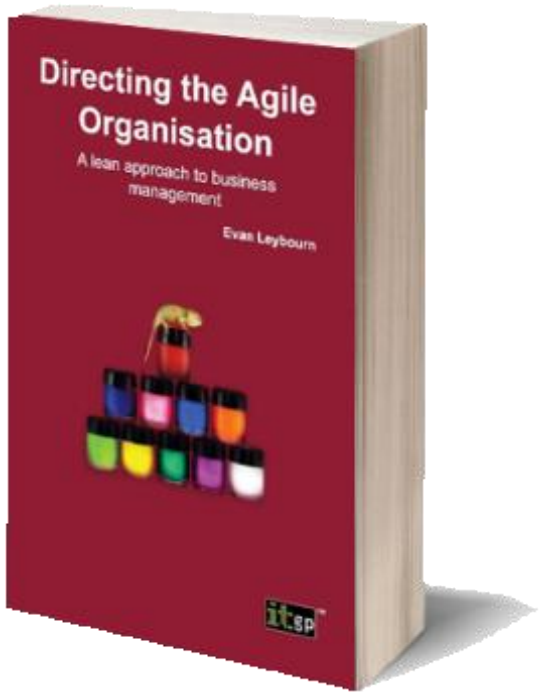
Lean Kanban Practitioner

A Lean Approach to Efficient Workflow Management

Student Guide

Lean, Agile & Kanban Processes for Software Projects by Evan Leybourn is licensed under a Creative Commons Attribution-ShareAlike 3.0 Australia License <http://creativecommons.org/licenses/by-sa/3.0/au/>

Evan Leybourn
evan@theagiledirector.com
Twitter: @eleybourn



OTHER WORKS BY EVAN LEYBOURN

DIRECTING THE AGILE ORGANISATION – BY EVAN LEYBOURN

http://theagiledirector.com/book

- Embrace change and steal a march on your competitors
- Discover the exciting adaptive approach to management
- Become the Agile champion for your organisation

Business systems do not always end up the way that we first plan them. Requirements can change to accommodate a new strategy, a new target or a new competitor. In these circumstances, conventional business management methods often struggle and a different approach is required.

Agile business management is a series of concepts and processes for the day-to-day management of an organisation. As an Agile manager, you need to understand, embody and encourage these concepts. By embracing and shaping change within your organisation you can take advantage of new opportunities and outperform your competition.

Using a combination of first-hand research and in-depth case studies, Directing the Agile Organisation offers a fresh approach to business management, applying Agile processes pioneered In the IT and manufacturing industries.

TABLE OF CONTENTS

WHAT DOES AGILE MEAN?

'On two occasions I have been asked, "Pray, Mr Babbage, if you put into the machine wrong figures, will the right answers come out?" [...] I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.'

Charles Babbage, 1864

Notes:

THE AGILE MANIFESTO

The “Agile Software Development Manifesto” was developed in February 2001, by representatives from many of the fledgling “agile” processes such as Scrum, DSDM, and XP. The manifesto is a set of 4 values and 12 principles that describe “What is meant by Agile".

THE AGILE VALUES

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

THE AGILE PRINCIPLES

1. Our 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 customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time-scale.
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.
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.

Notes:

THE LEAN PRINCIPLES

In addition to the principles from the Agile manifesto, there are 7 principles defined for lean development.

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

AGILE METHODS

The term Agile actually refers to a concept, not a specific methodology. There are many, and sometimes conflicting, methods that can be used under the Agile umbrella. These include;

- Agile Unified Process,
- Behaviour Driven Development (BDD),
- Crystal Clear,
- Dynamic Systems Development Method (DSDM),
- Extreme Programming (XP)
- Feature Driven Development (FDD),
- Kanban
- Lean Development,
- Rapid Application Development (RAD),
- IBM - Rational Unified Process (RUP),
- Scrum,
- Test Driven Development (TDD),

Notes:

KEY POINTS

All of the above methods have four key points in common.

1. Iterative design process
2. Continuous stakeholder engagement
3. Aims for quality and reliable systems
4. Short development cycles (up to a month) allows to regular delivery of improvements

This shows that an agile approach is appropriate in contexts where the outcomes are not known (or can't be known) in advance and where the delivery of the outcomes cannot be fully controlled. This is especially relevant in business intelligence environments given the natural ambiguity around reporting requirements, data quality and information management.

The following figures[1] are an excellent example of the differences between traditional (or phased) development vs. the agile approach of iterative development.

FIGURE 1: THE TRADITIONAL APPROACH (PHASED DELIVERY OF KNOWN OUTPUTS)

FIGURE 2: THE AGILE APPROACH (ITERATIVE DELIVERY TO MEET CHANGING EXPECTATIONS)

[1] Images with thanks from Jeff Patton: http://www.agileproductdesign.com/

Notes:

THE ORIGIN OF LEAN

Lean Manufacturing, often called lean production, or just 'Lean', is a streamlined manufacturing and production technique, as well as a philosophy that aims to reduce production costs, by eliminating all 'wasteful' processes. Put another way, Lean focuses on 'getting the right things to the right place, at the right time, in the right quantity, to achieve perfect workflow'.

Lean Manufacturing provides a set of techniques to identify, and eliminate, waste which, in turn, improves quality, and reduces overall production time and cost. In addition, Lean Manufacturing also improves the 'flow' of production through the system. These techniques include:

- Value stream mapping: Analysing and planning the flow of materials and information that is required in the production process.
- 5S: This is an approach to quality and continuous improvement. The five S's are: Sort (to clean and organise the work area), Set in Order (arrange the work area to ensure easy identification and accessibility), Shine (mess prevention and regular maintenance of the work area), Standardise (create a consistent approach for carrying out production processes), Sustain (maintain the previous four S's through discipline and commitment).
- Kanban: This will be covered later.
- Fail-proofing: Prevent human errors before they occur.
- Production levelling: Ensure that each step in the production process delivers at a constant rate, so that subsequent steps can also deliver at a constant rate. No step in the production process should produce goods at a faster rate than subsequent steps, or consume goods at a slower rate than preceding steps.

Finally, Lean Manufacturing emphasises Kaizen (改善) or Continuous Improvement; the ongoing, incremental and regular technique of improving all processes, products and functions relating to production, resources, organisational management, and the supply chain.

It should be noted at this point that many of the terms in Lean Manufacturing have been translated from the original Japanese. As such, they often lose the context, or secondary meanings, of the term. Where possible, this context is described throughout this course.

Notes:

UNDERSTANDING WASTE

The techniques and frameworks within Agile & Lean aim to increase development efficiency, by eliminating all 'wasteful' processes. Drawing on the successful concepts from the Lean manufacturing frameworks, we can define 3 major forms of waste.

- Mura (Unevenness): Mura exists where there is a variation in workflow, leading to unbalanced situations, most commonly where workflow steps are inconsistent, unbalanced, or without standard procedures.
- Muri (Overburden): Muri exists where management expects unreasonable effort from personnel, material or equipment, most commonly resulting from unrealistic expectations and poor planning.
- Muda (Waste): Muda is any step in the production workflow that does not add direct value to the customer. The original seven wastes, as defined by the Toyota Production System (TPS), were:
    1. Transport,
    2. Inventory,
    3. Motion (moving more than is required),
    4. Waiting,
    5. Overproduction,
    6. Over Processing (from poor design), and
    7. Defects (the effort involved in inspecting for, and fixing, defects).

  Additional and new wastes are not meeting customer demand, and are a waste of unused human talent. There is further differentiation between Type 1 (necessary waste, e.g. government regulations) and Type 2 (unnecessary waste).

Notes:

CRITICAL SUCCESS FACTORS

The successful application of an agile methodology depends on the relative maturity of an organisation in relation to Customer Engagement, Staff Resources, Technology, and Processes. These measures are defined as follows:

- Customer Engagement – Customer Representatives involved in teams' daily activities, defines user stories, drives the prioritisation of stories, and has decision making delegation of authority.
- Staff – have experience in an agile method, are skilled in the Standard Operating Environment (SOE) toolsets, have an understanding of the underlying data and technical infrastructure, and are conversant in the development, testing, and configuration and release procedures.
- Technology – a stable and well documented technology stack, with clearly defined ownership and service levels, providing discreet development, testing and release environments that are sized and supported for the delivery of projects, and controlled through rigorous configuration and release management.
- Processes – business processes exist for all domains, with cross stream interdependencies defined and service levels agreed, and clear business ownership and delegations of authority identified.

COMMON MISCONCEPTIONS

Being a generic term, Agile means different things to different people. Therefore, before we go much further, I should clarify some of the more common misconceptions surrounding Agile.

- Agile is ad hoc, with no process control: First of all, Agile isn't a lack of process. Agile provides a range of formal processes, and methods, to inform work processes, customer engagement and management models. Conversely, Agile isn't about blindly following the prescribed 'agile' methods and processes. Agile is about using your common sense to apply processes, as determined by the current situation, and shaped by the agile philosophy.
- Agile is faster and/or cheaper: Agile isn't significantly faster, or cheaper, than alternative frameworks. Put another way, in most cases you can't get significantly more effort out of your teams by moving to an agile approach. While there is an overall

Notes:

efficiency gain when utilising agile methods, well-managed Agile and non-Agile teams will deliver products and services in approximately the same time and effort.

- Agile teams do not plan their work or write documentation: Agile is not an excuse to avoid appropriate planning or writing documentation. It is an on-demand, or Just-In-Time, approach that encourages continuous planning and documentation, but only when needed for specific stories. This allows customers and teams to determine if the planning, or document, adds value to the process or product. It creates an opportunity to emphasise valuable documents, and eliminate anything that isn't useful.
- An Agile project never ends: While this may be true in some situations, the benefit of Agile is that work will continue while the customer continues to gain business value, and that value is worth more than the cost of developing it. Most projects, in any industry, have a point of diminishing returns. This is the ideal time for an agile project to end.
- Agile only works for small organisations: Agile works for projects, teams and organisations of any size, not just small projects. That is not to say that it will work for all organisations, but size is rarely a factor. Large and complex projects and organisations are often excellent candidates for Agile transformation, where it is difficult, or impossible, to know all your customer's requirements in advance.
- Without upfront planning, Agile is wasteful: This assumes that your customer knows the detail of all of their stories in advance. If this is true, then by all means, undertake comprehensive upfront planning. However, in reality this is rare, and usually leads to the greater 'waste' of having undertaken design and development work that was ultimately unnecessary. Agile Business Management encourages minimal upfront planning, ensuring everyone is working towards the same goal, and reduces the risk of miscommunication.
- Finally, Agile is not the solution to all your problems. It is a change in approach and culture that comes with its own set of benefits and issues.

Notes:

AN OVERVIEW OF KANBAN AS A SOFTWARE DEVELOPMENT METHOD

The original concepts of Kanban (カンバン) were developed in the 1940s and 50s by Taiichi Ohno[2] as part of the Toyota Production System, as a mechanism to control Just-In-Time (JIT) production and manufacturing processes. Kanban, which approximately translates as 'signboard', is described as a 'visual process management system that tells what to produce, when to produce it, and how much to produce'. The modern Kanban method, as formulated by David J Anderson in 2007[3], is an adaption of the original JIT approach, with an emphasis on staff welfare and continuous process improvement practices. This is ultimately a strategy that strives to improve a business return on investment by reducing waiting inventory and associated carrying costs.

At its simplest, each prioritised task (or card) on a Kanban Board passes through a visualisation of the team's process, or workflow, as they happen. Each primary activity in the team's workflow is visualised as columns on the Kanban Board, usually starting at task definition, and finishing with delivery to the customer. Of course, being Agile, these cards and activities are visible to all participants, including the customer.

