Agile Business Intelligence

Student Guide

|  |  |
| --- | --- |
| C:\Users\Evan\Dropbox\Speaking and Presentations\Training\cc-by-sa.png | Agile Business Intelligence 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](mailto:evan@theagiledirector.com)  Twitter: @eleybourn |

# C:\Users\Evan\Dropbox\Writing\directing the agile organisation\Promotion\Book Cover - Directing-the-Agile-Organisation-338x400.pngOther Works by Evan Leybourn

**Directing the Agile Organisation – by Evan Leybourn**

[**http://theagiledirector.com/book**](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

[Other Works by Evan Leybourn 2](#_Toc420956678)

[Table of Contents 3](#_Toc420956679)

[Introduction to Agile 5](#_Toc420956680)

[The Agile Manifesto 6](#_Toc420956681)

[Lean principles 7](#_Toc420956682)

[Agile methods 7](#_Toc420956683)

[Key points 8](#_Toc420956684)

[Understanding waste 9](#_Toc420956685)

[Common misconceptions 9](#_Toc420956686)

[Business intelligence project team 11](#_Toc420956687)

[Critical success factors 15](#_Toc420956688)

[Flow 16](#_Toc420956689)

[The origin of Lean 17](#_Toc420956690)

[An overview of Kanban as a BI method 18](#_Toc420956691)

[Task workflow 19](#_Toc420956692)

[Value Stream Mapping 22](#_Toc420956693)

[Kanban boards 24](#_Toc420956694)

[Project velocity 27](#_Toc420956695)

[Lag and lead indicators 28](#_Toc420956696)

[Cumulative flow diagrams 29](#_Toc420956697)

[Cycle time run charts 33](#_Toc420956698)

[Burndown & Burnup Charts 36](#_Toc420956699)

[BI basics 39](#_Toc420956700)

[The data warehouse 40](#_Toc420956701)

[ETL 44](#_Toc420956702)

[Reporting 47](#_Toc420956703)

[Technical Excellence 49](#_Toc420956704)

[Business intelligence: creating information from data 50](#_Toc420956705)

[Incremental data acquisition 52](#_Toc420956706)

[Incremental design and enterprise architecture 59](#_Toc420956707)

[Constraining principles 65](#_Toc420956708)

[Continuous integration, automated unit testing and continuous deployment 67](#_Toc420956709)

[Agile Project Planning 70](#_Toc420956710)

[Beginning the process 71](#_Toc420956711)

[Fixed time, cost and scope 72](#_Toc420956712)

[Scrum as a business intelligence product and project framework 73](#_Toc420956713)

[Product backlog 76](#_Toc420956714)

[Kaizen (改善) 79](#_Toc420956715)

[Product review 81](#_Toc420956716)

[Daily stand-up 81](#_Toc420956717)

[References 82](#_Toc420956718)

[Books and links 83](#_Toc420956719)

[Tools 83](#_Toc420956720)

# Introduction to Agile

‘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

## 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". These values and principles apply regardless of the product being produced, in this case data warehousing and business intelligence products (just replace software with business intelligence).

### 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.

## 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)

## 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]](#footnote-1) are an excellent example of the differences between traditional (or phased) development vs. the agile approach of iterative development.

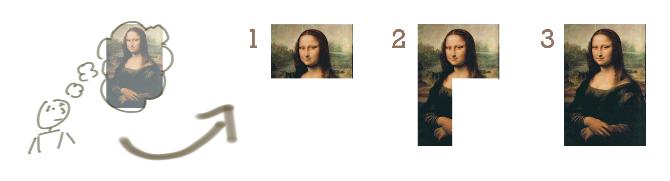


Figure : The Traditional Approach (Phased delivery of known outputs)



Figure : The Agile Approach (Iterative delivery to meet changing EXPECTATIONS)

## 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).

## 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 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 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.

