, tasks in execution and closed tasks, normally implemented as "To Do", "Being Done" and "Done", respectively. Normally units in the board are represented by post-its.

Board customization is the most common adaptation to the SCRUM framework, variable from the columns on the board, the decompositional level of units in the board to usage of swim lanes. With the usage of SCRUM support applications, the physical board is sometimes replaced by a virtual one, or even by a spreadsheet.

But one fact is indistinct, the physical board is regarded by the community as one of the most useful artefacts to SCRUM.

3.2.4 Techniques and Concepts

User stories

Requirement Engineering activities are generally hard to do in any development life cycle. User Stories are one of the most groundbreaking innovations to traditional IT Requirement Engineering. A User Story is a simple and commonly understandable functionality description of the customer desired product. It's the result of a direct and clear communication between the Team and the Customer or end-user. User Stories are the means used by the Customer to express his product wishes on the form of envision use-cases, establishing a commonly understandable form of communication between Customers and the Team. The Product Backlog is mostly composed by User Stories.

User Story definition is one of the most customized of the SCRUM concepts, any company using SCRUM has its different concept, but a few principles should always be present: existence, detail and acceptance criteria [Cohn2004]. A simple written description is imperative to then be part of the Product Backlog. Prioritization, estimation and User Story detail come afterwards, by that order. This short User Story description may be sufficient to proceed with development.

The need to have User Story low-level detail can be a consequence of not having clear and direct communication to the Customer or end-user. Sometimes the User Story is used only as a reminder to the detailed communication that took place, with both the Team and the Customer having the recollection of the detail requested. A User Story can also be a remainder to a future conversation, which may indicate that that User Story is important, but not urgent at the time. This bi-understandable detail level may be sufficient if both Customer and the Team are comfortable with it. There are no magic formulas, and adaptations should be done case-by-case.

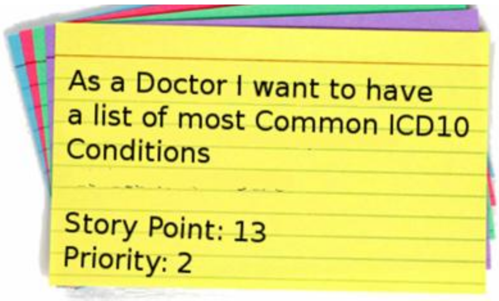
A User Story also needs an acceptance criteria, some agreement which may also be part of the User Story detail description, which is both accepted by the Customer and the Team as the criteria that both acknowledge that the deliverable completes the initial Costumer request. This concept is linked to Software Testing and Customer Validation, and generally known as de Definition of Done.

An Extreme Programing author came up with an acronym, INVEST, generally used to describe how to create good User Stories [Cohn2004]:

- **Independent** – dependencies between User Stories should be avoided to ensure a simpler prioritization, estimation, and Sprint Planning. Detail levelling and User Story split can be common tactics. On a best case scenario, all User Stories are independent.
- **Negotiable** – The Agile mindset is Customer-focused, therefore User Stories may change, be edited or cancelled easily. Since they are the communication understanding between Customer and the Team, they are in constant negotiation and should be written in such a matter that embraces and amplifies easy User Story negotiation.
- **Value to the users or customers** – this is on the pillar of Agile - focus on what the customer really wants. User Stories must bring value to the customer and he should easily acknowledge it. Customer value is very useful information when it comes to User Story prioritization.
- **Estimable** – regardless of the unit used or the level of confidence on the value, a User Story should be estimated in size, shape or complexity. The units used may vary but the objective is always the same, estimation is held in consideration with User Story

Value by the Customer and Team. One User Story may have a high value but its size can push the User Story to future Sprints.

- **Small** – User Story decomposition is almost a mitigation of Scope Creep[7] and other scope related risks. By decomposing big User Stories to smaller, more easily manageable User Stories, the Team and Customer can avoid scope and size uncertainty behind a big User Story and also making it small enough to decompose into smaller tasks. Again, no size is defined as optimal, but User Stories should be decomposed as a rule, and the Team has to find its sweet spot based on Sprint duration, Team size and customer decomposition ability.
- **Testable** – User Stories must be understandable both development and test-wise. If it's not clear how to test a User Story, probably the User Story isn't well written in the first place. The Customer may have some difficulty to describing test scenarios, but the Team needs to provide support and encounter common ground.



As a Doctor I want to have
a list of most Common ICD10
Conditions

Story Point: 13
Priority: 2

Figure 30 - User Story example

User Story usage, detailing and estimating are one the most adapted concepts of SCRUM. From the well-known "As X, I want to Y so that Z" template to more complex requirement engineering approaches, from very accurate estimation methods to not as simple or no estimations at all, each SCRUM implementation will have a different storytelling and use.

Small, Co-located teams

SCRUM is based on easy access to the Customer, a direct all-stakeholder communication, and small development cycles. Two serious negative factors are Team size and proximity or lack thereof.

The SCRUM Guide states that "Optimal Development Team size is small enough to remain nimble and large enough to complete significant work within a Sprint" [SGuide2013]. Again, there are no magic numbers but many point as a 5 to 7 element Team to be a good interval. Choreography-wise, if too small, productivity will not compensate the choreography weight, and if too big, it will be hard to manage planning and other choreography ceremonies. A maximum of 9 is also a common ground as being too hard to manage.

[7] Scope Creep – deviation from defined scope duo to uncontrolled changes and/or continuous growth in a project's scope

In the other hand, location is also a risk factor. Having a war-room and a co-located Team will be the perfect scenario, this will increase successful communication and other factors. Unfortunately, today's IT World has not only distributed teams but sometimes overlapping work hours. Other negative factors of close-proximity based models include different time zones, different cultures and languages, among others. Some tools and techniques may be used to address this issue, but impact should always be expected and mitigated [WSG2010].

Story points

Although the SCRUM Guide and other similarly-official frameworks don't specifically mention this technique, the Story Points estimation technique is one of the most talked about groundbreaking innovation of the last decade, introduced by James Grenning in 2002 [Grenning2002].

Estimation is always hard, and estimation accuracy will always be influenced by the estimation units used, who is doing the estimation and who is performing the actual work. The main idea behind the technique is reducing the influence of non-Team members and promote positive discussion within the Team to agree on a size-based scope estimation. Basically, a technique for estimating effort or work size, which follows some concepts [Rubin2012]:

- Consensus based
- Expert opinion Intense discussion
- Relative sizing
- Accurate grouping/binning
- Leverage estimating history

The assumption is that traditional cost, time or effort estimations, sometimes set by management, lead to wrong planning and execution. In this case, the estimation has to be a consensual one, based on input from the most subject-expert Team member, and then discussed by all.

To avoid traditional approaches, the Team uses a Fibonacci based scale that represents size or/and complexity of the User Stories. The scale representation is different from Team to Team, with each Team discovering her own understanding. It's common to set one User Story to be a baseline, a previously estimated User Story, used as comparison to help estimate the remaining ones.

The estimation procedures have their own choreography. Due to the similarly to the Poker card game, it's generally known as "Poker Planning", or something similar. The Product Owner or the expert element presents one User Story at a time, and each Team element picks a card from their own deck face down. All the cards are turned at the same time. If values are different, the parties discuss the source of the different values, therefore aligning some details that may be forgotten by some. Consensus has to be encountered, until everyone in the Team is comfortable with the final voting value. These procedures are normally done in the Planning Meeting or the Grooming Meeting.

As before, Poker Planning, its estimation scale and estimation unit-level used are only good concept recommendations. Many visions exist and no two teams do this technique alike, but if its concepts are followed, time will grant the Team the accuracy on their estimations.

Sprint Velocity

As described before, the User Stories are estimated in Story Points, this means at a certain moment in time some of the User Stories on the Product Backlog have an estimation value, which relates to the User Story size or complexity. Even if we may not map that size with effort, schedule or cost, the Team knows its meaning and how much User Stories can be implemented in a time-box. Sprint Velocity is the amount of Story Points a Team can deliver in one Sprint, therefore the average amount of work a Team can complete each Sprint.

As the Poker Points, the Velocity is a concept easy to understand but difficult to begin using. Because Sprints are of fixed length, with time the Team starts to achieve a more accurate Story Points estimation, therefore achieving a more consistent Sprint Velocity.

Sprint Velocity main goal is consistency and not evolution. Consistency will bring scale stability that can be used for forecasting the remaining Product Backlog using historical Sprint Velocity. This can only be done in SCRUM mature teams. When starting in SCRUM is normal to have an unsteady velocity, but in time the value will converge to a smaller deviation interval (range).

Velocity consistency is linked to good estimation and Team commitment and maturity. Several factors may influence Sprint Velocity. Changing Team elements or vacation periods may affect Sprint Velocity but they should not be seen as a performance decrease indicator [Rubin2012].

Scope Decomposition

The concept of incremental and iterative development also implies that the scope is to be progressively refined and decomposed throughout the several Sprints. Although Sprint scope is normally composed by User Stories, which represents features requested by the Customer, that's not the only level of scope decomposition used in SCRUM. The User Story is normally sized to be included in a Sprint, therefore they are small.
Because the Product Backlog is in constant update, it's normal to need different representation for customer requests. Some will be a User Story-like detail request, or something bigger that will need to be decomposed in User Stories in the future.

Epics are larger User Stories, ones that can take several sprints to complete. They are used to communicate large groups of requirements with the Customer, and act as a reminder for future decomposition. When they have enough priority, they will be decomposed in smaller, more manageable User Stories [Rubin2012].

Tasks are normally technical work, decomposed in the Sprint Planning as set of activities to complete a User Story. They are commonly small, daily activities, performed in a few hours by the Team. It’s not uncommon to being estimated in effort-hours during decomposition [Rubin2012].

Releases are also a used grouping concept. It’s is a set of features that when packaged together make for a coherent deliverable to Customers or Users [Rubin2012].

Other decomposition scope units may exist, but again its part of adaptation. Epic-User Story-Task is the most used, Release is adaptation towards Release Management practices.

3.3 Detailing KANBAN

3.3.1 Origins

As referred in sub-chapter 3.3, the origins of KANBAN are derived from the Toyota production System, with Lean being the cornerstone. A simple visualization-based card system with a workflow aimed to manage production and potential problems in-between manufactory productions steps. Much could be said regarding potential implementations, but in this sub-chapter we will focus on the IT related model instead of the manufacturing one.

As said before, Taiichi Ohno is regard as the father of Lean Management, and maybe even of KANBAN, at least manufacturing-wise. However, the adaptation of KANBAN to Software Development is recent, some describe it as a second generation Agile approach [Shall2011], based on the original KANBAN from Toyota, and Agile based models like XP and SCRUM. That said, many refer David Anderson as the father of the KANBAN Method for Software Development, after his publications in the 2000s.

Based on the visual card system and adding agile principles like incremental building and self-organized teams, Anderson introduced a model that we will use in the course of this document as a possible KANBAN application to Software Development.

By changing the manufacturing board processes with common Software Development phases and using card-signs to represent work items decomposed from the requirements, we end up with a simple Software Development visual workflow, with all the original KANBAN benefits.

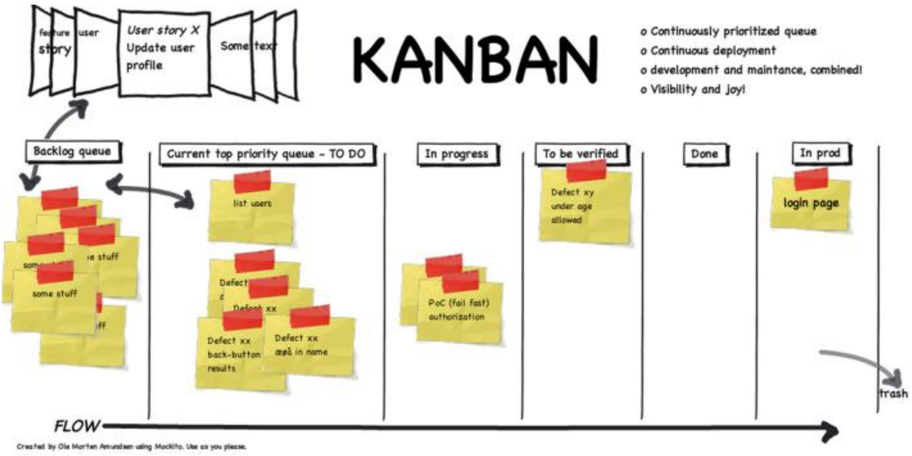

feature story
user
User story X Update user profile
Some text
KANBAN
o Continuously prioritized queue
o Continuous deployment
o development and maintance, combined!
o Visibility and joy!
Backlog queue
Current top priority queue - TO DO
In progress
To be verified
Done
In prod
list users
some stuff
some stuff
Defect xy under age allowed
login page
Defect
Defect xx
Defect xx back-button results
Defect xx æøå in name
PoC (fail fast) authorization
trash
FLOW
Created by Ole Morten Amundsen using Mockito. Use as you please.

Figure 31 - KANBAN Board example [Morten2013]

Development cycle details may differ by making some changes in the KANBAN board columns, but with this easy adaptation development phases can be easily mapped and a workflow can be instantly acknowledged.

As a variation of Toyota's KANBAN, IT KANBAN is based on some core pillars [Anderson2010]:

- **Visual-based workflow**, simple and easy to understand
- **Just-in-time** (JIT) management, were wasteful activities are identified and eliminated, building only what is needed when needed
- **Limit Work-In-Process** (WIP) based on the Team capacity, revised but not exceeded at all times
- **Manage and optimize the flow/process** constantly seeking ways to improve Lead-time and remove flow bottlenecks
- **Process and Policies need to be explicit**
- **Improvements are done collaboratively** by the Team

The core basis of both Agile and original KANBAN concepts and practices is easily acknowledged above, from the Just-in-Time philosophy, in which the complete elimination of waste is attained, by making only what is needed when it is needed and in the amount needed, to the Agile collaboratively improvement approach [Toyota2012].

Nevertheless IT KANBAN, by itself, is not a Software Development lifecycle methodology or an approach to Project Management. It can be seen as a framework, but unlike SCRUM, it does not lay roles, artefacts and concepts besides a Board, a workflow and some core principles. It can also be seen as a performance amplifier for an existing process or practice already in place, which will improve cycle-time, predictability and continuous improvement, whichever the improvements may be.

This means that we can create an IT KANBAN over core Software Development practices, using the core KANBAN principles to focusing on flow management improvement, without having any fixed complicated Software Development frameworks. Keep in mind that this would mean that only flow management and continuous improvement can be identified as results of this approach [Anderson2010].

In the following course of this document, we will refer to the IT KANBAN simply as KANBAN.

3.3.2 Lifecycle and Activity flow

Contrasting SCRUM's time-box incremental lifecycle, the KANBAN lifecycle is defined as needed, with the KANBAN board taking form to represent the Software Development process and its phases. The sequence of all these activities is called KANBAN Workflow.

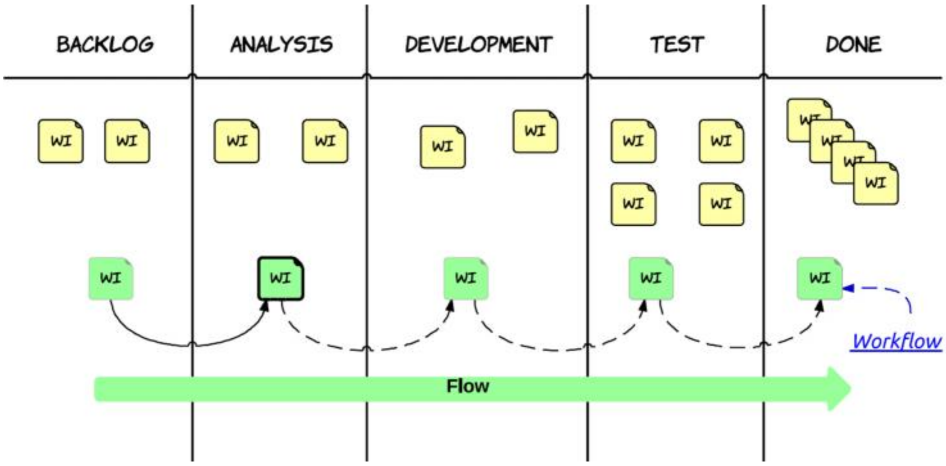


Figure 32 - KANBAN Workflow

As a result, Work Items travel through the Board, only doing so by completing the activity on each column. The example above is a simple one, other columns may be added as needed. This concept will be further detailed during the rest of this chapter regarding KANBAN Board and flow.

3.3.3 Roles, Ceremonies and Artefacts

Unlike SCRUM and other Agile Frameworks, KANBAN does not set any Roles or specific meetings during the workflow of activities.

3.3.3.1 Artefacts

KANBAN Board

The KANBAN Board, also known as card wall, is the most important artefact in the KANBAN framework. Not only is it a physical-tool, it also includes activity flow, work control and performance indicators.

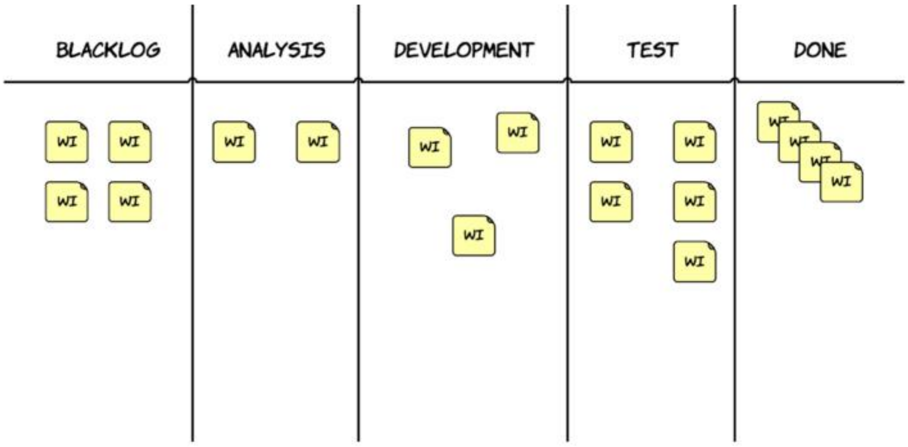


Figure 33 - KANBAN Board

Normally on the KANBAN board each column represents a phase of the Software Development process. The Board's complexity may range from simple to complex. It's normal to start with a simple Board, but the continuous improving nature of KANBAN will probably evolve it as needed. The flow of cards in the board dictates the development workflow itself, normally flowing from left to right.

Each column can have input queues or buffers, and is marked with a WIP limit, which represents the maximum Tickets number (Work Items) in that phase of the development process at any given time. The Team decides the order in which to pull the Tickets from one column the next, based on prioritization and the next column's WIP limit.

Normally the first column is a stack of all the existing Tickets, resulting in being similar to an Agile Backlog, and the last column represents a "Done" state [Budau2012]. Priority and other information regarding pulling order may be written on the Tickets themselves, or be represented as the position of the Tickets in the Board. It's normal to have high priority Tickets positioned in a higher position in the Board. From finding meaning in the Tickets' position, to a system of Tickets colour, the possibilities are endless.

KANBAN Tickets

By definition, in the adapted IT KANBAN, the KANBAN Tickets represent Work Items, similar to User Stories from SCRUM or other Agile methods. The concept is the same; usually bigger requirements are progressively refined and decomposed to smaller, preferably independent, more easily manageable User Stories, or even the Tasks to implement them. Although not a rule, all Work Items should be decomposed to similar levels, resulting on each ticket being of equal weight, which foments measurement accuracy. This will be better explained later on.

3.3.4 Technics and Concepts

Pull system

KANBAN is a visual board-based Pull system, meaning that elements on the board, Work Items, flow from the beginning to the end of the KANBAN Board by being pulled from one column to the next. The pull movement represents Work Item completion in the past column, therefore being a fulfilment of the Definition of Done on that specific column. Each column's Definition of Done is written in the bottom of the column, as a reminder to which conditions need to be fulfilled.

