Heavy vs Light Methods

Heavy Process

- Predictive making it difficult to respond to change.
- Elaborate, long-term, detailed planning
- Disciplined, detailed process
- Lots of documentation produced in a bureaucratic environment
- Process oriented

Light(Agile) Process

- Adaptive – embrace change.
- Iterative and incremental.
- Fast cycle/frequent delivery.
- Tacit knowledge – project knowledge in the participants' heads rather than in documents.
- People oriented.

Predictive vs Adaptive

- A Predictive process attempts to plan and predict the activities and resource (people) allocations in detail over a long period of time, -the waterfall model.
- This works well until things change so their nature is to resist change.
- An adaptive process accepts change as an inevitable driver and encourages flexible adaptation

Agile methods

- Dissatisfaction with the overheads involved in design methods led to the creation of agile methods. These methods:
  - Focus on the code rather than the design;
  - Are based on an iterative approach to software development;
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- Agile methods are probably best suited to small/medium-sized business systems or PC products.

What is an Agile Process?

- An agile software process addresses three key assumptions about the majority of software projects:
  - It is difficult to predict in advance which software requirements will persist and which will change. In addition how will customer priorities change as the project progresses?
  - For many types of software design and construction are interleaved.
  - Analysis, design, construction and testing are not as predictable (from a planning point of view) as we would like.

What is an Agile Process?

- An agile process must be :
- Adaptable – to changing project and technical conditions.
- Incremental – to allow adaptation to keep pace with change.
- Agile Methodologies
  - XP (Extreme Programming)
  - Scrum
  - Feature-Driven Development
  - DSDM (Dynamic System Development Method)

Agile Methods

- Agile Development
- Values/trade offs of the agile manifesto.
- Principles of the agile manifesto.
- Reasons to go agile.
- Problems with agility.
- How to make agile modeling work.
- Scrum

Agile Methods

- In the '70s and '80s there was a widespread view that the best way to achieve better software was:
  - Through careful project planning.
  - Formalised quality assurance.
  - Use of analysis and design methods supported by case.
  - Controlled rigorous software process
- Necessary because systems were large, long lived systems e.g. government or aerospace:
  - Large teams working for different companies and geographically dispersed.
  - Critical systems.
  - Long life i.e. maintenance.
- Not suited to small/medium business applications.

The Agile Manifesto

- The cost of change grows through the software lifecycle, the question now is not how to stop change early in a project but how to better handle the inevitable changes throughout the lifecycle.
- Software engineers cannot eliminate change, driving down the cost is the only viable strategy.
- While embracing change, quality must be retained.
- Group of industry experts met in 2001 to discuss ways of improving current software processes that were document driven and process heavy. Goal- speed up development and respond to change. The Agile Alliance.

The Agile Manifesto

- Agile Methodologies emphasise the following key values:
  - *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.

Pros & Cons of the Manifesto

Individuals and interactions over processes and tools.

- Pros: capable people can figure out how to accomplish tasks without a need of process.
- Cons: without process productivity gains cannot easily be repeated on future projects.

Working software over comprehensive documentation.

- Pros: without the need to document extensively software can be developed more quickly.
- Cons: harder for people who did not develop the system to maintain it.

Pros & Cons of the Manifesto

Customer collaboration over contract negotiation.

- Pros: can adapt more easily to changing requirements.
- Cons: customer representative may not have adequate knowledge of what is needed by their organisation.

Responding to change over following a plan.

- Pros: adaptability leads to producing what the customer really wants.
- Cons: customer may keep changing the requirements leading to longer delivery time

How the values of the agile manifesto agile methods lead to the accelerated development and deployment of software.

Individual and interactions over processes and tools.

By taking advantages of individual skills and ability and by ensuring that the development team know what each other is doing, the overheads of formal communication and process assurance are avoided. This means that the team can focus on the development of working software.

Working software over comprehensive documentation.

This contributes to accelerated development because time is not spent developing, checking and managing documentation. Rather, the programmer's time is focused on the development and testing of code.

How the values of the agile manifesto agile methods lead to the accelerated development and deployment of software.