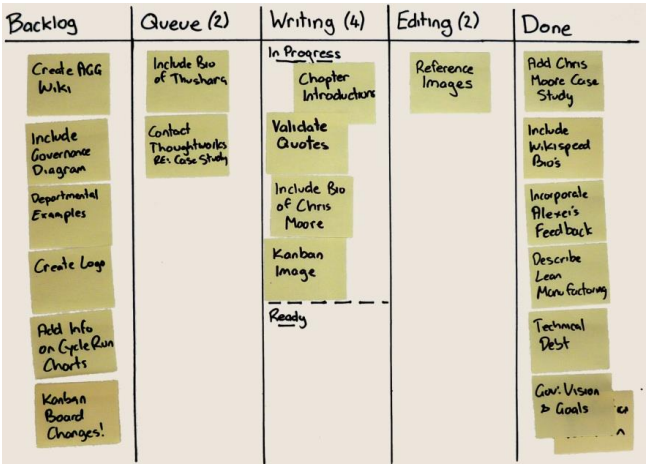


FIGURE 3: EXAMPLE KANBAN BOARD

[2] Toyota Production System: Beyond Large-Scale Production, Ohno (1988).
[3] Kanban: Successful Evolutionary Change for Your Technology Business, Anderson (2010).

Notes:

While a Kanban workflow can become very complex, the simplest visualisation of workflow has only 4 different states which a task would progress through during its lifecycle. These states are;

1. Backlog – tasks that are waiting to be worked on
2. In Progress – currently being developed by a team member
3. Testing – undergoing integration, system or UAT testing
4. Done – complete and ready to be demonstrated and/or deployed

To identify, and control, bottlenecks and process limitations, each workflow state (or column) has a limit, called a WIP, or Work In Progress Limit, to the number of currently active tasks. This allows managers and team members to regularly monitor, and measure, the flow of work.

In total, there are 6 core elements to Kanban:

1. Visualise (Kanban / Card Wall)
2. Limit WIP
3. Manage Flow (and reduce bottlenecks)
4. Make Policies Explicit
5. Feedback Loops
6. Improve Collaboratively

SCRUM OVERVIEW

Scrum is described as a 'framework within which you can employ various processes and techniques', rather than a process, or a technique, for building products. The Scrum framework is primarily team based, and defines associated roles, events, artefacts and rules. The three primary roles within the Scrum framework are:

1. The product owner who represents the stakeholders,
2. The scrum master who manages the team and the Scrum process
3. The team, about 7 people, who develop the software.

Each project is delivered in a highly flexible and iterative manner where at the end of every iteration of work there is a tangible deliverable to the business. This can be seen in the following diagram.

Notes:

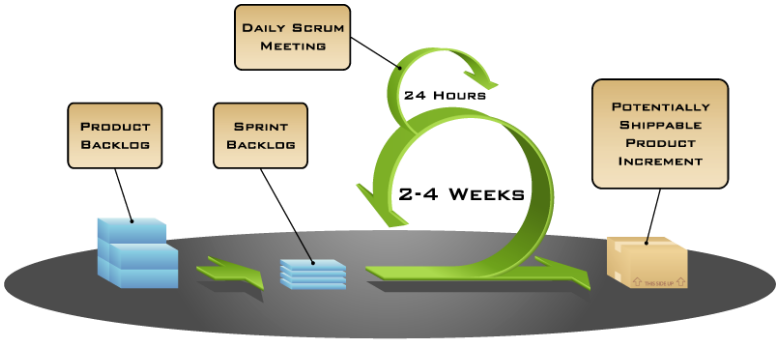


FIGURE 4: SCRUM FRAMEWORK

The requirements that form the basis of the project are collated into what is called a Project Backlog, and is updated regularly. The features that are associated with these requirements are termed user stories. This relationship is illustrated in the following diagram:

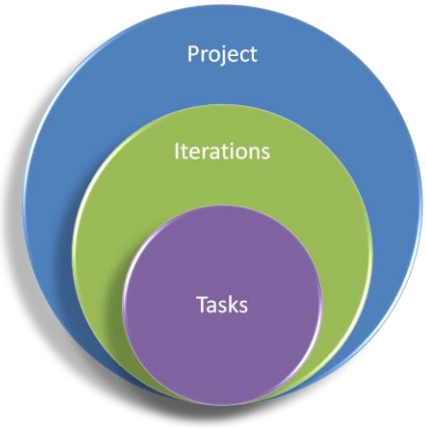


FIGURE 5: SCRUM PROJECT STRUCTURE

The work is time-boxed into a series of 1 to 4 week cycles where the business and project team estimate which user stories in descending priority order are achievable each cycle, or iteration. This subset of user stories from the Project Backlog form the basis of the Iteration Backlog planned for delivery over that two week period.

Under Scrum, there are 3 timeboxed (or fixed duration) meetings held during an iteration plus a daily stand-up meeting for the team, scrum master and (ideally) the product owner. At the

Notes:

beginning of an iteration, features to be developed during the iteration are decided during the iteration planning meeting. At the end of the iteration are another 2 meetings, the iteration review and iteration retrospective where the team reviews the product and demonstrates the use of the software, as well as reflect on, and improve, the iteration process itself.

After the iteration is complete, the next set of user stories is selected from the Project Backlog and the process begins again. Burn rate is monitored to determine when funding will be exhausted.

TABLE 1: KEY SCRUM CONCEPTS

Concept	Description
Project	Discreet set of end user requirements that have been grouped, prioritised and funded.
Requirement	The end user statement that outlines their information need.
Iteration (also known as a "Sprint")	An iteration is a 1 to 4 week time-boxed event focused on the delivery of a subset of user stories taken from the Project Backlog.
Project Backlog	The Project Backlog is the current list of user stories for the Project. User stories can be added, modified or removed from the Backlog during the Project.
Iteration Backlog (also known as a "Sprint Backlog")	Subset of user stories from the Project Backlog that are planned to be delivered as part of an Iteration.
User Stories	The user story is a one or two line description of the business need, usually described in terms of features.
Tasks	Tasks are the activities performed to deliver a user story.
Technical Debt	This refers to items that were either: • missing from the Planning meeting; or • deferred in favor of early delivery.

Notes:

TEST DRIVEN DEVELOPMENT (TDD) OVERVIEW

Test-driven development is a development methodology that requires developers to create automated tests before writing the code or function itself. If the tests pass, then the code is displaying correct behaviour. These tests should be run automatically using one of the xUnit applications.

There are 5 steps to TDD.

1. Create a test
2. Add the test to the test catalogue
3. Write the code
4. Run the tests (all of them)
5. Clean up the code as required. (Refactor)

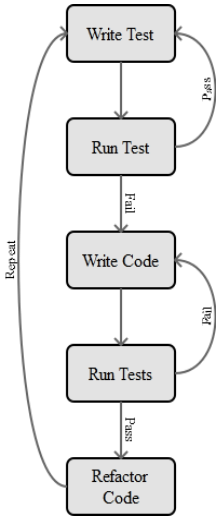


FIGURE 6: TEST-DRIVEN DEVELOPMENT FLOWCHART

Notes:

Extreme Programming (XP) Overview

XP is an agile development methodology, intended to accept and respond to changing customer requirements.

XP describes four basic activities within the software development process.

1. Writing the software
2. Testing the software, regularly and automatically
3. Listening and understanding the customer
4. Designing, or refactoring, an application framework to reduce unnecessary dependencies between features.

XP also includes development methodologies such as;

1. Pair programming, with two developers working together
2. Common code standards (documentation, naming conventions, whitespace)
3. An understandable system metaphor, where all classes and functions names should be understandable.

Feature Driven Design (FDD) Overview

FDD is a model-driven process that consists of five basic activities. For accurate state reporting and keeping track of the software development project, milestones that mark the progress made on each feature are defined.

Processes

- Domain Object Modelling: Define the domain of the problem to be solved and create an object framework.
- Developing by Feature: Functions are broken into the smallest possible part and developed in the order of priority.
- Individual Class Ownership: Each code class is assigned to a single owner who is responsible for the consistency, and performance of the class.
- Feature Teams: A small, dynamically formed team that designs and develops each feature.
- Inspections: Carried out to ensure good quality design and code.

Notes:

- Configuration Management: A log of all features and the source code that have been completed to date.
- Regular Builds: Ensures there is always a fully functional system.

ACTIVITIES

Develop Overall Model:

- High level scoping
- Create domain walkthroughs for each modelling area
- Peer review of each model
- Merge into complete system model

Build Feature List:

- Split domain into subject areas
- Separate subject areas into business activities
- Separate business activities into individual features
- e.g. Confirm the password of user

Plan by Feature:

- Assign features to classes
- Assign classes to developers (or development teams)

Design by Feature:

- Create a set of features to be developed within two weeks
- Build sequence diagrams for each feature
- Refine model for each feature
- Inspect and review the design

Build by Feature:

- Develop the code for each feature and class
- Unit test
- Promote to build

Notes:

PROJECT ROLES

'So Mr Edison, how did it feel to fail 10,000 times?'
'Young man, I didn't fail, I found 9,999 ways that didn't work'

Thomas Edison, anecdotal (on his invention of the incandescent light)

Notes:

PROJECT TEAM

The project team is a self-governing group capable of independently delivering to a customer's requirements. As a result the team requires cross functional representation of skills and knowledge in the data, tools and infrastructure domains.

A typical project team can be comprised of the following:

- Customer Representative or Product Owner (Scrum)
- Team Facilitator or Scrum Master (Scrum)
- Architects / Analysts
- Designers
- Developers

The project team characteristics and responsibilities are as follows:

- 7 ± 2 (5 to 9) resources who are allocated full-time to the team
- Cross functional in nature across skills, applications, data and organisational knowledge
- Self-empowered
- Responsible for delivering the product
- Determine the tasks required to deliver each feature
- Estimate the effort for each task
- Develop the features
- Resolve issues

Ideally the project team, including the Customer Representative are co-located.

INTERESTED AND COMMITTED

Interested roles are individuals who have an "interest" in the software development. Whilst they should be kept informed of progress, they do not have the same level of responsibility and input into the development as committed roles. Interested parties include the Users, the customers and the Customer Representative.

Notes:

Committed roles are responsible for the software development, and are the people who "do" the work. Committed parties include the Team Facilitator, the team and testers.

FIGURE 7: PIGS AND CHICKENS

PRIMARY ROLES

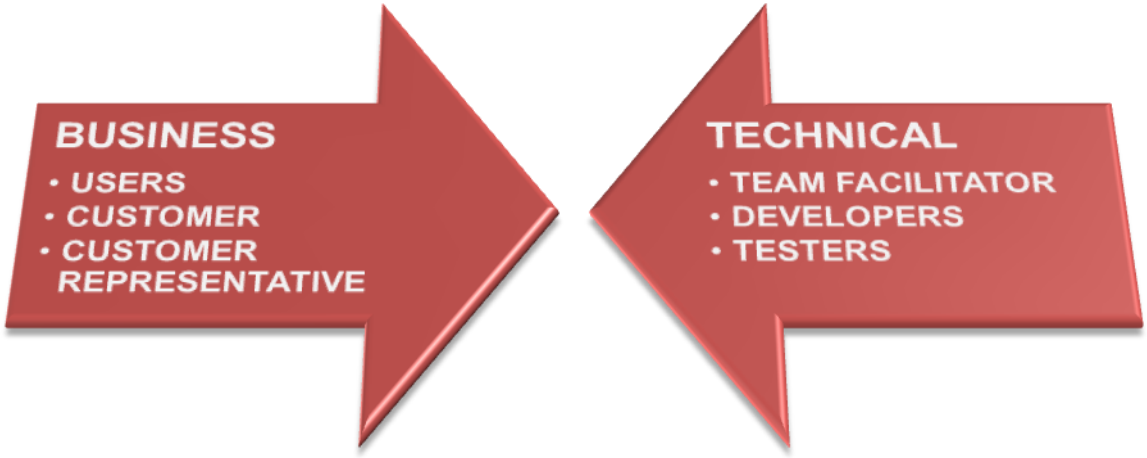


FIGURE 8: BUSINESS VS TECHNICAL ROLES

Notes:

TABLE 2: PRIMARY TEAM ROLES

Role	Primary Responsibility	Typical	Does Not
Users *Interested Role*	• Use the software • Identify issues & • Provide feedback	• There are no typical users.	• Set Scope • Test Work
Customers *Interested Role*	• Define, start and end the project	• Internal managers • External Clients	• Direct Work
Customer Representative *Interested Role*	• Manage the product backlog • Set the scope • Approve Releases	• Project Manager • Product manager • Customer	• Manage the team
Team Facilitator *Committed Role*	• Manage the Agile process • Report on progress	• Project manager • Team leader • Team member	• Prioritise features
Developers *Committed Role*	• Develop features • Resolve issues	*cross functional* • Developer • Designers • Writers • Administrators	• Prioritise features
Testers *Committed Role*	• Test • Approve or reject features for release	• Existing developers • Dedicated testers	• Test their own code

DEMING'S 14 POINTS FOR MANAGERS

I'd like to end this section with a look at some of W Edwards Deming approaches for lean managers to transform business effectiveness. These points aim to change the focus of management from growth through financial returns, to the more Agile approach of growth through investment, innovation and strong staff engagement.