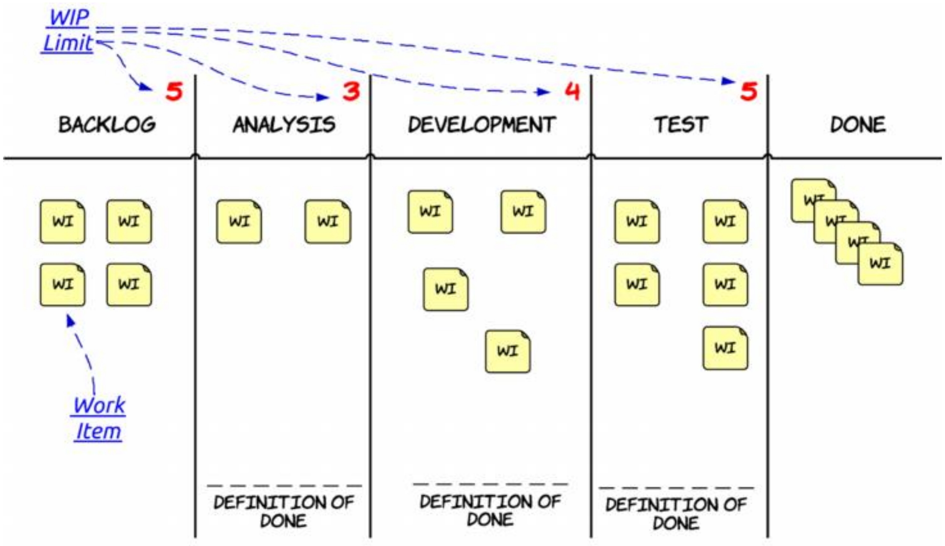


Figure 34 - KANBAN WIP Limits

Limited Work-in-Progress (WIP)

WIP is one of KANBAN's fundamentals. It's the amount of work being done at any moment in time. KANBAN advocates a WIP limited pull system, meaning that WIP should be limited and therefore be enforced on all the KANBAN Workflow. Common implementations limit the KANBAN columns with a maximum Work Items count at any giving time, like a threshold, resulting in constrains when pulling new Work Items if the limit is exceeded. Work Items can only be pulled to the next column if the WIP limit for that column is not exceeded [Anderson2010].

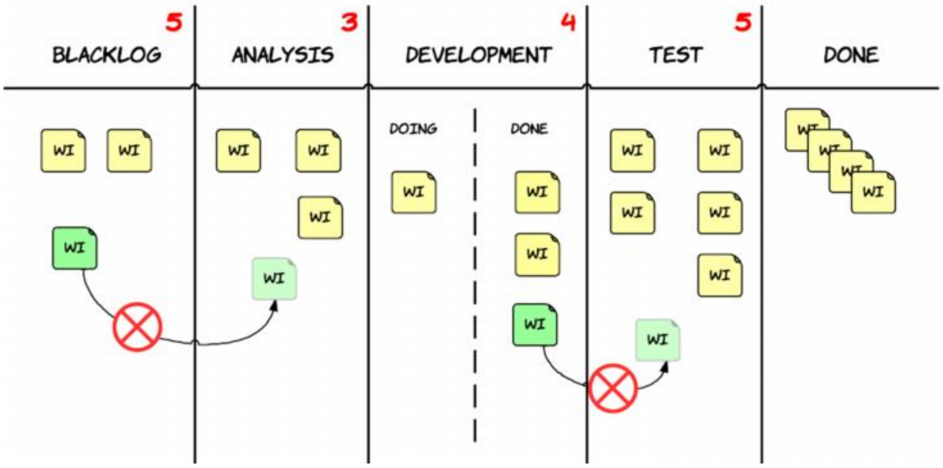


Figure 35 - KANBAN bottlenecks

In the example above, both the "Analysis" and "Test" WIP Limit was reached, new Work Items cannot be pulled until some move-on to the next column. Limited WIP also exposes KANBAN flow bottlenecks, the above example shows Work Items already concluded in the "Development" column, but they cannot advance because the "Test" column reached the maximum WIP. These situations are analysed by the Team in order to find a solution, Continuous improvement at is best.

Cycle-time

It is a time measure, starting when a Work Item begins to be implemented until being considered done. Normally begins when the Work Item is pulled from "Backlog" column to the next, until it reaches the "Done" column. It's a basically a metric of the KANBAN system capacity, and a performance completion indicator [Budau2012].

Lead-time

It is also a time measure, measuring the time from the customer request creation to the delivery of such request. It's a customer request response time, a metric that gives us the customer's own vision, and can be used to forecast future response times and give timeframe delivery commitments. Lead-time includes the Cycle-time and is basically a request arrival indicator.

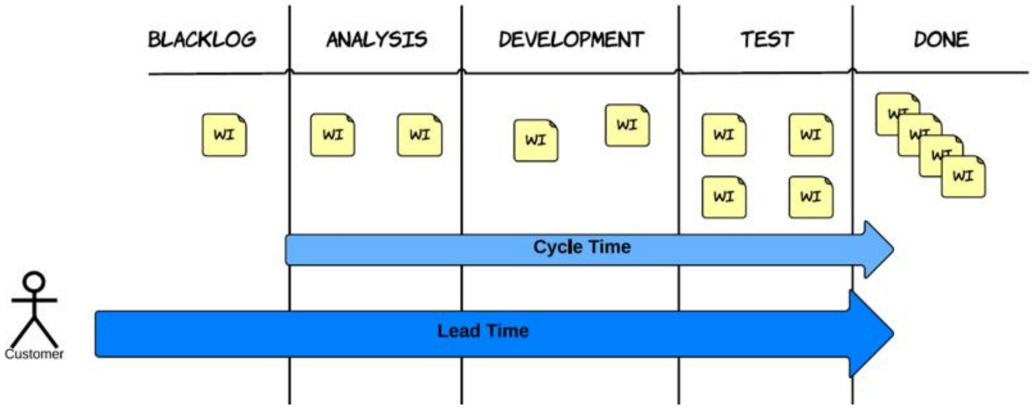


Figure 36 - KANBAN Cycle and Lead Time

Class of service

Service diverseness is a common concept in IT requests. From emergency-fix, code-bug or optimization, the list is endless and differentiation may come as a benefit in more mature KANBAN systems. Prioritization and differentiation is the main objective of Class of Service. By classifying Work Items we can define acceptable levels of response or risk management constrains, leading to different policy definitions for each Class of Service. Class of Service Policies regulate priority and how a Work Item should be pulled throughout the KANBAN system [Anderson2010]. Classes of Service are typically defined based on Cost of Delay, and the general practice is to use different Work Item card colours to distinguish the different Classes of Service. Presented below are examples of the mostly common used Classes of Service [Anderson2010]:

- **Expedite** – known as the silver-bullet, it's basically an emergency that needs immediate response. It can have a different workflow in order to jump some of queues or reassign some resources.
- **Fixed Delivery Date** – a Work Item which may be delayed until closed to its deadline date. Normally used for contractual obligations or regulatory requirements that may have some slack time.

- **Standard Class** – standard Work Item, with an immediate delay cost and lesser customer urgency. Has a linear delay cost, to be delivered as soon as possible. It's the most common Class and can also have sub-classes if needed.
- **Intangible Class** – despite the potential value, it's a Work Item with a delay cost that cannot be established in the timeframe that it might take to be delivered.

Cumulative Flow diagram

The Cumulative Flow Diagram is an innovation enabled by digital KANBAN tools available on the market today. This diagram shows real-time WIP for each Column, Cycle-Time, Lead-Time and other metrics as shown below. This is a very useful tool to make sure KANBAN Workflow is operating properly and to support the continuous improvement actions and reviews.

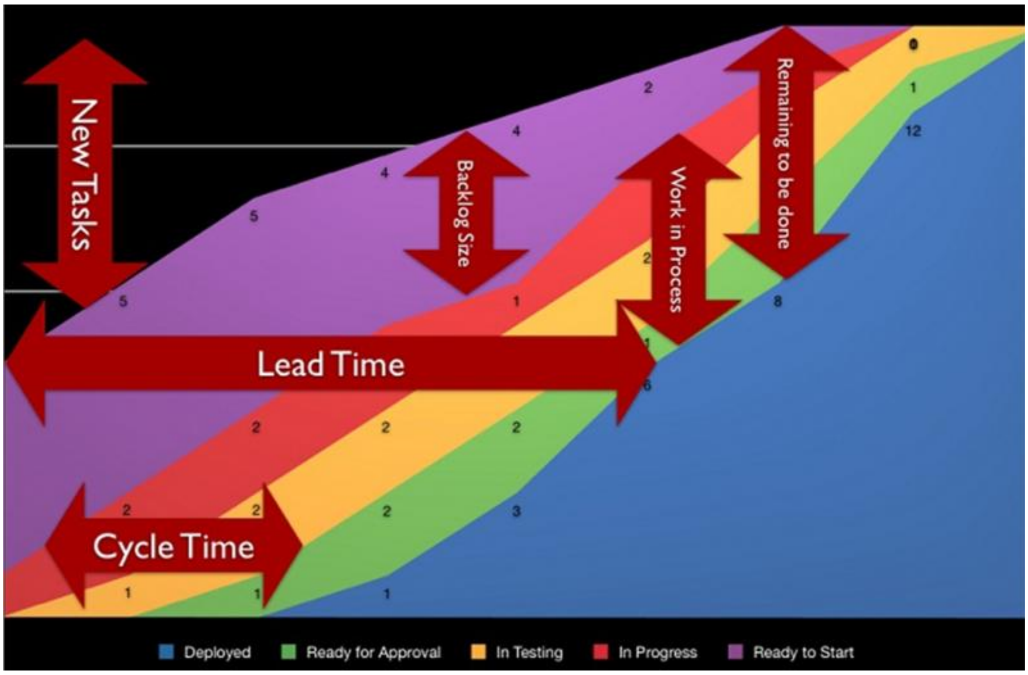


Figure 37 - KANBAN Cumulative Flow diagram [hunskaar2013]

Throughput

Throughput is the amount of Work Items done in a given period of time. It is basically the output ratio of work delivered to the customer. The KANBAN's main goal should be improving Throughput, which is constrained by bottlenecks. It is the balance between WIP and Cycle-Time [Nain2004].

Swim lanes

Swim lanes are horizontal rows that intercept several columns of the KANBAN board. Its main objective is to organize Work Items by limiting them to specific swim lanes. One common use of this is to distinguish product scope, by having a different swim lane for each product developed by the Team.

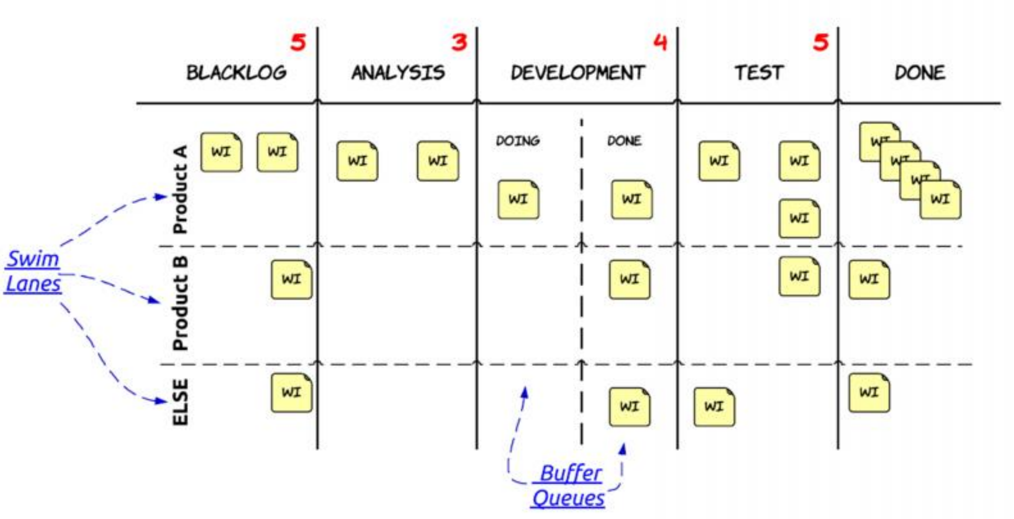


Figure 38 - KANBAN Swim Lanes and Buffer Queues

Buffer Queues

Buffer Queues are subsets of a given column, usually created to visually divide the Work Items in a given column, but also to find or correct bottlenecks easily.

3.4 Pros and Cons

As George E. P. Box said “essentially, all models are wrong, but some are useful”. If we think SCRUM and KANBAN not as frameworks but as Software Development process support tools[8], none is perfect but they have strengths and weaknesses.

The following sub-chapter will compare both, cross them with other pairs and advise application situations.

3.4.1 Shared Concepts of both models

3.4.1.1 Just-In-Time (JIT)

SCRUM and KANBAN are both pull scheduling systems. This concept is aligned with the JIT inventory management principle of Lean, focused on reducing in-process inventory and its associated carrying costs. In terms of SCRUM and KANBAN this means that we focus on the Work Item most valued by the customer, and that the right things are in place at the right time so we can develop the Work Item and deliver them to completion. Teams choose when and how much work to commit to, and they “pull” work when they are ready, instead of having it “pushed” in from the outside by management.

3.4.1.2 Limited WIP

Both SCRUM and KANBAN champion the limited WIP has one of its core pillars, a Lean heritage. Michikazu Tanaka said “by reducing work-in-process we motivate people to do better than they ever thought they could do, resulting in a raise in productivity and illuminating problems. By limiting the WIP we also reduce multi-tasking and task-switching, resulting in more Work Items being stated and ended without interruptions”.

SCRUM uses Sprints to limit WIP, in a time-box approach. Based on the Sprint Velocity, the Team uses the Sprint Planning to limit the work capacity for each Sprint. That commitment defines an almost sacred amount of WIP.

This WIP limitation results from two factors: estimated User Stories, using Story Points, and Sprint Velocity history from past Sprints. Literature says that when the Team finds itself finishing all User Stories before the end of the Sprint, they should pick other User Stories from the Product Backlog and try to finish them, reflecting this as a Velocity increase for the next Sprint. Although, Student Syndrome[9] contradicts this practice, SCRUM supporters say while feasible, even if the brings velocity uncertainty, it will become more stable in the long run.

[8] Tool = anything used as a means of accomplishing a task or purpose. Process = how you work [KniSka2010]

[9] Student syndrome - phenomenon which states that many people will only start a task at the last possible moment based on its deadline and duration. This makes estimations and scheduling less valuable [Goldratt1997].

KANBAN uses WIP limits in each KANBAN column to limit the amount of WIP at all times in a flow approach. Because normally Work Items are not estimated, the Team relies on the average Cycle-Time, the Class of Service and other measurements to manage and adjust Work Items flow based on each Column's limited capacity.

3.4.1.3 Deliver smaller units of work

Other Lean concepts used by both SCRUM and KANBAN are decompose big requirements to simple, more manageable units and deliver then incrementally. Instead of receiving big deliverables that take months to finish, customers will value sliced fully-working product parts delivered in small periods. Names and techniques may vary but the principle is the same in both models. This brings benefits and reduces risk at many levels, from customer communication to WIP management.

SCRUM delivers chunks of features at the end of the Sprint. KANBAN will deliver features independently with no time-box interval, with continuous flow as the key.

3.4.1.4 Too simple Development Lifecycle

As said before, both KANBAN and SCRUM have a simple Development Lifecycle. Considering the generally accepted approaches of both Software Engineering and Project Management practices, they are focused on scope, team activity and schedule management to deliver the product (SCRUM with the Sprint cycle and KANBAN with the Board Workflow) assuming that other activities exist before and after [Larman2004].

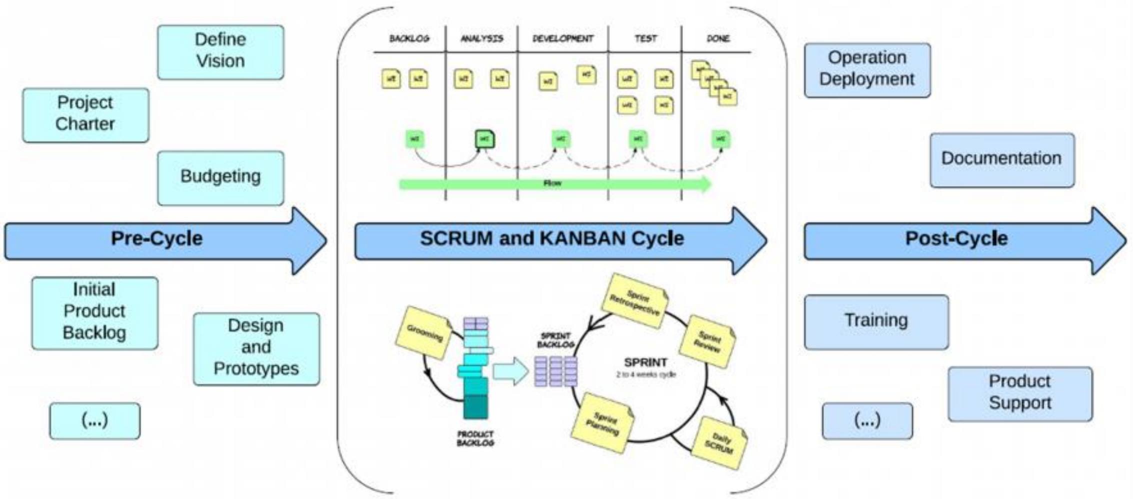


Figure 39 - SCRUM and KANBAN Pre-Post-Cycle [Larman2004]

The phase names or activities may vary, but we can easily agree that a pre-Development and post-Development activities/phases need to be added to the simple not exhaustive Sprint or KANBAN Workflow cycle to ensure some needed Software Development, Project Management and IT Governance practices [Larman2004].

These pre and post cycle activities are usually done regardless of company, project or Team, but they are not always implemented or managed by the SCRUM/KANBAN lifecycle or even the SCRUM/KANBAN Team. They are many times too distant from the SCRUM/KANBAN lifecycle and are not considered Product or Project scope.

Benefits will come by having Project Management, Release Management, Deploy Management and other activities to create a richer life cycle. If adding those activities to the SCRUM and KANBAN lifecycle is impossible, alignment should always be assured and empowered. Some may say that some of these activities are easily mapped as Work Items and consumed on the Sprint or KANBAN Workflow, which may also be a possible approach.

3.4.1.5 Lower cycle time

One of the core principles in Agile is to have shorter cycle-times which combined with delivering smaller units of work, results in faster delivery to the customer and therefore having early feedback for the delivered product. Theoretically this results in less rework and less waiting by the customer.

KANBAN cycle time is based on already existing items on the Board and the ability of pulling new Work Items when possible. SCRUM manages this in the Sprint time-box and relies on SCRUM Review to validate the product and receive feedback.

3.4.1.6 Self-organizing and Cross-functional teams

Both KANBAN and SCRUM focus self-organization and cross-functional skills as a success factor. On both models self-organizing is key, teams choose the best possible way to complete Work Items and adjust processes and procedures without the intervention of upper management. It’s a power-to-the-people continuous improvement concept.

Cross-functional Team vision allows not only having all the needed skills to complete Work Items, but also works as a resource levelling technique, as every element can pull any Work Item [SGuide2013]. In the real word this is not easy to accomplish but efforts should be done to encourage it. Both models will benefit if any Team element has the skill to complete every Work Item, but this not mandatory in KANBAN.

Cross-functional teams are a bigger necessity to SCRUM due to all-team-member commitments during the SCRUM lifecycle. User Story estimations should be done collaboratively by everyone in the Team in order to achieve a consensual Sprint Goal. Having different internal roles like developer and tester may impact the consensus and therefore the commitment of the Team as a whole.

3.4.1.7 Delivery Cadence

Both models deliver the product to the customer, but they do it in two different ways. A delivery cadence establishes the capability of a Team to reliably deliver working software at a dependable velocity [Poppen2005].

SCRUM will deliver features in a Sprint, a time-box cadence, managing its planning with estimations and Sprint Velocity.

KANBAN decouples cadences (or "Iterationless" Development) by using Throughput and Cycle-time to manage Delivery Cadence [Anderson2010].