Customer collaboration over contract negotiation.

Rather than spending time developing, analysing and negotiating requirements to be included in a system contract, agile developers argue that it is more effective to get feedback from customer's directly during the development about what is required. This allows useful functionality to be developed and delivered earlier than would be possible if contracts were required.

Responding to change over following a plan.

Agile developers argue(rightly) that being responsive to change is more effective than following a plan-based process because change is inevitable whatever process is used. There is significant overhead in changing plans to accommodate change and the inflexibility of a plan means that work may be done that is later discarded.

The Agile Manifesto 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.**
- Business people and developers **work together daily** throughout the project.

The Agile Manifesto Principles

- 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.

The Agile Manifesto Principles

- 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 effective, then tunes and adjusts its behaviour accordingly.

MoSCoW

- The practice of critically reviewing the requirements and prioritising them in the context of what it would take to produce a successful system.
- We **MUST** have these features, the system is of no use without all of this functionality.
- We **SHOULD** have these features, these features will lead to an excellent system but we could succeed without them.
- We **COULD** have these features, if we have more time this functionality would be very useful.
- We **WON'T** have these features, they are not needed and have been dropped.

Practical Problems With Agile Methods

- The informality of agile development is incompatible with the legal approach to contract definition that is commonly used in large companies.
- Agile methods are most appropriate for new software development rather than software maintenance. Yet the majority of software costs in large companies come from maintaining their existing software systems.
- Agile methods are designed for small co-located teams yet much software development now involves worldwide distributed teams.
- Active user involvement.

When to go Agile

- You are working iteratively and incrementally.
- Uncertain or volatile requirements.
- Primary goal is software development.
- Stakeholder support and involvement.
- An AM champion exists.
- Developers are responsible and motivated.
- Adequate resources are available.

http://agilemodelling.com/When Does(n't) Agile Modeling Make Sense?

When not to go Agile

- Large team - over a hundred located in different places.
  - New approaches have been used on a small scale and are difficult to scale up. They have also been created with an emphasis on small teams.
- Critical systems development where a detailed analysis of all of the system requirements is necessary to understand their safety or security implications.
- Fixed price, or more correctly a fixed scope, contract.

What is agile modelling (AM)?

- AM is a practice-based methodology for effective modeling and documentation of software-based systems. Simply put, Agile Modeling (AM) is a collection of values, principles, and practices for modeling software that can be applied on a software development team in an effective and streamlined manner.
- AM is not a prescriptive process, in other words it does not define detailed procedures for how to create a given type of model, instead it provides advice for how to be effective as a modeler.

What is an Agile Model

- Agile models are more effective than traditional models because they are just barely good enough, they don't have to be perfect. A paper prototype could be an agile model, a whiteboard sketch could be an agile model, a Visio diagram could be an agile model, or a physical data model (PDM) created using a CASE tool could be an agile model. What is important is that the model is just good enough for the task at hand.

What are the values of AM?

- **Communication**. Models promote communication between your team and your stakeholders as well as between developers on your team.
- **Simplicity**. It's important that developers understand that models are critical for simplifying both software and the software process"
- **Feedback**. By communicating your ideas through diagrams, you quickly gain feedback, enabling you to act on that advice.
- **Courage**. You need to make important decisions and be able to change direction by either discarding or refactoring your work when some of your decisions prove inadequate.
- **Humility**. Developers recognize that they don't know everything, that their fellow developers, their customers, and in fact all stakeholders also have their own areas of expertise and have value to add. Everyone involved with the team has equal value and therefore should be treated with respect.

Agile Project Management

- The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project.
- The standard approach to project management is plan-driven. Managers draw up a plan for the project showing what should be delivered, when it should be delivered and who will work on the development of the project deliverables.
- Agile project management requires a different approach, which is adapted to incremental development and the particular strengths of agile methods.

Agile Planning

- Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments.
- Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development.
  - The decision on what to include in an increment depends on progress and on the customer's priorities.
- The customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes.