1. Create constancy of purpose toward improvement of product and service, with the aim to become competitive, stay in business and to provide jobs.
2. Adopt the new philosophy. We are in a new economic age. Western management must awaken to the challenge, must learn their responsibilities, and take on leadership for change.

Notes:

3. Cease dependence on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place.
4. End the practice of awarding business on the basis of a price tag. Instead, minimize total cost. Move towards a single supplier for any one item, on a long-term relationship of loyalty and trust.
5. Improve constantly and forever the system of production and service, to improve quality and productivity, and thus constantly decrease costs.
6. Institute training on the job.
7. Institute leadership. The aim of supervision should be to help people and machines and gadgets do a better job. Supervision of management is in need of overhaul, as well as supervision of production workers.
8. Drive out fear, so that everyone may work effectively for the company.
9. Break down barriers between departments. People in research, design, sales, and production must work as a team, in order to foresee problems of production and in use that may be encountered with the product or service.
10. Eliminate slogans, exhortations, and targets for the work force, asking for zero defects and new levels of productivity. Such exhortations only create adversarial relationships, as the bulk of the causes of low quality and low productivity belong to the system and thus lie beyond the power of the work force.
11. a. Eliminate work standards (quotas) on the factory floor. Substitute leadership. b. Eliminate management by objective. Eliminate management by numbers, numerical goals. Substitute leadership.
12. a. Remove barriers that rob the hourly worker of his right to pride of workmanship. The responsibility of supervisors must be changed from sheer numbers to quality. b. Remove barriers that rob people in management and in engineering of their right to pride of workmanship. This means, inter alia, abolishment of the annual or merit rating and of management by objective.
13. Institute a vigorous program of education and self-improvement.
14. Put everybody in the company to work to accomplish the transformation. The transformation is everybody's job.

Notes:

PROJECT INITIATION

aka Feasibility

aka Sprint 0 (Scrum)

aka Iteration 0 (XP)

'It is always wise to look ahead, but difficult to look further than you can see.'

Winston Churchill, ~1960

Notes:

SPECIFICATIONS IN AGILE?

Contrary to common opinion, it is very important to have a good, but lean, specification before starting an agile project. By building this specification (or backlog in agile terminology) a project will;

- Reduce Risk & Uncertainty
- Improve Decision Making and integrate With Long term Goals
- Improved cost planning (including Staff Hiring)
- Prioritise research and information Gathering

BEGINNING THE PROCESS

Unlike traditional waterfall methods, the specification phase of an agile project is very short; usually no more than 1 or 2 days, and the full team should be available at inception. During the period the customer should be made fully aware of their role. The design should contain the following;

- Problem statement that needs to be addressed
- Desired business objectives, outcomes and benefits for this project
- Identified key stakeholders
- High level business requirements
- Architectural and technical scope
- Testing requirements

OUTCOMES

The outcomes from Project Initiation are:

- The team should be identified and brought together
- If not part of the team, identify and train the Customer Representative
- Create the backlog in low detail. Allow customers to slowly build the product requirements throughout the process.
- Estimate the product backlog.
- Add any team training to the backlog as tasks.

Notes:

Cost / Time / Scope

"How much is this going to cost?" - "As much as you're willing to spend."

"How long is this going to take?" - "As long as it necessary."

"What am I going to get?" - "Whatever you tell us you want."

Fixed Cost

Where a customer asks for a fixed price quote prior to agreeing to project commencement, but is flexible on what is delivered and how long it takes.

- Work in absolute customer priority order – reducing the time spent on technical helper tasks will help meet short-term budget constraints (at the cost of long term, post-project, efficiency)
- Monitor cycle time – this is your key indicator of cost

Fixed Time

Where a customer asks for delivery by a certain date, but is flexible in scope and cost.

- Work in absolute business value order – increases the number of user stories complete in a given time period (high business value = simple, moderate-high priority)

Fixed Scope

Where a customer asks for a fixed set of deliverables, but is flexible in the time it takes to deliver and the cost of delivery. This is sometimes known as "heavy agile".

- Focus on backlog definition and estimation during project initiation to ensure accurate scope definition

Fixed Cost and Scope

Where the customer asks for a fixed price quote for a fixed set of deliverables. In this situation, the final date for delivery is flexible. As well as the points in fixed cost and fixed scope;

- Increase the estimate risk during project initiation – to ensure your quote for the project allows for unexpected delays (which would impact on your cost to deliver)
- Update delivery date as required

Notes:

Fixed Cost and Time

Where the customer asks for a fixed price quote by a fixed time. In this situation, the exact set of features (or scope) for delivery is flexible. As well as the points in fixed cost and fixed time;

- Calculate total cost based on average cycle time – which makes your quote to the customer very simple.

Fixed Time and Scope

Where the customer asks for a fixed set of deliverables by a fixed time. In this situation, the total cost to the customer is flexible. As well as the points in fixed time and fixed scope;

- Allocate additional time into the schedule to cater to unexpected defects or technical debt
- Increase the size of the team prior to the end of the project if required – to ensure the set of features are completed in time.

Fixed Cost, Time and Scope

Where the customer gives no flexibility in the project.

Cancel the project – this is not an agile project. This should be run using a waterfall methodology such as PRINCE2 (and even they are likely to fail without some flexibility)

Notes:

STORIES, TASKS AND THE BACKLOG

'Make everything as simple as possible, but not simpler.'

Albert Einstein (paraphrased), 1933

Notes:

PRODUCT BACKLOG

FIGURE 9: PRODUCT BACKLOG

The backlog contains all the user stories (features) for the product. The Customer Representative is responsible for defining the user stories, and assigning each with a priority for delivery. The order may also be influenced by story dependencies or business value e.g. a lower priority story may need to be done first before a higher priority story can be started.

The user stories describe the set of features that will satisfy each requirement. The high priority user stories that are candidates for development in the short term require sufficient detail for the team to solution, and should be sized to fit within a few days. As a guideline, tasks and stories that involve waiting should be split into separate tasks and research tasks should have a high estimate risk. This is done to enable accurate tracking.

Notes:

 (cc)-by-sa – Evan Leybourn

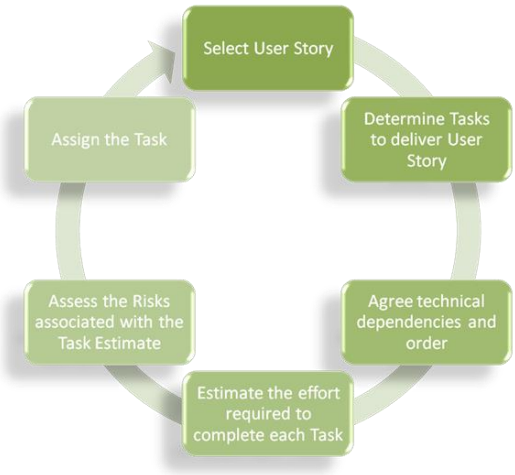


FIGURE 10: DESIGN AND PLANNING WORKFLOW

Each user story should meet the INVEST characteristics, as defined by Bill Wake.

- Independent: Each story should be as self-contained as possible, with minimal dependencies on any other story. This allows for easy reordering or removal, as customer stories change.
- Negotiable: The customer can change a story at any time, up to the point it enters development.
- Valuable: Each story should deliver a tangible, and measurable, benefit to the customer.
- Estimatable: The definition of each story is such that the team can estimate it.
- Small: The estimate, and delivery, of a story, should be between half a day to a few days.
- Testable: Each story should have appropriate quality control and quality assurance metrics, so the customer can validate their deliverables against the original story.

Each feature should contain, at a minimum, the function, priority, an estimate of the effort to develop and the estimate risk (0 - 100%) based on how accurate the team feels the estimate is.

Notes:

In can be helpful to structure each user story in the following format.

As a [role]

I want a [goal/desire]

So that [benefit]

The [role] is the expected end-user of the user story, who is usually different from the customer. The [goal/desire] of each user story describes what should be delivered, and the [benefit] provides the context, or the Why.

TABLE 3: KEY STORY ELEMENTS

Planning Element	Description
User Story	The user story is a description of the business need, usually expressed as a feature.
Story Identifier	Every user story will be assigned a unique identifier for tracking purposes.
Tasks	A task is typically a single activity that can be described in one sentence that contributes to the delivery of a user story. • Generally a task takes no longer than 4-16 hours of effort to complete • There may be one or many tasks per user story • The task can only be assigned to and owned by one person at a time
Task Identifier	A unique identifier will be assigned to track each task, and show which user story they are associated with.
Project Function	This describes the architectural layer where the task activity will be performed.
Assignee <Optional>	This is the person who will be responsible for delivering the task. The person assigned to the completion of the task may also change at any point.
Estimate	The estimate in hours is the amount of effort the team agrees is required to complete the specified task. The estimate may include: • Analysis • Build

Notes:

	• Unit Test • Migrate from DEV to TEST • Integration Testing • Documentation
Estimate Risk Modifier	This is a measure of the confidence level associated with the estimate provided and represented as a numeric modifier.

REVIEW THE BACKLOG

The product backlog should be regularly reviewed by the Customer Representative and the delivery team to ensure that the listed stories remain relevant, are appropriately ordered and the estimates are still valid. This includes clarification, prioritisation and in some cases investigation of the feasibility of the collated user stories. This activity also needs to take into consideration any technical debt inherited from previous work.

This also provides the Customer Representative with the opportunity to communicate the required scope of delivery, provide the business context and priority, and address any questions the project team may have to assist with performing the solution decomposition and estimation steps. Like all lean techniques, this is a JIT process and should not take more than a few hours every month.

Ensure that sufficient notice is given to the team, so that they have sufficient adequate time to consider solution options for discussion during the review, and prepare clarification questions for the Customer Representative.