3.4.2 Comparison

Aligned with Agile and Lean, both SCRUM and KANBAN are pull scheduling systems based on continuous and empirical process optimization, a Kaizen[10] principle [KniSka2010].

SCRUM is about delivering chunks, KANBAN is about continuous flow, but both are focused on the customer and respond with some degree of agility to change, even if in different ways. Both limit WIP, release software often, rely on delivery measurements like Sprint Velocity and Lead-time, progressively refine scope as they proceed and are based on self-organized teams.

Artefacts

SCRUM	KANBAN
Board only contains the Sprint User Stories and tracking information; Always rearranged for the next Sprint	Board is persistent and contains vast amounts of tracking and management information
A groomed, estimated and always prioritized Product Backlog is mandatory	An input queue is needed (start column); Empirical event-driven prioritization
Burndown Chart is a mandatory artefact due to the importance of commitment management during the Sprint	No specific diagram is prescribed

Table 3 – SCRUM and KANBAN comparison – Artefacts

[10] Japanese for "improvement", or "change for the better", philosophy focused on quality enhancement and waste reduction through continual improvement in manufacturing, engineering, and business management.

Roles and Team

SCRUM	KANBAN
Product Owner, Scrum Master and Team are mandatory and without them SCRUM simply does not work	No roles are defined as a KANBAN necessity or even as a good practice
Team must be cross-functional, collaboratively working to achieve the Sprint Goal	Specialists in the Team are allowed and managed accordantly to achieve flow
Team is stable (fixed) preferably with full allocated elements. Reinforcements, one-time helpers or other resource changes will impact future Sprint Velocity	Team elements may be shared with other teams or projects. Team reinforcing can be done in a specific KANBAN column (activity)
Small teams, from 5 to 7 elements, preferably co-located	Can be used for big teams and even more than one Team. Co-location is not required but it's a bonus regarding Board visualization benefits

Table 4 - SCRUM and KANBAN comparison – Roles and Team

Ceremonies

SCRUM	KANBAN
Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective are mandatory, Grooming sessions and others are option	No defined ceremonies, the same SCRUM principles are done continuously throughout the flow

Table 5 - SCRUM and KANBAN comparison - Ceremonies

Techniques and Concepts

SCRUM	KANBAN
Same-duration Sprint iterations are mandatory (time-box-driven)	Continuous Flow focused and event-driven, no Iterations are prescribed
Team commits with Sprint Goals	Commitments are optional

Estimations are mandatory to ensure Velocity	Estimations are non-compulsory
Velocity is the main performance indicator	Lead-time and Cycle-time are the main performance indicators
WIP is limited by Sprint and revised on upon Sprint Retrospective	WIP is limited in each KANBAN column, constantly revised based on capacity and event changes
Changes that interfere with the Sprint Goal will wait for the next Sprint	Changes are added immediate if KANBAN column WIP limit allows it

Table 6 - SCRUM and KANBAN comparison – Techniques and Concepts

The purpose of this exercise is not to find a winner, they are both winners in their own way. But this brings some other questions: which one should be used?
In the next sub-chapter we will address this question. However, finding an easy answer is problematic, because every project, Team and company is different in its own way.

3.4.3 What to use?

Ok, they are both good and have legions of enthusiasts, but what should you use?
The easy answer is: it depends!
A good analogy is that KANBAN and SCRUM are like a Fork and Knife meal-wise [KniSka2010]. You know the potential of both, you need both, but sometimes you need one more than the other. You want both, but sometimes you only get to pick one. But, if you pick one, can you adapt it to become both, like a Swiss army knife?

3.4.3.1 How much can be adapted

Let's analyse the issue from an adaptation degree point of view. As seen in the previous sub-chapter, both models have mandatory artefacts and some techniques and concepts. Some can be bent, adapted, or simply not used.
SCRUM, with mandatory roles and ceremonies, is clearly more prescriptive than KANBAN, but is clearly less prescriptive then other frameworks or models, therefore it is adaptable in some degree.

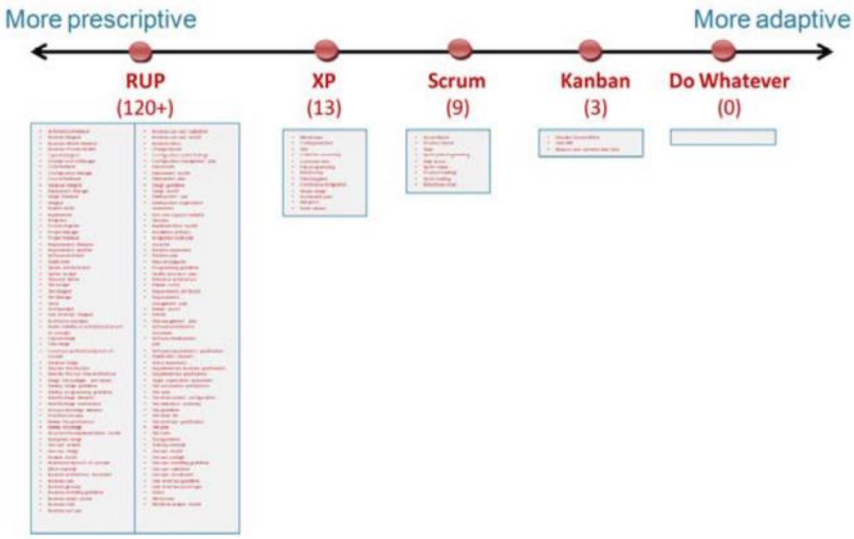


Figure 40 - Agile adaptability comparison [KniSka2010]

IBM Rational Unified Process (RUP) is very prescriptive with its more than 120 roles, activities and artefacts. XP is a bit more prescriptive then SCRUM, but KANBAN is less prescriptive [KniSka2010].

Keep in mind that both founders of SCRUM revised the framework definition in 2013 in little more than 10 pages, a few meetings, roles and concepts. This opened a community revelation, one that never should be forgotten: as designed by Schwaber and Sutherland, SCRUM pillars are as simple as a few pages of text, and the remaining is adaptation. This so called adaptation created space for dozens of books, thousands of blogs, training events worldwide, certification programs and millions on consulting fees.

Maslow said, if all you have is a hammer, everything looks like a nail. For a correct, responsible and beneficial adaptation of models and frameworks, we really need to understand all of them to some degree, only then can we list all possibilities of adaptation and even merge practices from different frameworks. Remember that SCRUM has some practices from XP, so this experiment was already done.

The takeaway is, before choosing KANBAN or SCRUM, invest some time on knowing both to some extent, only then can we chose the framework to use, or better yet, choose only parts of each framework.

3.4.3.2 Scope-wise Application

What should we use and when should we use it? Below we aim to list some of the characteristics that should be considered when choosing one of these models.

The following Venn diagram frames some aspects to keep in mind. It crossed two axes:

- **Scope Focused:** when we have the possibility of totally focusing a Team to the project scope, without any external interference
- **Interrupted and divergent needs:** the opposite of the above. The Team is constantly being interrupted or has several needs of different projects/products
- **Repeatable or specialist activities:** which are repeatable (like manufacturing process steps), or have to be done by one particular Team member.
- **Exploratory or Innovation activities:** never seen necessities, requiring Team brainstorming, debate, imagination and resourcefulness

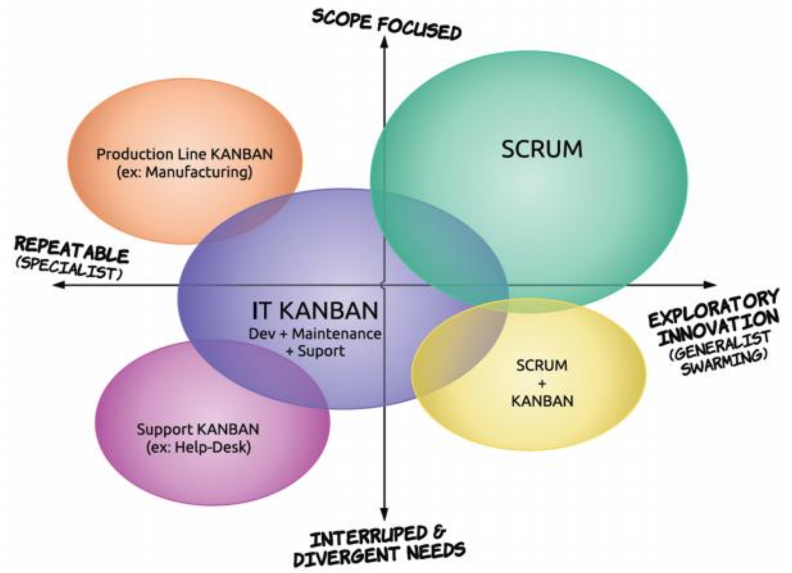


Figure 41 - SCRUM and KANBAN applicability [Sahota2010]

As can be seen above, SCRUM is more applicable when collaborative team work is needed to create something never created before, especially when the Team can be fully allocated to that task without interruptions. If we have a new project and can fully allocate a Team, SCRUM will be the way to go. Create a war-room and buy a drawing board just in case.

History has shown us that KANBAN is excellent to process step manufacturing environments, with always repeatable steps and step-assigned Team elements. This is not a very collaborative and cross-functional Team, but it will be a productive one.

Another good example of KANBAN application is one of a Help-Desk, always flooded with new requests from different areas, all with constant reprioritizations. This is more Team collaborative then the above, but not lacks in cross-function also.

If we take a normal development team, one that creates new products, but also needs to respond to emergencies, urgent code bugs and priority changes, KANBAN will be a good choice. With a model similar to the one described in sub-chapter 3.3, development activities can be done without time-box iterations in order to deal with unexpected events like unplanned maintenance and other support issues [Sahota2010].

Another alternative can be a merge of both KANBAN and SCRUM. This is particularly useful if doing exploratory innovation while struggling with the necessity to manage interruptions and/or divergent needs. This merge is known as SCRUMBAN and it's a new trend without a clear definition, basically it's a mixture of best practices from both models that should be adapted for each the specific necessity. The result is a model that can be more SCRUM-like or KANBAN-like depending on what we choose to implement. Due to being a novelty with no clear definition, almost a consultant mother lode, SCRUMBAN will not be focused in this document.

So, if we think in simple terms, the takeaways are [KniSka2010, Ramrak2011, SGuide2013, Sahota2010]:

Use KANBAN if:

- you have many and totally unpredictable interrupts which make planning difficult
- Team is too big (more than 7) or have specialized roles with divergent skill sets
- Have requests from different areas with constant reprioritizations, external dependencies that may stall activities or different levels of customer feedback/validation
- Activities are generally repeatable, like maintenance, support or service-desk

Use SCRUM if:

- Creating new product which needs focused brainstorming collaboration
- Can create a safe haven and avoid any external influence to the Team
- The Team is rather small, between 5 and 7 elements
- Scope is totally Team dependent, with almost no external dependencies that may jeopardize the Sprint Planning

3.5 Conclusions

Unfortunately, this exercise is not without flaws, so discretion is advised when it comes to choosing KANBAN or SCRUM. One may say that choosing is just the first step, some warnings should also be referred when it comes to applying both models. Again, knowing both models in some depth is crucial not only for choosing but for a correct implementation of them in real world teams.

SCRUM can be easily corrupted to a Micro-Management approach, with the Scrum Master having a team leader role and not allowing Team self-organization. Estimations activities may become overkilling, perverted for a too much low-level approach, in itself a remnant from Micro-Management control. A truthful Project Owner role and customer commitment are difficult challenges, choreography meetings are sometimes misunderstood or skipped… the list is endless, and as referred before is known as SCRUM-But.

Duo to its simplicity, KANBAN can easily became a tracking system ran by rush and without self-organization or continuous flow improvement. Teams tend to over-generalize the already simple concepts, ending up with Tickets on the wall, no clear workflow, acceptance criteria or rules.

Models application and pitfalls are not the objective of this analysis, since this was is just a reminder that lessons learned exist and are of great importance.

4 Mapping PMBOK® Processes with Agile Practices

4.1 Introduction

In the previous chapters we took the time to analyse in-depth both SCRUM and KANBAN, their roles, artefacts, board approaches and also made some remarks on their real world applicability. Clearly many benefits can come from the application of both models, but taking in account the PMBOK® 10 Knowledge Areas it's clear that these cannot be seen as Project Management frameworks.

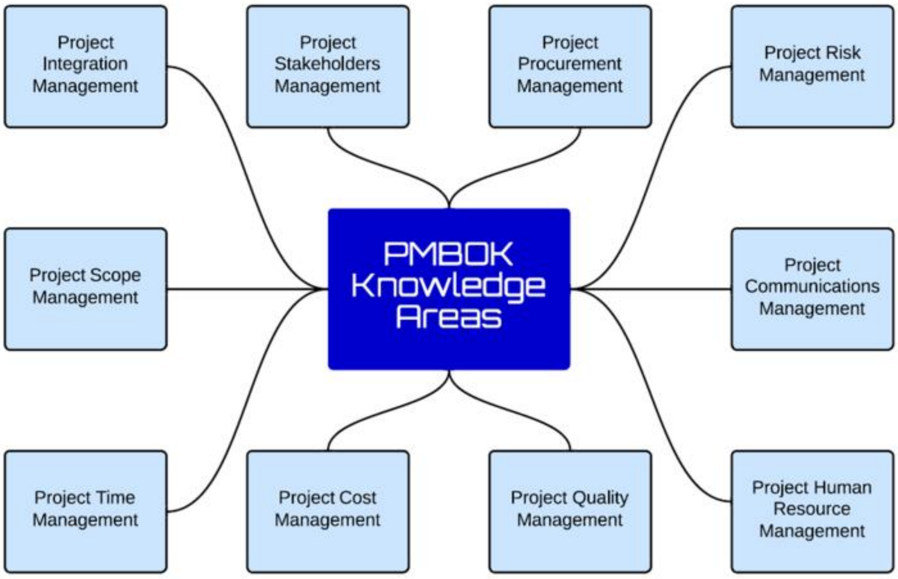

Project Integration Management
Project Stakeholders Management
Project Procurement Management
Project Risk Management
Project Scope Management
PMBOK Knowledge Areas
Project Communications Management
Project Time Management
Project Cost Management
Project Quality Management
Project Human Resource Management

Figure 42 - PMBOK® Knowledge Areas [PMBOK®2013]

Both KANBAN and SCRUM address some practices related to processes of the above Knowledge Areas, but clearly some are left out. Scope, Time and Human Resource Management are addressed by both in some degree, but Cost, Procurement and Risk Management may be set aside and regarded as not relevant regardless of importance in the Project Management discipline.

Some studies have been made to map Agile frameworks on PMBOK processes, done especially for SCRUM. But those studies, almost always, were made to prove that SCRUM can be an alternative to PMBOK Processes, presenting a vision of PMBOK vs. SCRUM of sorts.

We agree that frameworks like KANBAN and SCRUM are process tools and not Project Management processes by themselves, making them the perfect tool to manage activities at Development Team level. But how can we address the remaining Project Management activities?

PMBOK® is the most commonly used framework to address Project Management activities in organizations world-wide. Even organizations with no defined Project Management processes or PMOs use the framework as a basis to systematize processes and/or procedures related to Project Management. As PMI® claims, every project and performing organization is different in its own way, making it crucial to best tailor the process and procedure necessities of each case. This does not mean that every 47 processes should be used in a given project, or that a PMBOK® based Project Management methodology for a given performing organization should take in account every aspect of the 47 processes. Tailoring[11] activities are imperative for any project and/or performing organization, with value analysis done pragmatically.

That said, the concept for the remaining exercise is:

- Agile frameworks, like SCRUM and KANBAN, should be used to create procedures to ensure agility at Team level (Micro-Management)
- The PMBOK® framework should be used to create Project Management processes that ensure overall project control at higher level (Macro-Management)

Good Tailoring is imperative, on both levels, for each project or performing organization defined processes. One should not implement a Procurement process if acquisitions are non-existent, or persist in agile practices that cannot be implemented in company culture. Processes and procedures should be tailored targeting project or performing organization's real needs.

The current chapter will deliver Tailoring Diagrams and a Tailoring Matrix between SCRUM, KANBAN and PMBOK® to aid the difficult activity of Tailoring new Project Management process, procedures and practices for projects and/or performing organizations.

4.2 Macro and Micro-management approach

As stated before, the concept behind this exercise is the existence of two different Project Management layers: Macro-management and Micro-management.

Micro-management is the lower management layer, the one closest to the Development Team and where Agile frameworks application can create optimum benefit and value to the project, Team and/or performing organization. This is where SCRUM and KANBAN best practices will ensure an Agile-oriented management at Software Development Team level. The key is maintaining a self-organized and cross-functional Team with focus on participation and collaboration on the project objectives at that level.

The Macro-management layer will, in turn, be addressed by the remaining non-Agile areas that cannot be ignored as project constrains: Risk, Procurement, Integration, etc. The focus is the

[11] Tailoring - the act of making, altering, or adapting something for a particular end [CMMIDEV2010]

PMBOK® Knowledge Areas, since they represent traditional Project Management areas that cannot be left out. The key is maintaining a bigger overall-managed project which assures that non-agility areas are not left out and are managed accordantly.

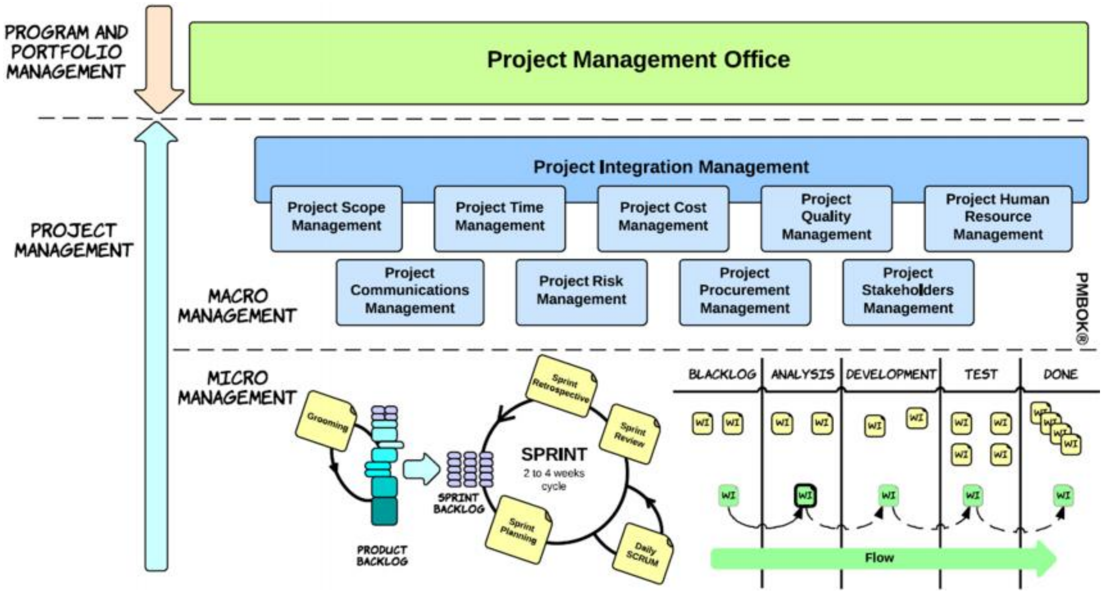


Figure 43 - Macro/Micro-Management approach

As referred in sub-chapter 1.3, we want to trace alignment between these two management approaches and find contact points between SCRUM/KANBAN (Micro-Management oriented) and PMBOK® processes (Macro-Management oriented).

Unfortunately, SCRUM, KANBAN and PMBOK® are very different definition-wise. PMBOK® is a complete, organized, well defined and appropriated detailed Project Management Standard. On the other hand, SCRUM and KANBAN are empirical process approaches based on sparse definition, with general concepts but with dubious detail, making each implementation different than the next. They are not defined as a Standard.
This conceptual difference makes it difficult to find an accurate way to organize an alignment. Because of this, a one-way mapping had to be found.

Our approach will be based on the more organized PMBOK® process definitions, with its 47 processes grouped in 10 Knowledge Areas, which will receive the inputs from the low-level KANBAN and SCRUM executions. These inputs are based on the detail laid down for both frameworks on chapter 3.

