How to **outsource scrum projects**

xsolve.

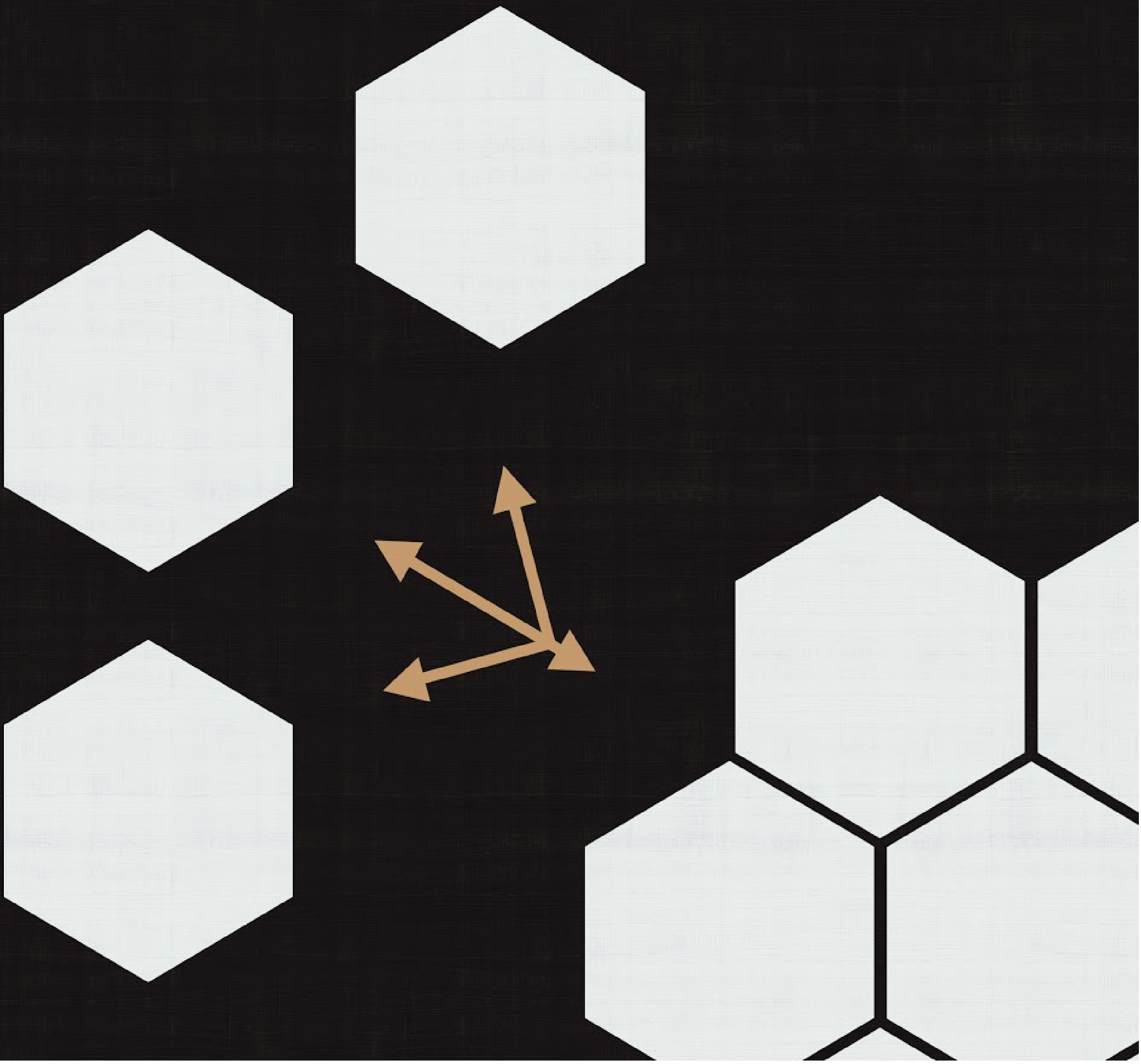


Table of Contents:

1 Introduction

This is a document which fulfils the need for information. It answers the questions that inseparably go together with modern challenges of running a business. Modern business more and more often creates a vision and influences consumers towards following this vision and the product value it implies. Rather than creating a product alone, it creates a need and value.

That's why this free e-book is for business owners and managers. For decision makers, who need tailored software which realises the initial idea for the application.
As the authors of this publication, we believe in Scrum, we are its everyday users. If there's a need to get familiar with it, this document provides the necessary knowledge and practical tips.

We have made it to help decide on choosing the best possible way to outsource software projects. We have also addressed the issue of Agile and Scrum and the very important company's culture, as a part of natural conjunction between the quality expectations and the final outcome.

