SYMETRI
ADDNODE GROUP

November 2020
Manufacturing

WHITE PAPER

Approaches to Configuration Systems Development

Overview

Systems and systems development are common across many industries. In engineering we have many examples of processing systems, production systems and manufacturing systems. In computing, systems development is a common phrase and in the CAD environment we commonly involve ourselves in developing or helping to develop configuration systems.

What is a system?

Typically, we can think of a system as a set of components brought together in an organised way to carry out some function. This function will take inputs, carry out some form of manipulation, and produce outputs to achieve a specific purpose.

A system is something that is greater than the sum of its parts. Take a configuration system – we can bring together iLogic rules, Inventor component models and the Inventor environment running on a Windows PC to form our configuration system. Each in isolation will not give us our required output until we configure them to work together to perform our specific task.

Although we are going to use a configuration system as our example, there are many commonalities about how we can engineer any system and the steps and processes involved in doing so.

Characteristics of a good system

A high-quality system that meets the requirements of its users, has particular characteristics and attributes:

- The system must be of use. You would expect that to be a given as we are developing to suit our requirements, however not all systems are developed on demand. There are many off the shelf systems available to use. We should never expect to use all the functionality in an off the shelf system but as a minimum it should have the capabilities in place to meet our needs (and not get distracted by those features we don’t need regardless of how attractive they are).
- The system should be robust and reliable. Although every possible error can never be fully trapped and avoided, there must be a level of error the system must not exceed to be deemed reliable for productive use.
- The system should be able to support change. Change may come during the development of the system, where requirements have been missed in the specification stage, or after the system goes into production use. Opportunities for improvement may be identified only once the system becomes operational. In a configuration system it is possible that new options become available for use, or components updated, that then require a system change to accommodate. A system that does not allow a sufficient level of flexibility may result in hindering rather than assisting innovation.
- A system must be usable. The users of the system must be able to operate it with their level of skill and knowledge. Often this is where our interface to the system is crucial for users to be able to use the system satisfactorily.
- The system costs, both for initial development and ongoing change and maintenance, must return sufficient value to be viable.
- The system needs to be accessible and able to be used where and when needed by the users that should have access.

Approaches to Configuration Systems Development

System development

Developing a configuration system has much in common with engineering, where we apply a systematic, quantifiable approach to the development, operation, and maintenance of a product.

- Development is generally a disciplined process which implies project management as a necessary element of the process.
- In common with engineering, our configuration system will have requirements to meet.
- There will be a defined process that we follow to produce the required solution and within that process, distinct identifiable phases.
- Each phase will generate one or more deliverables of some form.
- Within the process there will be quality processes to ensure the system is performing as expected through testing. There will also be verification that standards are being correctly applied, and validation that the system meets the requirements.

Development Processes

A development process is a set of directions that define how a development project should be carried out. Each activity undertakes some clearly defined process, starting with a number of inputs from any preceding activities. On completion of an activity there may be one or more outputs. The order the activities follows is dependent on the process model being followed but within a development cycle there are likely to be four elements:

- **Analysis** – This includes requirements analysis to understand what the system is to do and its functional requirements. Analysis of the domain the system is to be used in will highlight non-functional requirements.
- **Design** – Working out and deciding how we are going to achieve the requirements.
- **Implement** - Carry out the activities required to implement the design.
- **Test** that the implementation matches the design and meets the requirements identified.

Other activities will also take place around these activities, relating to project management and quality management. Maintenance activities will also be carried out as requirements change or errors identified.