Essentially, for each of the 47 PMBOK® Macro-Management processes, we will identify the inputs it can receive from both SCRUM and KANBAN Micro-Management. The inputs for each process will be mapped graphically and explained individually in some detail.

4.3 Input Tailoring Diagrams

4.3.1 Project Scope Management

Scope is one of the key aspects in any project. Project Scope Management comprises 6 processes in order to guarantee that all requirements are identified, and work is performed to achieve those requirements and not others [PMBOK®2013].

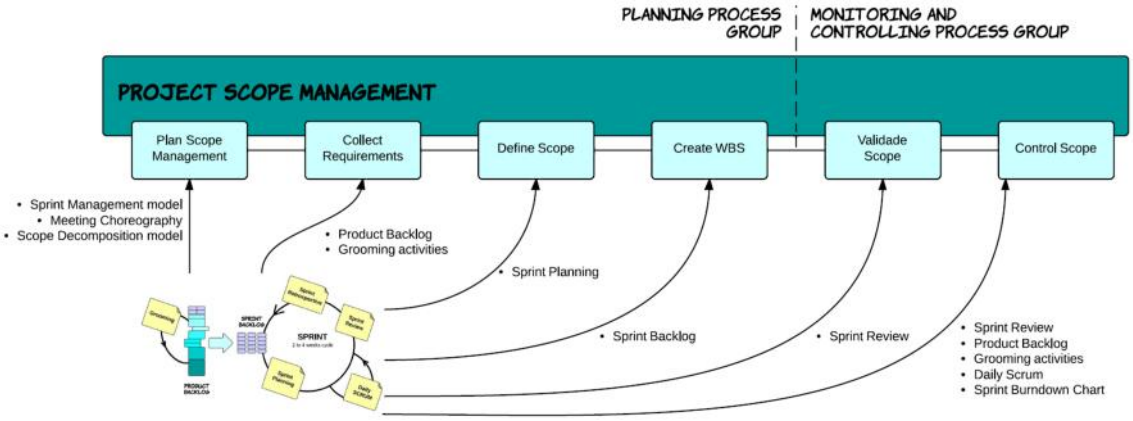


Figure 44 - SCRUM inputs to Scope Management processes

SCRUM has an incremental and iterative Life-cycle. Here, Scope is progressively refined through the course of Sprints. This might be seen as different from the traditional first-define-all-requirements approach and not easy to crossbreed, but we beg to differ. Scope decomposition is a SCRUM concept, and it is normally composed of Epics, User Stories and other smaller units of work. It's normal to use transversal grouping units, like Releases, to manage a set of work unit as a single bigger one. This decomposition ensures activities, choreography meetings, metrics and artefacts that can be used as inputs to the Scope Management processes. All choreography meetings should be documented as needed.

Plan Scope Management defines the way Project and Product Scope will be managed in the course of the project, therefore it needs to know how Sprints will be managed, which meetings will be taking place during the Sprint and how Scope will be decomposed in the course of the Sprint chain. SCRUM tends to underrate Project Scope, focused primarily in Product Scope, all SCRUM activities as meetings and choreography should be also considered as Project Scope. The inputs above will be necessary to define how both project and product scope will be managed in the incremental and iterative vision, with the following processes being executed when needed and more than once.

Due to the recurrence of the following processes, higher scope units like Epics should be used for maintaining a macro-scope management, focusing the Scope Management processes in that higher hierarchy of requirements, aligning with SCRUM to manage the lower end of the User Stories and Tasks. Epics are normally identified and discussed in the beginning of the project, making it the ideal unit to manage at a macro level since day one. Additional Epics should be added accordantly.

Collect requirements will receive all the information from the Grooming activities and the Product Backlog items. Those grooming activities are key to identifying all project requirements and clarifying units that will not be considered in the project scope. Because Grooming activities will be iterative, new information will be received from time to time, executing the Collect Requirement process again if needed (ex: adding new Epics). The Team will constantly use the Product Backlog to track requirements and their future changes.

Define Scope and Create a WBS will also be recurrent processes, always aligned with the Sprint Planning of each Sprint. In that meeting the Team will define scope for the next Sprint and clearly state it in the Sprint Backlog. This event marks the Team commitment. Work-Breakdown-Structure (WBS), an important artefact, will receive the inputs from the Sprint Backlog, which should be organised as a WBS itself. The project WBS will be progressively refined at lower level, but the higher levels should be identified as soon as possible and managed accordantly.

We can also use the Product Backlog as an input to the WBS, but some caution is needed. The Product Backlog may contain some requirements that will never be implemented, therefore it is very important to add to the WBS only previously agreed requirements.

Control Scope will constantly receive information from the several activities, artefacts and meetings from the SCRUM lifecycle. These activities are SCRUM's golden rule. Sprint Reviews may be used to manage requirement acceptance, but together with Grooming meetings they light out requirement changes and also new requirements. These changes should be managed accordantly and addressed in future sprints. Daily Scrum and the Sprint Burndown charts will return information for constant management of performance and will allow to actively report produced scope at higher levels.

The SCRUM Micro-Management will also act as a constant natural measure against Scope Creep and Gold Plating[12].

Validate Scope will receive information from the Sprint Review meeting at the end of every Sprint. The key concept is that we will validate a set of scope in the end of the each Sprint, always based on the acceptance criteria defined in the User Stories (Definition of Done). The meeting itself is an excellent evidence of Customer acceptance. In this meeting some requirement changes or defects may present themselves, and they should be accounted for and added to the Product Backlog and documented, if needed.

[12] Gold Plating - continue working on a work product well past the point where the extra effort is worth the value it adds (if any).

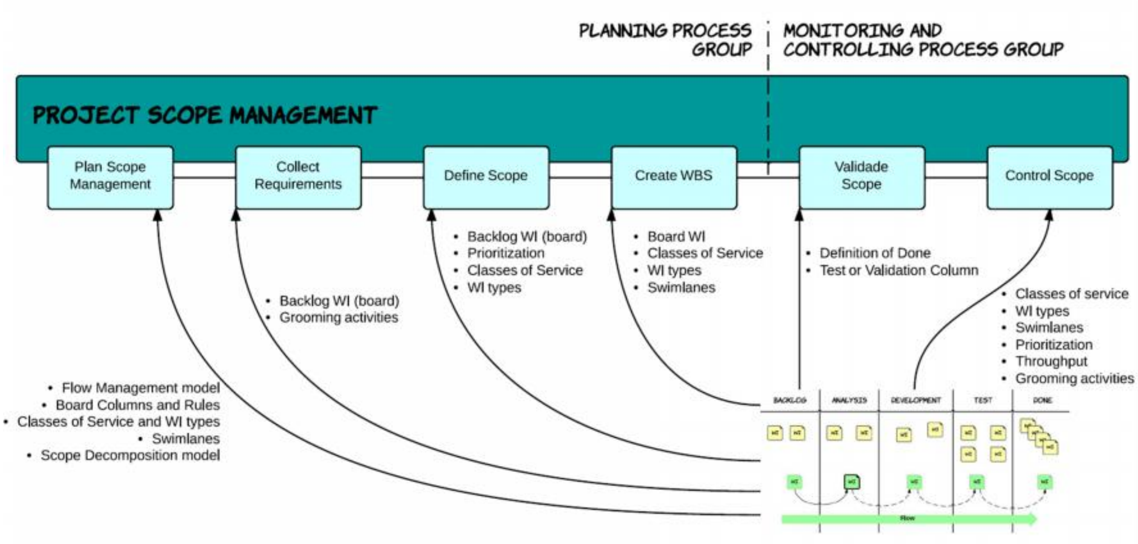


Figure 45 - KANBAN inputs to Scope Management processes

KANBAN, like SCRUM, has the same incremental life-cycle, Scope is progressively refined, however the Sprint iteration approach does not exist. KANBAN is based on flow, its optimization and response time, are largely different from a planned time-box iteration in which the Team commits to deliver a set of features. Because it's event-driven, the approach is even further from the traditional first-define-all-requirements approach than SCRUM. Creativity is advised.

Similar to SCRUM, Plan Scope Management processes will have to receive information on how KANBAN will be set and managed. KANBAN Workflow and Columns need to be defined as well as the KANBAN rules and the Definition of Done for each column. The decomposition model should also be known in order to clarify the decomposition unit that will be used in next processes.

Classes of Service and its rules should also be defined as they will have a big role in managing scope later on. One of the key aspects of KANBAN is the continuous improvement of KANBAN flow and other aspects. Plan Scope Management will need to be reviewed if changes prove to be necessary.

On a similar position to SCRUM, KANBAN Workflow will manage in a Micro-Management way, making it critical to define ways to distinguish scope from non-scope in event-based models, and link it with the higher level requirements. Align Classes of Service, Work Unit types and Swim lanes with the customer before Collect Requirement may help to create a link between higher and lower level requirements.

KANBAN may not have a Grooming meeting, but being an event-based approach, Grooming activities are made constantly to collect or refine requirements. These are one of the key inputs to Collect Requirements process, which will culminate with the Backlog on the board.

Define Scope activity is also based on flow and not on time-box as SCRUM, therefore it needs to be based on already accepted Work Items on the board and Work Item types defined with the customer. Prioritization plays a crucial part in order to make sure the Team is focused on the most valued Work Items. Again, with no time-box concept, the commitment needs to be based on the customer aligned Classes of Service and their rules. Work Unit types and Classes of Service will avoid Scope Creep by confining the Work Items on board with agreed scope types. Those Work Unit types, Classes of Service and Swimlanes may be used to construct and organize input to the global WBS in the Create WBS process.

Because KANBAN has no Sprint Review, it's common to create a column for User Acceptance Testing or Validation. The name may vary but the objective is to have a customer validation on every Work Item passing by that column. Therefore the Validate Scope process needs to receive that information and register it for scope validation purposes. Other requirements may be found when performing these activities, from requirement modification to new requirements, all of which need to be accountable and placed on the KANBAN Board accordantly.

Regarding Control Scope, KANBAN is all about controlling the flow of Work Units entering the board, passing by the column workflow until "Done". Throughput will be monitored constantly and is a key input for the process. Prioritization will dictate order and be crucial when coming to requirement changes and its urgency. Grooming activities are constant and will bring new Work Items that need to be added to the board. Another key aspect is the use of Classes of Service, Work Item types and Swimlanes to manage flow, scope changes and Throughput.

4.3.2 Project Time Management

From milestones to phases, a project is bound by Time is several ways. It's the most remembered constrain in Project Management, time scheduling is implemented in some degree even in the simplest project. Project Time Management comprises 7 processes in order to guarantee that project scheduling is planned and controlled as needed [PMBOK®2013].

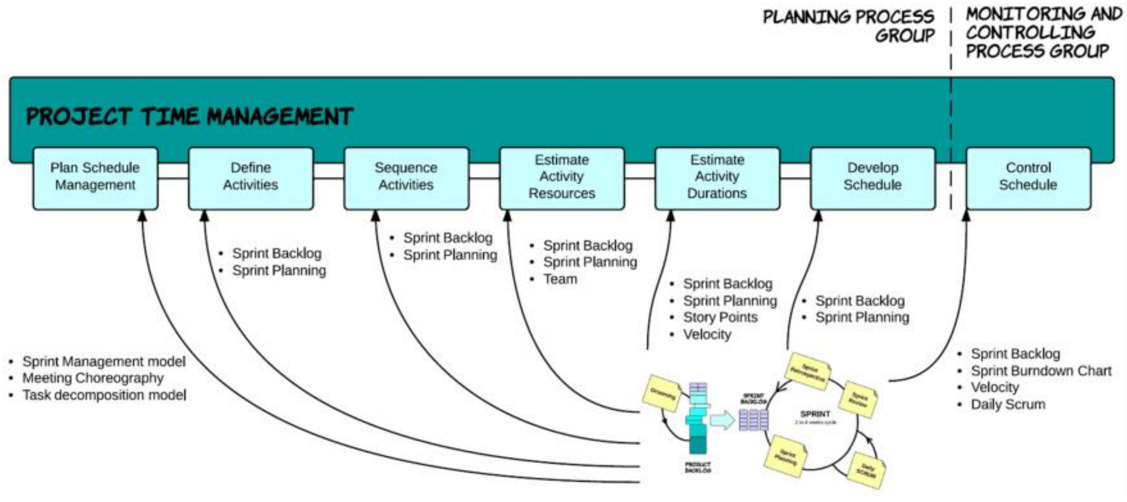


Figure 46 - SCRUM inputs to Time Management processes

Time Management is about identification, estimation, plan execution and control execution of the needed activities to complete the project requirements. Unlike a waterfall approach, SCRUM has an incremental and iterative Sprint life cycle. This repeatable time-box style means that common Time Management activities are repeated in every Sprint regarding its defined scope.

This iterative management means that Sprint activities regarding estimations, choreography meetings, Sprint Velocity, and other Time related matters should be input to Time Management processes. Due to the importance of some choreography meetings regarding the Time subject, they should be documented as needed.

As said before, SCRUM is also a Just-In-Time scheduling system focused on customer value, therefore the Team should only invest time in scheduling when they are sure of such a value, i.e. in the Sprint Planning activities. Because on this, inputs for the Time Management processes, except initial Planning, are mostly generated beyond the Sprint Planning.

Plan Schedule Management establishes the plan for how Time will be managed throughout the project. Therefore, information regarding the meeting choreography, Sprint size, Sprint velocity and Burndown chart management and other Time related information needed for planning the overall Project Scheduling will be input to this process.

As said before, SCRUM tends to underrate project activities that are not related to producing product. It is common to have excellent time plan detail on activities that implement User Stories, but SCRUM meetings and other choreography activities are not always regarded as relevant time-wise. It’s very important that Plan Schedule Management receives inputs for all needed Time related activities in order to better understand the SCRUM Time related necessities in the overall project planning and how will they be managed during execution.

Sprint length, Story Points and Sprint Velocity usage have a major role in Time Planning. Defining their usage, metrics and control will lay the ground to how the Sprints will manage Time and which inputs will be used in the other Time processes. This is a crucial part of Time management at a Team Micro-Management level that should be planned to make sure an understanding is created and followed in the other processes. The Sprint iterative approach obligatorily executes some of the next processes in each Sprint.

Define Activities will receive information from the Sprint Backlog content and the Planning Sprint meeting. The Sprint Planning is the meeting where the Team will decide which Product Backlog items will be implemented in the current Sprint and details the activities to implement them, making this information crucial as input to this process. The Sprint Backlog will contain not only the identification of the Scope being implemented on that Sprint but also all the detail activities that need to be performed to implement such Scope. These decomposed activities, normally called Tasks, are the technical work needed to implement each User Story.

Sequence Activities, similar to Define Activities, will basically receive input from the Sprint Backlog content and the Planning Sprint meeting. Planning Sprint will not only define the activities for that Sprint, but also expose the sequence in which they need to be executed based on their dependencies.
Although User Stories should be independent from each other, these decomposed activities may have dependencies (technical, resources, etc), making a plan sequence an asset for the Sprint execution.

Estimate Activity Resources will receive inputs from Sprint Planning meeting, especially regarding Team elements and their allocation to the Sprint. During the Sprint Planning the Team will review the activities in the Sprint Backlog, how they can be executed with the resources available and identify time constrains, if any. With the Team being cross-functional and normally fully allocated to the Sprint, human resources dependencies are fairly reduced. All non-human resources should also be discussed on the Sprint Planning and will be an input to this process.

Estimate Activity Durations will receive all the information regarding estimations used for planning the Sprint. Like the last 3 processes, this will also be a recurrent process aligned with the Sprint Planning of each Sprint.
SCRUM advocates the usage of Story Points as estimation technique, but Story Points should not be a representative of Time estimation - this is one of the most common SCRUM misinterpretations. But with Sprint length being a fix time-box, we can use this to calculate relative activity duration based on Sprint Length, Sprint Velocity and the Story Points estimated for that Sprint. Even not being a direct Time measure, this can be used as an input for this process.
Although not a rule, it's not uncommon for the Team to estimate effort for the User Stories decomposed activities. This normally takes place during Sprint Planning and serves as a ratification of the Sprint feasibility regarding the resources and Team allocation. If done, these measurements can be also an important input for this process.

Develop Schedule will also need to be a recurrent process because it will receive input from each Sprint Planning. The Sprint Planning meetings outputs will progressively serve as input for the overall project Schedule. The Sprint Backlog will have the details of the activities to be done in each Sprint, adding this information to the overall project Schedule will be crucial in maintaining the overall project Schedule vision and controlling the project Schedule at Macro-Management level while SCRUM micro-manages Schedule aligned with each Sprint.

Control Schedule will assess the Scheduling execution, therefore will need every information that results of managing the Sprint execution.
As said before, SCRUM's ability to manage Scheduling is based on the ability to use the Sprint time-box interactions and manage it accordantly to what the Team commitment is. The Team

uses the Daily Scrum to keep track of the of this commitment, they constantly review the Sprint Burndown Chart as a performance forecast to check if all the activities can be executed by the end of the Sprint. Any deviation or Sprint change should be also input for this process to make sure a Change Request is created if needed.

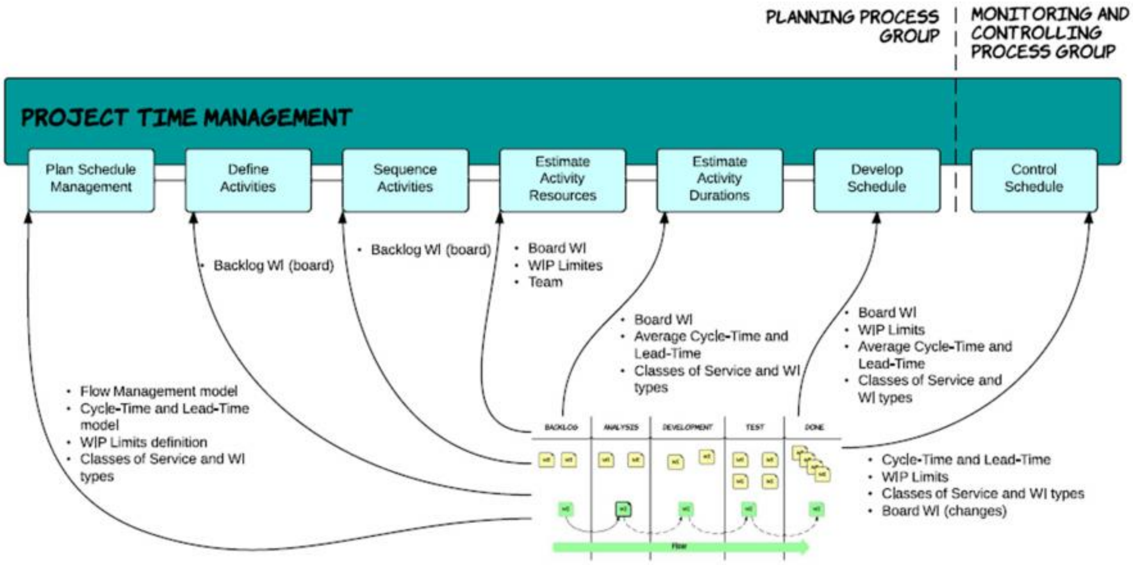


Figure 47 - KANBAN inputs to Time Management processes

Unlike SCRUM, KANBAN does not have an iteration time-box approach based on estimations and velocity to limit WIP, instead it's focused on improving flow and response time and limiting WIP in the Board columns. Although Lead-time and Cycle-time are time duration measurements, it does not have a define estimation practice for the Work Items in the Board. Estimation techniques are not unheard in KANBAN, nevertheless we are assuming that there are none for the course of this exercise.

Similar to SCRUM, Plan Time Management process will have to receive information on how the KANBAN will be set and how will manage Time during the activity Workflow.
KANBAN Workflow and Columns WIP Limits need to be defined as well as the usage and management for Cycle-Time, Lead-Time and Throughput.
Time measurements like Cycle-Time and Lead-Time can be used in together with Classes of Service to come up with performance indicators that can express how time control will be done at a Micro-Management level. The Macro-Management level will then use this information to report performing and Scheduling fulfilment.