The envisaged result of reading this document would be a better understanding of the outsourcing itself and the challenges it predicates.

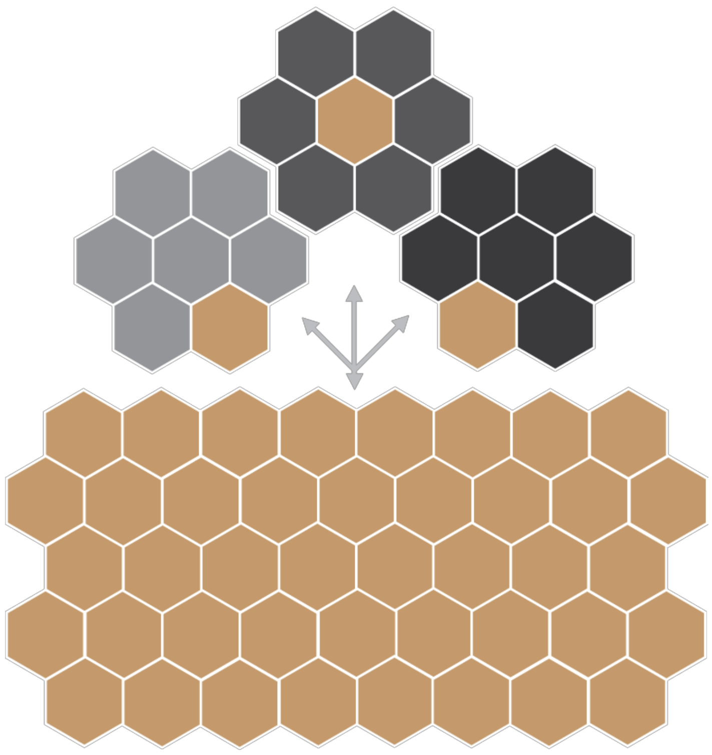
The process of making this e-book was based on the knowledge of the topic itself. After researching the common and available knowledge (some of which was used here, along with some graphic materials), we decided on making something of our own authorship.

If the Reader knows Scrum already, we suggest skipping the contents and heading straight to section 4.

The Authors

2

Why do outsourcing?



Outsourcing is today’s prime tool for managing risk, resources, and projects. By outsourcing software development one can find companies which understand the value of the Partners’ business and product, help develop and elevate it, cutting noncrucial functionalities and introducing new ones. That makes the outsourcing company both a developer, consultant, and often business advisor.

The other perspective is that over 85% of startups and enterprises have problems with attracting and retaining tech talents[1]. And those are just few of the problems companies face within the development field. Outsourcing, a cheaper and more convenient way, solves those problems. Due to this, process costs of HR, maintaining workplaces and leaves, are irrelevant. Moreover, software development is complex and outsourcing companies solely focused on software services are often more effective.
Those are merely statistics. Important, but not essential. What really matters is how an outsourcing company will fit into the organisation of the Partner both culture and process wise.