## Business intelligence project team

In high maturity agile environments, the business intelligence 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 agile business intelligence project team can be comprised of the following:

* Customer representative or product owner (Scrum)
* Team facilitator or scrum master (Scrum)
* Data analysts and architects
* Data modellers
* ETL developers
* Report developers

Note that these are roles, not necessarily people. An individual may (and in many cases should) hold more than one of these roles.

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.

Table : Example Team Roles

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Primary Responsibility | Typical | Does Not |
| Users  *Interested Role* | * Use the reports * Identify data 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 * Test features | *cross functional*   * Developer * Designers * Writers * Administrators * Testers | * Prioritise features * Test their own code |

### Interested and committed

Agile distinguishes between business and technical roles; generally referred to as interested or committed.

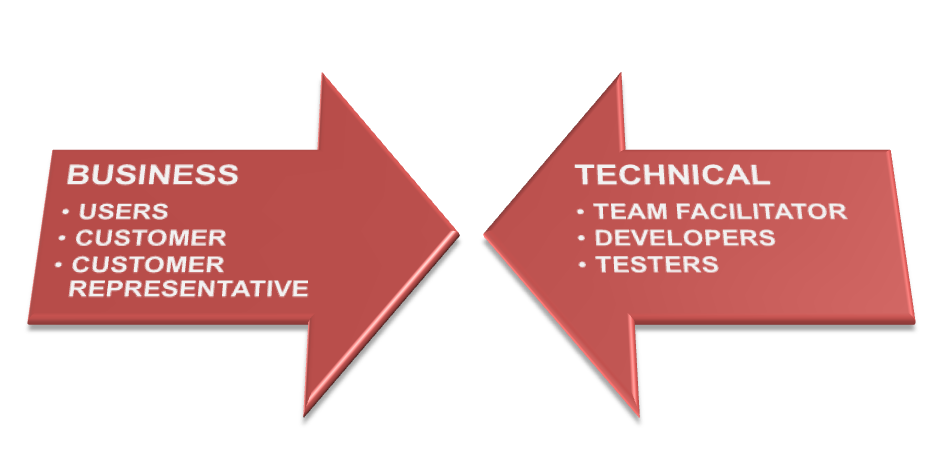


Figure : Business vs Technical Roles

Interested roles are individuals who have an “interest” in the 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.

Committed roles are responsible for the development, and are the people who “do” the work. Committed parties include the team facilitator, the team and testers.

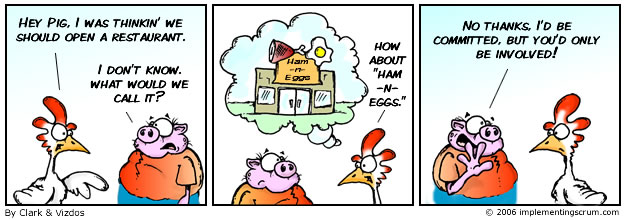


Figure : Pigs and Chickens

### Deming’s 14 points for managers

I’d like to end this section with a look at some of W. Edwards Deming’s 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.
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.

## Critical success factors

The successful application of agile to business intelligence 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.

# Flow

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

Sun Tzu, ~6th Century BCE

## 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’.

This is perfect for many business intelligence workflows.

Lean 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 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.

## An overview of Kanban as a BI method

The original concepts of Kanban (カンバン) were developed in the 1940s and 50s by Taiichi Ohno[[2]](#footnote-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]](#footnote-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.