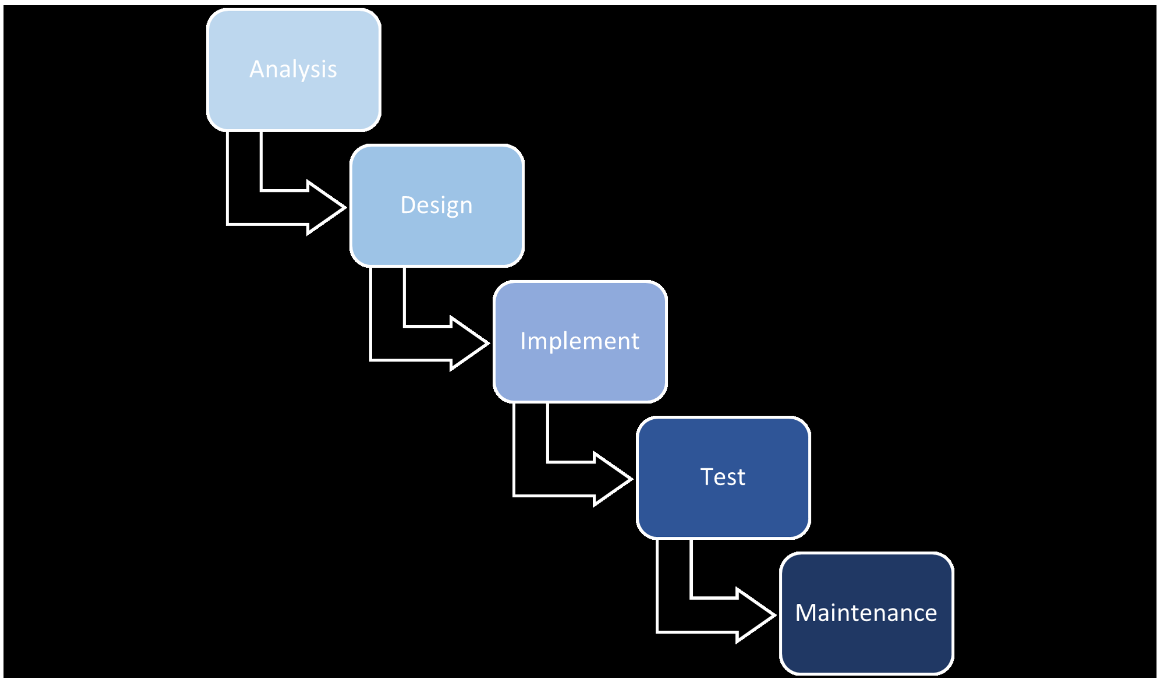
Development Process models

Although the activities are common for any development process, the way that they are applied can differ dependent on a number of factors. Those factors can include the complexity of the system required, the people carrying that out, and how well defined the requirements are.

There are a number of process models that we can apply to the development of a configuration system. We will discuss three of the most common ones.

Waterfall Model

In a waterfall model our process follows a single sequence from analysis through to finished system, with ongoing maintenance as required.



This requires each activity to be carried out correctly in one phase prior to the system going live.

This model may be used when the requirements are reasonably simple or can be very clearly defined with little chance of change. Also, if you're working with outside providers this approach may be the only available option.