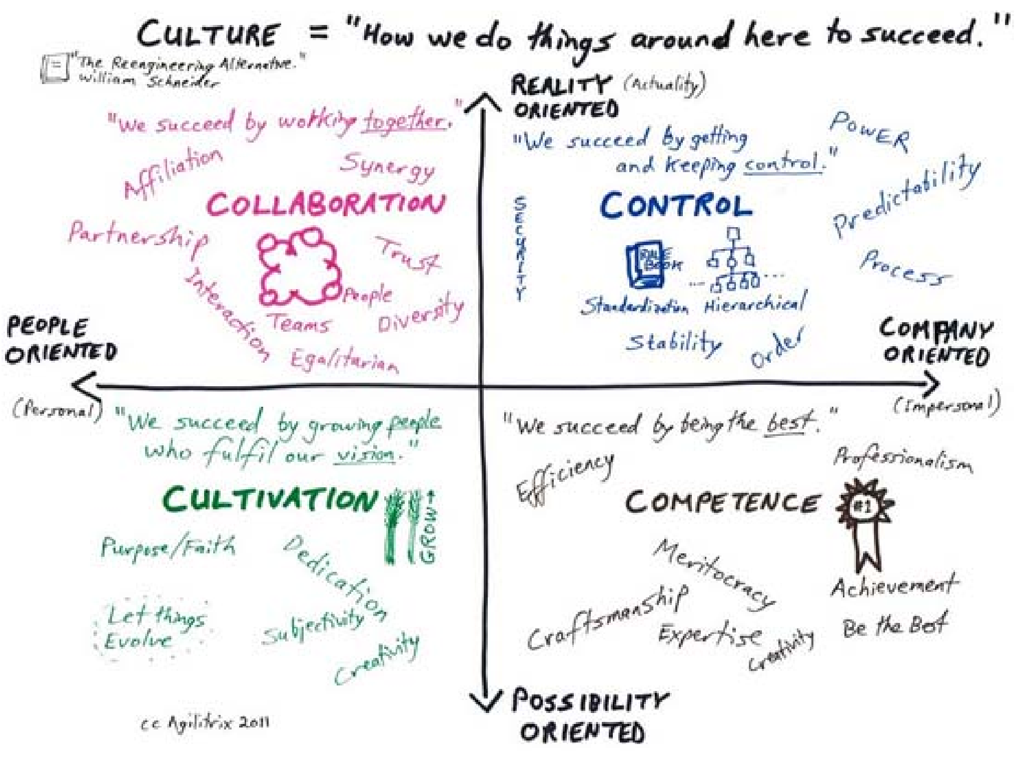
[1] Venture Pact

Software development outsourcing could be demanding in terms of research, time, and risk. That's why it's important to choose the right developer.

Organisation's culture is the first step

To achieve the best results you have to start with the organisation's culture. What is your culture and what is the culture of the company you want to outsource your development to?

But let's come back to what organisation culture is: this is how it's done.[2] The picture above shows four different organisational cultures.



Schneider's Culture Model, source: http://www.methodsandtools.com/

There is no wrong model of collaboration and culture. Every single model featured above has its own limits, environment, and scope. In Agile, though, collaboration culture is desired. Control is a big no-no, competence is for companies driven by a single goal or set of goals. Cultivation is for visionaries. Collaboration is for… well, everyone. Everyone who is prepared for it, that is. **Trust, diversity, partnership - these are the words that fuel Agile projects.**
If your organisation's Culture is Control and Competence, does the outsourcing company have a similar one? If your outsourcing company of

[2] http://bit.ly/1H7r83H

choice has no Culture of Collaboration and your idea is to share the knowledge across in-house and outsource teams, can you make it? If you work in iterations, does the outsourcing company as well? Communication - how are you doing it? Through Project Manager, or is the team used to communicate with everyone by themselves?

In the process of outsourcing, many Partners find themselves awaken. Developers making the software product can point to new solutions, correct the view on particular functionalities, suggest changes. Forwarding the process to other companies can also free the company's employees and make time for future endeavours[3].

The primary concern regarding software development outsourcing should be minimising functional disjunction. This is the threat of lacking proper communication, project management, and general vision due to geographically dispersed development. Outsourcing everything should be based on the ability to properly manage and receive the feedback, not only the code itself. **Working in the same cultural context gives the Partner and Scrum Team the ability to hear the same message that's getting across, not the words themselves.** Project management is equally vital. It means the necessity to attend the meetings (even via Skype or Google Hangouts) by the Partner and Scrum Team. General vision should be respected across the board and influence every part of the project.

This is also the case of motivated vs engaged developers. Those motivated by money will be simply making code. Those motivated by creating the best possible business value will overcome the presence of possible disjunction.

The best way of making the software work is to make it correctly, with the Partner, all the way. This is the way of Agile, the philosophy of success. It covers the basic principles of everchanging circumstances. Agile developers believe that changes in technologies, trends, and most important - needs of users - demand quick reactions. That's why they operate on a set of Agile guidelines[4]. One of them is Scrum, a framework to organise and optimise the process of development.

Quote: **'Never was so much owed by so many to so few'.**

Winston Churchill

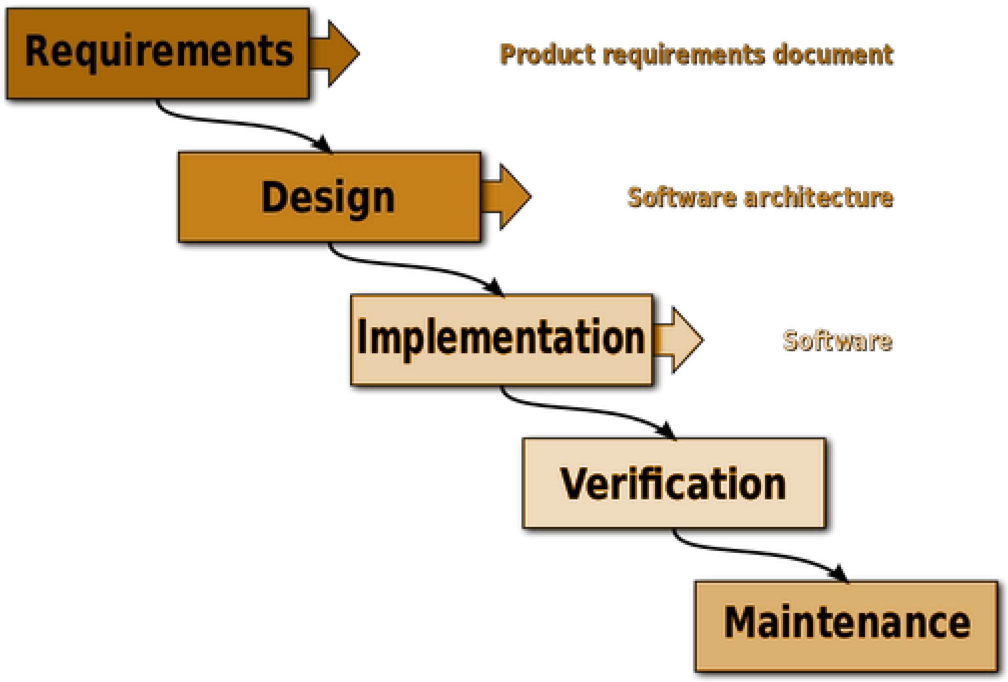
[3] http://bit.ly/1ELHrSX
[4] http://bit.ly/17RD3H2