The participants for this session are as follows:

- Customer Representative
- Team Facilitator
- Team
- Testers

Notes:

ACCURACY

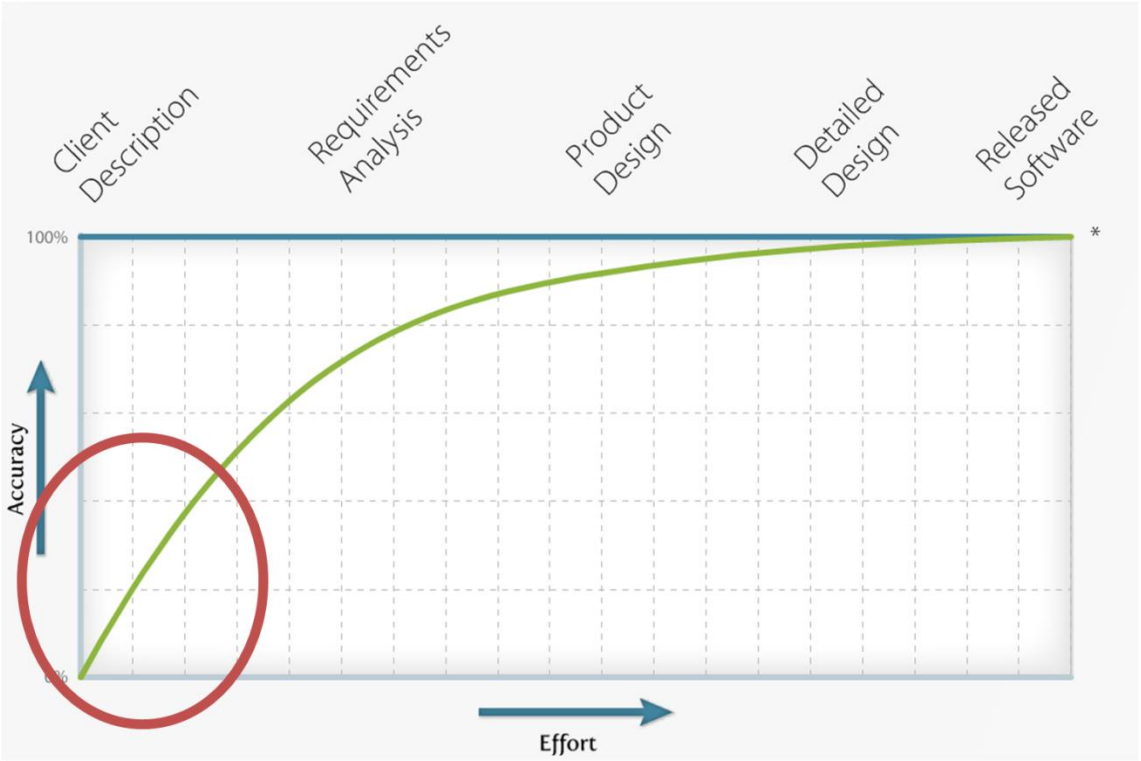


FIGURE 11: ESTIMATE ACCURACY OVER TIME

It is important to note that to increase the accuracy of any estimate, the effort involved increases exponentially. In Agile, we are only interested in the initial estimation.

ESTIMATING EFFORT

An Agile estimate is single number intended to represent the effort (or man-hours) required to deliver a component, or user story, of the final product. As a result, to perform agile estimation requires cross functional representation of all the requisite skills and knowledge in the systems, data, tools and infrastructure during the estimation workshops.

All stories should be assigned as estimated effort, or cost, to implement. We use a modified Fibonacci series, such as 1, 2, 3, 5, 8, 13, 20, 40, and 100, to represent effort. This encourages

Notes:

features to be split into the smallest task possible, and provides a more realistic estimate range.

As large stories bubble up to the top of the ready queue, they are broken down into their composite tasks as re-estimated. The process of solution decomposition may reveal additional tasks, or more complex tasks, that were not apparent during the initial estimation. These are added, and the total estimate of the project is updated.

How?

Tasks can be estimated in 4 ways.

1. Expert opinion: The team member with specific understanding, or who is most likely to develop the task, can provide a more accurate estimate of effort.

   E.g. A database administrator can better estimate effort for database tasks.

2. Comparison: Comparing a task to another, already estimated, task.

   e.g. "Task A is about twice the effort of Task B"

3. Components: If a task is too large to accurately estimate, break it into small sub-tasks.

   e.g. User management can be broken into interface, login, ACL, etc.

4. Planning poker: If using Planning Poker, estimates must not be mentioned at all during discussion to avoid anchoring. A timer may be used to ensure that discussion is structured; any person may turn over the timer and when it runs out all discussion must cease and a round of poker is played.

   Each person lays a card face down representing their estimate of the task, and then simultaneously turns their cards over.

   People with high estimates and low estimates are given an opportunity to discuss their estimate before anyone else can speak.

Notes:

Repeat the estimation process until a consensus is reached.

FIGURE 12: PLANNING POKER CARDS

Estimating Time

Converting an estimated cost into estimated time is very simple. There are 2 primary modifiers that we use. Staff overhead and estimate accuracy (or risk).

Staff Overhead

This is a percentage modifier for staff availability to work on specific project tasks. It allows you to take into account factors such as estimated leave, illness, breaks, daily stand-up meetings etc. The industry standard modifier is 25%-40%, though you should modify this as required. To calculate staff overhead, use the following process;

Staff Overhead = Average (actual hours worked per day)/total days - 1

Calculation

Story Cost x (Staff Overhead + 1) x (Estimate Risk + 1)

e.g. 4 x (25%+1) x (50%+1)

= 4 x 1.25 x 1.5

= 5 to 7.25 hours

Notes:

CONTINUOUS DELIVERY

'Treat your men as you would your own beloved sons. And they will follow you into the deepest valley.'

Sun Tzu, ~6th Century BCE

Notes:

DAILY LIFECYCLE

The daily lifecycle of team activities is as follows;

1. Team members select the next task to work on
2. Undertake the task as described
3. Commit and share the completed task with the rest of the team
4. Write and run the tests that will be used to verify they have been completed successfully. Verification, unit testing and documentation need to be completed prior to migrating the deliverable from DEV to SIT.

The assignee for a task may change at any of these steps. Team members will proactively interact will their colleagues and any internal parties as required to progress the assigned task to completion, including any quality assurance and review.

The governance of the daily lifecycle is through the daily stand-up.

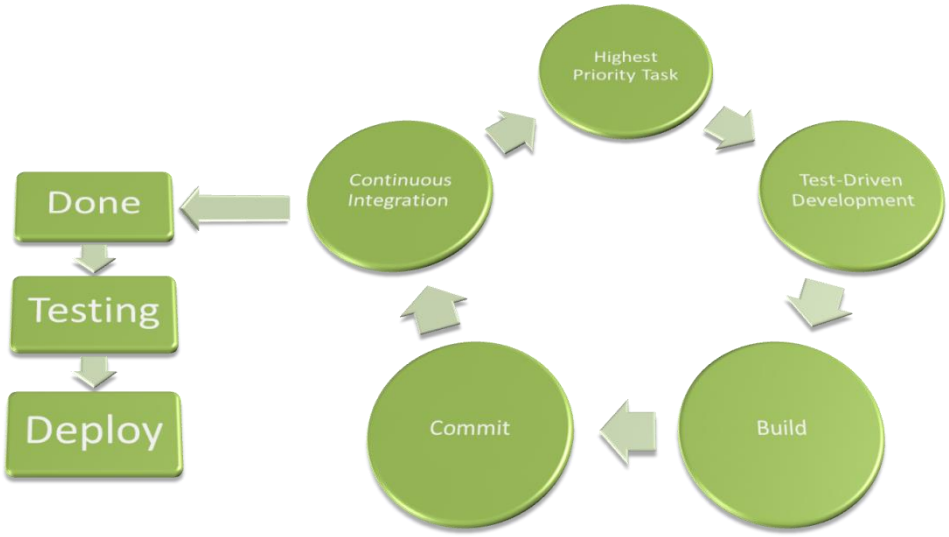


FIGURE 13: WORK LIFECYCLE

Notes:

Development Hints

Features

Get the highest priority feature from the backlog. Allow developers to choose their work, don't assign it.

Develop

Agile also makes some suggestions on improving the development process. These are;

- Pair Programming: Two developers working together, the first as a coder and the other as a reviewer. These roles should change regularly, and the pairs themselves should switch pairs each day
- Code Standards: A common coding style (Documentation, Names, Whitespace, etc)
- System Metaphor: All classes and functions should be named such that their purpose is understood.

Commit

Everyone must commit every day, and should never commit broken code. (Continuous Integration)

Transparency

Key to Agile is transparency between the product, the team and the customers.

Customers can:

- Attend daily stand-ups. However they should not talk. Questions should be directed to the Customer Representative or Team Facilitator. An alternative is to record the daily stand-ups and make the recording available to the customer.
- See the backlog in its current state.
- See the state of each task via a Kanban board or integrated dashboard.
- Access a test version of the software from the development environment.

Notes:

Test Driven Development

Key Points

Tests are written by the customer and developer together and are written before the code. Both automated unit tests and user acceptance tests should be written. There is no issue with using standards such as IEEE 829 and IEEE1008 to document and write tests.

By using TDD, the team can prove how well a project is going and how close to completion. This in turn, allows customers and Customer Representative to make informed decisions about the project.

Test Coverage

The tests should cover;

- Software functions,
- Boundary cases,
- User interface,
- User experience,
- Non-functional components,
- Performance

Test Types

There are 4 types of tests that can be written.

1. Defect
2. Functionality
3. Usability
4. Data

Notes:

TDD IN DEVELOPMENT

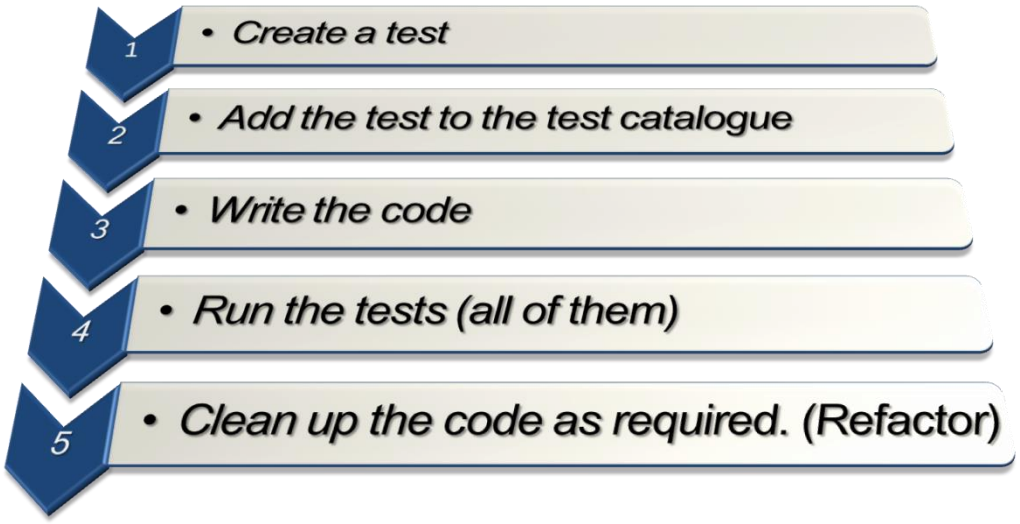


FIGURE 14: TDD WORKFLOW

CONTINUOUS INTEGRATION

UNIT TESTING

Runs predefined tests to identify software defects

- Create tests for each class and function
- Create tests for all parameter combinations
- Create tests for all edge cases
- Create tests to examine the database for logical errors
- Create tests to detect interface defects (Selenium)
- Tests should be kept in the version control repository
- Test in a clone of the production environment

Notes:

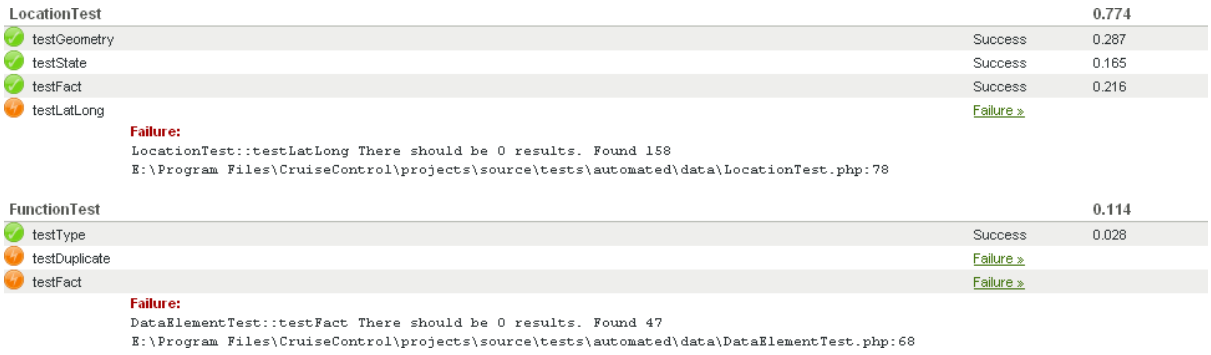


FIGURE 15: UNIT TEST SCREENSHOT

CODE STANDARDS

Inspect the developed code for deviations from the internal code standard

- Check for correct inline documentation (docblock)
- Check for correct variable naming conventions
- Check for correct whitespacing conventions
- Check for complex code that may require refactoring
- Check for incomplete or unused functions