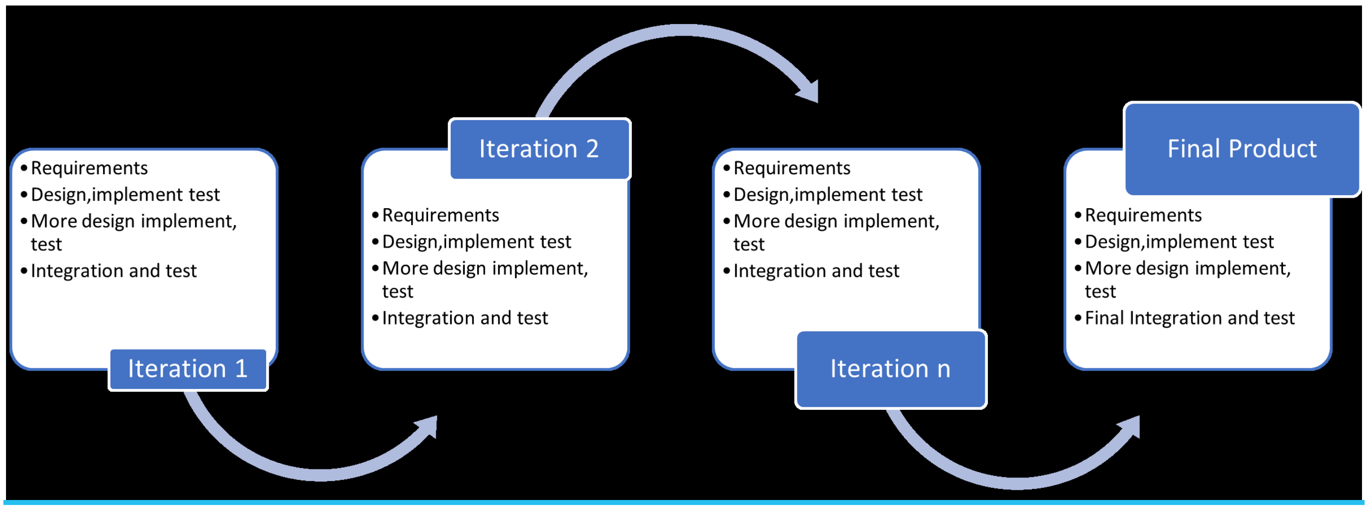
A disadvantage of this process is that a working version of the system is not available until late in the testing activity. Any validation errors identified with the working version of the system may be too late to change.

Incremental iterative process Model

In reality, anything other than a very simple configuration system will be split into several iterations with review stages incorporated in the process.

A method that can be followed is to iterate around subsets of requirements. The outputs from one iteration can be reviewed and feedback obtained, which then influences the next iteration.

Typically, each iteration will be limited to a subset of requirements but also, commonly, each iteration will be time limited.



Visibility at each iteration gives greater possibility to incorporate change based on feedback and incrementally grow the solution system.

It can be difficult to fully define all requirements unequivocally at the start of the process, particularly on more involved systems. With an iterative process we can start with a minimum subset of the requirements as the starting iteration and grow the system with each iteration implementing additional requirements and adjusting based on feedback gained.

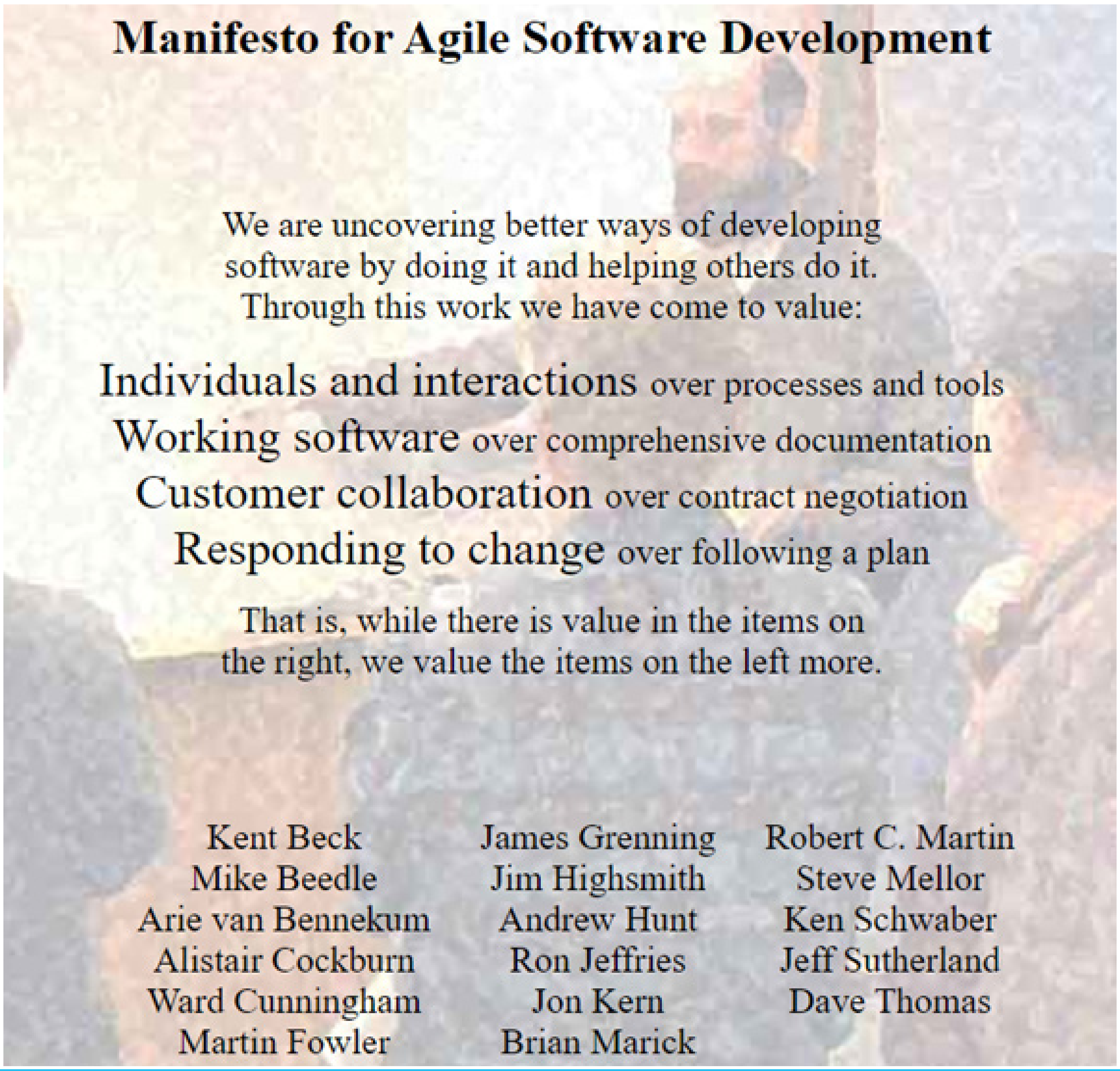
Agile Development

Agile development is an approach that has gained popularity in recent times. It is a catch-all term for a variety of methods that follow practices that promote simpler, lightweight, faster turnaround and more agile development processes that can adapt to changes in requirements.

Originating in software development, agile processes are now applied across many different sectors.

Agile is an approach that puts people and responding to changing requirements at the focus of the process, in opposition to more prescriptive, highly documented, and restrained methods.

Agile development is more than a process model. There was a manifesto created by some of the key protagonists in 2001 to identify the values and principles of agile development:



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.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Principles behind the Agile Manifesto

We follow these principles:

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.

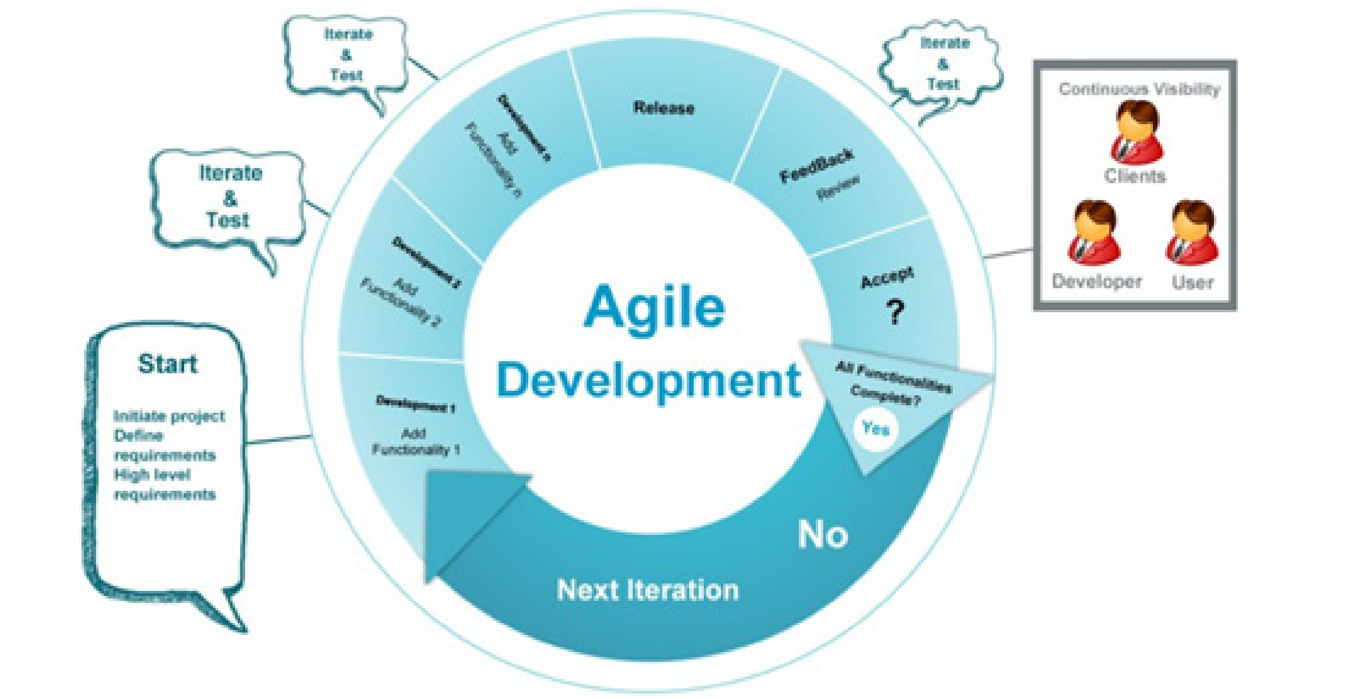
Continuous attention to technical excellence
and good design enhances agility.

Simplicity - the art of maximising the amount
of work not done is essential.

The best architectures, requirements, and designs
emerge from self-organising teams.

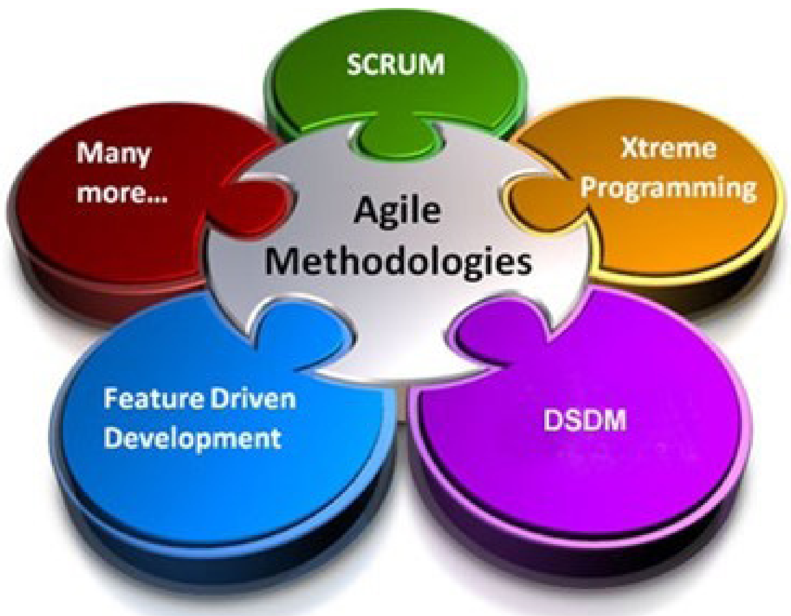
At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behaviour accordingly.

Agile development processes are similar to the iterative development process discussed previously, however, with agile, the emphasis is on the involvement of people in the process, working much more collaboratively across all stakeholders, and applying specific rules to the process depending on the agile methodology being followed. Agile is light in documentation though still involves planning and recording of important information for a level of traceability.An agile approach to software



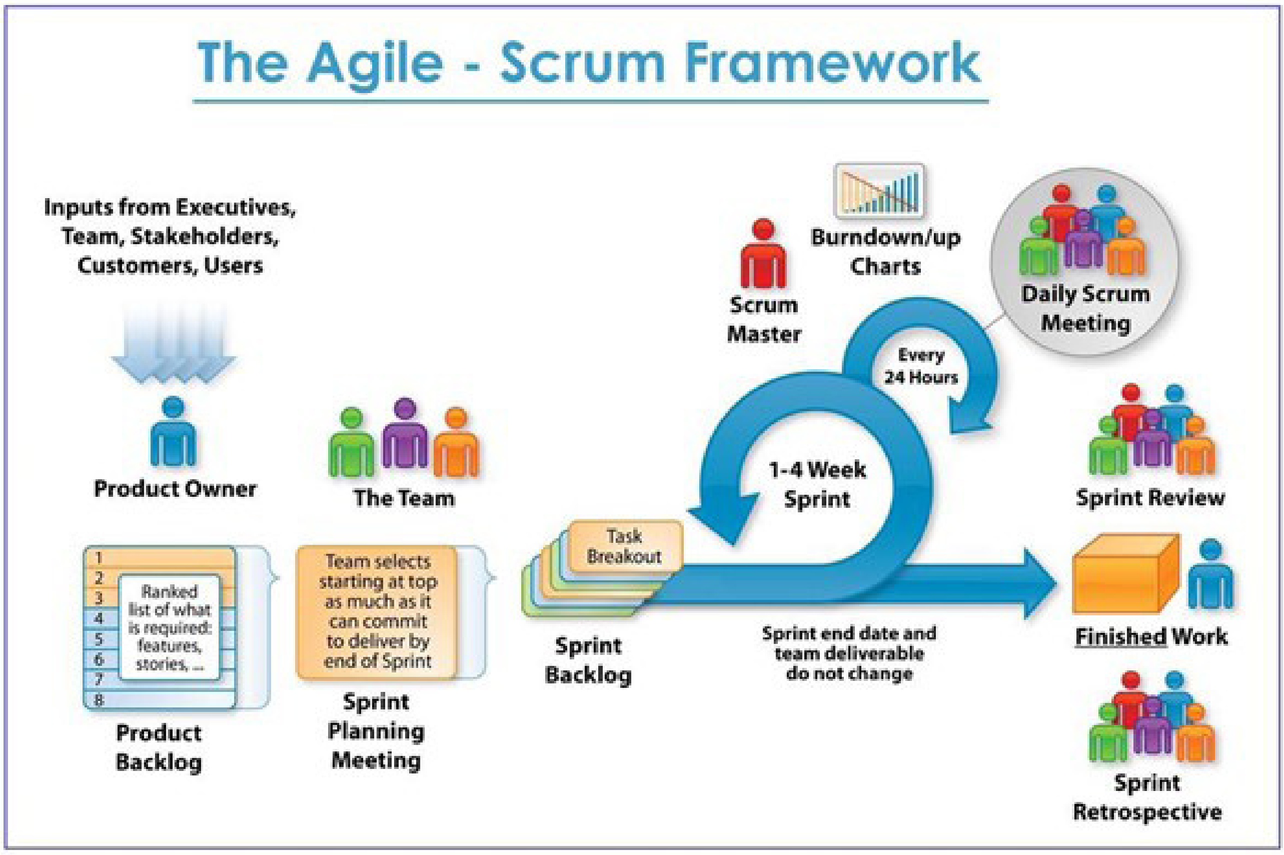
development also puts an emphasis on short iterations and quickly working software, and on the acceptance that systems change. It encourages practices that promote cooperative work.

An agile project has documentation in the form of user stories and tests. A user story describes some functionality required by a user, while a test is an executable form of a user story and therefore directly related to it. Tests are written before coding and are also directly related to the code produced. An agile approach to a requirements document may take the shape of a product backlog (as in Scrum process), which is an ordered, structured list of what needs to be done to a system.



There are various methodologies of agile development each with their own particular rules and methods.

The most widely used agile process is Scrum (software development), fully detailed on Wikipedia.


The Agile - Scrum Framework
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

In Scrum, all events are timeboxed and have well-defined rules:

- **The sprint** – a development phase, no longer than a month, that has, as a deliverable, a useable working increment.
- **The sprint planning meeting** – establishes a sprint goal and the required product backlog items to be covered in that sprint. The length of sprint planning should be in relation to the sprint length with a 2-week sprint having no more than 4 hours for the sprint planning meeting.
- **The daily scrum** – a daily meeting, no longer than 15 minutes, looking at what has been done and planning the work for the next 24 hours. This is also the time to identify and raise any issues that may prevent reaching the sprint goal.
- **Sprint review** - review the work that was completed and the planned work that was not completed. Present the completed work to the stakeholders and collaborate with them on what to work on next.
- **The sprint retrospective** – to reflect on what went well or not so well on the sprint, and what can be improved for future sprints.

When and where to use

“Development today is a race between engineers striving to build bigger and better idiot-proof systems, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning.” – Rich Cook

Because there are such a wide variety of configuration systems, in different environments, with different drivers, goals and stakeholder positions, there isn’t one development process suited to every situation. Different situations will determine what it means to have a system that is useful, usable, reliable, flexible, available and affordable.

Often industry and product will determine what appropriate levels of specification, formality and traceability are required.

Take, for example, a configuration design for a product to be used in the nuclear industry, where components are manufactured by others. It will be expected that this kind of system will need to have a high level of traceability to ensure the system provided is fit for purpose, highly reliable and ensure correct instructions are provided to the manufacturers. In this example there is a need for a formal development process. The process may be slower and more heavily prescripted, however the resulting system must be able to demonstrate that each aspect of the standards the system must adhere to has been incorporated. Depending on the complexity of the product, it may be prudent to create a number of iterations to ensure our developed system is valid for purpose. If the system being implemented is very simple, a waterfall approach may be appropriate with one iteration of design, implement and test. It can also be appropriate for when the system being developed, is well understood by all involved and is unlikely to change during the system development.

Let’s take another example where the product configurator is for a product with a much shorter “shelf-life” and knowledge on what’s required is not fully known or understood, but some form of automation is required for financial viability. If we spend a long time documenting and putting together the perfect ‘plan’ for the ‘perfect’ system, we may have missed our opportunity and our competition may overtake us in our market for that product, or we may even build the wrong system. In this scenario, time to having access to a minimum viable system is more important. By employing an agile approach, we can get started much quicker with collaborative input from all involved to shape the system over short sprints. After the first sprint we would expect some form of resulting system that will begin to give us return and benefit. Each successive sprint will give us incremental benefit while allowing for change and refinement.

The examples here show 2 different scenarios where differing approaches are best suited.

Each process we have discussed has different advantages, disadvantages, and risks. Here are the most obvious:

- **Waterfall**

  - **Pros** – Full traceability and up-front prescription.

  - **Cons** – Late visibility of product.

  - **Risks** – Changes during the process or testing can be accommodated less easily.

  - **Examples of when to use** – Where legal conformance is required, within simple systems, and with well understood requirements with little risk of change.

SYMETRI
ADDNODE GROUP

- **Incremental Iterative**

  - **Pros** – Traceability, review possible earlier in the process, adaptable to change in requirements, modularised delivery elements, and is adaptable to change. It can prioritise requirements and delivery.

  - **Cons** – Variance in time required to accommodate changing requirements makes it more difficult to predict time and cost at outset. There are more administration and management overheads, and it is likely there will be longer processes for reviews and appraisals.

  - **Risks** – Scope creep.

  - **Examples of when to use** – When major requirements are understood at the beginning, working with outside contractors, and when the systems are larger and more complex.

- **agile**

  - **Pros** – Open to change, documentation is light, and can evolve as understanding grows.

  - **Cons** – Requires regular and sustained input from project stakeholders. Scalability may be an issue.

  - **Risks** - Access to required people at required time, dependent on how requirements evolve, may not result in viable product in timescale or dead ends with major code rework, and scope creeps away from the original goals.

  - **Examples of when to use** – Domain knowledge and development knowledge across different groups which are non-transferable, requirements are not clear, and incremental return required across project.

These are just some examples, there are many others.

What is important is to recognise the nature of the system being developed, and the other outside influences in determining which may be the best method to follow. The expected life span of the product should be a big consideration regarding the resource and effort to invest, along with the amount of effort to expend on maintainability.

We have looked at three different approaches here, with agile being a philosophy that is across a number of different methodologies.

One size does not fit all, regardless of how proponents of the different methodologies may argue the case otherwise.

It's also important to recognise that these are not standards that must be strictly followed, so are open to interpretation. Interpret them however best gives you the system you require in the time available and to the level of functionality required.

"The gap between theory and practice is not as wide in theory as it is in practice." – Author Unknown

If you need assistance deciding the best Configuration Systems development processes or help along the way, talk to us about what you are looking to achieve.

Visit our website here: www.symetri.co.uk

References and influences

- CAST (2020) Goals of Software Engineering[Online].Available at https://www.castsoftware.com/glossary/goals-of-software-engineering-best-practices-theory-principles (Accessed 15 September 2020)
- Open University (2014) Approaches to Software Development. (p28,36-37)TM354 Software Engineering – From Domain to Requirements - http://www.open.ac.uk/courses/modules/tm354
- Manifesto for Agile Software Development. [Online]Available at https://agilemanifesto.org/ (Accessed 17 September 2020)
- Principles behind the Agile Manifesto. [Online]Available at https://agilemanifesto.org/principles.html (Accessed 17 September 2020)
- Wikipedia (2020) Scrum (software development) [Online] Available at https://en.wikipedia.org/wiki/Scrum_%28software_development%29 (Accessed 17 September 2020)
- Schwaber, K. and Sutherland, J. (2011) The Scrum Guide [Online]. Available at https://www.scrumguides.org/scrum-guide.html (Accessed 17 September 2020).
- Airbrake (2016) Iterative Model:What is it and When Should You Use it?[Online] Available at https://airbrake.io/blog/sdlc/iterative-model#:~:text=The%20iterative%20model%20is%20a%20particular%20implementation%20of,the%20design%20and%20implementation%20of%20each%20new%20iteration. (Accessed 16 September 2020)
- Issues and Challenges in Scrum Implementation" (PDF). International Journal of Scientific & Engineering Research. 3 (8). August 2012. [Online] Available at https://www.ijser.org/researchpaper/Issues-and-Challenges-in-Scrum-Implementation.pdf

At SYMETRI, We empower our customers with expertise, leading edge technology and services, so they can enjoy the journey from product concept to implementation.

We work with you to tailor digital BIM, product design and lifecycle solutions to help you work smarter and do more with less. That's why our partnerships are long lasting.

With over 30 years' experience, more than 10.000 Customers worldwide we work together as a trusted partner and extension of your team.

Our 450 people strong team work from 27 locations around the world, across vertical industries, delivering a premium service with a global infrastructure and a local presence. We adopt the latest technology and agile methodologies so, even
as technology changes, our relationships last a lifetime.

SYMETRI
ADDNODE GROUP

Symetri
Design Technology Centre
8 Kinetic Crescent
Innova Business Park
Enfield
Middlesex
EN3 7XH

T: +44 (0) 345 370 1444

E: info@excitech.co.uk

W: www.symetri.co.uk