Because KANBAN has no Sprint Planning like SCRUM, Define Activities will receive information based on the Backlog on the KANBAN Board. Like SCRUM, KANBAN is a Just-in-Time system, so inputs will only be available when Customer value is recognised, i.e when the Work Items exist on the Board due to Customer prioritization.

Sequence Activities will receive some input regarding the Work Items on the Backlog on the Board, but because of the low decomposition of the Work Items, the dependency information will mostly be valued by the Team at a Micro-Management level.

Estimate Activity Resources will receive not only the Team elements but also the resources linked to certain Board Column, this normally take forms as a WIP limit.
WIP limits can be used to limit capacity for both human and not-human resources. Column WIP limits can be set in a certain set for limiting capacity because we have limited man power for such a step, or because a needed non-human resource has a limited capacity. One common example of this usage is limiting a test stress step (column) that can only handle one Work Item at once.

Again, KANBAN does not define techniques for estimation, but Estimate Activity Durations can receive input derived from Cycle-Time and Lead-Time together with Class-of-Service and Work Item type.
Historical average Cycle-Time and Lead-Time, crossed with Class-of-Service and Work Item type will return valued information for average activity durations. This information will become more and more valuable over time, and it will converge to a regular interval that can be used for forecasting by the Macro-Management level.

Similar to SCRUM, Develop Schedule will also need to be a recurrent process because it will receive constant input from the Board and its Work Items. These items will progressively serve as input for the overall project Schedule.
WIP limits and Classes-of-Service should be used to bind the limitations of the KANBAN Workflow columns and priority of the Work Items, together with an average Cycle-Time and Lead-Time as a valued information for Scheduling the project Macro-Management. Meanwhile, Micro-Management Scheduling is done by the Team on Board level.

Control Schedule will receive information on all controls available, those being based not so much in Scheduling itself but much more in managing Cycle-Time and Lead-Time for all Classes of Service and Work Item types.
The ability to manage scheduling is based on the ability to manage Cycle-Time and Lead-Time of each Work Item in the Board. By doing so, the Team will be managing the Time Schedule even if by a different approach than a regular one, based on managing deviation from the average Cycle-Time and Lead-Time.
By also managing the WIP limits, flow and bottle neck are also accountable, making this also an insurance measurement against average Cycle-Time and Lead-Time deviation, therefore Schedule deviation.
Because KANBAN is event-driven based, all changes to prioritization should be accounted for on the Board, with all necessary actions taken. This is also an input can be helpful in identifying needed Change Requests.

4.3.3 Project Cost Management

Cost is a very important constrain in all projects, together with the already seen Time and Scope, they are normally revered as the triple constrain in Project Management. From planning to cost control, these processes are designed to accommodate all cost management necessities. The Project Cost Management comprises 4 processes [PMBOK®2013].

Unfortunately, budgeting, funding and other cost related subjects are commonly distant from development teams. Normally the development team is focused on time and effort estimations for all needed activities to develop the product. Cost management is about managing real money costs for all project requirements.

Managing the actual cost requirements is generally done at a Macro-Management level. The owners of such activities receive all information for budgeting the project, not only at development level, but all levels. This control is normally done using information from the Micro-Management level using metrics of work performance activities performed by the development team and other teams involved.

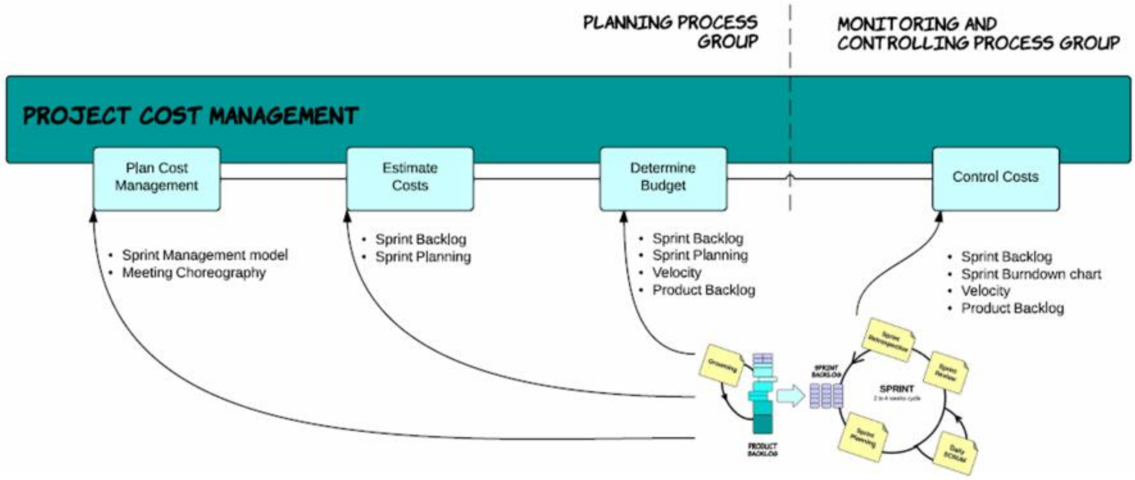


Figure 48 - SCRUM inputs to Cost Management processes

Because Plan Cost Management is the process responsible to plan all aspects of cost, it will need to receive information regarding the Sprint model and the meeting choreography.

Due to SCRUM's incremental cycles, all the Sprints and meeting choreography will need to be accounted cost-wise. This information will be needed to define how Sprints, effort to produce product, validation, among other work will be budgeted and controlled throughout the several Sprints. The Macro-Management process will receive these details and create mechanisms to calculate cost and progressively detail the overall project budget.

Depending on Sprint practices, different cost management approaches may be implemented. The most common is to account the Team-direct manpower costs in each Sprint, making Determine Budgeting a recurrent process in each Sprint. Unfortunately this will negatively affect the overall project budget due to the fact that detail budgeting would be directly linked to the Sprint Planning , which is only done in a Sprint-to-Sprint basis. This would mean that

overall project budget would never be completed during the project execution, nor even would it contain some degree of forecasting estimation regarding future Sprints.
To ease this lack of overall project budget vision, it's common to use the Story Point estimation and average Sprint Velocity to have a top-down account of work still not delivered in Sprints. This can be used as a top-down approach for forecasting overall project budget regarding future Sprints, which is later confirmed by the bottom-up approach from each Sprint.

Estimated Costs will receive information from the Sprint Planning meeting, which contains the Sprint Backlog, therefore all cost related necessities found when detailing Sprint activities. At this time, Sprint Backlog items should be sufficiently detailed to give accurate measurement to be used in cost estimation for the Sprint. This bottom-up approach relies on the effort needed to complete the Sprint commitments with the already identified Team elements.

Determine Budget will sum the overall project budget, and therefore will receive not only detail estimations of the Sprint Planning, but also the means to use that information as baseline to extrapolate parametrically future Sprint estimations based on the known Product Backlog and therefore arriving at an estimation of the overall project budget.
Like Estimate Costs, this process should be revised in each Sprint Planning to make sure parametric estimations factors are still valid or need to be adjusted. This top-down approach will maintain some degree of accuracy in the estimating of how may Sprints will be necessary to implement the Product Backlog Items, used to construct the overall project budget.

Control Costs will manage deviation from the overall project budget, which consequently will need the information from the Sprint execution and commitment fulfilments.
Like other processes, the Team will micro-manage cost deviations when managing Sprint execution. By sending this information to the main process, the Macro-Management level will be able to manage deviation of the overall project budget, both present and forecasted. The Sprint Burndown Chart will give accurate detail of Sprint execution and will identify if any Sprint deviated. If deviations occur, an analysis is mandatory to make sure the parametric extrapolation remains valid for the overall project budget.
Because changes identified during the Sprint or Sprint Review will be addressed in future Sprints, those should be analysed to identify needed Change Requests.

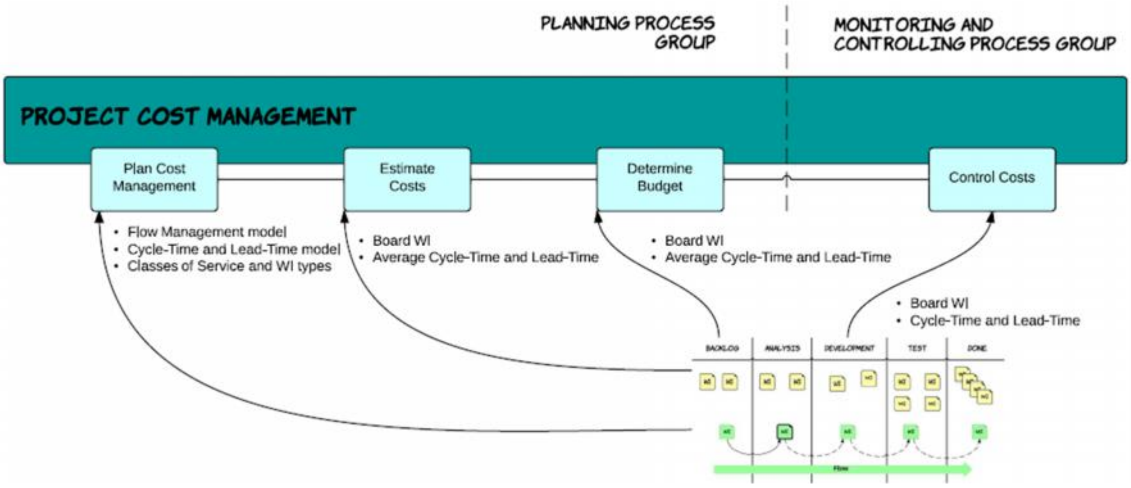

PLANNING PROCESS GROUP
MONITORING AND CONTROLLING PROCESS GROUP
PROJECT COST MANAGEMENT
Plan Cost Management
Estimate Costs
Determine Budget
Control Costs
• Flow Management model
• Cycle-Time and Lead-Time model
• Classes of Service and WI types
• Board WI
• Average Cycle-Time and Lead-Time
• Board WI
• Average Cycle-Time and Lead-Time
• Board WI
• Cycle-Time and Lead-Time
BACKLOG
ANALYSIS
DEVELOPMENT
TEST
DONE
Flow

Figure 49 - KANBAN inputs to Cost Management processes

As said earlier, KANBAN does not have a systematic estimation approach, making the Cost Management somewhat hard to accomplish. Like SCRUM, the Team controls work performance activities, making those metrics the only asset to be used in cost estimation, forecast or control.

Because Plan Cost Management will outline how Cost will be planed and controlled, it will need to receive all the needed information to how the KANBAN Board will be assembled, the rules for managing it and the types of units managed by it.

Additionally to the KANBAN Workflow model, the Cycle-Time and Lead-Time is the closest factor to estimate, control and forecast cost. Because Team and resources cost for the KANBAN Workflow are known, we can use average Cycle-Time and Lead-Time as a cost indicator based on Throughput history. This indicator can be differentiated by using categorization for Classes of Service and Work Items types. Like SCRUM, the Team will micro-manage cost by managing deviation from average Cycle-Time and Lead-Time, and provide necessary data to ensure some degree of forecasting to macro-manage the overall project cost.

Estimate Costs and Determine Costs will both receive inputs of Work Items on the Board, and by using the average Cycle-Time, Lead-Time and Throughput, a cost for each unit can be estimated.

Cost estimation will be more accurate if Work Items on the Board can be differentiable between themselves. By using differentiable Classes of Service and Work Item types and crossing them with independent average Cycle-Time and Lead-Time for each one, we can estimate the cost of each based on history. Swimlanes and other Work Items differentiation may also be used as described, and while adaptation is endless, it will be always as good as the average history data. Accuracy will improve over time in this bottom-up derived estimation.

In the top-down perspective, forecasting the overall project cost is limited by the event-driven nature of KANBAN: we can only forecast based on the Work Items that exist on the Board.

Some creative forecasts can be calculated if the average of incoming requests is historically reliable.

Control Costs will receive real-time information of the KANBAN Workflow execution in order to manage deviation from the overall project budget.
By actively controlling Cycle-Time, Lead-Time and Throughput at all times, the Team will micro-manage cost deviations for each Work Item in the board.
On a Macro-Management perspective, history of average measurement will be passed to the Control Costs process to ensure that cost deviations does not occur at the overall project budget. Cost indicators should be revised frequently to make sure they remain valid for forecasting.

4.3.4 Project Quality Management

Quality is vital to both project and product being produced, therefore planning and managing it is of crucial importance. Project Quality Management is comprised of 3 processes to assure that quality requirements are planned, executed and controlled [PMBOK®2013].

Quality is conformance to requirements, those being product or project ones. Both SCRUM and KANBAN, with their iterative and incremental approach, have almost naturally embedded some quality assurance practices. Frequent deliveries, fast change feedback, customer collaboration and frequent reviews are some of the Agile Quality related practices defined in both models. Because this is micro-managed by the team, we will map how these practices can result as an input to the Macro-Management processes.

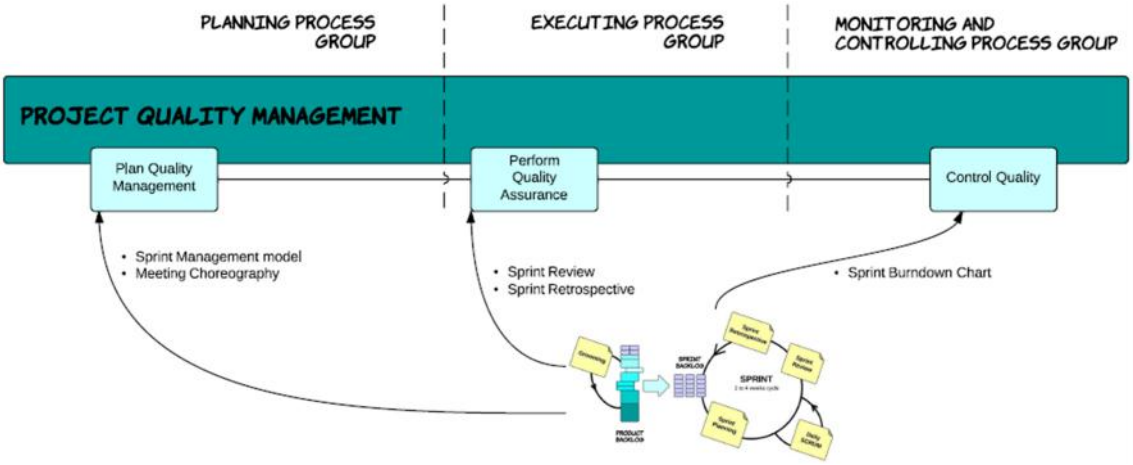


Figure 50 - SCRUM inputs to Quality Management processes

SCRUM has specific practices regarding product and project quality requirements. From the Definition of Done to Sprint Reviews for Validation or Sprint Retrospectives for continuous improvement, we can see practices for assuring both product and project quality throughout the SCRUM lifecycle.

Plan Quality Management will need to receive all the information needed to define how Quality will be planed, executed and controlled. The Sprint Management model will contain not only how Sprints will be executed, but also how concepts like Definition of Done are used with all the ceremonies in the Meeting Choreography. Together, those techniques and meetings lay the ground for how quality activities will be planed, executed and controlled.
Product quality requirements are linked to the effort invested in customer collaboration in decomposing all the needed requirements. INVEST User Stories are key.
Project quality requirements are assured by the iterative and incremental Sprint approach, which enforces sort development cycles with validation and quick feedback. Stakeholder communication and customer commitment are key aspects.

Performance Quality Assurance will receive the inputs that attest to the execution of activities aligned with the Quality assurance practices of the SCRUM cycle.
The Sprint Review at the end of the Sprint is basically a validation by the customer of the developed features during the Sprint. The meeting will attest compliance to requirements at a product level, and will also provide feedback to possible adaptations needed (Change Request) or identify additional work and add it to the Product Backlog, therefore lessening Scope Creep and Gold Plating impacts.
During the Sprint Retrospective the Team will analyse their own performance regarding the Sprint execution, i.e. analysing the project requirements execution. This practice reflects the importance of continuous improvement during the Sprint cycle, and can result in quality assurance activities for future Sprints. This process should be revised at every Sprint.

In order to Control Quality, this process will need to receive the Sprint Burndown chart. This will provide real time information on the status of Sprint activities, and, as seen above, these activities assure quality to both Product and Project. Adjustments should be addressed if deviation from planning occurs.

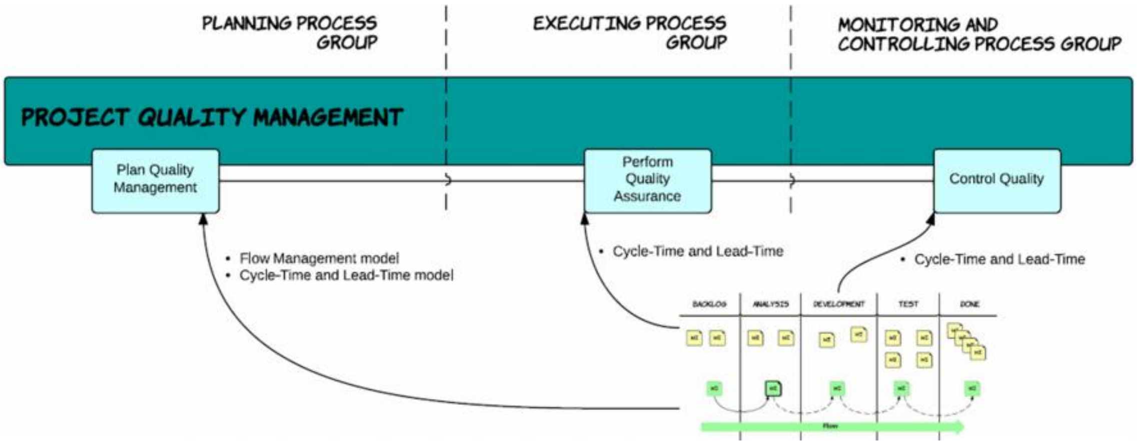

PLANNING PROCESS GROUP
EXECUTING PROCESS GROUP
MONITORING AND CONTROLLING PROCESS GROUP
PROJECT QUALITY MANAGEMENT
Plan Quality Management
Perform Quality Assurance
Control Quality
• Flow Management model
• Cycle-Time and Lead-Time model
• Cycle-Time and Lead-Time
• Cycle-Time and Lead-Time
BACKLOG
ANALYSIS
DEVELOPMENT
TEST
DONE
Flow

Figure 51 - KANBAN inputs to Quality Management processes

Contrasting SCRUM, the KANBAN does not have specific meetings with a quality agenda. Instead the quality is focused on improving Cycle-Time and Lead-Time, making continuous improvement a constantly done activity and not contained in a specific timeframe. If deviation or blocking points occur, the Team will immediately make the arrangements to find the root cause and eliminate the issue.

Quality assurance is flow driven, a manufacturing reminiscence. Each column on the board has a Definition of Done to ensure all Work Item complies with the definition or it will not advance in the Board.

But adaptations can also be done. Another common practice in KANBAN is the creation of a Test column. The name may change, but normally acts as a verification and/or validation step for all Work Items. Generally the customer is involved.

Plan Quality Management will need to receive all the information of the KANBAN Workflow model and how Cycle-time and Lead-time will be used during execution and control. Clear Definitions of Done will need to be defined to all Columns. If a Test column exists, the results of their execution will serve as evidence of quality assurance activities.

Both Performance Quality Assurance and Control Quality will receive in real time Cycle-Time and Lead-Time metrics for all Work Items on the board. Deviation analysis or blocking point elimination will be done by the Team continuously; therefore by controlling Cycle-Time and Lead-Time, you are assuring quality.

4.3.5 Project Human Resource Management

Regardless of project or related industry, people are the work-force for executing project activities and implementing requirements. Hence, planning how to identify, acquire and manage the Team is a universal necessity. For this, Project Human Resource Management comprises 4 processes [PMBOK®2013].

Both KANBAN and SCRUM rely on people to execute activities and manage Sprints or flow. Activities to identifying the needed skills, number of elements and their responsibilities will need to be addressed when managing project human resources.