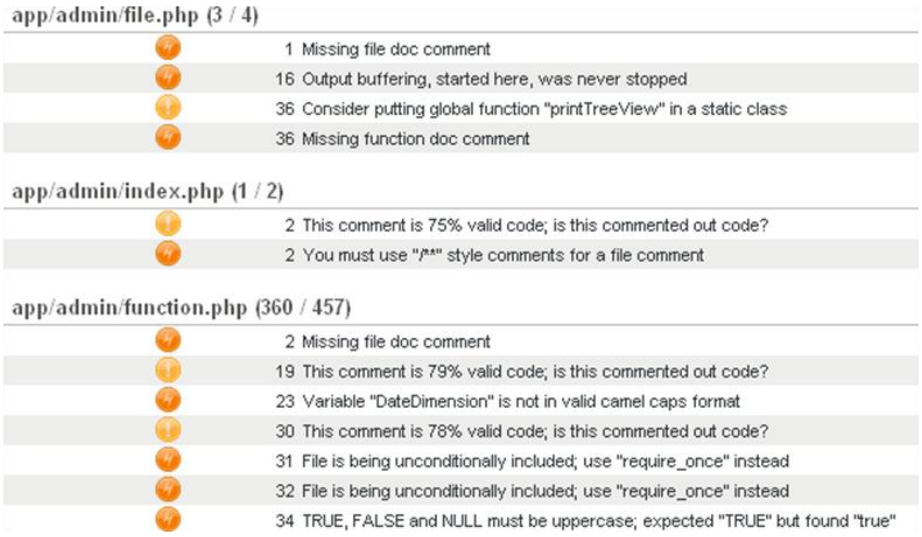


FIGURE 16: CODE STANDARD SCREENSHOT

Notes:

Documentation

The following should be commented;

- Files
- Classes
- Functions
- Class Variables
- Complex Structures

Comments should contain;

- Description
- Author
- Usage
- Parameter description
- Return description
- References to other functions
- Copyright (file comments)

Code Coverage

Calculate and display how much of the software is covered by unit tests. Aim for 80-90% code coverage.

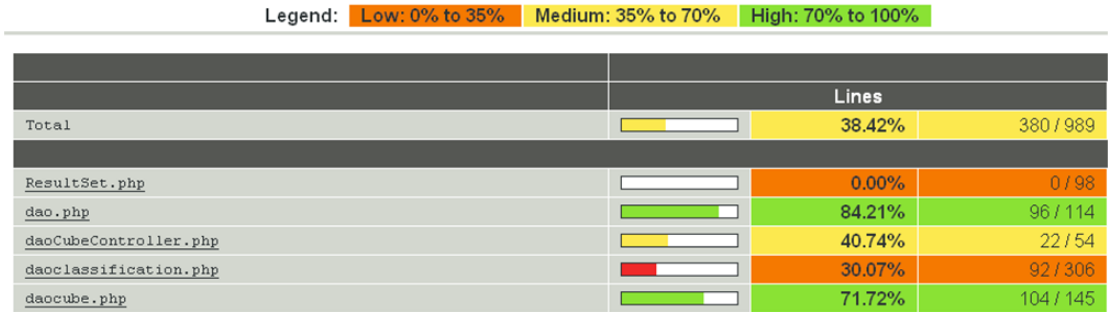


FIGURE 17: CODE COVERAGE SCREENSHOT

Compile

Run any compile or make scripts. All commits should compile.

Notes:

5S

5S is an approach to waste reduction, quality & continuous improvement (housekeeping) defined by the lean approach. The 5 S's are:

- Seiri (Sort): to clean and organise the work area
- Seiton (Set in Order): arrange the work area to ensure easy identification and accessibility
- Seiso (Shine): mess prevention and regular maintenance of the work area
- Seiketsu (Standardize): create a consistent approach for carrying out production processes
- Shitsuke (Sustain): maintain the previous 4S's through discipline and commitment

DAILY STAND-UP

The purpose of the daily stand-up meeting is to provide a consistent focus on incremental progress and delivery demonstrated to the Customer Representative. It is intended to be informative and interactive and align the team's understanding of what is being worked on, by whom and its current status.

This team meeting is strictly time-boxed to 15 minutes. All participants should answer the following three questions:

1. What did you achieve yesterday?
2. What will you achieve today?
3. What impediment may prevent you from achieving your goal today?

It is the objective of the Team Facilitator to remove any impediment identified by the team.

Projects with multiple teams should hold a summary stand-up, also timeboxed to 15 minutes, after the initial stand-ups. This meeting should bring together Team Facilitators from multiple teams to answer the same 3 questions as before, but relating to their teams.

Notes:

PRODUCT REVIEW

At regular intervals, the Team Facilitator should hold a product review meeting to demonstrate to the customer representative and customer (if different) the completed user stories for final review for release to production. As the customer representative should have been involved in the development and verification process on a daily basis this step should be straightforward.

During the meeting the team should:

- Present the completed work to the customer representative
- Review the work that was completed to date
- Review the work that is scheduled to be completed.

Notes:

KANBAN

'Simplicity, carried to the extreme, becomes elegance.'

Jon Franklin, 1994

Notes:

TASK LIFECYCLE

Agile processes are designed to promote a sustainable workload, where your teams, management, and customers, are able to maintain a constant pace indefinitely. Teams have the authority to design, plan, and estimate, each story, as well as the responsibility and accountability for delivery. This level of ownership for work, combined with integrated customer engagement, significantly improves workload management, which in turn reduces overtime and stress. However, for this to be efficient there needs to be a simple mechanism to manage, and level out, workflow within each team. This is where Kanban comes in.

Depending on your process workflow, a task will progress through a minimum of 4 different states during its lifecycle. Each task and state should be visible to the team, Customer Representative and customer; commonly this is done through a Kanban board (card wall) or integrated dashboard.

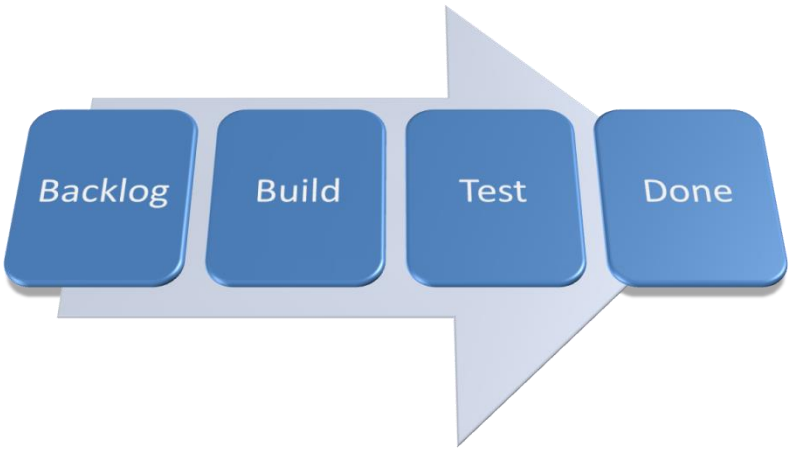


FIGURE 18: BASIC TASK LIFECYCLE

The Assignee for a task may change at any of these steps. It is important to understand that each state (and thus column on the Kanban board) represents a state within the development workflow, not a handoff between team members. Team members will proactively interact with their colleagues, and any internal parties, as required, to progress the assigned task to completion, including any quality control and review.

Notes:

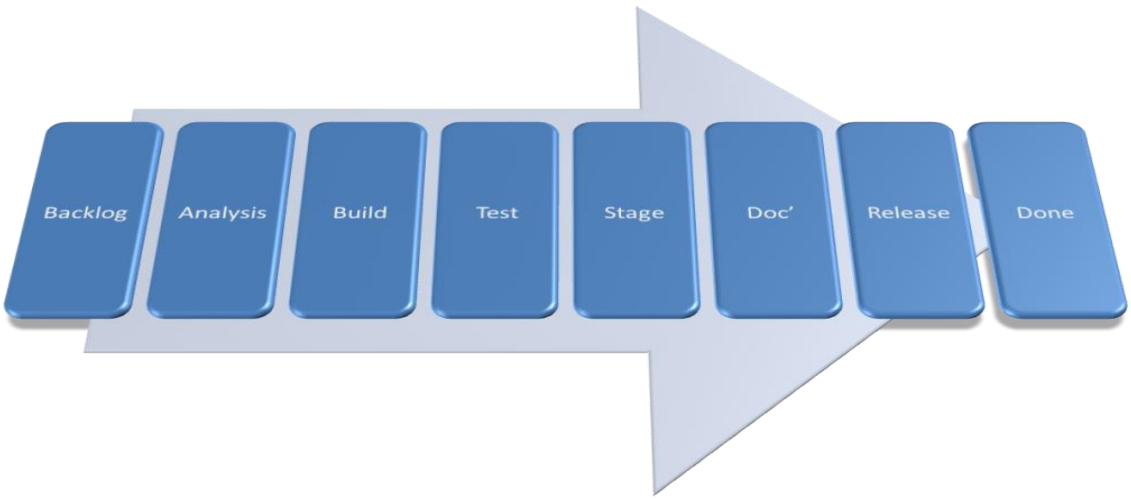


FIGURE 19: COMPLEX TASK LIFECYCLE

BACKLOG

The Backlog is the list of user stories (or tasks) that the Customer Representative maintains. There may be a second backlog column "Ready" which contains those stories which have been technically designed and are ready to be developed. Based on the logical sequencing of tasks and agreed prioritisation, the project team members select the next task to work on and promote this to the "In Progress" state.

IN PROGRESS

In Progress items are tasks that are actively being worked on. This includes both development and unit testing activities. Once the task has been completed it is promoted to the "Testing" state.

In Progress tasks include the following types of activity being performed:

- Analysis
- Build
- Unit Test
- Documentation

Notes:

When a task has been completed the deliverable will be changed to the "Testing" state. In the case of code based artefacts these will be promoted from the development environment to the test environment.

Special Type: Blocked

Blocked items are stories and or tasks that have upstream or downstream dependencies external to the project team that are preventing progress. These impediments are moved to this holding state to highlight these issues to the Team Facilitator and Customer Representative for escalation and resolution.

Testing

Testing, in this context, is performed by the team's specialist testing resources. Unit testing is expected to be undertaken by the developers.

Done

Tasks are considered "Done" when:

- Code has been produced, meets development standards, and has been checked in and run against current version in source control
- Unit tests written and passed
- System tested and passed
- Quality Assurance reviewed
- Builds without errors for deployment
- Relevant documentation, including diagrams have been produced or updated and communicated

The decision over whether a user story is done is based on whether all the pre-requisite tasks associated with this story have been completed. The completed user stories are presented to the Customer Representative for acceptance and sign-off. This can either be done individually, or in a batch.

Not Done

Any tasks that are "not done" are reviewed in the context of the user stories that they belong to and if this impacts whether the user story can be considered delivered. The not done tasks may be rolled into a new user story, accrued as technical debt, or it may be decided that they are no longer required and are removed.