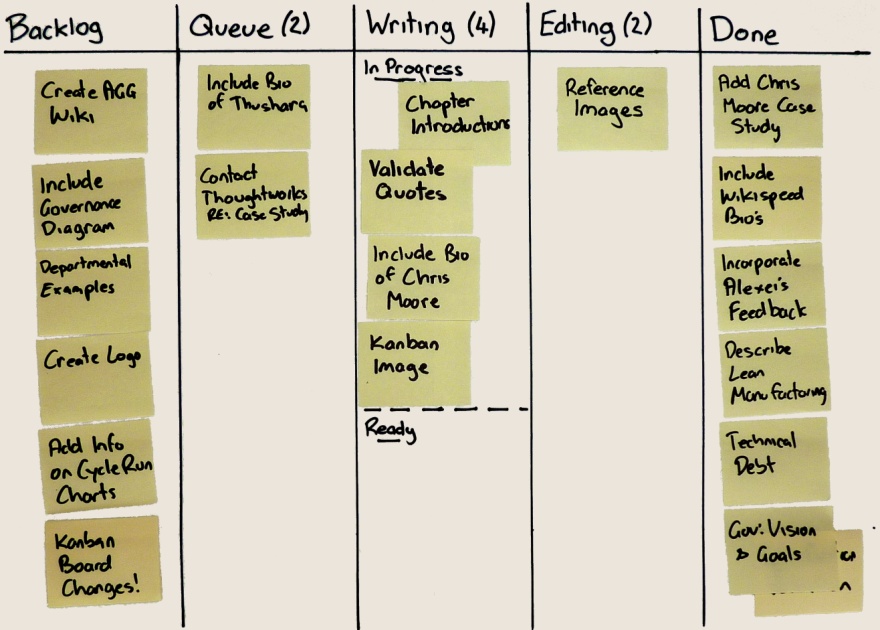


Figure : Example Kanban Board

While a Kanban workflow can become very complex, the simplest visualisation of workflow has only 3 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. 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

## Task workflow

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 several 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.

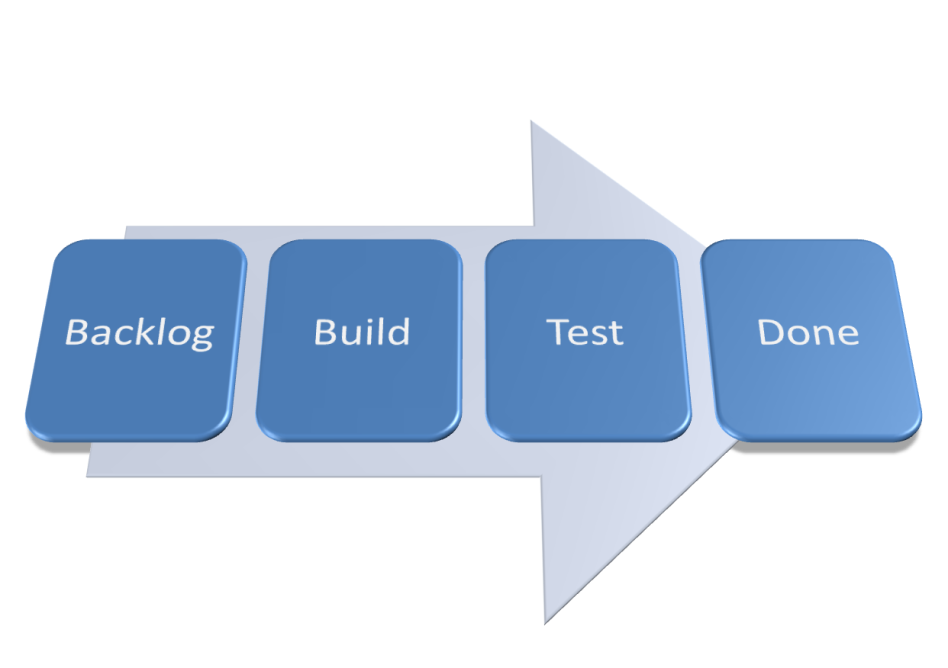


Figure : Basic Task Lifecycle

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.