3

Why Scrum?

The immanent characteristic of software is that it’s open to constant development. Years ago, programmers noticed that traditional methods of making software were less efficient than expected. Then some of the developers started to find more effective frameworks to what they did everyday. Nobody wanted to waste time and money.

On the other hand, a popular approach to outsourcing is still the Waterfall: highly structured physical environments in which after-the-fact changes are prohibitively costly. It is a sequential design process in which progress is seen as flowing through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance. Once it’s done - it’s done. Changing or shifting things during the period of making the software brings consequences - longer time of making the software, longer exposure to stress produced over the process of change. In Scrum it’s possible to make changes on-the-fly and mandatory to cooperate.



By Peter Kemp / Paul Smith (Adapted from Paul Smith's work at Wikipedia).

A key principle in Scrum is its recognition that customers can change their minds about what they want and need. This allows to recognise the requirements late and cleverly respond to emerging business changes without budget overruns.[5]

Scrum adopts an empirical approach: a problem cannot be fully understood and defined, therefore the focus should be on delivering quickly and responding to changes.

Scrum fulfils the *Manifesto for Agile Software Development*[6], which says:

'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**

Following this come **12 Principles of Agile Software Development**[7]

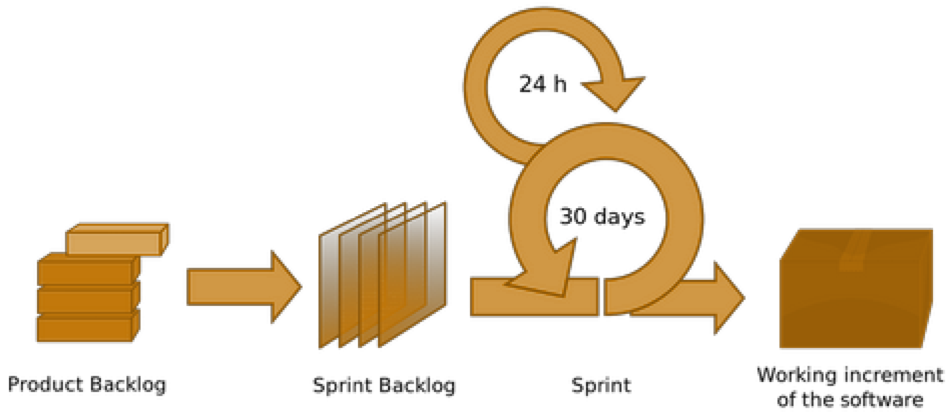
1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customers' competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

[5] http://bit.ly/1GlOq9S
[6] http://bit.ly/1vDH6iO
[7] http://bit.ly/1aLadtO

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximising the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

How does Scrum work?

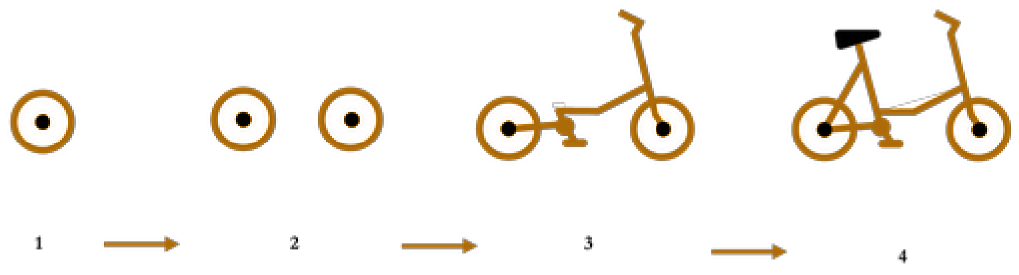
Software development processes should be built around the three pillars of Scrum method: Transparency, Inspection, and Adaptation.
Scrum Team is focused on understanding the Partner's vision and expectations at first. Customer and Provider become partners in development. Throughout the project, the team is flexible for changes. They use iterative approach, where the Partner gets 'increments' of his or her product in less than 30 days that are called Sprints. Developer takes care of the Partners' investment by avoiding the 'big design up front' and 'technical debts' traps.