Notes:

VALUE STREAM MAPPING

Teams can create Value Stream Maps to defines the 'As-Is' steps & roles for each task lifecycle. That is, to create the workflow that will be used by the teams and visualised on the Kanban board below.

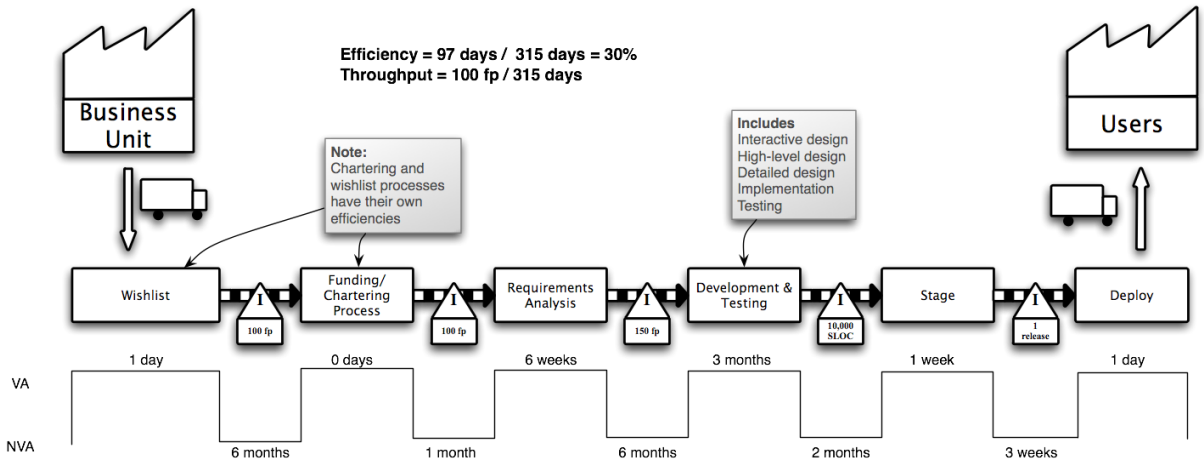


FIGURE 20: EXAMPLE PROJECT VALUE STREAM MAP[4]

A value stream can be defined as all the steps – both value added and non value added – required to take a product or service from its raw materials state into the waiting arms of a happy customer. Each step is defined, articulated and mapped on the value stream (above). The average time taken to complete each step (value add time) is measured, as well as the time taken to move between steps (non-value add time).

In the example above, it takes 6 weeks to complete Requirements Analysis, and an additional 6 months from completing this to starting the Development & Testing.

Lean defines 10 steps needed to accurately define the value stream maps for each process within a product family group. If there is only one process to be modeling, you can start from step 5.

[4] Image thanks to: http://www.nexusis.com/solutions/cloud/cloud-consulting/

Notes:

1. Gather Preliminary Information
2. Create a Product Quantity Routing Analysis
3. Group Customers and Sort Materials
4. Sort Product Families by Build Sequence
5. Choose One Value Stream to Begin With
6. Create an Operations Flow Chart
7. Walk the Shop Floor
8. Collect the Data
9. Construct the VSM
10. Summarize the Data and Get the Big Picture

Any process can be subject to a value stream map. Even eating cake;

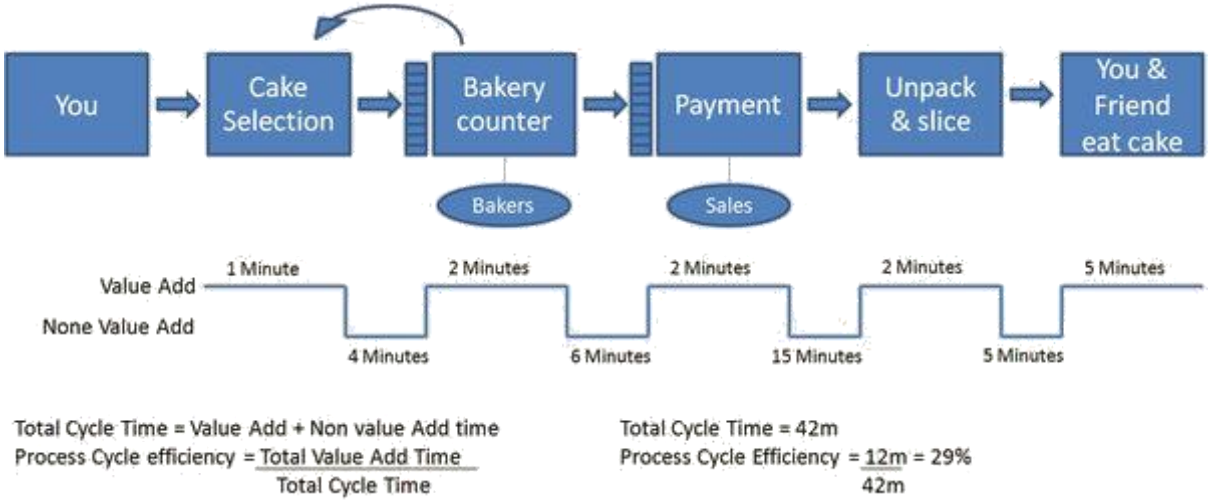


FIGURE 21: VALUE STREAM MAP FOR BUYING AND EATING CAKE[5]

By adding the total value add time to the total time (VA+NVA), it is possible to define the total process efficiency. In the example above, 29%.

[5] Images thanks to: http://leadinganswers.typepad.com/leading_answers/2011/09/pmi-acp-value-stream-mapping.html

Notes:

KANBAN BOARDS

A Kanban board is a useful visualisation and control mechanism for both continuous product delivery. Starting at the Backlog, and finishing at done, each team will define the intervening states and workflows that make up the lifecycle (Value Stream Mapping) of their tasks (the Kanban). This can be as simple, or complex, as required. Teams working on several different types of tasks may have multiple Kanban Boards, visualising the different states and workflows for each type. When there is available capacity in a state, it will 'pull' any 'ready' tasks from the preceding state, thus moving tasks through the workflow.

The visualisation component, or cards, of a Kanban, helps identify the state of each task, when a task is ready, where there is spare capacity, and if there are any bottlenecks or impediments.

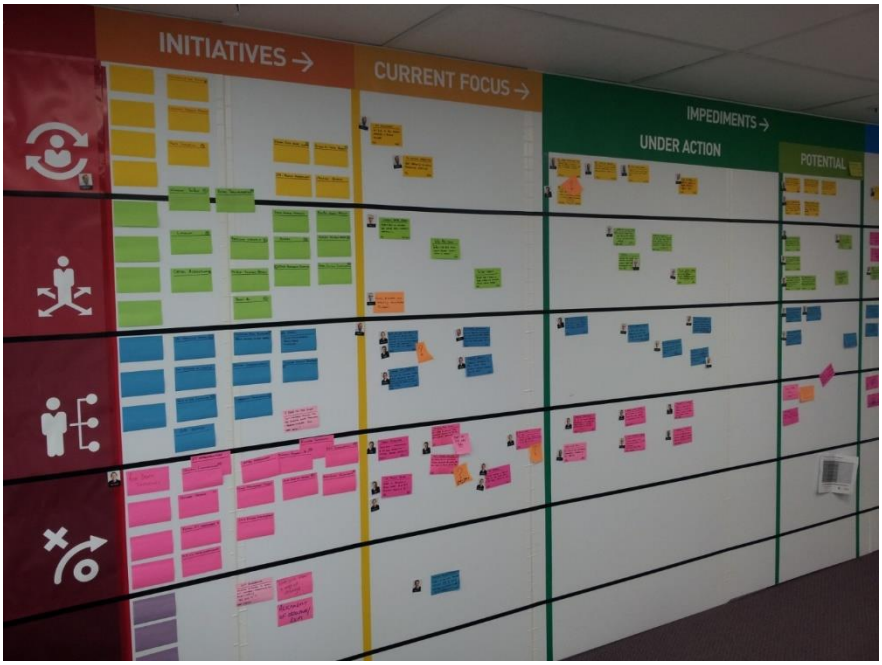


FIGURE 22: EXAMPLE KANBAN BOARD[6]

Tasks or stories that have identified defects, need rework, or have upstream or downstream dependencies external to the team that are preventing progress, are marked as 'blocked', but

[6] Image thanks to: New Zealand Postal Group via Directing the Agile Organisation

Notes:

do not change state. By leaving the task in the current state, the team can see where the blockage is, and identify where the process should resume once it is resolved. Similarly, by making all blocked tasks and stories visible on the Kanban board, customers and management are aware of the issues, and this simplifies any escalation and resolution processes.

Each task and state should be visible to your teams, customer, and customer representative; commonly achieved through a physical Kanban board, or integrated virtual dashboard. Each card describes a single task or story, as well as its estimate, and who is currently working on it. Keep cards simple, with additional information stored elsewhere (usually the workload management system). Divide the Kanban board into multiple, labelled columns, each representing a single state. Then further divide each column in half, the first half being 'In Progress' and the second half being 'Ready'.

Some versions of Kanban also provide a single 'Expedite' track, at the top of the board, for urgent stories and tasks. There can only ever be one card at a time in this track, and it has the highest priority, above all other cards. If possible, team members should finish their current cards before moving onto the expedited card.

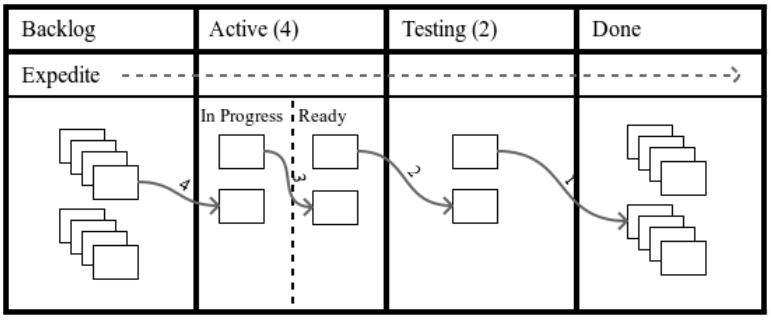


FIGURE 23: KANBAN BOARD AND FLOW

Except for the Backlog and Done states, the number of cards allowed at any single time, in each state, is restricted. Called the Work In Progress (WIP) Limit, it includes both the 'In Progress' and 'Ready' Cards in any state, and matches the team's work capacity. In general, smaller WIP Limits reduce lead times, help uncover problems in the workflow, and drive continuous improvement (Kaizen), whilst higher WIP Limits help to absorb variation and exploit opportunities. Teams using pair programming will have lower WIP Limits, as there is

Notes:

less simultaneous work in progress. By experimenting with various WIP Limits, you can identify, and resolve, process bottlenecks, adjust the impact of lead time, and create a predictable and efficient workflow. As a rule, your WIP Limit is too low if you hit bottlenecks every week, and too high if blockers don't cause an issue.

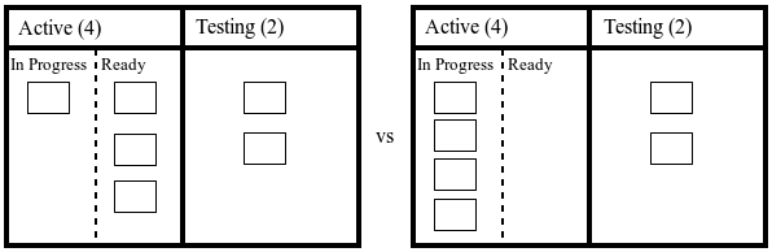


FIGURE 24: BOTTLENECKED VS. CLEAR KANBAN BOARDS

Kanban applies a form of Production Levelling to the process flow. This ensures that each step in the production process delivers at a constant rate and that no step in the production process produces goods at a faster rate than subsequent steps. Additionally, Kanban uses the five focusing steps from the Theory of Constraints as the incremental process improvement model. These are:

1. Identify the system's constraints: A bottleneck is an extreme example of a constraint. The slowest part of any process, no matter how smoothly it is working, will limit the throughput of the rest of the process.
2. Decide how to exploit the system's constraints: Keep the constrained state focused and busy at all times, by focusing on value adding work, removing impediments, and providing high-quality tools and materials.
3. Subordinate everything else to the above decision: All other states should not produce more than the constraint can process. This means that they have available capacity that can support the constrained state to focus on its core responsibilities.
4. Elevate the system's constraints: Once the constrained state has been fully exploited, the team, or organisation, needs to invest in additional capability, in order to increase its overall capacity.
5. If, in a previous step, the constraint was broken (e.g. it is no longer a constraint), go back to step one: At this point, the overall system throughput will have increased, and

Notes:

the system constraint will move to a new point. This encourages continuous improvement (Kaizen) within each team's processes.