User Story Template

One particular template, often referred to as "As a… I want to… So That…" is the most commonly recommended aids for teams and product owners starting to work with user stories and product backlog items in general:

As a [type of user], I want [an action] so that [a benefit/a value]

User Stories

As a user, I want to reserve a hotel room online so that I am not constrained by opening hours and don't have to make a phone call.

As a user, I want to cancel a reservation online so that I am not constrained by opening hours and don't have to make a phone call.

The Details

As a user, I can cancel a reservation.

- Does the user get a full or partial refund?
- Is the refund to their credit card or is it site credit?

How far ahead must the reservation be cancelled?

- Is that the same for all hotels?
- For all site visitors? Can frequent travellers cancel later?

Is a confirmation provided to the user? How?

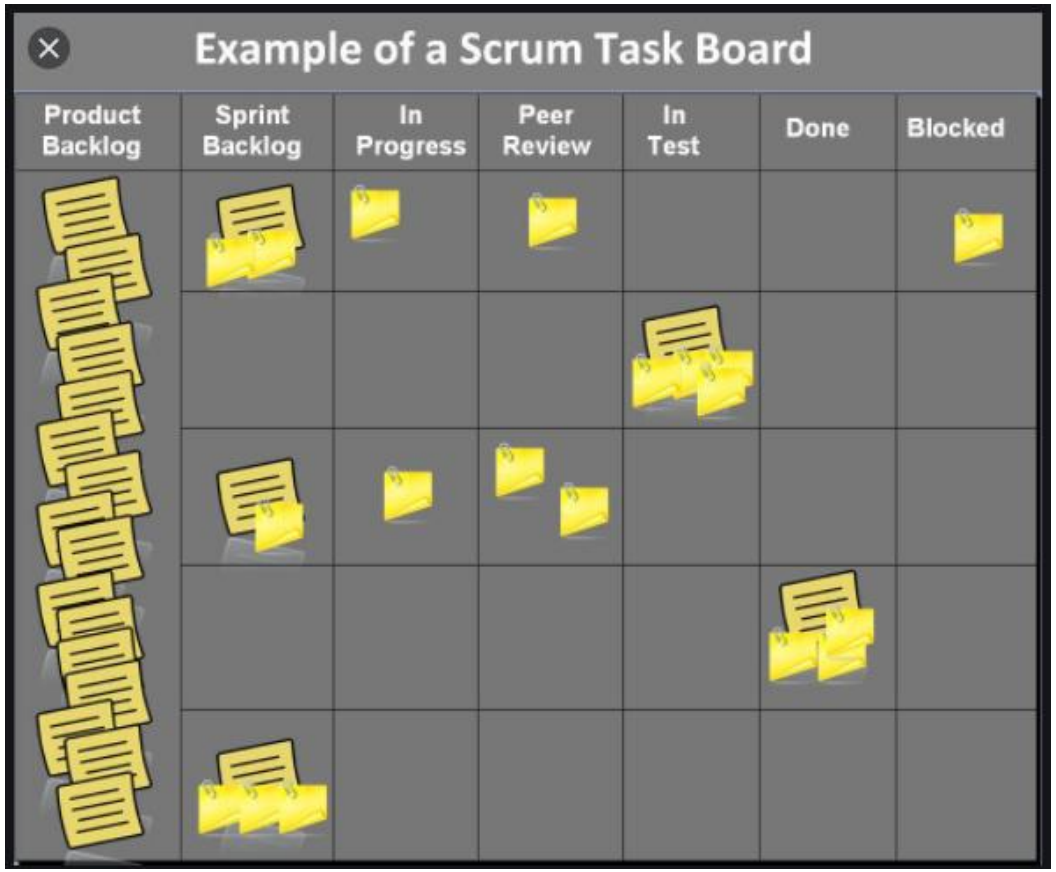
Release and Iteration Planning

- Release planning involves selecting and refining the stories that will reflect the features to be implemented in a release of a system and the order in which the stories should be implemented.
- Stories to be implemented in each iteration are chosen, with the number of stories reflecting the time to deliver an iteration (usually 2 or 3 weeks).
- The team's velocity is used to guide the choice of stories so that they can be delivered within an iteration.