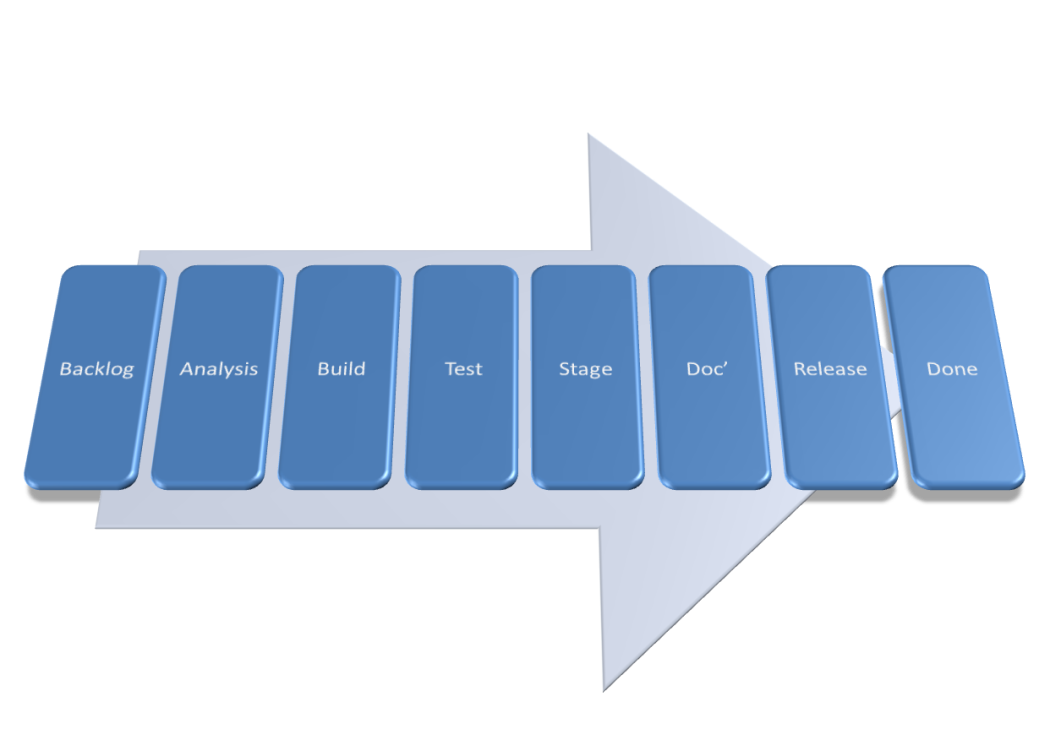


Figure : Complex Task Lifecycle

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

* Analysis
* Data modelling
* ETL development
* Unit testing
* Integration testing
* Report writing
* Documentation
* Deployment

Depending on the complexity of the workflow, a task may move through these steps individually, or they may be treated as a single step followed by Done.

### Done

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 based on the agreed definition of done. 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.

## 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.

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. 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).