Note: Each team will go through different iterations of their Kanban board, as their workflow and work processes improve and evolve. It is not unusual for a productive team, one that has embraced continuous process improvement, to go through several Kanban board designs a year. A team that does not change their Kanban board is probably only using it to track work, not as a method for improvement.

INSPECTION

As well as visualising progress, it is important to measure the time a task or story sits in the Backlog before being actioned, and the time task or story sits in each state (e.g. the time taken to move from Active to Testing, and from Testing to Done). The primary measures for this are Lead Time and Cycle Time. Lead Time and Cycle Time do not measure effort, but the elapsed time (or duration).

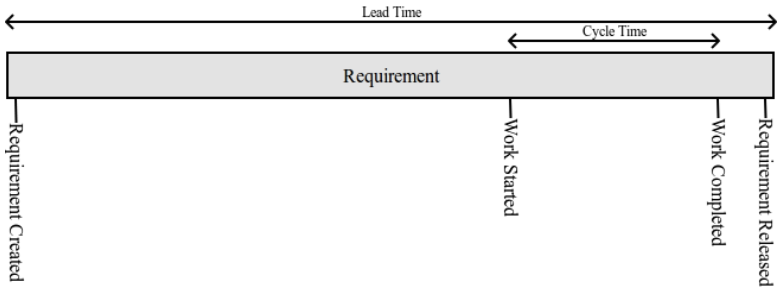


FIGURE 25: LEAD TIME VS CYCLE TIME

Lead Time is defined as the time taken between adding a story to the Backlog, and releasing it to the customer. Whereas, Cycle Time is defined as the time taken between starting, and completing, work on a story.

But, numbers alone don't provide enough useful information to manage your teams. Using Cumulative Flow Diagrams, and Cycle Time Run Charts, you can represent, and visualise, the scope of work, planned delivery, and actual delivery of tasks and stories. To ensure full transparency between your teams and customers, these charts should be available to everyone even remotely involved with the team.

Notes:

CUMULATIVE FLOW DIAGRAMS

A Cumulative Flow Diagram (CFD) visualises the flow of work over elapsed time and the number of tasks or stories remaining in each state. This is important in verifying the efficient delivery of work.

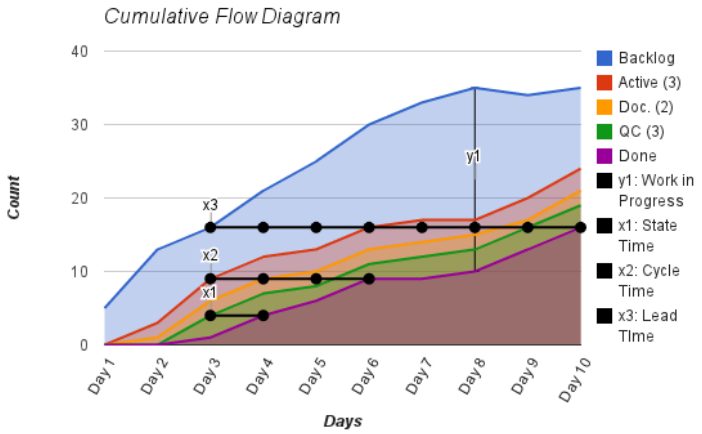


FIGURE 26: EXAMPLE CUMULATIVE FLOW DIAGRAM

The CFD tracks each state in your workflow separately, from when a card (or task) enters that state, to the time the card enters the subsequent state. The vertical distance (y1) in each charted line shows the number of tasks currently in progress. This distance should never be greater than the WIP limit for the state. The horizontal distance (x1) shows the time taken for a task to progress to the next state. The horizontal distance (x2) shows the Average Cycle Time, the time taken from when a card leaves the Backlog state, until is it done. The final horizontal distance (x3) shows the Average Lead Time, the time taken from when the card enters the Backlog, until it is done.

Each line on the Cumulative Flow Diagram should appear smooth; any flat vertical or horizontal generally indicates impediments, or an uneven flow of work. You can quickly, and easily, identify bottlenecks, when the area between two bands narrows or, in the worst case, reduces to zero. Keeping low WIP limits simplifies the identification of bottlenecks, when analysing Cumulative Flow Diagrams.

Notes:

BOTTLENECK

Identified by: A band reducing to zero.

Identified issue: The Documentation state is a bottleneck in the process, and has starved the Quality Control state of any work.

Resolution: Improve the delivery through the bottlenecked state by exploiting, subordinating, and elevating the constraint.

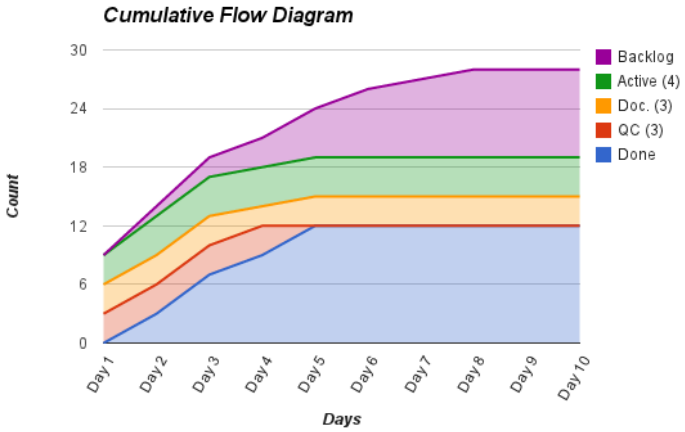


FIGURE 27: PROBLEM CFD (STARVED STATE)

POOR FLOW

Identified by: Jagged, widening, and narrowing bands, between two or more states.

Identified issue: Caused when there is not a smooth flow of work through the system. States that jump to the maximum WIP, and back down again, can also be indicative of bottlenecks, or other impediments, throughout the work processes.

Resolution: Identify the cause of the impediments of bottlenecks, and remove them, to improve the flow of work.

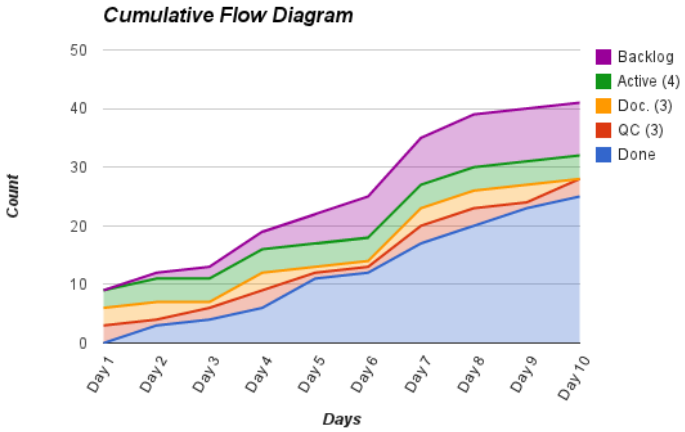


FIGURE 28: PROBLEM CFD (POOR FLOW)

Notes:

No, or large, WIP limit

Identified by: A large distance between each band, causing a false sense of smoothness.

Identified issue: This example actually shows the same variation as the 'Bottleneck' example above. However, as the WIP Limit is very large, it is difficult to identify any fluctuations in the chart.

Resolution: Reduce the WIP Limit to an appropriate number.

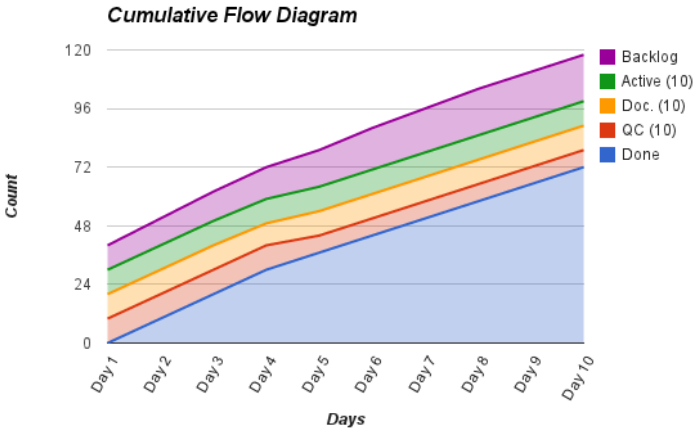


FIGURE 29: PROBLEM CFD (LARGE WIP)

Long lead time

Identified by: A very slow, and shallow, rise in all states.

Identified issue: There is a long lead time (and cycle time) between raising a story, and it being delivered to the customer.

Resolution: Reduce the WIP Limit, or reduce the size of the stories, to improve the speed of the workflow.

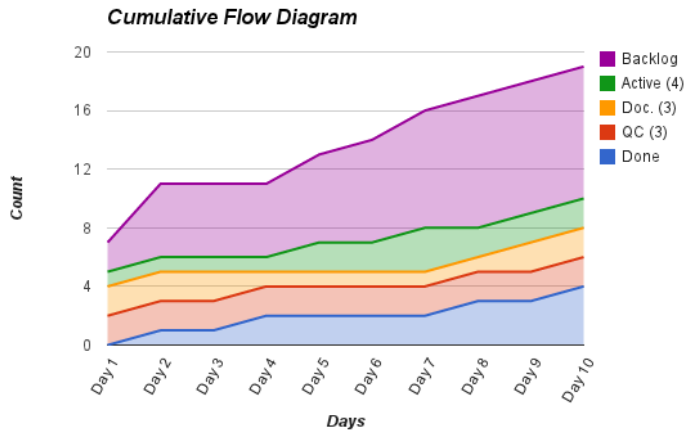


FIGURE 30: PROBLEM CFD (LONG LEAD TIME)

Notes:

PLATEAU

Identified by: Tracking well, followed by no visible progress for several days.

Identified issue: The flow of work has stopped (or dramatically slowed), usually caused by critical production issues, large-scale staff absences (e.g. Christmas holidays), or waiting for customer sign off.

Resolution: Identify what is causing the halt of workflow, and (if appropriate) resolve the underlying issue.

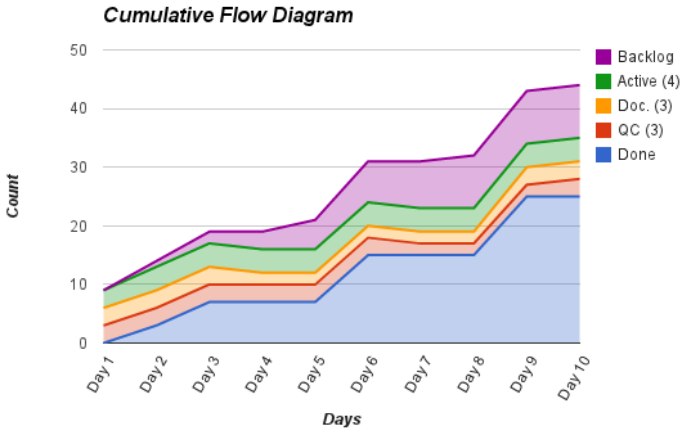


FIGURE 31: PROBLEM CFD (PLATEAU)

NOTHING WRONG

Tracking well in terms of consistent rise, and no major widening or narrowing of bands.