Task Allocation

- During the task planning stage, the developers break down stories into development tasks.
  - A development task should take 4–16 hours.
  - All of the tasks that must be completed to implement all of the stories in that iteration are listed.
  - The individual developers then sign up for the specific tasks that they will implement.
- Benefits of this approach:
  - The whole team gets an overview of the tasks to be completed in an iteration.
  - Developers have a sense of ownership in these tasks and this is likely to motivate them to complete the task.

Information Radiator



Software Delivery

- A software increment is always delivered at the end of each project iteration.
- If the features to be included in the increment cannot be completed in the time allowed, the scope of the work is reduced.
- The delivery schedule is never extended.

Scrum

- Scrum is an agile method that focuses on managing iterative and incremental development rather than specific agile practices.
- There are three phases in Scrum.
  - The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  - This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
  - The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

Scrum Team

Small team size:

- Typically 5 to 9 team members.
- Cross-functional- **have all competencies needed to accomplish the work without depending on others not part of the team.**
- Design, coding, testing, etc.
- Members must have a broad range of skills.
- Every team member is an expert in his/her field but can also take over responsibilities of other team members.

Scrum Team

Self organising - **choose how best to accomplish their work, rather than being directed by others outside the team**

- Decides which requirements to include in next Sprint (i.e., team has power to reject too many requirements).
- Decides independently which tasks to perform to implement the requirements.

Members should

- Work in close distance (ideally in the same room).
- Members should be full-time.

Membership should change only between sprints .

The team model in Scrum is designed to optimize flexibility, creativity, and productivity.

Scrum:Roles

Product Owner

Decides which requirements are implemented for a product version.

Decides about when product increments will be shipped.

Team

Implements requirements.

Decides how many requirements are implemented in a Sprint.

Organizes its activities (-> tasks) independently.

Scrum Master

Takes care of the proper implementation of Scrum .

Supports the team in process-related issues.

Teamwork in Scrum

- The Scrum master is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.
- The whole team attends short daily meetings (Scrums) where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.
  - This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.

Scrum:Backlogs

Product Backlog

Collection of requirements (user stories) for the product – at project start

- a few, detailed user stories; collection evolves over time and requirements will be refined over time

Managed by the **Product Owner**

Sprint Backlog

Collection of requirements (user stories) that are selected for implementation during next sprint

Managed by the **Team**

Scrum:Sprints

Sprint

Period (2-4 weeks) in which a shippable product increment (executable, tested, and documented) is created by the Team