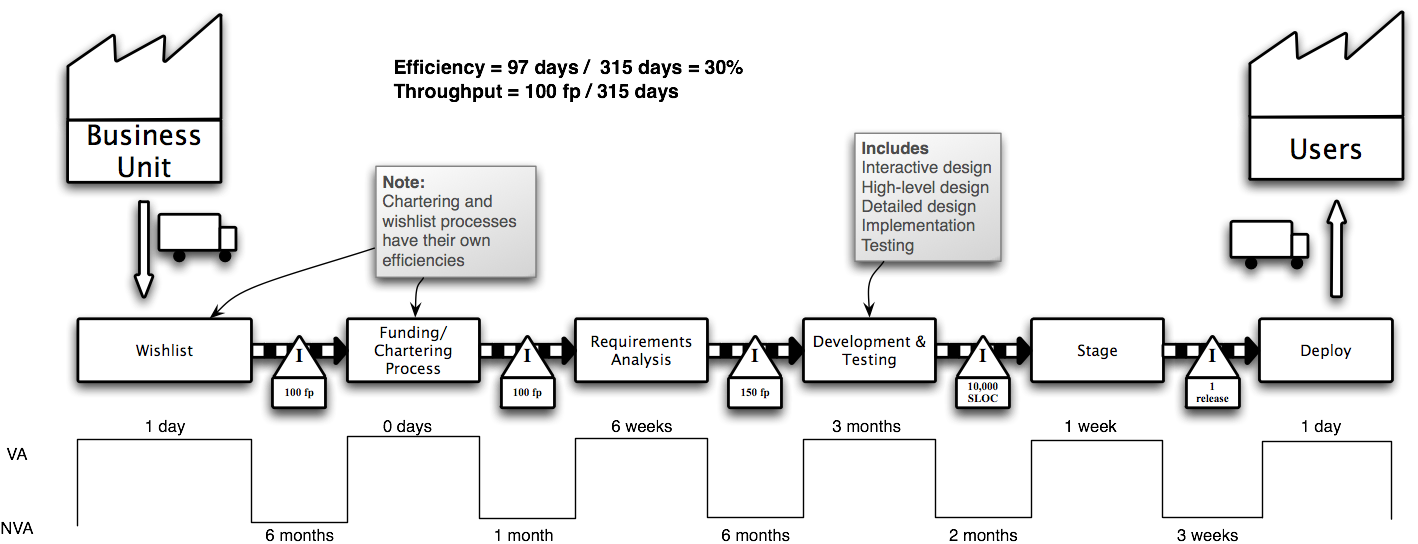


Figure : Example Project Value Stream Map[[4]](#footnote-4)

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.

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;

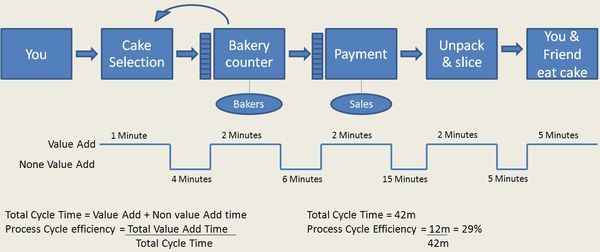


Figure : Value Stream Map for Buying and Eating Cake[[5]](#footnote-5)

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%.

## 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 : Example Kanban Board[[6]](#footnote-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 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.

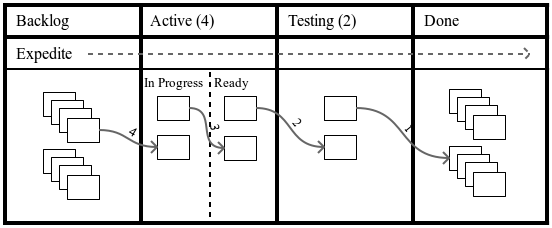


Figure : Kanban Board and Flow

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 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.

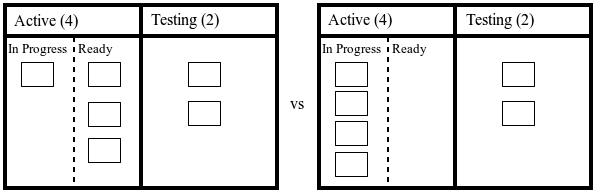


Figure : Bottlenecked vs. clear Kanban Boards

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 needs to invest in additional capability, 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 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.

## Project velocity

Project Velocity is a measure that determines the capacity and resulting output from the project team over time. In this case it refers to how many user stories the project team estimate that they can deliver in the project.

The methods available to determine Project Velocity are as follows:

* Use historical values - this assumes similarity between previous user stories in terms of relative size, data, infrastructure etc
* Run an Iteration – derive the estimate based on observed Velocity over 2-3 Iterations
* Make a forecast – this approach is used where we do not have existing historical values and it is not feasible to run several Iterations to observe the team’s velocity. User stories are disaggregated and estimated at a task level.

The scope of the project, any changes to the scope, and the planned and actual delivery of user stories can be represented in a burndown or burnup chart. This chart can also be used to forecast when the project team will have completed all the user stories in the product backlog.

## Lag and lead indicators

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).

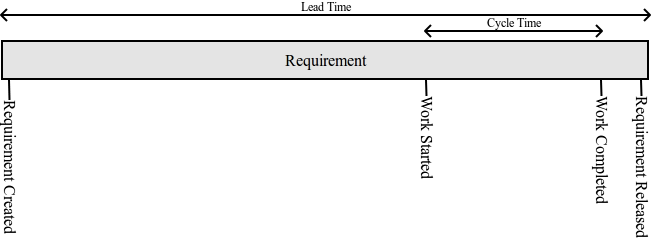


Figure : Lead Time vs Cycle Time

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.

## 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.

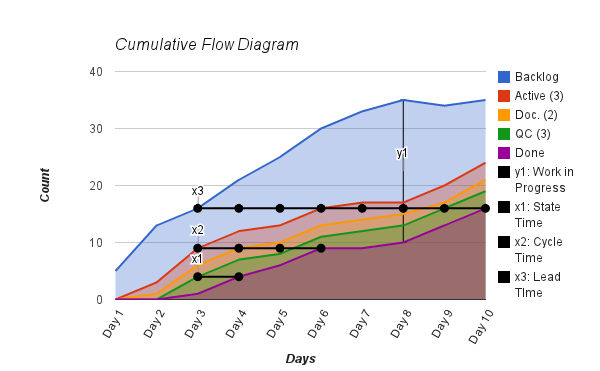


Figure : Example Cumulative Flow Diagram

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.

|  |  |
| --- | --- |
| 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. | J:\Downloads\cfd_bottleneck.png  Figure : 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. | J:\Downloads\cfd_poor_flow.png  Figure : Problem CFD (Poor flow) |
| 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. | J:\Downloads\cfp_no_wip.png  Figure : 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. | J:\Downloads\cfp_long_lead.png  Figure : Problem CFD (Long lead time) |
| 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. | J:\Downloads\cfp_plateau.png  Figure : Problem CFD (Plateau) |
| Nothing wrong Tracking well in terms of consistent rise, and no major widening or narrowing of bands. | J:\Downloads\cfd_nothing_wrong.png  Figure : Normal CFD |

## 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.

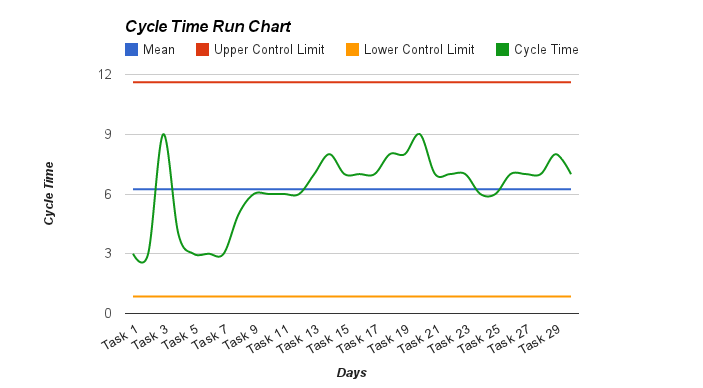


Figure : Example Run Chart

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.

|  |  |
| --- | --- |
| 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 : 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 : Problem Run Chart (Process shift) |
| 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 : Problem Run Chart (Extreme process variation) |
| Nothing wrong No major variations in process flow. | Figure : Normal Run Chart |

## Burndown & Burnup Charts

Burndown and Burnup charts help managers and customers to view progress against the release, improve future estimates and identify problem trends early. The Burndown (or burnup) chart should be available to everyone involved in the project.

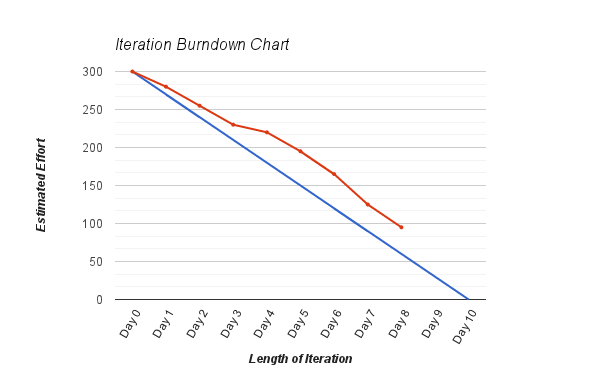


Figure : Example burndown Chart

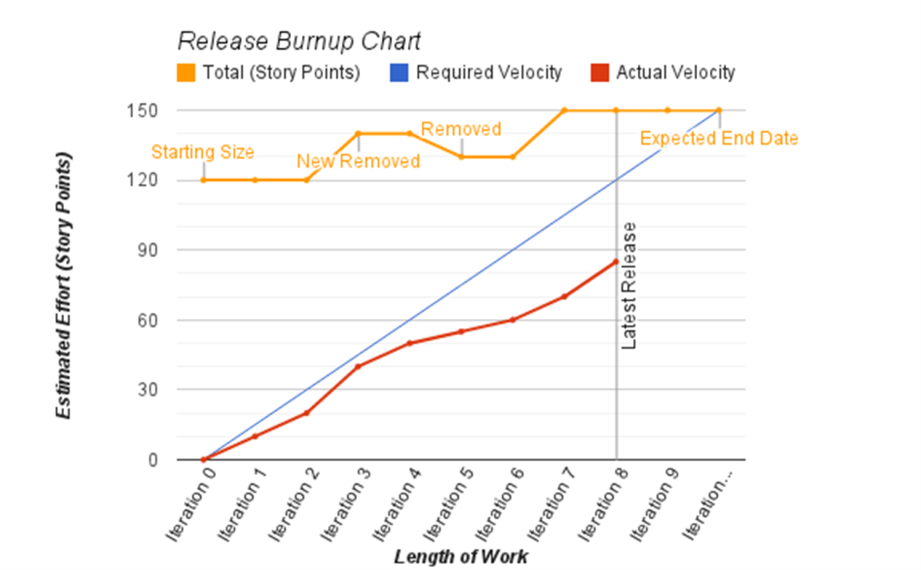


Figure : Example Burnup Chart

|  |  |
| --- | --- |
| Discovery Issues identified after the iteration begins or refined estimation after the iteration begins. Watch the progress carefully. If necessary review the tasks in the Iteration | burndown_discovery.png  Figure : Problem Burndown (Discovery) |
| Scope creep Tasks are being added mid-release. Identify who is adding tasks and stop this behaviour. | burndown_scope_creep.png  Figure : Problem Burndown (Scope Creep) |
| Plateau Features are more difficult than estimated or unexpected staffing issues. Review the tasks in the iteration. | burndown_plateau.png  Figure : Problem Burndown (Plateau) |
| Too many features Features are more difficult than estimated. Review the estimation process and remove tasks from the iteration. | burndown_too_much.png  Figure : Problem Burndown (Too Many features) |
| Tracking epics Individual stories are too large and difficult to track. Keep each task under 1 day of work. | burndown_tracking_epics.png  Figure : Problem Burndown (Tracking Epics) |

# BI basics

‘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)

## The data warehouse

Fundamentally the overall design of a data warehouse does not change if developed using an agile or non-agile approach. All data within a business intelligence system is stored in a central, read-only, repository called a “data warehouse” or “consolidation database”.

A data warehouse allows an organisation to;

* Gain access to valuable data from multiple sources from a central location, increasing efficiency and reliability.
* Standardise and simplify the way information is managed within an organisation.
* Develop cross-application reporting to improve business intelligence.
* Identify obsolete, duplicate and redundant data across multiple applications.

Periodically, the extraction system will extract data from your sources and insert it into the data warehouse. As this will store all the data from every source it will become very large. Historical information from these sources must be kept, so an incremental extraction/insertion system is the best solution.

As one of the most important aspects of a business intelligence system is the ability to combine multiple sources into a single data warehouse there are some issues that should be considered.