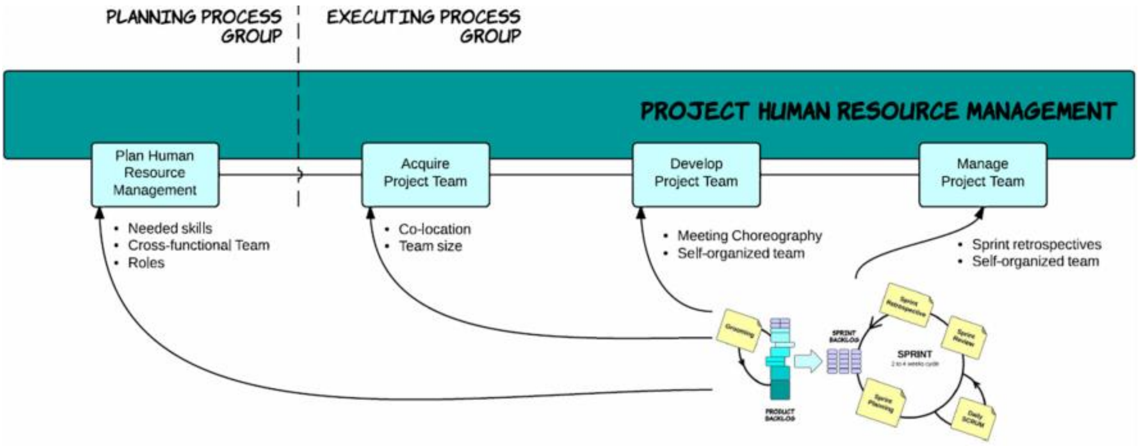


Figure 52 - SCRUM inputs to Human Resource Management processes

Plan Human Resource Management will plot how human resources will be dealt with in the overall project, thus it must receive information on specific necessities regarding the Team elements. SCRUM defines roles and responsibilities to Team members, these can serve as an input to the overall staffing plan. Skill-wise, keep in mind that SCRUM also fosters Team members cross-functional skills, making the skills identification a very important aspect.

Acquire Project Team will actively select Team members. Because SCRUM promotes a stable Team with a fixed size throughout the project, it's very important to identify how many elements the Team will need. Reminder: enforce the co-location of the team.

SCRUM defines that the Team should be self-organized, resulting in internal growing. The meeting choreography will also build the Team and strive individual improvement. The Develop Project Team process should receive these inputs.

Sprint Retrospectives are an excellent input for the Manage Project Team process. They will give valuable input on how the Team reviews itself and outlines a plan for improvement. Keep in mind that the self-organized Team concept makes for frequent adaptations at Team organization level. Those adaptations should be reported if the change affect plans already defined in the Plan Human Resource Management process.

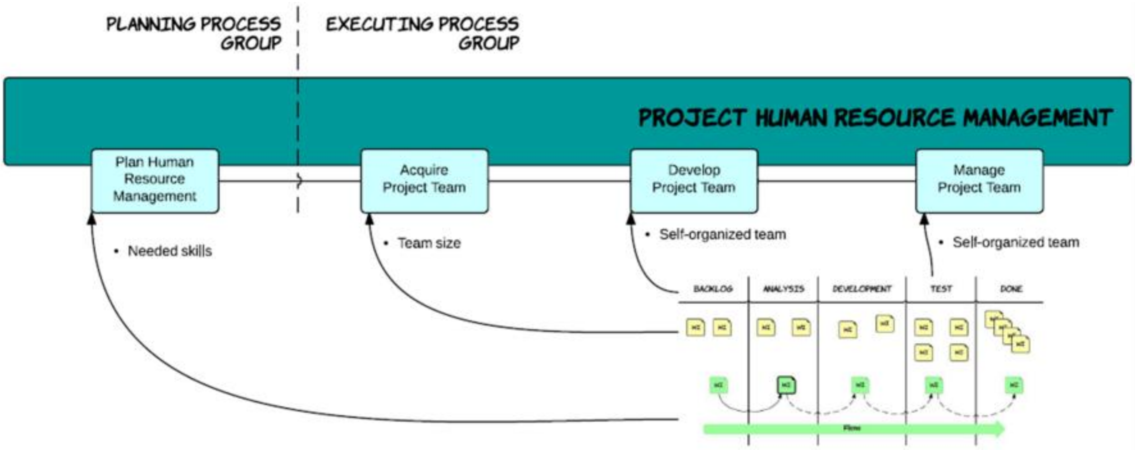


Figure 53 - KANBAN inputs to Human Resource Management processes

KANBAN on the other hand, does not have some of SCRUM's constrains regarding Team elements. Skill identification and Team size are necessary, but co-location, roles and others are not compulsory.

Because no Roles or Cross-functional aspects are mandatory, Plan Human Resource Management will receive only the needed skills for completing the necessary work.

Acquire Project Team will select Team members based on the skills defined in the Plan Human Resource Management process and estimated Team size.

Develop Project Team and Manage Project Team processes can count with the self-organization concept as an assurance of both Team development and management. Because continuous improvement is a key concept, self-organization will influence Team development as needed, the management aspect will be addressed by the always present improving KANBAN flow which also affects Team elements.

4.3.6 Project Communications Management

Project Management professionals not only agree that good communication is a critical project success factor, but also that communication related activities represent a large part of Project Management activities. In fact, these activities have such importance that in PMBOK 5th edition PMI® divided Project Communications Management and also created a separated Project Stakeholders Management Knowledge Area, allowing Project Communications Management to focus solely in communication management [PMBOK®2013].

Open, circular and valuable communication is one of Agile's foundations. Almost every Agile pillar is reliant or influenced by having or not a good internal and external communication. In this case, both KANBAN and SCRUM value direct customer communication to assure rapid feedback, making internal communication a must have in any Agile self-organized team.

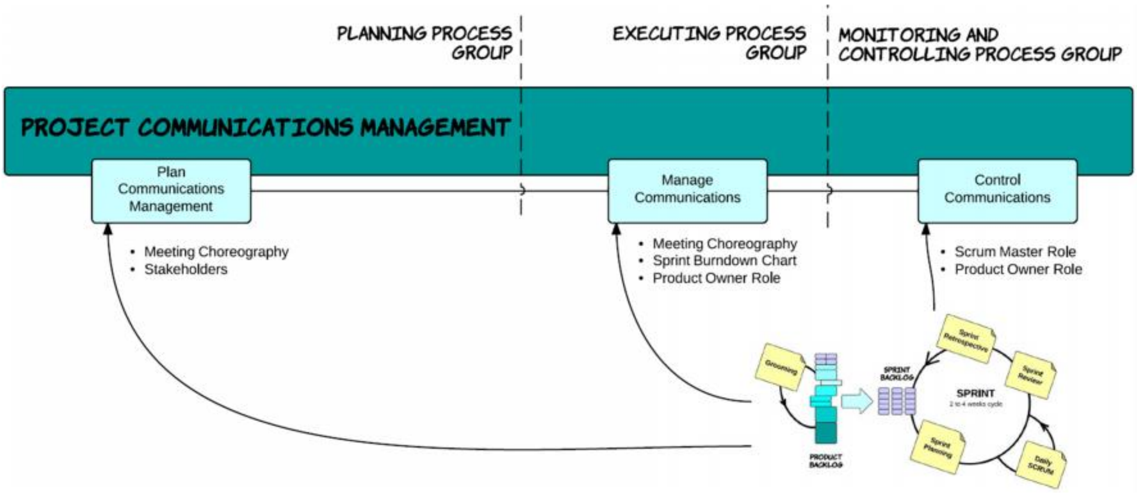


Figure 54 - SCRUM inputs to Communications Management processes

SCRUM framework addresses communication is a very enforced manner. By providing roles and meeting choreography, the Team and Customer are drawn to a circular communication ring. Apart from being a Customer representative and/or having privileged access to him, the Product Owner also bound a direct communication channel to all Team members and other external stakeholders. It's almost the external spokesperson that brings news, objectives and vision.

On the other stand, mandatory meeting choreography ensures that specific communication moments are known by all and exploited to the limit. Remember that the Scrum Master is the responsible for coaching the team, assuring such ceremonies really take place and bring value to the product or Team development.

Hence, the Plan Communication Management will receive not only the identified Stakeholders but also how the meeting choreography will channel communication to and by them during the project execution.

During execution, Manage Communication will benefit from inputs of all the Choreography meetings for internal and external communication. Sprint Planning, Sprint Review and Grooming will deal will external communication, making Daily Scrum and Sprint Retrospectives more focused in internal communication. Artefacts like the Sprint Burndown Chart are also tools the Product Owner can use to report performance.

Control Communications will be assured by both Scrum Master and Product Owner as defined in their responsibilities. The Product Owner will ensure the communication channels are working and the Scrum Master will assure that the Team follows the meeting choreography and SCRUM best practices.

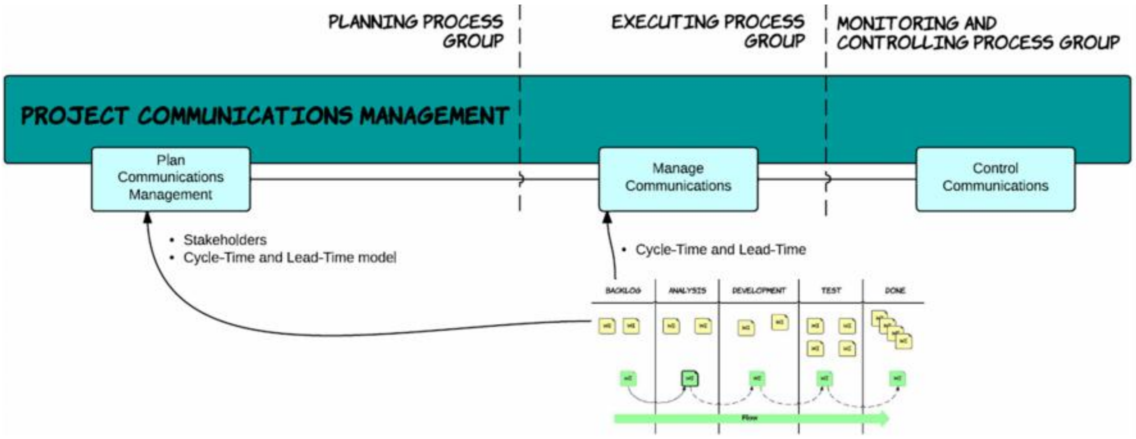

PLANNING PROCESS GROUP
EXECUTING PROCESS GROUP
MONITORING AND CONTROLLING PROCESS GROUP
PROJECT COMMUNICATIONS MANAGEMENT
Plan Communications Management
Manage Communications
Control Communications
• Stakeholders
• Cycle-Time and Lead-Time model
• Cycle-Time and Lead-Time
BACKLOG
ANALYSIS
DEVELOPMENT
TEST
DONE
Flow

Figure 55 - KANBAN inputs to Communications Management processes

On the other hand, KANBAN does not have a Meeting Choreography or customer representative, Product Owner-like. Yes, the model sponsors fluid communication, but it is mainly focused on internal communication to safeguard the self-organization aspect and the continuous improvement concept.

Plan Communications Management will only receive the Stakeholders list, relying on the team's self-organization pledge to exploit communication capabilities. Because the Cycle-Time and Lead-Time will also be used as a communication tool, the model should also be input to confirm inclusion in the Communication Management Plan.

Manage Communications will receive the input from both Cycle-Time and Lead-Time in real-time, and this information should be used to report project performance.

Because no role or practice addresses Control Communications, this process will receive no input.

4.3.7 Project Risk Management

Risk Management is one on the most overrated project constrains. All advocate its importance, but tend to forget to apply the processes in the course of managing their projects. Risk Management takes time, budget and special skills to do right, which might be a reason many tend to leave it out [PMBOK®2013].

Both KANBAN and SCRUM have no defined Risk Management activities, even though they have some practices that can indirectly be considered Risk Management related activities, and therefore can be used as input to Risk Management processes.
Another aspect is the known risk mitigation derived from the iterative and incremental approach of both models. Scope decomposition, short development cycles and direct customer communication are examples known to affect project success and/or product conformance to requirements. We can say that it’s a casual and informal risk management strategy, nevertheless Agile teams report benefits and results.

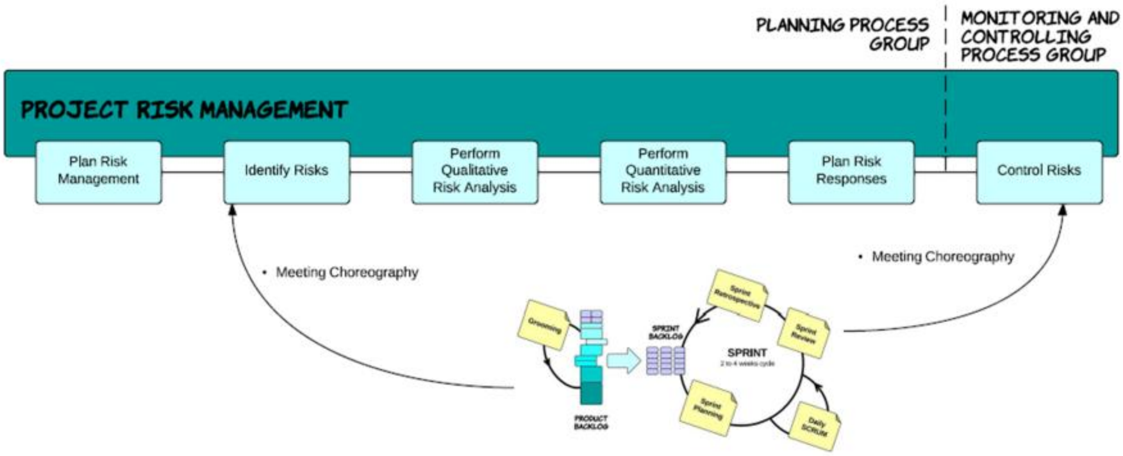


Figure 56 - SCRUM inputs to Risk Management processes

Even not having a systematic approach to Risk Management, SCRUM has some activities and practices that directly mitigate known project risks, hence some outputs may be gathered and passed to Risk Management processes that will manage risk at a Macro-Management level.

The majority of these are derived from the output generated on all the meetings that take place in the Sprint cycle. Even if the Team does not call them risks, it’s common to identify risks at Grooming activities or in Sprint Planning. These are normally not registered in any form, but they are outspoken between the Team and managed and reviewed during the Daily Scrum or/and in the Sprint Retrospective. These activities can easily be used as input for both Identify Risks and Control Risk processes.

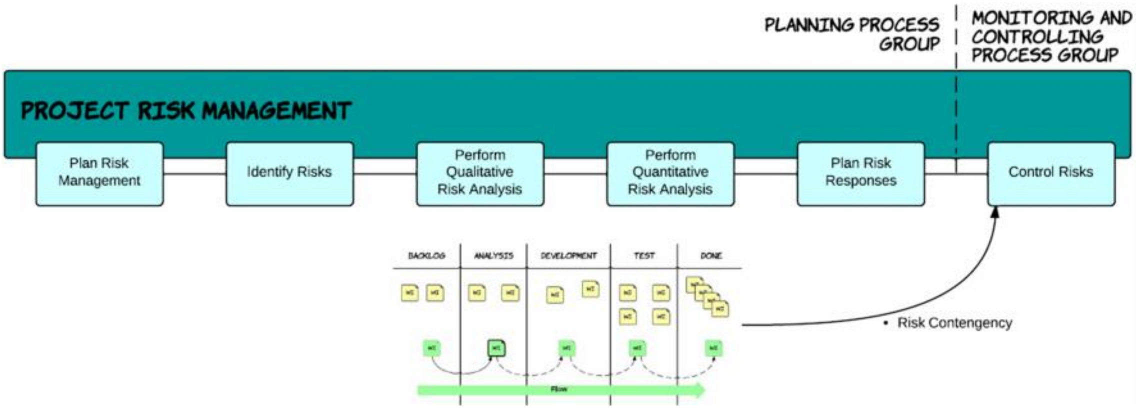


Figure 57 - KANBAN inputs to Risk Management processes

On the other hand, KANBAN does not have a defined meeting choreography or a planned development cycle, making the approach different.

KANBAN is focused on identifying risks that impact normal flow or impact response times. Unfortunately, because KANBAN is mostly event-driven, this approach is almost limited to risk contingency and not risk mitigation. Thereby, the Team will mostly control risk mitigation as risks happen and impact flow or response time. Inherited from manufacturing, it’s common to have specific contingency plans already defined for specific situations, ex: when WIP limit is reached or SOS are reprioritized.

4.3.8 Project Procurement Management

The modern IT World is most demanding. Resorting to outsourcing of services or products, consulting or products purchasing is done on a daily basis. Regrettably, even directly impacting projects or its produced products, Procurement Management is commonly done at project Macro-Management level or even done by a special company department with a feeble connection to the project team [PMBOK®2013].

This fact is even clearer in the Micro-Management Agile reality. Customarily, procurement activities are planned, executed and managed far from SCRUM or KANBAN influence, at most the teams are informed but not involved or accountable. For this reason, both SCRUM and KANBAN have no practices or common activities to address Procurement Management practices.

Therefore, no significant inputs can be drawn for the 4 processes proposed by the Project Procurement Management process area.

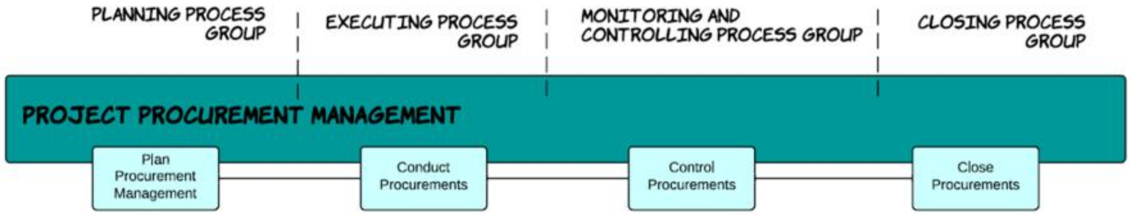


Figure 58 - Procurement Management processes

Something resembling an input to Plan Procurement Management would be necessities identified by the Team and/or customer when collecting or detailing requirements, but again it's too farfetched to be considered as such.

4.3.9 Project Stakeholders Management

The Project Stakeholder Management Knowledge Area was introduced in the PMBOK® 5th edition. The subject of managing stakeholders and their expectations is not new, but in the older version of PMBOK® that subject was included in the Project Communications Management Knowledge Area. The main reason for this is enhancing customer engagement regarding Project Management activities [AppX12013]. Consequently, the new Project Stakeholders Management Knowledge Area consists of 4 processes [PMBOK®2013].

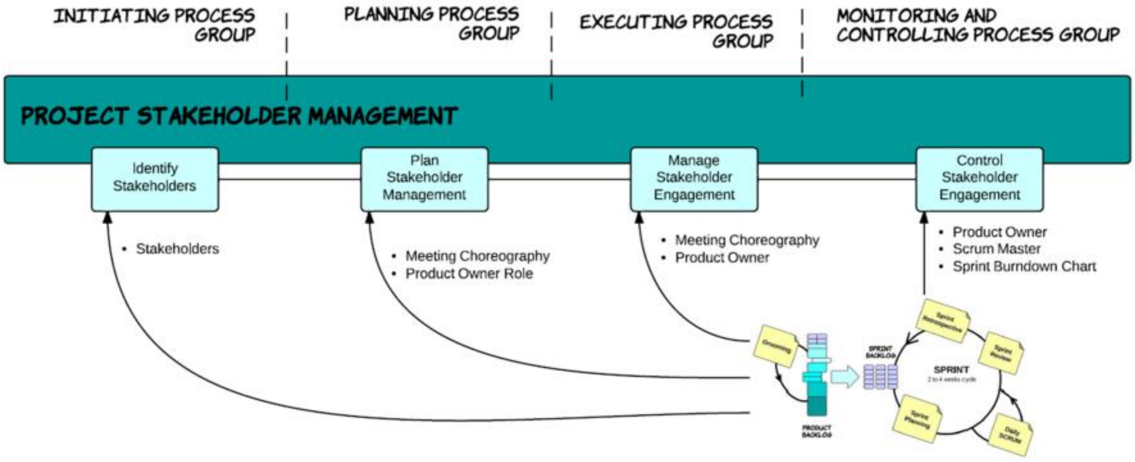


Figure 59 - SCRUM inputs to Stakeholder Management processes

As referred on Project Communication Management, SCRUM's mandatory meeting choreography and frequent validated deliveries, bounds Customer involvement to the development cycle. Hence, engaging the Customer in being committed on project and product success. Customer engagement is indispensable for SCRUM to work, the existence of the Product Owner role is evidence of Customer engagement preponderance.

The Identify Stakeholders process is one of the firstly executed processes on the Macro-Management perspective, therefore it should receive from the Team the list of identified Stakeholders. Surely this will not list all Project Stakeholders, but will list the ones directly involved in the SCRUM cycle.

Analogous to Plan Communication Management, Plan Stakeholder Management will receive input on the used meeting choreography, in order to plan how Stakeholder engagement will be guaranteed by the meeting that will take place. Although focused on the Customer, engagement should also address end-users, solution architects or steering committees. The Product Owner is a centrepiece of managing Stakeholder engagement.

During project execution, the Manage Stakeholder Engagement process will receive as input the evidences of the meetings taking place during the Sprints. On the other hand by doing his job, the Product Owner will create and maintain the appropriate channels to maintain Stakeholder engagement.

In the controlling perspective, Control Stakeholder Engagement will receive input of control activities done by the Product Owner and Scrum Master as well as additional performance information. The Product Owner will control and assure the level of Stakeholder Engagement, adjusting the Stakeholder Engagement strategy if needed. On the other hand, the Scrum Master will control the meeting choreography effectiveness, guiding the Team adjustments to the meeting choreography. Regarding performance information, the Sprint Burndown Chart is an excellent tool to report Sprint status.

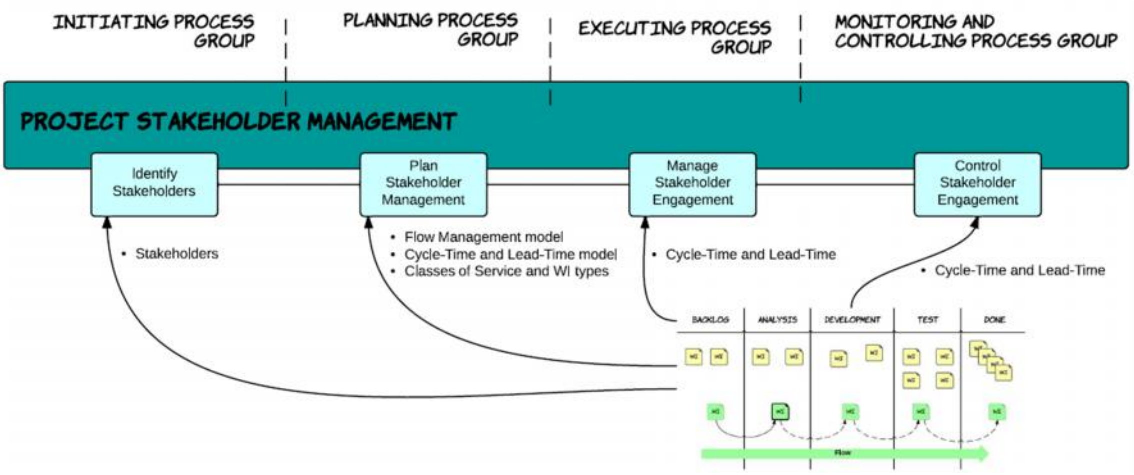

INITIATING PROCESS GROUP
PLANNING PROCESS GROUP
EXECUTING PROCESS GROUP
MONITORING AND CONTROLLING PROCESS GROUP
PROJECT STAKEHOLDER MANAGEMENT
Identify Stakeholders
Plan Stakeholder Management
Manage Stakeholder Engagement
Control Stakeholder Engagement
• Stakeholders
• Flow Management model
• Cycle-Time and Lead-Time model
• Classes of Service and WI types
• Cycle-Time and Lead-Time
• Cycle-Time and Lead-Time
BACKLOG
ANALYSIS
DEVELOPMENT
TEST
DONE
Flow

Figure 60 - KANBAN inputs to Stakeholder Management processes

KANBAN, not having a mandatory natural binding meeting choreography, relies on flow and results communication to create and maintain Stakeholder engagement.

Such like SCRUM, the Identify Stakeholders process should receive from the Team the list of identified Stakeholders.

On the other hand, Plan Stakeholder Management will receive how the KANBAN Workflow will be managed, including usage of Cycle-Time, Lead-Time, Classes of Service and Work Item types in the KANBAN Board. Differentiation of performance reporting will assure that all stakeholders, internal and external, receive value information, thus binding their engagement to the KANBAN flow and continuous improvement.

Due to not having specific roles to centralize Stakeholder communication and engagement, performing information will be the sole input to both Manage Stakeholder Engagement and Control Stakeholder Engagement.

4.3.10 Project Integration Management

Project Integration Management processes are the glue that binds all the process pieces from the previous Knowledge Areas. As the name suggests, they integrate all process and activities to achieve unification and coordination of all assorted processes. Such activities are vital for maximizing project performance and successful project completion. Project Integration Management is composed of 6 processes and they are extended throughout all 5 Project Management Process Groups [PMBOK®2013].

Develop Project Charter is without a doubt the first process executed in every Project, but because this activity is generally done by upper management, no significant input can drive from the SCRUM or KANBAN Team. It's clearly a sole Macro-Management process, and at that time it's even possible that upper management has no idea how many or what teams will be created to execute the project.

Develop Project Management Plan is process that merges all the adjuvant Project Management Plans of all Knowledge Areas in to one all-inclusive Project Management Plan. Because inputs for all other processes were already laid out for both SCRUM and KANBAN, we will assume no more inputs at this time. In opposition to Develop Project Charter, Develop Project Management Plan is a process that should involve Team collaboration, thus receiving inputs from Micro-Management activities.

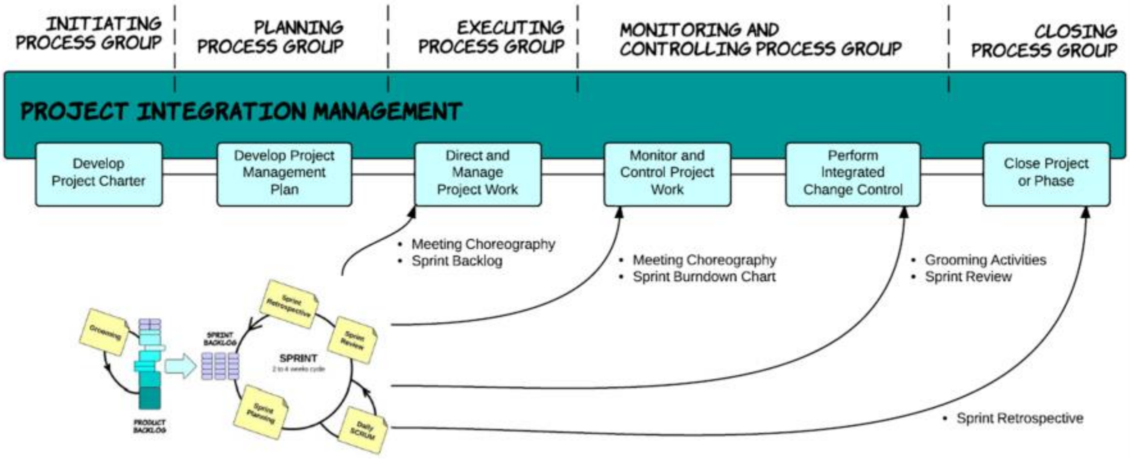


Figure 61 - SCRUM inputs to Integration Management processes

Direct and Manage Project Work as an executor of planned activities will receive the input from the various SCRUM meetings take place in the SCRUM cycle and the Sprint Backlog.

From the Sprint Planning crucial commitments to the Daily Scrum constant alignment, these meetings are an important factor to report the activities being done by the Team at all times. Although some of these meetings don't produce evidences, a way needs to be found to report SCRUM Micro-Management to the Macro-Management process.

On the other hand, the Sprint Backlog contains detailed information on the work being performed in each particular Sprint, with vital information to manage project execution.

Monitor and Control Project Work will also receive the same information reports of various SCRUM meetings and also the Sprint Burndown chart for tracking proposes.

Sprint Reviews, Sprint Retrospectives and Grooming activities are particularly important, due to the generated output. These meetings are capable of identifying possible Change Requests to product and project, and these should be monitored at both Micro-Management and Macro-Management levels.

Perform Integrated Change Control will approve the identified Change Requests, therefore receiving feedback from Sprint Reviews and Grooming activities are of extreme importance to better detail the final decision.

Close Project or Phase will not receive much input from the SCRUM cycle, SCRUM contribution is due before this phase. The activities of this process as mainly Macro-Management ones, aligned with the overall project official closure.

Nevertheless, Sprint Retrospective can generate some information resembling Lessons Learned, and if so, these should also be added to the overall project Lessons Learned.

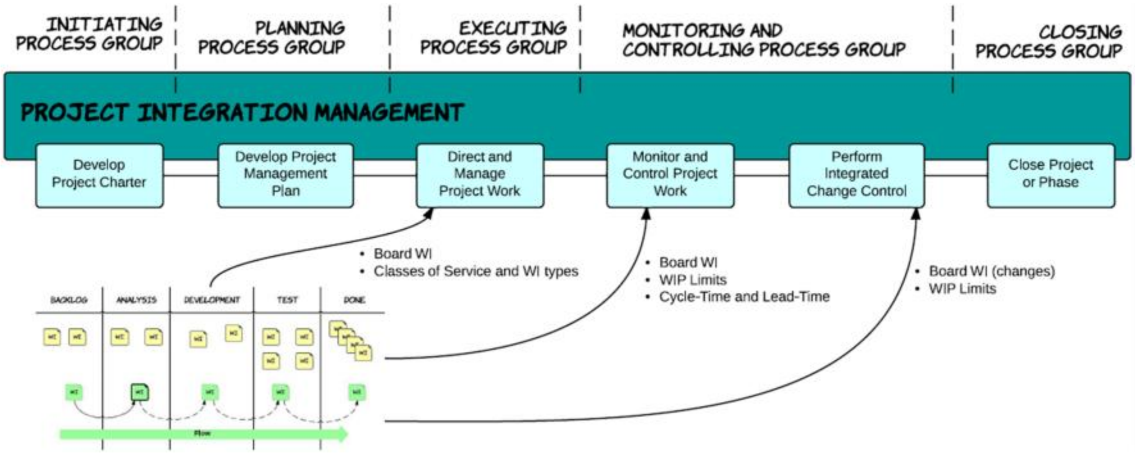

INITIATING PROCESS GROUP
PLANNING PROCESS GROUP
EXECUTING PROCESS GROUP
MONITORING AND CONTROLLING PROCESS GROUP
CLOSING PROCESS GROUP
PROJECT INTEGRATION MANAGEMENT
Develop Project Charter
Develop Project Management Plan
Direct and Manage Project Work
Monitor and Control Project Work
Perform Integrated Change Control
Close Project or Phase
• Board WI
• Classes of Service and WI types
• Board WI
• WIP Limits
• Cycle-Time and Lead-Time
• Board WI (changes)
• WIP Limits
BACKLOG
ANALYSIS
DEVELOPMENT
TEST
DONE
Flow

Figure 62 - KANBAN inputs to Integration Management processes

KANBAN, on the other hand, relies on the KANBAN board, WIP limits and Lead-time and Cycle-time control to manage project execution.

The KANBAN Board, Classes of Service and Work Items types should be considered as input to the Direct and Manage Project Work. The Board contains all Work Items at all times and Classes of Service and Work Items provide valuable information that may ease the execution tracking of the KANBAN Workflow. Thereby, the Team will be focused on micro-managing the KANBAN Workflow, and by providing such information, will allow the Macro-Management process to focus on direct and manage the overall project work.

On the other hand, Monitor and Control Project Work will receive the input to control execution. WIP Limits, Cycle-Time and Lead-Time will be vital information to track the development of the Work Items in the Board.

These metrics provide Work Items execution status and spot possible Change Requests to already existing Work Items, mostly product related. Continuous improvement activities derived from Team self-organization can also induct Change Requests, generally at the project level. Although being micro-managed by the Team, these changes will need to be reviewed at a Macro-Management level for acceptance and compliance with the overall plan.

The Perform Integrated Change Control process will approve the identified Change Requests, hence needing input regarding changes on the Board and the WIP limitations for reprioritization purposes.

In a way similar to SCRUM, Close Project or Phase will not receive much input from the KANBAN cycle. The self-organization concept will, without a doubt, spawn Lessons Learned regarding product and/or project execution, but due to the nonexistence of a Sprint Retrospective-like ceremony, this valuable asset will be of difficult gathering.

4.4 Input Tailoring Matrix

The last sub-chapter detailed the tailoring input diagrams for the 47 PMBOK processes coming from both KANBAN and SCRUM Micro-Management approaches. In order to analyse the overall results we will categorise the level of input for each process in order to find the processes and Knowledge Areas with tendency to receive more input.
A detailed categorization of each input type is difficult to encounter due to the differences and lack of definition of both KANBAN and SCRUM frameworks. SCRUM Roles have no parallel in KANBAN and its crucial importance in SCRUM has no meaning in KANBAN. Input type definition would be of difficult determination and would not provide accurate comparison.

Because of this, we will define a simple scale that measures the input level, from abundant input to no input at all. Therefore we came up with a 3 step scale [Campos2013]:

HI	High Input

- Means that the process receives several different inputs from KANBAN or SCRUM

LI	Low Input

- Means that the process receives at least one input from KANBAN or SCRUM

NI	No Input

- Means that the process receives no input from KANBAN or SCRUM

This simple scale will help us identifying the input tendency at least in quantitatively terms, but unfortunately it will not weight the qualitative value of the inputs themselves. This will be identified as a possible future work.

The result is the following:

Knowledge Area	Process	SCRUM	KANBAN
Scope Management	Plan Scope Management	HI	HI
Scope Management	Collect Requirements	HI	HI
Scope Management	Define Scope	LI	HI
Scope Management	Create WBS	LI	LI
Scope Management	Validate Scope	LI	LI
Scope Management	Control Scope	HI	HI
Time Management	Plan Schedule Management	HI	HI
Time Management	Define Activities	HI	LI
Time Management	Sequence Activities	HI	LI
Time Management	Estimate Activity Resources	HI	LI
Time Management	Estimate Activity Durations	HI	HI
Time Management	Develop Schedule	HI	HI
Time Management	Control Schedule	HI	HI
Cost Management	Plan Cost Management	HI	HI
Cost Management	Estimate Costs	LI	LI
Cost Management	Determine Budget	LI	LI
Cost Management	Control Costs	LI	LI
Quality Management	Plan Quality Management	HI	HI
Quality Management	Perform Quality Assurance	HI	LI
Quality Management	Control Quality	LI	LI
Human resource Management	Plan Human Resource Management	LI	LI
Human resource Management	Acquire Project Team	LI	LI
Human resource Management	Develop Project Team	HI	LI
Human resource Management	Manage Project Team	HI	LI
Communications Management	Plan Communications Management	HI	LI
Communications Management	Manage Communications	HI	LI
Communications Management	Control Communications	HI	NI
Risk Management	Plan Risk Management	NI	NI
Risk Management	Identify Risks	LI	NI
Risk Management	Perform Qualitative Risk Analysis	NI	NI
Risk Management	Perform Quantitative Risk Analysis	NI	NI
Risk Management	Plan Risk Responses	NI	NI
Risk Management	Control Risks	LI	LI
Procurement Management	Plan Procurement Management	NI	NI
Procurement Management	Conduct Procurements	NI	NI
Procurement Management	Control Procurements	NI	NI
Procurement Management	Close Procurements	NI	NI
Stakeholder Management	Identify Stakeholders	LI	LI
Stakeholder Management	Plan Stakeholder Management	HI	HI
Stakeholder Management	Manage Stakeholder Engagement	HI	LI
Stakeholder Management	Control Stakeholder Engagement	HI	LI
Integration Management	Develop Project Charter	NI	NI
Integration Management	Develop Project Management Plan	HI	HI
Integration Management	Direct and Manage Project Work	HI	LI
Integration Management	Monitor and Control Project Work	HI	HI
Integration Management	Perform Integrated Change Control	HI	HI
Integration Management	Close Project or Phase	LI	NI

Table 7- Input Tailoring Matrix

4.5 Conclusions

With information from both Input Tailoring Diagrams and Input Tailoring Matrix, lets analyse the information revelled.

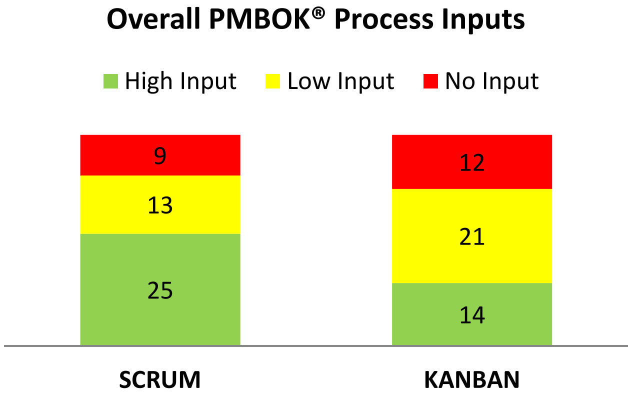


Figure 63 - Overall PMBOK® Process Inputs

Overall high-level analysis shows that SCRUM generates more input to the 47 PMBOK® processes than KANBAN. This is mainly due to the SCRUM mandatory choreography, roles and artefacts. Those more palpable definitions allow for an easier identification of possible inputs. The KANBAN choreographyless and non-rigid approach proves to be of more difficult input identification.

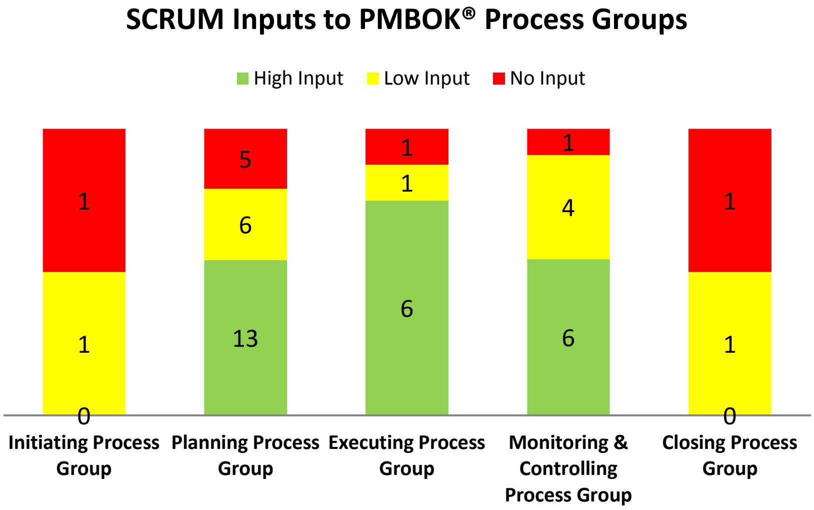


Figure 64 - SCRUM Inputs to PMBOK® Process Groups

SCRUM has little to contribute to the Initiating and Closing Process Groups. This comes as no surprise, these two Process Groups contain manly high-level management activities much distant from the Team, mostly managed by the high-level Project Management or performing organization.

On the other hand, Planning, Executing and Monitoring and Control Process Groups receive many inputs. These result from the SCRUM choreography execution, that plan, execute and control Sprints throughout the development cycle.

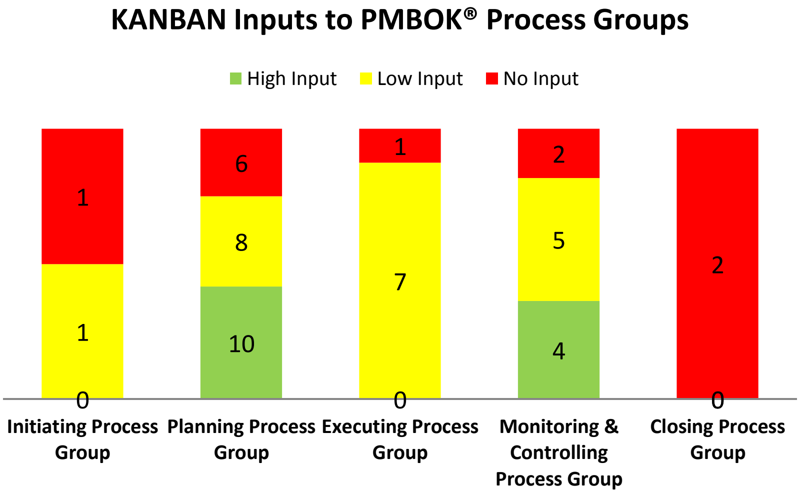


Figure 65 - KANBAN Inputs to PMBOK® Process Groups

KANBAN demonstrates almost no input to the Initiating and Closing Process Groups. Like SCRUM, being a Team level focused approach, KANBAN contributes with little value input to such high-level Project Management practices.

KANBAN shows many inputs to the Planning, Executing and Monitoring and Control Process Groups, however not as much as SCRUM. This is mostly the result of not having a clear mandatory choreography and support artefacts that can easily be used as input to the Macro-Management process.

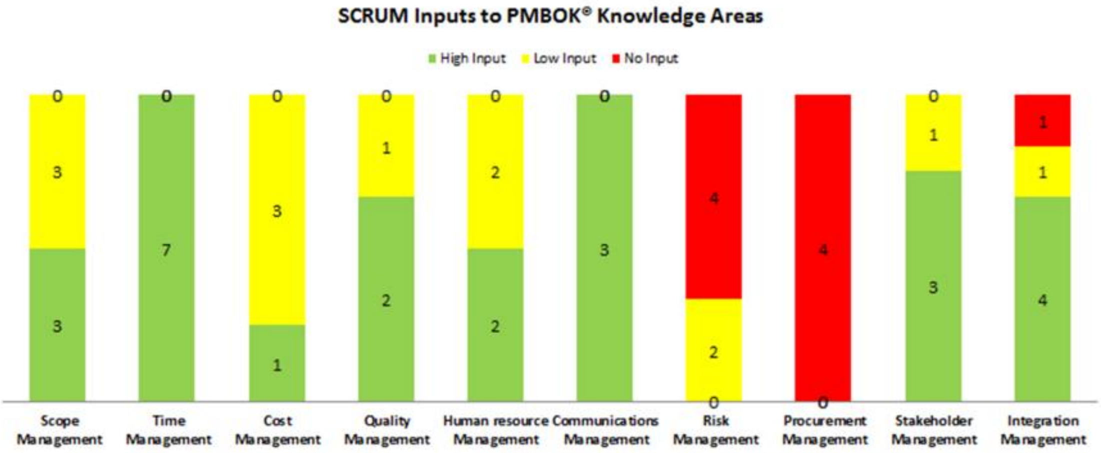


Figure 66 - SCRUM inputs to PMBOK® Knowledge Areas

Scope and Time Management processes benefit from the iterative and incremental approach (Sprint time-box), their activities and mandatory meetings, providing much input for processes of both Knowledge Areas.

Customer proximity, mandatory roles and meetings provide Communications, Human Resource and Stakeholder Management with much input.

Quality Management profits from SCRUM work validation procedures and a lower cycle time, ensuring value Quality planning and control inputs.

As explained before, SCRUM does not provide much input to Risk and Procurement Management. The same could be said for Cost Management processes, but even not being in the SCRUM nature it provides some degree of valued input to be used in the overall Project Cost Management activities.

Integration Management processes receive some input from SCRUM, although not as much as other Knowledge Areas, nevertheless the mandatory choreography and constant Sprint control generate much input that allows to direct and control work execution. The Micro-Management choreography approach is affluent in providing Sprint status and deviation from plan (bottom-up), allowing the Macro-Management use that detailed information for appropriate management of the overall Project.

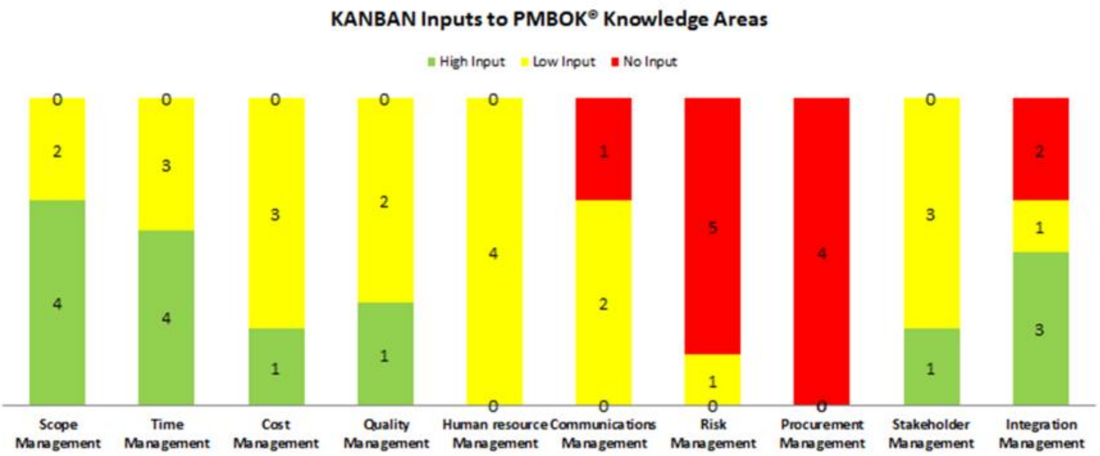


Figure 67 - KANBAN inputs to PMBOK® Knowledge Areas

As expected, the event-driven orientation penalises the inputs for the estimating and planning related processes. Scope and Time Management receive much input for controlling related processes, due to the WIP Limits and other metrics, but the planning processes receive seldom inputs.

Not having mandatory roles, meeting choreography and customer engagement toolsets result in lesser inputs to Human Resource, Communication and Stakeholder Management.

Like SCRUM, and largely for similar reasons, scarce input is provided to Risk and Procurement Management processes, with Cost Management processes having some input that allows some degree of cost related control.

Quality Management processes will receive some degree of input, but because no mandatory validation is defined in KANBAN, it is basically dependent from quality related activities being done in the defined Workflow, but not by the framework itself.

Integration Management processes receive input from the event-driven concept, with direct and control work execution being based on the metrics reported and the Work Items on the KANBAN board. This allows a Micro-Management approach with real-time metrics and deviations report (bottom-up) allowing it to be used to manage the overall Project on the Macro-Management perspective.

5 Conclusions

Every project is different. Every company or project struggles to some degree with existential management questions regarding the process and procedures used to plan, execute, control and close the project.
This exercise aims to shed some light on how to choose and adapt some of the most used frameworks on today's market. Due to its quantitative-only nature, it should not be seen as carved in stone but as a pragmatic approach enhanced by Tailoring aids and lessons learned.

As a Tailoring disclaimer, one needs to know in some detail the application of all the 3 frameworks and pragmatically identify the approach to take. Our assumption, as mentioned in sub-chapter 1.2, is that a significantly large company with numerous and/or complex projects would benefit from an integrated Macro/Micro-Management approach. This may not be true for every project or performing organizations.
Some small projects or companies may encounter additional overhead with such an approach, one that does not make up for the value created in the end run. Tailoring discretion is advised in all situations: don't embellish what could be managed with a straightforward solution.

There are different reasons for the choice of these specific 3 frameworks. As referred before in chapter 2, both SCRUM and KANBAN are on currently hyped-up in the IT industry, representing the peak of agility nowadays. From the two, SCRUM is the most well-known, owing its recognisance to world-wide event promotions, certification programs, gainful consultants and shared success stories. Nevertheless, KANBAN is a small rising star, making its way from deep Lean Management to the IT industry. Even obfuscated by the SCRUM hype and success, it was never out of the race. PMBOK® is by far the most known beacon of Project Management knowledge. Created collaboratively with the professional community and being a hub for lessons learned, it's truly a reminder of all the areas that can easily be dangerously overlooked. Choosing the three was a no-brainer.

As researched, both KANBAN and SCRUM have its strengths and pitfalls, and their application can be framed based on some key concepts regarding industry, scope, response cadence, etc. Even contrasting in approach, both are based on Agile and Lean concepts and offer an excellent approach for Micro-Management activities at development team level. This management level can be sufficient in many realities, but as observed it clearly lacks Macro-Management activities when compared with PMBOK®'s vast processes.

It's our understanding that the resulting Tailoring Diagrams, the Tailoring Matrix and enclosed details are valuable tools to identify dependencies between the three frameworks, making somehow lighter the heavy burden of tailoring the processes, procedures or methodology for your project or performing organization. Please use them wisely.

5.1 Summary

The Introduction chapter explained our motivations, framed our Challenge and defined our three objectives, allowing us to set our goals to this exercise, ones we will superficially review below.

Chapter 2 “Project Management and Agile Methods” brought to light some important aspects regarding the History of Project Management, its success - or lack of it - and some Agile principles. Agile roots were explained by Project Management evolution and Project Success, and mapped with the origins of SCRUM, KANBAN and PMBOK®.

This research exercise was especially helpful in understanding the Project Management history unravelment and some aspects that tend to be forgotten in time. Those aspects allowed us to better understand the motivations behind the creation on the three frameworks, important information for the next chapter.

Chapter 3 “KANBAN vs. SCRUM” laid the ground for a detailed comparison of both KANBAN and SCRUM using the same approach. Because of the non-existent official framework definition for both, the comparison was focused on community accepted concepts and personal past experience. This resulted in the identification of shared concepts which are important facts to the following work done. As a conclusion, Pros/Cons comparisons of both models mapped the differences between them, enriched with explained details for adaptation and application, given different environments or projects.

This information was not only vital for framing the new chapter analysis, but can also be seen as a benefit to the ones deciding on what Agile approach to use on their project or performing organization: KANBAN or SCRUM.

Chapter 4 “Mapping PMBOK® Processes with Agile Practices” not only detailed the Micro/Macro-Management approach but also mapped every PMBOK® Process with both KANBAN and SCRUM practices, detailing the Micro/Macro-Management approach individually for each process by way of Tailoring Diagrams. This resulted on Input Tailoring Diagrams and the Input Tailoring Matrix, with the first ones being detailed to some extent. Conclusions were also drawn from a quantitative analysis regarding the possible PMBOK® process inputs received from the KANBAN and SCRUM practices.

We truly believe that the triplet Input Tailoring Diagrams, the details for each diagram and the overlap Input Tailoring Matrix is intended to be used as a tool that will benefit professionals struggling with initial adaptation and/or tailoring of both Agile and traditional Project Management activities and frameworks.

5.2 Future work

The nature of this exercise opens considerable future work possibilities, both practical and theoretical.

From a theoretical analysis prism, as referred before, we could enhance this exercise with more a qualitative analysis of the inputs identified, and this would without a doubt increase the value of the Tailoring aids. Unfortunately, because both KANBAN and SCRUM frameworks don't have a world-accepted and clear-defined definition, qualitative analysis and input categorization is difficult to accomplish… "difficult" can mean debatable and controversial, but it does not mean impossible.

A more practical exercise would be to try the Macro/Micro-Management approach in the field, in a real performing organization, and measure the value created by such approach. This would probably result in momentous lessons learned to other performing organizations, Project Managers, PMOs and Process Engineering professionals.

Another interesting but monumental exercise, with a theoretical and practical essence, would be a detailed and precise two-way mapping between KANBAN/SCRUM inputs/outputs/techniques and each PMBOK® process ITTOs (Inputs, Tools, Techniques, Outputs). This would not only include the already referent qualitative input analysis, but also a detailed mapping of inputs, outputs, tools and techniques from both levels. This would be an overwhelming production, one that would need to be aligned with both PMI®, for guidance and copyrights, and with key representatives of the Agile community. The result could be a detailed all-containing Tailoring framework, covering from low-level Agile Project Management activities to the high-level Project Management PMBOK® processes.

References

[Weaver2007] Patrick Weaver, April 2007, *"The Origins of Modern Project Management"*, (Lecture, Fourth Annual PMI College of Scheduling Conference, Vancouver, Canada)

[Chiu2010] Y.C. Chiu, 2010, *"An introduction to the History of Project Management"*, Eburon Academic Publishers

[Taylor1911] Frederick Winslow Taylor, 1911, *"The principles of Scientific Management"*, Republished 2008 by Forgotten Books

[PMBOK®2009] Project Management Institute, 2009, *"A Guide to the Project Management Body of Knowledge (Fourth Edition)"*, Project Management Institute

[Charvar2003] Jason Charvat, 2003, *"Project Management Methodologies Selecting, Implementing, and Supporting Methodologies and Processes for Projects"*, JOHN WILEY & SONS, INC

[ICA2006] International Project Management Association, 2006, *"IPMA Competence Baseline (Version 3.0)"*, International Project Management Association

[MSPP2009] Office of Government Commerce, 2009, *"Managing Successful projects with Prince2TM"*, Office of Government Commerce

[PMNJul2007] PM Network magazine, July 2007, *"Top 9 causes for project failure"*, Project Management Institute

[WDBT2008] William Dow, Bruce Taylor, 2008, *"Project Management Communications Bible"*, Wiley

[CHAOS1994] *"CHAOS Report"*, 1994, Standish Group,

[SumCHAOS2009] *"CHAOS Report Summary 2009"*, 2009, Standish Group

[EvVer2010] J. Laurenz Eveleens, Chris Verhoef, January/February 2010, *"The Rise and Fall of the CHAOS Report Figures"*, IEEE Software, IEEE Computer Society

[RLG2005] Robert L. Glass, *"IT Failure Rates 70% or 10–15%?"*, 2005, IEEE Software, IEEE Computer Society

[DFAN2010] Daojin Fan, July 2010, *"Analysis of critical success factors in IT project management"*, 2nd International Conference on Industrial and Information Systems

[Spalek2005] Seweryn Spalek, 2005, *"Critical Success Factors in Project Management. To Fail or not to Fail, That is the Question!"*, 2005 PMI Global Congress Proceedings

[JKB1996] James J. Jiang, Gary Klein, Joseph Balloun, December 1996, *"Ranking of system implementation success factors"*, Project Management Journal

[JH2002] Jim Highsmith**,** 2002 **,** ***"****Agile Software Development Ecosystems"***,** Addison Wesley

[AM2001] Several authors, 2011, *"Agile Manifesto"*, Agile Manifesto

[TSUH2009] Thomas Stober and Uwe Hansmann, 2009, *"Agile Software Development"*, Springer

[PMI®2011] www.pmi.org, 21/04/2011, Project Management Institute®, Inc.

[SA2011] www.scrumalliance.org, 15/03/2011, SCRUM Alliance

[Cimo2013] Stephen Cimorelli, 2013, *"KANBAN for the Supply Chain: Fundamental Practices for Manufacturing"*, CRC Press

[HevCha2010] Hevner, A., & Chatterjee, S., 2010, *"Design Research in Information Systems - Theory and Practice"*, Springer

[Lu1989] David John Lu (translated), 1989, *"KANBAN Just-in-time at Toyota: Management Begins at the Workplace"*, Productivity Press

[Kenji2008] Kenji Hiranabe, 2008, *"KANBAN Applied to Software Development: from Agile to Lean"*, InfoQ Enterprise Software Development

[Takeuchi1986] Hirotaka Takeuchi and Ikujiro Nonaka, 1986, *"The new new product development game*", Harvard Business Review

[Rubin2012] Kenneth S. Rubin, 2012, *"Essential SCRUM: A Practical Guide to the Most Popular Agile Process"*, Addison-Wesley

[MBB2011] Scott Millett, Jerrel Blankenship, Matthew Bussa, 2011, *"Pro Agile .NET Development with SCRUM"*, Apress

[SMY2013] www.SCRUMmethodology.com , 16/08/2013

[hunskaar2013] hunskaar.com/kanban-in-a-non-linear-flow/, 21/08/2013

[Charvat2003] Jason Charvat, 2003, *“Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects”*, John Wiley & Sons

[SGuide2013] Ken Schwaber, Jeff Sutherland, *"The SCRUM Guide: The Definitive Guide to SCRUM: The Rules of the Game"*, 2013, www.SCRUM.org

[Schwaber2011] Ken Schwaber, 2011, *“The enterprise and SCRUM”*, O'Reilly Media, Inc.

[Keith2010] Clinton Keith, 2010, *“Agile Game Development with SCRUM*”, Pearson Education

[RisJan2000] Linda Rising and Norman S. Janoff, 2000, *“The SCRUM Software Development Process for Small Teams”,* IEEE SOFTWARE July/August 2000

[AppX12013] *“Appendix X1 - Fifth edition changes”*, 2013, Project Management Institute®, Inc.®

[Sadd2012] Peter Saddington, 2012, *“The Agile Pocket Guide: A Quick Start to Making Your Business Agile Using SCRUM and Beyond”*, John Wiley & Sons

[Cohn2004] Mike Cohn, 2004, *“User Stories Applied: For Agile Software Development”*, Addison-Wesley Professional

[WSG2010] Elizabeth Woodward, Steffan Surdek, Matthew Ganis, 2010, *“A Practical Guide to Distributed SCRUM”*, Pearson Education

[Grenning2002] James Grenning, 2002, *“Planning Poker or How to avoid analysis paralysis while release planning”*, renaissancesoftware.net

[Shall2011] Alan Shalloway, 2011, *“Demystifying KANBAN”*, NET OBJECTIVES, INC

[Anderson2010] David J. Anderson, 2010, *“KANBAN: Successful Evolutionary Change for Your Technology Business”*, Blue Hole Press

[KniSka2010] Henrik Kniberg, Mattias Skarin, 2010, *“KANBAN and SCRUM: Making the Most of Both”*, InfoQ Enterprise Software Development, C4Media, Publisher of InfoQ.com

[Ko2013] Neville Ko*, 2013, "Pre-development & Iterative Design Activities in SCRUM: An Approach to Agile Software Development"*, Corel Corporation

[Ikompass2013] www.ikompass.co.nz/images/ACP/burn-down-chart.png, 3/09/2013

[TBoard2013] www.teamscrumboard.com/images/appwnd.png, www.mountaingoatsoftware.com/system/asset/file/33/LabelledTaskBoard.jpg, 3/09/2013

[Toyota2012] www.toyota-global.com/company/vision_philosophy/, 4/09/2013

[Budau2012] Tiberiu Marian Budau, 2012, *"KANBAN in a nutshell"*

[Nain2004] Philippe NAIN, 2004, *"Basic elements of Queuing Theory: Application to the Modelling of Computer Systems"*, INRIA

[Poppen2005] Mary Poppendieck, 2005, "Managing the Pipeline", www.leanessays.com/2010/11/managing-pipeline.html (10/09/2013)

[Sahota2010] Michael Sahota, 2010, *"SCRUM or KANBAN? YES!"*, http://agilitrix.com/2010/05/SCRUM-or-KANBAN-yes/ (11/09/2013)

[Ramrak2011] Nitin Ramrakhyani, 2011, *"Implementing SCRUMban, SCRUM Bangalore"*, "Pecha Kucha" Presentation

[Morten2013] olemortenamundsen.wordpress.com/2010/03/19/kanban-and-scrum-combined/, 10/09/2013

[CMMIDEV2010] *"CMMI® for Development (version 1.3)"*, Software Engineering Institute

[Larman2004] Craig Larman, 2004, *"Agile and Iterative Development: A Manager's Guide"*, Addison-Wesley Professional

[Goldratt1997] Eliyahu M. Goldratt, 1997, *"Critical Chain"*, North River Press

[Campos2013] Carina Daniela Batoca Campos, 2013, *"Estudo de Interdependências e Rastreabilidade no Processo RUP: Análise do Micro-Processo V-Model e do Método 4SRS"*, Universidade do Minho - Portugal