The Scrum Process, source: http://en.wikipedia.org/

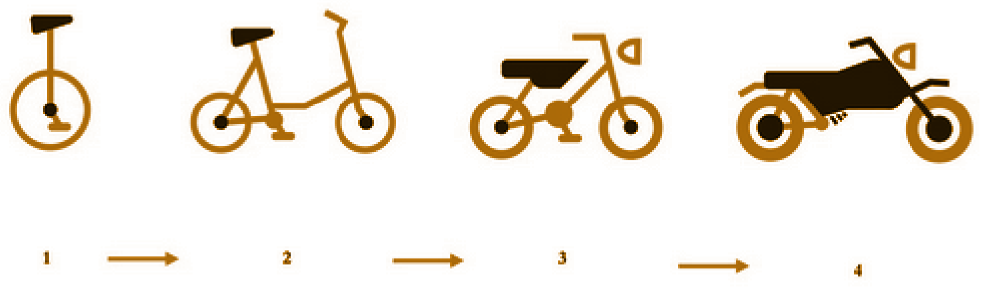
Building complex products for Partners is an inherently difficult task. Scrum provides a framework that allows teams to deal with this difficulty. This process is very simple, however difficult to master, especially for young development teams. The process is governed by the three main roles of Scrum Team: Product Owner, Scrum Master, and Development Team. Product Owner takes care of the business value of the product as well as helps in communication with the Partner. Scrum Master provides the right pipeline for communication between all the developers, Product Owner, and the Partner, and makes sure the process supports Transparency, Inspection, and Adaptation. And there goes the Development Team: in Scrum it's a self-organised team, which delivers a working product every Sprint. The progress of work of the Development Team is transparent and visible to all product's stakeholders.

The main advantage of iterative approach is that the Partner receives a working, and therefore valuable product, every Sprint. In Waterfall, a part of the system can be delivered during the project, but these parts are not designed to give business value before the project is finished:



Waterfall Process, credits: XSolve

In Scrum, Product Owner is responsible for prioritising requirements in Product Backlog for the Partner to receive a (business) valuable working product every Sprint:



Scrum Process, credits: XSolve

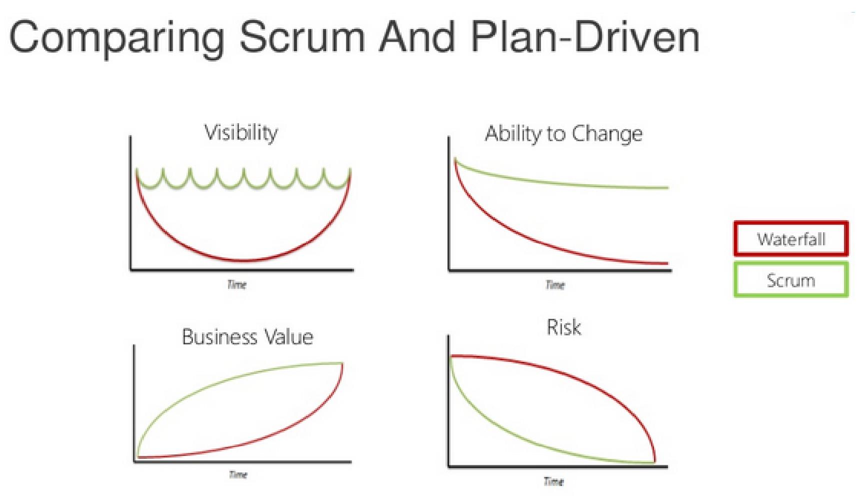
Scrum vs Scrum-but

Scrum is easy to learn, but difficult to master. And Scrum is the most wanted method in the modern software development. These reasons make some developers unable to deliver Scrum Teams, even if they want to prove they can. Sometimes the Partner can hear software companies saying: 'Yes, we work in Scrum, but we found that only parts work for us…'. This might be the first signal that such development company didn't master Scrum. **Scrum-but isn't Scrum and might be dangerous** because it puts Scrum Team off the balance and often results in Development being not predictable and not accountable at all.

How to check if a given developer really mastered Scrum? Ask for a number of successfully finished projects in Scrum; ask for certificates, ask for experience with Scrum, check their company blog. Ask them how their organisation supports their teams and develops them. Ask them if they have Scrum Coaches.
One of the most powerful questions might be asking them what they think about Scrum Retrospective and what value it brings to the Partner. If you don't have the knowledge to verify that by yourself, we strongly encourage you to use one of the Scrum Experts that can be found here:
https://www.scrum.org/Find-a-Scrum-Expert

To Scrum or not to Scrum?

Not every project needs Scrum: for some, the natural environment is a more traditional approach, which puts the primary focus on specifications and detailed project plans. For example, the Waterfall method is founded on sequential development model with clearly defined deliverables accompanied by a task plan for every phase.

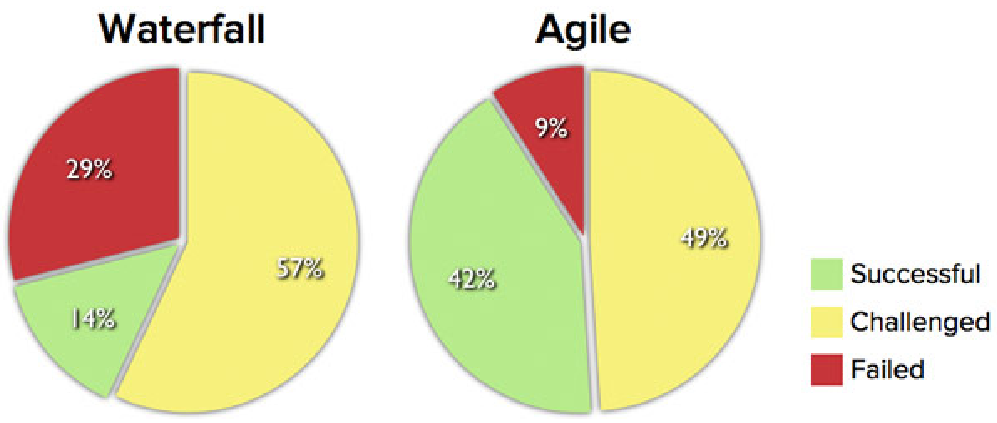


Comparing Scrum and Plan-Driven, source: http://www.slideshare.net/gerardbeckerleg/

In Waterfall there's a slim chance of making alterations during the project; it's very difficult and can cause a lot of problems. Another drawback is that the effect of long-term working is always invisible until the end. In Waterfall everybody must strictly respect the established plan, schedule and changes can be adapted during the product development but it's a disturbing process, costly and complicated. Therefore, it works for environments where business and technical requirements, technology and deliverables are known, well defined, and do not change throughout the project.

The results of the executed projects depend on dozens or hundreds of factors, most of which we do not even begin to realise. **Both Scrum and Waterfall are effective if they are applied correctly**, suited to the circumstances, organisations, projects, and teams. However, applying the adequate work method to the circumstances is very often extremely challenging and therefore so many Software Development projects fail or face challenges.

In most cases, business requirements and technology (API, languages, libraries) may change during the product development. The rule of thumb for such complex projects might be using Scrum as a process framework. In general, the success ratio of Scrum projects is higher than in the Waterfall method[8].



Waterfall and Agile, source: The Chaos Manifesto. http://www.standishgroup.com/

Which companies use Scrum?

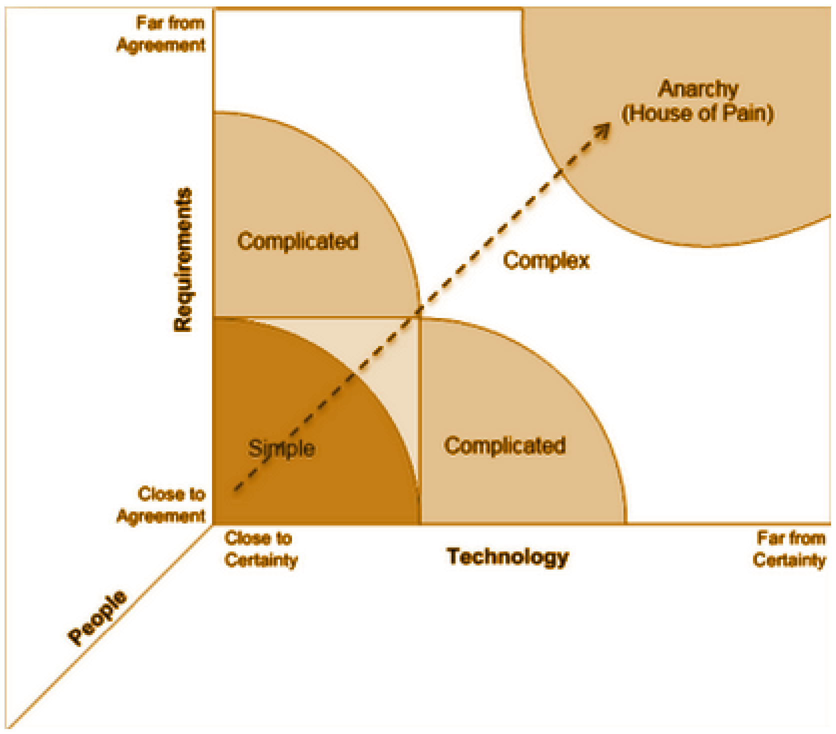
Intel, Microsoft, Amazon, Salesforce, Google, Yahoo, Facebook, Adobe, Nokia, Siemens, BBC, CNN, General Electric, Bank of America, Novell, Unisys, and many more all over the world.

[8] We are showing this example, but the outcome of cases above depends on research methodology and we encourage readers to look into the matter.

4

How to choose the right team?

The chart below shows the correlation among technology, requirements, and people building a software product. Based on their nature, software projects are complex, rarely complicated. But if a good team is found, ‘complex’ might become ‘complicated’, because most of the times technology is not certain, requirements are not certain, and people are changing. And a good team means a permanent one. They’ve had the time to develop communication patterns. They can provide reasonable increment every single time, in every single Sprint. This can lower the ‘chaos factor’ and a complex software project suddenly becomes just complicated. A great team can build the software from the bottom up[9].



Stacey Graph, source
http://pm.stackexchange.com/questions/389/when-to-use-waterfall-when-to-use-scrum

[9] http://bit.ly/1EsoWoC

Here are some key factors worth considering when looking for the right Scrum Team:

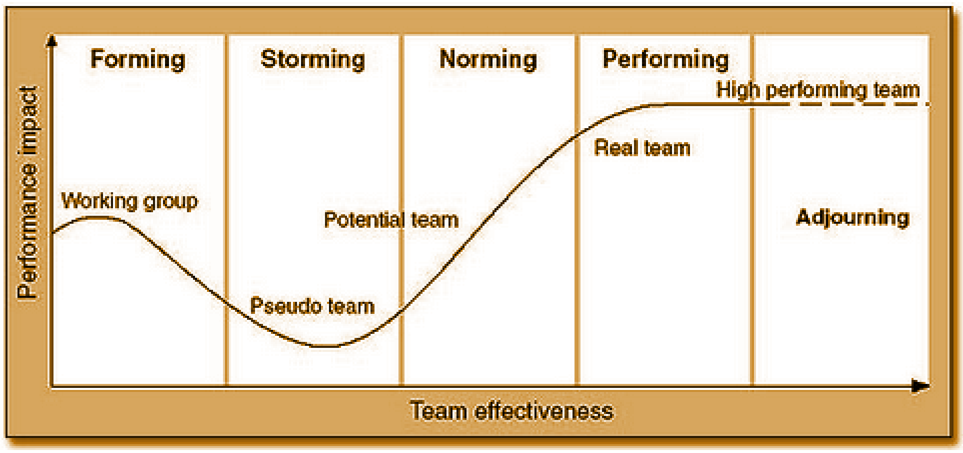
- Knowing and accepting the complicated nature of technology and possible changes in the requirements regarding the product, the team is capable of addressing challenges along the way properly. Can they prove it by showing a number of completed, successful projects? How would they approach a problem if the business needed to change; when would they introduce the change? What is their standard Definition of Done?

- How responsive is the team and to what level does it welcome the changes to the vision, direction, and the code itself? Do they treat email/chat/video communication as a part of the job, or as a distractor from programming? Do they welcome changes during a Sprint and if so, would they later blame you for not delivering the software? Who do they think is able to change requirements besides Product Owner?

- Is the team focused enough on a specific technology? What is their technology stack of choice? How long have they used it? What is their quality process? Can they name the biggest downsides of the development stack? How do they assess if the given technology is able to solve a business problem efficiently? What is their most remembered problem when it comes to the technology, and how did they deal with it?

- Are the members tight together and have they got cohesive skills, or are they a bunch of individuals? Do they understand consequences of the changes in the code provided by themselves and the team[10]? How long have they worked with each other? How many projects have they left behind finished? In Agile, understanding each other is the key. Can the outsourcing company tell you whether the team assigned to the project will be a project team (the one that is built for the project) or a permanent team? If it is a permanent team, at which stage of teams development is the particular team now?

[10] http://bit.ly/1ajdqk1



Stages of a team development, source:
https://programmentor.wordpress.com/tag/team-development/

- Tuckman's stages development model[11] shows the characteristics of an effective team. In the 'forming' stage, team members get to know each other. In the 'performing' stage, everything is right in the place. If your team is to be formed for the project, you have to be aware that these stages will affect your project and the velocity of development of your Development Team. A properly formed team becomes a High Performing Team and can be 100% more productive than a Working Group (Forming Team).

'A collocated self-organising team is 100% more productive than otherwise' - Boston Consulting Group.

- What percentage of developers have an engineer degree? How many of them can speak English and/or any other popular language? Statistically, developers with higher education and languages are more mature and disciplined, they can bring the software to the end. On the other hand, what are the team members software development years of experience? A reasonable median might be 3 to 6 years.

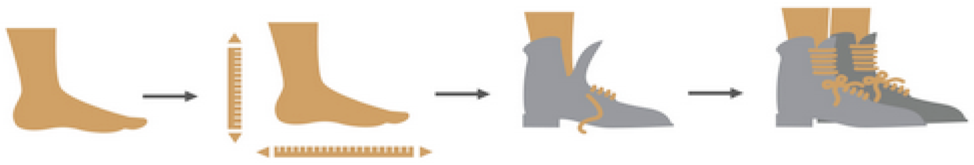
- Do developers run in the same cultural circles as the Partner? Laughing at the same jokes, understanding the same context and background brings people closer. More importantly, the same context helps developers understand the Partners' business[12].

- Price. Lower prices don't mean bad quality, it's just the geographic, cultural, and economic reality. However, unexpectedly good prices often mean a 'too-good-to-be-true' situation and contract. And as a result - the lower quality of the product itself.

[11] http://bit.ly/1emDuLI
[12] http://bit.ly/1MQQaZL

5

The art of cooperation


Bespoke craftsmanship, credits: XSolve

How to cooperate?

When a decision to outsource is made, there is the matter of how to work with a subcontractor. The goal is to make your project proceed in the right way. **Cooperating with the Partner benefits from understanding and trusting each other and not hiding problems**. Properly built cooperation is an important factor in achieving dedicated business goals, so it's important to find the right way of cooperation.

Product Owner might be the incarnation of the Partner or may work on the developers' team side. Product Owner is like the Partner's eye, ear, and hand. Thus, the Product Owner's principal task is to take care of the business like it was their own. That's why this is the first person with whom the Partner needs to build a relationship wisely.

If it is possible (and it should be), the Partner should meet Product Owner face to face, it would allow the PO to better represent the Partner's point of view. However, even if for some reason it's impossible to meet directly with the PO, there are many ways/tools of constant communication with the PO and the entire team.

A good opportunity to meet the team and to familiarise with them is **Project Workshop**[13].

Project Workshop is the analysis of the Partner's requirements, vision of the product, its needs and expectations. Does it have an impact on the business value? Of course – the better the team understands the Partner's needs and requirements, the greater is the chance for the realisation of

[13] http://bit.ly/1FgxbFh

business objectives. **There's no doubt that well thought software brings the desirable benefits**. The first step is to find the right way of cooperation.

Software development is a complex and multistage process extended in time. The process binds the cooperation of many people that have to go in one direction. Therefore, all participants must have a sense of community. If you let one another to know your goals, it will be a successful and satisfying journey for all of you.

Cultural fit

We must keep in mind that the team is not a set of robots, but a bunch of real people. **A team is more than the sum of its members. The Partner should treat the team as confident co-worker**s. Most of programmers are very ambitious, and if the Partner gains their reliance, they'll do what they can best and with high commitment. Cooperation should get smoothly through every Sprint and finally they should become well-oiled (moisturised) machinery that works on high speed with high quality. In a few Sprints they are going to work almost without any needless words. But most often it takes some time to upgrade on this level of cooperation. Obviously, it's easier to work together if the Partner and the provider are culturally aligned to each other.

Some of these aspects of cooperation matter a lot and must be highlighted: for example, fluent English and common culture. Insight in the technical and non-technical universum has its importance as well. Do the Partner and the developer have the same ground, laugh at same jokes, can they tackle challenges from the same perspective?

How to gain trust?

As in any business arrangement, outsourcing is risky, and here it's a risk for both parties. The process of building dedicated software is quite complicated and multistage, thus the trust means the desired values. Trust requires a lot of effort and commitment from both sides.

Very often at the early stage a business project doesn't have any specific requirements. Building software is like a journey to a destination you might know, but most often you meet unexpected obstacles and barriers, and sometimes there's a need to change or to choose another target. Business environment is dynamic, so it's worth building a team able to adapt flexibly to upcoming challenges. Moreover, a committed team is ready to give advice, not only of technical, but also of business nature. **Flexible project management is one of the most useful and advantageous attributes of the Agile approach**.