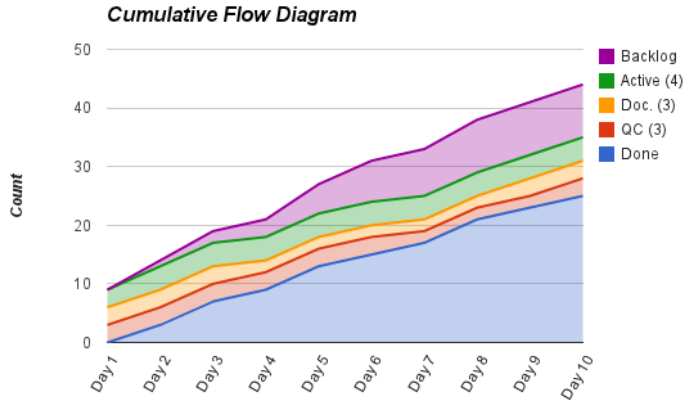


FIGURE 32: NORMAL CFD

Notes:

CYCLE TIME RUN CHARTS

Teams that measure Cycle Time and Lead Time can visualise these metrics using Cycle Time Run Charts (sometimes known as statistical process control charts). By looking for trends, cycles and outliers above expected tolerances, Cycle Time Run Charts will help you to identify both normal, and uncontrolled, variations in process flow.

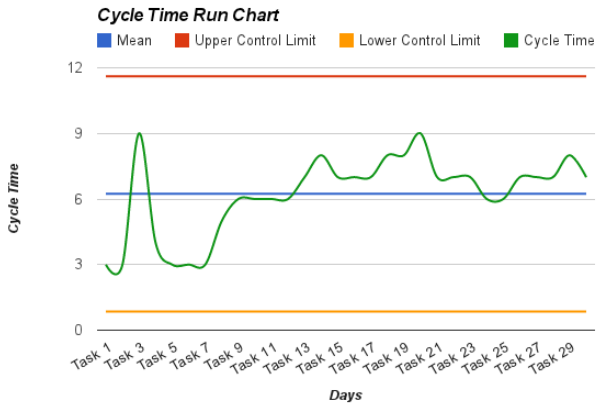


FIGURE 33: EXAMPLE RUN CHART

Run Charts plot the Cycle (or Lead) Time of each story, against the long-term average, known as the centre line. From a continuous improvement perspective, you should aim to improve your work processes, so the centre line (and thus your average Cycle Time) meets your customer's needs.

If you know the expected variance within your process (usually ± three standard deviations), you can plot the Upper Control Limit (UCL) and Lower Control Limit (LCL). These limits are the primary mechanism to identify special cause events. This means you should investigate anything beyond the limits, as they can indicate a process out of control.

It is simple to calculate Cycle Time Run Charts when all stories are of approximately equal size and effort. However, they can still be effective for stories of varying sizes, but will have higher Control Limits, and need a larger dataset to calculate a meaningful average.

Notes:

PROCESS TREND

Identified by: A run of points that are continuously increasing or decreasing.

Identified issue: There is a progressive trend in the Cycle Time, implying that something is gradually shifting over time.

Resolution: If the trend is in the right direction (usually down), it may be part of ongoing process improvement, and your team should sustain the change. Otherwise, your team needs to determine the cause of the variation, and resolve it.

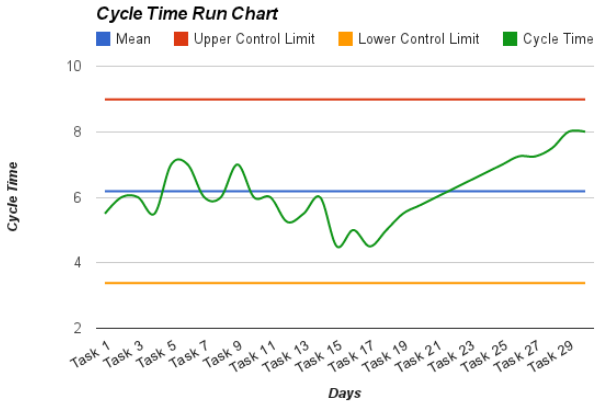


FIGURE 34: PROBLEM RUN CHART (PROCESS TREND)

PROCESS SHIFT

Identified by: A run of points on a single side of the centre line.

Identified issue: There is a sustained shift in Cycle Time, and may have reached a new equilibrium.

Resolution: If the shift is in the right direction (usually down), it may be part of ongoing process improvement, and your team should sustain the change. Otherwise, your team needs to determine the cause of the variation, and resolve it.

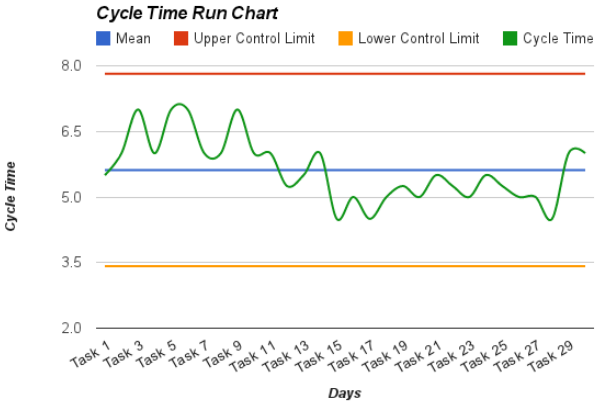


FIGURE 35: PROBLEM RUN CHART (PROCESS SHIFT)

Notes:

EXTREME PROCESS VARIATION

Identified by: Points above the UCL, or below the LCL.

Identified issue: Extreme variation (special causes) in the team's process that may indicate a process out of control.

Resolution: Identify the cause of the outliers, and if systemic, resolve the underlying issues.

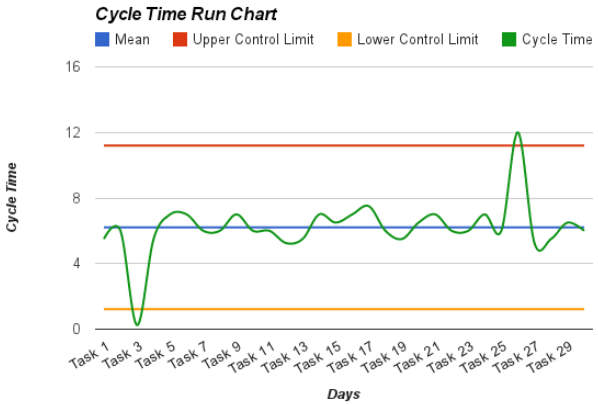


FIGURE 36: PROBLEM RUN CHART (EXTREME PROCESS VARIATION)

NOTHING WRONG

No major variations in process flow.

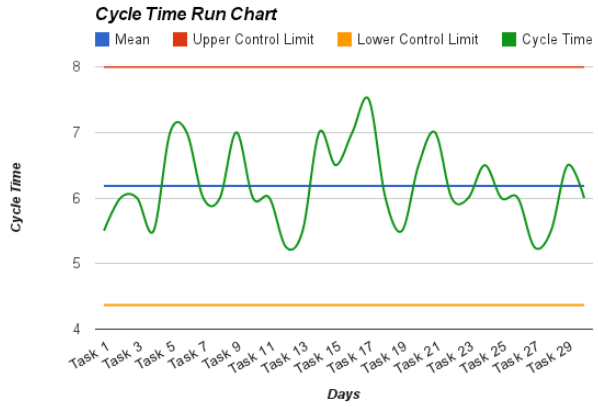


FIGURE 37: NORMAL RUN CHART

Notes:

KAIZEN

'Fall seven times. Stand up eight.'

Old Japanese Proverb

Notes:

All Agile processes are, by their very nature, cyclical, based on the inspect and adapt cycle. With very little additional effort, this continuous feedback can become ongoing improvement, and provide the mechanism for organisations to adapt to changing circumstances. This process of continuous improvement, or Kaizen, leads to a culture of continuous improvement.

There are five main elements to Kaizen:

1. Teamwork: All staff, regardless of rank or status, work together towards the common goals of your organisation. By extension, all staff, across every team, need to understand the implications of their work for the rest of the organisation, and share the responsibility for your organisation's success.
2. Personal discipline: Staff should be accountable for their actions. Not only for their core responsibilities, but for all aspects of their work, including quality control, time management, and their professional relationships with colleagues and customers. There is a corresponding requirement for organisations to set reasonable standards, and challenge staff to meet them.
3. Improved morale: Teams share responsibility to build an environment where they feel empowered, secure, and have a sense of ownership. An organisation with low morale, or conflict between managers and staff, will suffer from high absenteeism, poor engagement, and reduced productivity.
4. Quality circles/Retrospective workshop: The retrospective workshops are the primary forum for teams to suggest improvements to your corporate culture, delivery processes and management arrangements. Teams should be encouraged to hold cross-team retrospectives, as a means to share ideas, skills and technology improvements. Teams also need the authority to experiment with, and implement, local changes, and the organisation should be quick to respond to any large-scale suggestions that have implications beyond the team.
5. Suggestions for improvement: All business functions are candidates for Kaizen, and, as such, each team member has an obligation to participate in the continuous improvement process. Learning, observing, and putting forward new ideas, especially in relation to their core responsibilities, will help remove any impediments, and increase work efficiency.

Notes:

Kaizen is a truly continuous process, where teams should be seeking new ways of improving the business every day. Staff should be encouraged to experiment with different process changes, to drive continuous improvement.

The regular (weekly or biweekly), retrospective workshop, provides a formal forum for each of your teams for introspection, and reflection on the processes that support their development. The goal of this workshop is to suggest improvements, and the focus should be, in order of importance, people, relationships, process, and tools. The full team should be present for each retrospective, as they are ultimately responsible, and accountable, for driving process improvements in their team.

Teams should be free to discuss any relevant topic. During this short meeting, between an hour to half a day, the team should reflect on the processes since the last Retrospective. This may include:

- Discuss the processes that worked well, and were effective, or improved, since the last retrospective. By reflecting on the positive outcomes, the team can identify their strengths, and use those to overcome specific weaknesses. It is also important, as a mechanism, to provide positive feedback to the team.
- Discuss the processes that did not work as well as expected, and need improvement. By reflecting on the negative outcomes, the team can focus their effort on improving in that area, or make modifications to the process to better play to their strengths.
- Suggest any specific improvements to the processes used within the team. As mentioned, continuous improvement (Kaizen) is a core concept to Agile, and the team is responsible for driving most of this change. It is important to ensure that each improvement is actionable, and assigned an owner.

At the end of the Retrospective, the team should have a list of 'assigned' and 'actionable' improvements to the management processes.

Each retrospective provides the team with the opportunity to reflect on the time since the last retrospective, and drive continuous process improvement out of any learning's since then. Through this process of Kaizen, the delivery of each story should be more effective, and enjoyable, than the last.

Notes:

REFERENCES

Notes:

BOOKS & LINKS

Directing the Agile Organisation – Evan Leybourn

Kanban: Successful Evolutionary Change for Your Technology Business – David J. Anderson

Lean Software Development: An Agile Toolkit - Mary Poppendieck

Agile Estimating and Planning - Mike Cohn

Managing Agile Projects - Kevin J. Aguanno

http://en.wikipedia.org/wiki/Agile_software_development

http://agilemanifesto.org/

http://en.wikipedia.org/wiki/Agile_testing

http://en.wikipedia.org/wiki/Scrum_(development)

http://www.ambysoft.com

http://en.wikipedia.org/wiki/Iterative_and_incremental_development

http://en.wikipedia.org/wiki/Agile_software_development

http://www.agileadvice.com/

http://en.wikipedia.org/wiki/IEEE_829

http://www.ddj.com/architect/201202925?pgno=4

http://www.scrumalliance.org

TOOLS

http://trac.edgewall.org/

http://watir.com

Notes: