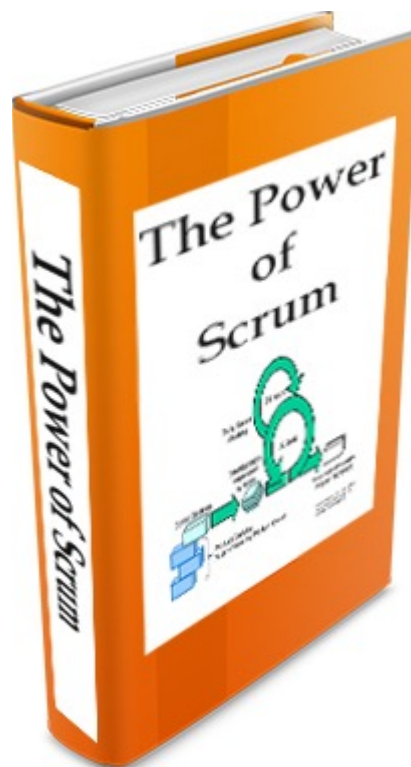


The Power of Scrum

Learn The Number One Reason Why Projects
Consistently **Deliver On Time** and Watch Your
Organisation's **Productivity Explode!**



By Paul VII

© 2012 by Paul VII

Published by

PASHUN PUBLISHING

www.pashunconsulting.co.uk

Introduction

Congratulations on downloading my free scrum

ebook. I will not waste any time and I will start giving you the facts. On the entry page where you downloaded this book, I said I would tell you the number one reason why projects consistently deliver on time and that you would watch your productivity explode if you used the method. In a sentence:

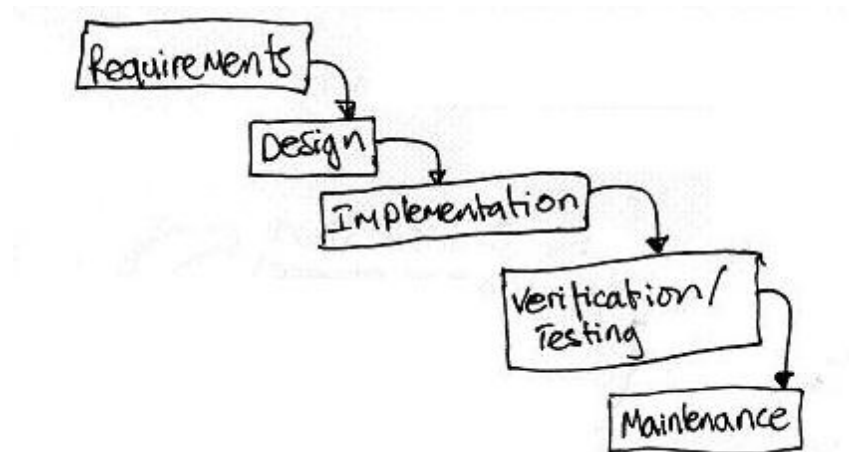
“People who deliver successful projects are sticking to a simple set of proven rules for project success!”

If there is anything I ask you to take away from this, it is the three words. Simple, Proven, and Success.

Now of course I am going to give you a lot more information than just these three words. Scrum is the exact framework that meets this criteria but knowing exactly how to run a Scrum project and stick to the rules is the difficult part.

The principles of Scrum are simple to understand, but the real challenge comes when helping others to understand and trust such a simple framework to deliver quality products, maximise return on investment and still be fun at the same time (yes, really!). So how can a framework be so simple, yet so powerful? How can it save failing projects and carve structure out of chaotic projects, even in complex blue chip companies? Well let's get straight into the next chapter and all will become clear. Then I will tell you how to get more information to put this knowledge to practice. Enjoy!

Chapter 1. The World Before Agile and Scrum



The Waterfall Model

Although the scrum framework can be used to deliver any type of project, it is most often used to manage software development projects. It is well-known that many such projects use what is known as the waterfall model to organise and deliver. This process consists of upfront contiguous phases. The requirements (or analysis) phase in which an analyst will usually gather the requirements for the product, the design phase in which the code and other artefacts are planned or modelled, the implementation phase where the designs are put into practice to build the product, the testing phase where testers make sure that the product meets the requirements to a high degree of quality and then finally the product is released to the public. After the product is released there is on-going support and maintenance in a live environment which can also be considered a phase (albeit a continuous one).

As the diagram shows, this process flows neatly from one phase to another like a waterfall (and believe it or not, that's where the name "waterfall" came from). The waterfall model has been seen as an industry standard process for running software projects for decades and on first glance it makes perfect sense. However, there are some fundamental flaws with this method.

If the requirements change after the requirements phase, then this has a knock on effect to the other phases. Therefore, the launch date becomes more difficult to hit. On top of that, the bulk of the defects and issues are not usually found until the test phase. This often delays launch, since more time is needed to fix bugs. How could you ever forecast exactly how many defects you will find? You cannot exactly, and so this situation usually leads to overtime, low moral and a last-minute scramble towards the end of the project. It is possible that a piece of software sitting on your tablet, phone or computer, has also been developed this way. Don't you feel sorry for the teams?

Most (if not all) people who work on software projects would agree that requirements rarely (if ever) remain fixed for the long-term. This means that either the process is interrupted to add requirements (causing annoyance and delays to the team), or that the business agree not to make any requirements changes. This is not an amiable solution, since change is often a reaction to market conditions and is often a good thing. Changing requirements can increase the business' return on investment, which usually has a good knock on effect for the company and employees. In practice, the former situation is usually the case, as businesses will usually aim to change requirements and worry about the problems later. However, neither situation is good, and in the end everybody involved in the project wants the best possible outcome. It builds everyone's self-esteem to see quality work in the public domain as opposed to a "software nightmare".

On top of these factors, the waterfall method has to deal with other obstacles (as any development method would) such as unclear requirements, unrealistic deadlines and inaccurate estimates. Even the number and calibre of people working on the project can change. A common culprit is known as "people pinching". Skilled team members can start off working on Project A, then due to either new priorities or "gentle pressure" from other project

managers, key people are poached by Project B. This reduces the ability to deliver Project A on time or to the scope agreed. In business, things can change rapidly, so change management is always key to success.

After decades of projects being run using the waterfall model, it was clear that many companies were facing these issues and some changes were needed to manage change keep projects running smoothly.

In scrum, these (and other) issues are known as impediments or blockers. Collectively, these blockers can cause chaos on projects if allowed to, and make the experience “less than a pleasure” shall we say.

Recognising these blockers, a group of thought leaders joined forces to create new, iterative, “agile” methods of working. One such method was scrum.

The Birth of Agile

The term “agile” is one that is often used and misused in the software development industry. Given that agile is so closely related to scrum, let us nail down exactly what agile means and how it is relevant in the context of scrum.

By the end of the 1990s there was a broad consensus of ‘thought leaders’ who recognised the short falls of waterfall (no pun intended) software development. Many of them founded their own new ‘iterative’ methods of software development. Iterative development is fundamentally different from waterfall. As opposed to upfront phases with lots of upfront requirements gathering, iterative methods contain mini phases of requirements, design, implementation, testing and delivery within a number of weeks. This allows the business to release a few features early and make some return on investment. They also get to discover potential issues early and change requirements far more often. Working in iterations also allows the project to react to “people pinching” through periodic re-planning.

Many of these iterative methods were also lightweight, since their founders believed in performing the simplest task possible to solve any given problem. Contrary to popular belief, before the term agile was coined there were already many such methods (of which scrum was one) such as XP, DSDM, crystal and FDD.

In the year 2000, seeing the range of iterative and lightweight methods, a group of industry thought leaders named the Object Mentor Group, called a meeting at Snowbird Ski Resort in Utah. In short, each invitee agreed on a consensus of principles that were common to all of them. This consensus was named the Agile Manifesto and reads as follows:

Manifesto for Agile Software Development

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

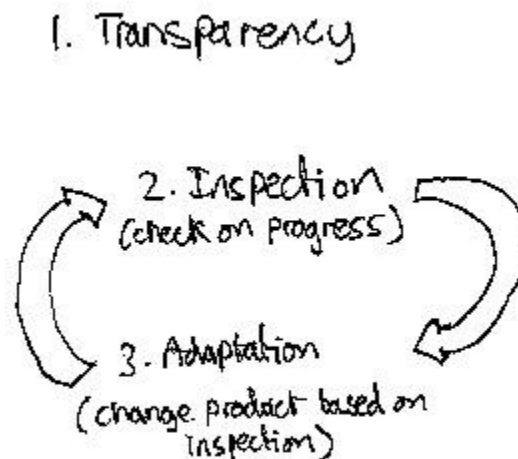
That is, while there is value in the items on the right, we value the items on the left more.

This manifesto was accompanied by a set of principles, agreed to by all. The detail of each of these principles is beyond the scope of this book, but needless to say, through this agreement the term agile was born.

In summary, agile is not an alternative to scrum, but an umbrella term for a set of methodologies and frameworks that share a manifesto and a set of principles. Scrum is one such framework.

Chapter 2. Introducing Scrum

In Ken Schwaber and Jeff Sutherland's (two of the original founders of scrum) Scrum Guide, they describe scrum as *"a framework for developing and sustaining complex products"*. Scrum consists of self-organising, cross-functional teams.



Simply put, this means that the teams consist of a group of people who each have different areas of expertise but work together for the same outcome. A project manager does not control them, since their expertise empowers them to make decisions collectively.

The teams work in iterations, which allows the business the flexibility to change their requirements but still gives the development team the certainty it needs to deliver a working piece of the product. This is one key thing that makes scrum powerful.

Scrum takes its name from the analogy to rugby where a team work together in a chaotic environment to keep control of a ball. This can be compared to a team working together in a chaotic environment to keep control of a project.

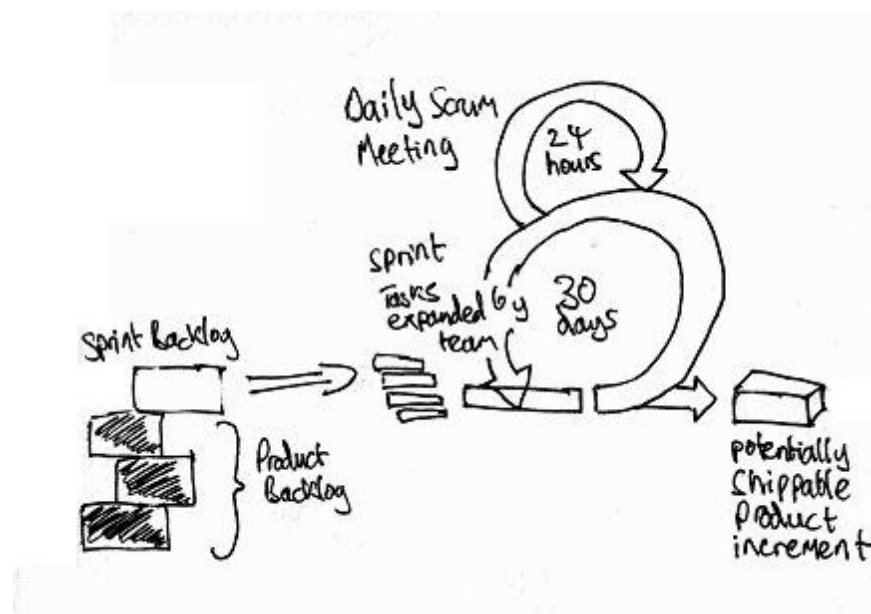
Scrum Theory

"History repeats itself, unless you do something about it!"

Scrum is based on empirical process control theory. The idea is very simple so do not let the name worry you. It consists of three principles: transparency, inspection and adaptation. The idea is that the scrum team, agree to be transparent (honest) in all that they do on the project.

Being transparent means that functionality is not 'done' until it meets the development team's definition of done. Transparency builds trust between the team members. Once the team have agreed on transparency, they agree to consistently check up on progress (inspection) and make improvements based on what they have seen (adaptation). These can be improvements in practices, sticking to values, communication or otherwise. This is powerful stuff in industry, the ability to consistently inspect and adapt. In that way they are improving time and time again before, during and after the release of a product. This is something that was not possible with the waterfall model of development.

The scrum skeleton is a very quick and easy way to explain the process to someone, so I will use it to explain



the process to you.

On the left side of the skeleton, we see the product backlog, which is nothing more than a list of all the features (and their acceptance criteria) that the business desires for the product. A subset of that backlog, called the sprint backlog is taken on by the team, broken down

into tasks, and worked on in an iteration called a sprint. A sprint is a period of time less than thirty days in length and in that time, the team work on their tasks until they develop a working increment of the product.

Remember those mini phases of the waterfall I described earlier? Well this is where it all takes place. There is some requirements gathering and specification update before the sprint, then design, implementation and testing. Above the large sprint circle, you will see a smaller circle. This represents the fact that every day the team meet to inspect on progress and adapt their plan for the day in a daily scrum meeting. At the end of a sprint, the potentially shippable increment of the product is delivered. The business can review the increment in a sprint review and then release the new feature(s) to the world if they so wish.

The team then discuss (transparently) their progress during the sprint in a sprint retrospective (inspect) so they can improve (adapt) on things that need improvement or retain things that are going well. The cycle then begins again and repeats until the product owner has nothing more to add to the product backlog.

The scrum skeleton demonstrates the simplicity and power of scrum as a mini factory, churning out shippable features each sprint.

Scrum Team Roles

Scrum simplifies projects down to only three roles. Remember? One of the benefits of this framework is keeping things simple. The three roles are:

- The scrum master
- The product owner
- The development team

These three roles form the scrum team.

The Scrum Master

The scrum master's purpose is to understand the scrum

rules and practices, remove any impediments or blockers to the team delivering and to help the team to understand how to self organise and work in a scrum manner. The scrum master facilitates for the scrum team wherever it makes sense to do so. The scrum master is your go-to guy in terms of how the scrum framework should operate, and this applies to anyone in the organization.

The scrum master usually understands how to aid the product owner in maximising return on investment from the business and he helps the team to work together to be as productive as humanly possible and deliver a shippable increment of the product.

The Product Owner

The product owner is responsible for creating requirements on behalf of the business. He prioritises based on business needs and is responsible for managing the product backlog, which is the list of all the features that the business requires in the finished product. The product owner is responsible for making decisions that maximise return on investment and for making priority calls or trade-offs to maximise the product's value.

The Development Team

The team are responsible for building a potentially shippable product increment in each sprint. Scrum is clear that there are only three roles in the scrum team. It does not go into the specifics of all of the different possible knowledge experts within the development team, because the idea is that if push came to shove, team members would collaborate to perform tasks outside of their role to deliver the product.

The development team are self-organising and collaborative, as well as skilled in whatever is needed to deliver the project. For example, in a typical technical project you might have developers, graphic designers, and user experience specialists working together in a sprint to

create a product increment.

One key difference between scrum and many other frameworks is that the development team are explicitly experts in their field as opposed to controlled resources. They look to the scrum master for coaching, guidance and the removal of impediments. They look to the product owner for clear requirements, prioritisation and trade off calls.

Development Team Size

One important fact that is often overlooked, is the optimum size of the team. Scrum teams are usually small because it helps them to be more cohesive, and communicate efficiently. The optimum size is between three and nine). This is not a number that just materialised over a chat in the local bar. It is based on experience of thought leaders who have been doing team based work for years. From my experience, having tried and tested these team sizes, I can stand by them as numbers that create highly productive teams. However, as you know, there is no substitute for common sense in these cases.

Time-boxes (Events) and Rules

You may not be familiar with the term 'time-box'. A time-box is a period of time dedicated to a specific event in scrum. In the new scrum guide, the term time-box has been renamed 'event', but as the term 'time-box' is prevalent at this time, I will continue to use it.

Part of the scrum master's role is to carry out time management very strictly. This means beginning and ending meetings and sprints on time and helps the team to maximise their productivity.

Scrum has a number of time boxes and I will outline them briefly, as there is far more detail on them in the remainder of this book. It is the scrum master's role to organise and facilitate all of these events:

Sprint

A sprint is a period of time less than four weeks in length, during which the team build a shippable increment of the product.

Sprint Planning Meeting

The team plan the work that they will do in the upcoming sprint. The meeting lasts no more than four hours for a two-week sprint. There are two halves to the meeting, the “what” and the “how”. In the first half of the meeting, the product owner presents the list of features that he would like the team to deliver from the product backlog. He explains them and the team ask questions. Eventually they pick the features they believe they can commit to in the sprint. In the second half of the meeting, the team break the stories into tasks and estimate them. In this way, they design their work and decide how they will build the product increment. They may adjust and negotiate which stories they can commit to with the product owner but finally they will make a commitment for the sprint.

Daily Scrum

This is a daily meeting, lasting no more than 15 minutes. Usually, one by one, each development team member answers the following questions (asked by the scrum master): What did you do yesterday? What do you aim to do today? do you have any impediments to delivery? The scrum master takes note of any impediments and aims to resolve them as quickly as possible. Anyone else at the daily scrum remains silent so that the meeting can be as productive as possible for the team. Any issues can be discussed afterwards.

The sprint backlog and sprint burn-down should be visible to draw attention to the team's progress or any impediments (see definitions in this chapter).

Sprint Review

This meeting is held at the end of each sprint and allows the team to demo the increment of the product to the product owner and stakeholders. The stakeholders ask questions and make suggestions to the product owner. The product owner makes notes to adapt the backlog if necessary based on suggestions or the output from the demo.

Sprint Retrospective

This is a meeting held after the review and before the next sprint. One by one, each team member answers the questions: “What worked in this Sprint?” and “What could be improved in the next sprint?”. This is a chance for the team to inspect and adapt. It generates continuous improvement.

Each of these events has a specific purpose and these are the only set of events that scrum defines in order to deliver a project.

Release Planning Meeting

The release planning meeting is mentioned in the first revision of The Scrum Guide. The purpose of this, is for the product owner to present a subset of features from the backlog and the team to agree what looks feasible to deliver in terms of scope or a release date. This, however, is not a commitment. Usually, after three to four sprints the team set a velocity (pace at which they work). This can be used to calculate how many features are likely to be completed by the release date (for date driven projects), or when the scope is likely to be delivered (for scope driven projects). See my section on release planning for more information.

Artefacts

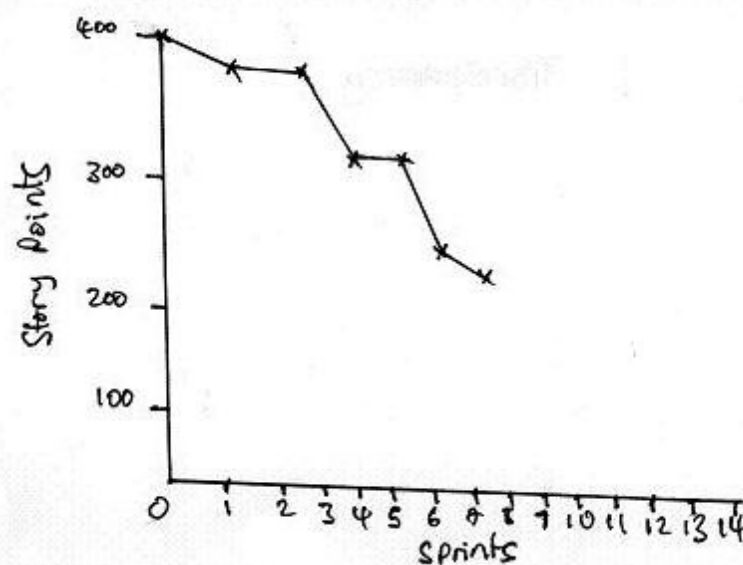
Product Backlog

The product backlog is a list of all the features that the

product owner would like to see in the finished product. This list constantly evolves and changes over time. The product owner maintains the backlog and works with the business stakeholders to form requirements. He also works with the team to get suggestions, technical input and estimates.

Since a product backlog contains features that apply to the lifetime of the whole product (as opposed to the release), a number of features that the product owner would like to release is referred to as a release backlog.

Monitoring the Progress of a Release – The Release Burn down



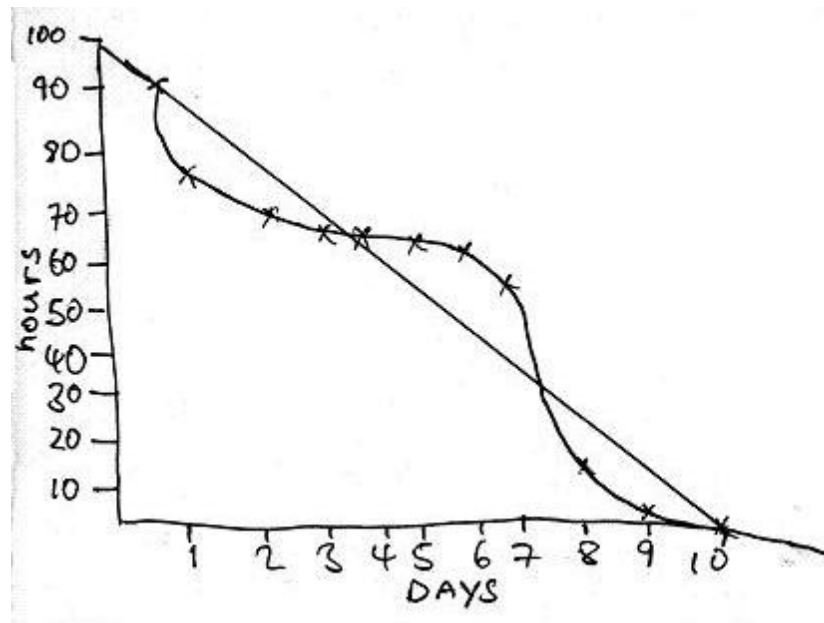
The release burn down is a common method of monitoring progress towards the release of a product. It shows the number of story points and sprints remaining till launch. Simply put, the burn-down makes it easy to see if a project is on track, since a line (or chart) tracks progress and if all is on track the line will be on or very close to its diagonal guide line.

Sprint Backlog

The sprint backlog is the set of items that the development team will work on in a sprint to deliver an increment of

functionality. It is a selection from the product backlog, initially picked by the product owner but finally committed to by the development team. It consists of features, tasks and their estimates.

Monitoring the Progress of a Sprint - Sprint Burn down



As with the release burn down, the sprint burn down helps the scrum team monitor progress within a sprint. The vertical axis usually shows 'hours' or 'number of tasks' and is related to the number of tasks remaining in the sprint Backlog. The horizontal axis shows the number of days in the sprint. A line tracks the team's progress (number of hours of tasks done each day) and ideally this should be a constant number resulting in zero hours of tasks left at the end of a sprint. As with the release burn-down, the line tracking progress should as closely as possible mirror the guiding line.

Shippable Product Increment

This is a piece of functionality delivered by the team at the end of each sprint. It should be potentially shippable and meet the team's definition of 'done' agreed at the start of the project.

Each of these artefacts either has the purpose of helping us to build a product, helping us to track the products in terms of progress or is the actual outcome of the team's work. We will explore them in more detail on the chapters to follow.

Summary

- Scrum allows us to release pieces of the product early by using sprints and getting early ROI
- Mastering the scrum rules is essential to deliver on time
- Retrospectives, Planning and Daily Scrum Meetings dramatically explode productivity
- Self organising teams dramatically reduce management overhead
- High quality is delivered by defining a definition of done, adhering to it in each sprint and reviewing in a sprint review

Congratulations on reading this far! You have read an overview of the basics of scrum. You know why it is so powerful.

So, what is the next step?

The next step is to get a more solid foundation in scrum. If you like the overview you read here. If you are serious about delivering on time and exploding productivity within your team or organisation, or you want to pursue a **certification** in scrum, You will need to get a solid grounding that will allow you to fully grasp the inner workings of each event, tasks of each role and value of each rule. You can get a solid grounding with the answer to the questions:

- How do I kick off a project with scrum?
- How do I carry out the scrum events?
- Who should play the role of Scrum Master, Product Owner to get the most out of the process?

- How do I become a Certified Scrum Master, Team Member or Product Owner?
- How do I master the rules in order to meet the deadlines?
- How do I convince the department or business to use scrum?

You can obtain exactly this by getting **the Udemy Top #10 Best-selling Video Training Package** based on my Amazon Top 10 Bestselling Books - the complete overview Scrum **“Scrum Mega Video Training”**



It is actually **7 online training courses in one package** and also at the time of this document's writing, the online video training edition is being sold for the special price of **\$29.97**. Here's what you get:

Course 1: Agile Scrum Training & Scrum Certification Preparation

Learn the foundation of scrum

Prepare for scrum certification

With emphasis on Scrum Master PSM 1 (value \$199)

[Click to buy \\$29.97 for a limited time](#)

[Click to buy \\$29.97 for a limited time](#)

Course 2: User Stories:

Learn how to write and update agile requirements (value \$50)

[Click to buy \\$29.97 for a limited time](#)

[Click to buy \\$29.97 for a limited time](#)

Course 3: Scrum Step by Step with Examples:

Learn exactly how to roll out scrum in your team or organization from the ground up with examples (value \$199)

[Click to buy \\$29.97 for a limited time](#)

[Click to buy \\$29.97 for a limited time](#)

Course 4: Project Deadline Training:

Learn how the pros deliver projects on time every time (value \$199)

Course 5: Scrum Retrospectives:

Learn how to continuously improve your projects with agile retrospectives / kaizen and how to organize sprint retrospectives for your teams (value \$199)

Course 6: Scrum Master Case Studies (Confessions):

Learn from 7 difficult situations / case studies based on my experience, and master how to improve scrum in your business (value \$199)

Bonus: Course 7: Scrum Advanced Training:

Learn the exact steps I took to get a scrum master job/role

Learn how to coach others with 72 reasons

Learn how to co-ordinate multiple teams using the scrum of scrums technique. (value \$199)

[Click to buy \\$29.97 for a limited time](#)

[Click to buy \\$29.97 for a limited time](#)

As I said, at the time of this document's writing the online video edition is being sold for the special price of **\$29.97**. So access it right now:

[Click to buy \\$29.97 for a limited time](#)

[Click to buy \\$29.97 for a limited time](#)

So I encourage you to take the step while the benefits are still fresh in your mind and I look forward to seeing you through the journey to explosive productivity and projects that deliver on time!

Your sincerely,

Paul VII

Pashun Consulting Ltd.