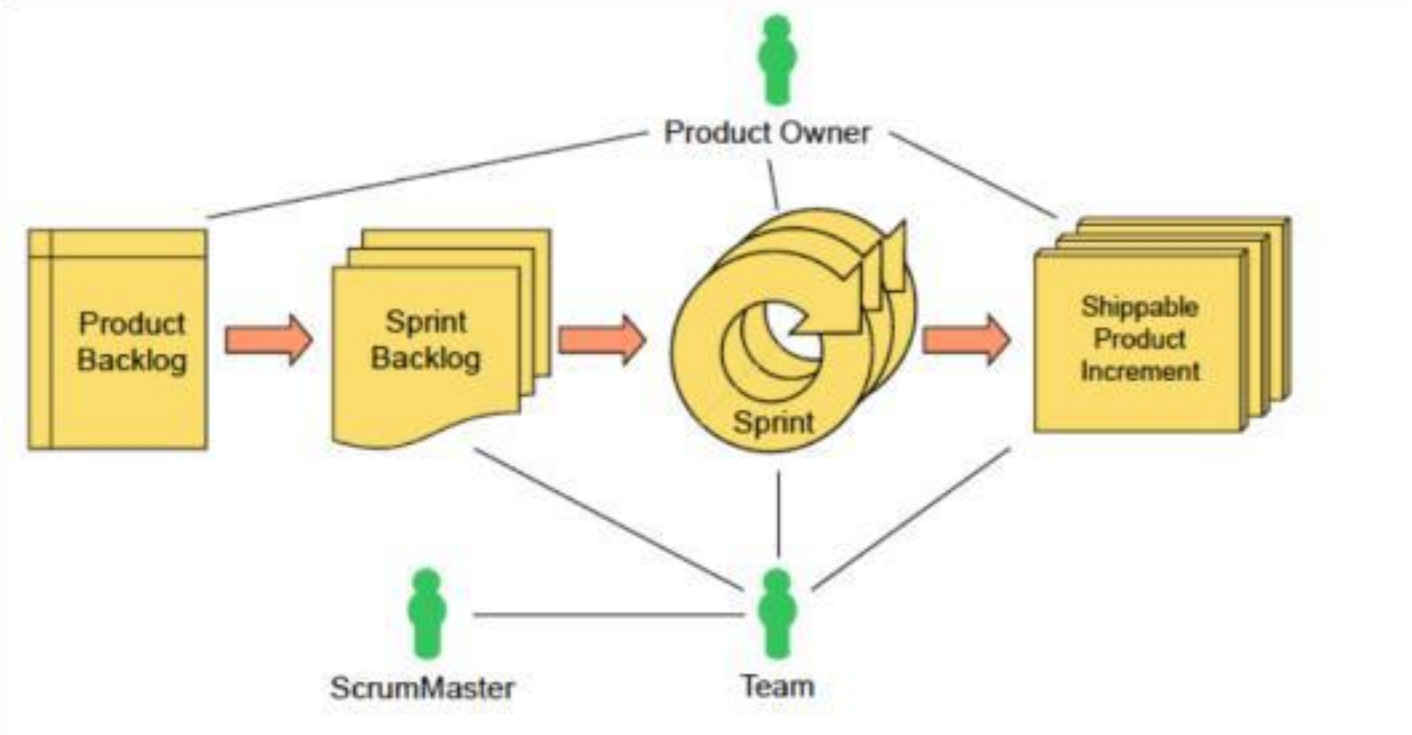
Time-boxed

- i.e. ends exactly at the scheduled time

At end of Sprint the product owner accepts/rejects the final results (i.e., the software)

- Partially completed or incorrect results will not be shipped (no compromise on quality) and go back to the product backlog for inclusion in the next Sprint (backlog)

Scrum Process



The Scrum Sprint Cycle

- The starting point for planning is the product backlog, which is the list of work to be done on the project. During the **assessment** phase of the sprint this is reviewed and priorities and risks assigned. The product owner is closely involved in this process and can introduce new requirements or tasks at the beginning of each sprint.

The Scrum Sprint Cycle

- The **selection** phase involves all of the project team who work with the product owner to select the features and functionality from the product backlog to be developed during the sprint.

The Scrum Sprint Cycle

- Once these are agreed, the team organise themselves to **develop** the software. Short daily meetings (scrums) involving all team members are held to review progress and if necessary reprioritise work. During this stage the team is isolated from external interference, with all communications channelled through the Scrum master. The role of the scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is **reviewed** and presented to the product owner. The next sprint cycle then begins.

Key Practices

Daily Scrum

- A 15-minute time-boxed event for the team. The team plans work for the next day, this optimises team collaboration and performance by inspecting the work since the last scrum and forecasting upcoming work.
- The team uses the scrum to inspect progress toward the sprint goal and to inspect progress toward completing the work in the sprint backlog. The scrum optimises the probability that the team will meet the sprint goal.

Key Practices

- Sprint Review
  - At the end of each sprint the software is demonstrated to the product owner, the goal is to assess the work results and obtain approval. The result is a revised product backlog.
- Directly after the review a retrospective is held:
  - What went wrong?
  - What went well
  - What could be improved and how?

  The goal is to improve team collaboration and the application of Scrum.

Scrum Benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Agile Planning Difficulties

- Agile planning is reliant on customer involvement and availability.This can be difficult to arrange, as customer representatives sometimes have to prioritise other work and are not available for the planning game.
- Furthermore, some customers may be more familiar with traditional project plans and may find it difficult to engage in an agile planning process.

Sources

- Sommerville Software Engineering
- Braude Software Engineering
- www.agilealliance.org
- www.agilemodeling.com