* How to resolve conflicts between data from different sources relating to the same record.
* How to correlate records that may have different primary keys to identify the record.
* How to accurately and efficiently store the data in the consolidation database, such that it does not duplicate existing records.
* How to differentiate identical primary keys that refer to different records in the same consolidation table. (E.g. companies and universities in an organisation table.)

When a business intelligence system is being designed, it is important to keep these problems in mind. How you resolve these problems is specific to each data source.

### Data marts

Data marts are small, usually aggregated, subsets of the data warehouse. Consolidation databases are generally slow to query. They are designed for bulk inserts whereas data marts are databases designed for queries and reporting. If done well, they can be orders of magnitude faster to query than the source systems.

Data marts are very specific to individual reporting needs, so they must be designed with those needs in mind. However remember these four things:

1. The data marts should also contain historical / incremental data, so the table structure should be similar to the consolidation database.
2. When designing data marts forget everything you know about normalisation. Merge table, duplicate data, anything goes. Although counter-intuitive, it will greatly improve performance.
3. Finally, make judicious use of indexes. As this is not a transactional system and is primarily used to report on data, the overhead of indexes on inserts and updates is insubstantial.
4. Your denormalised schema for the Data Mart should be purpose designed for reports from a subset of the data.

### Star schemas



Figure : Example Star Schema

The star schema is perhaps the simplest data warehouse schema. The centre of the star consists of a large fact table, with a compound primary key, with one segment for each dimension and additional facts.

The points of the star are the dimension tables, or lookup tables, each of which contains information about the entries for a particular attribute in the fact table.

A good reason for using star schema is its ease of understanding. Fact tables in star schema are mostly in third normal form (3NF) meaning that there is no duplicate data across any table, but dimensional tables are in de-normalised second normal form (2NF) which has some duplication of data. If you want to normalise dimensional tables, they look like snowflakes and the same problems of relational databases arise - you need complex queries and business users cannot easily understand the meaning of data.



Figure : Example snowflake schema

Snowflake schemas normalise dimensions to eliminate redundancy. That is, the dimension data has been grouped into multiple tables instead of one large table. While this saves space, it increases the number of dimension tables and requires more foreign key joins. The result is more complex queries and reduced query performance.

Also, many business intelligence systems are designed to be used by business users, and the added complexity of the snowflake schema will often preclude non-specialist users from forming their own queries.

### Historical data

Producing reports based on historic trends is critically important for most organisations and one of the primary advantages of a good business intelligence system. Each record in the data warehouse has a start and end timestamp attribute. These timestamps detail when a record was inserted and when it was updated or deleted from the source.

Each group of like records make up an incremental snapshot of a single data source record. If a specific record/row in a data source has been:

* Unchanged: Update the end timestamp of the current row to the current timestamp.
* Updated: Create a new row in the data warehouse with the start (and end) timestamp set to the current timestamp. The old row should stay the same with the end timestamp marking the date the record was obsoleted.
* New (Inserted): Create a new row in the data warehouse with the start (and end) timestamp set to the current timestamp.
* Deleted: This row will never be extracted from the data source (because it doesn't exist). Thus the end timestamp should mark the date the record was last extracted.

Many data warehouse systems will utilise a series of triggers to perform these incremental extracts. These triggers will run through the inserts and check them against the current row, updating the end timestamp as they go. The current row is the one where the primary key matches and there is no end timestamp.

### Indexes

A database index is a data structure that improves the speed of operations in a table. Indices can be created using one or more columns, and for every insert or update those columns get duplicated in the index in alphabetical order.

Indices increase the time taken to insert and update a record within the database, but significantly reduce query time. In general, indices are dropped during a bulk load (ETL) operation and recreated afterwards. A data warehouse is, by its nature, a highly query-based system and as such indices greatly increase efficiency.

Indices are automatically created for all unique columns and all primary and foreign keys. Additional indices can be added for any columns that are frequently used within query constraints.

## ETL

### Extraction

The first step in the ETL process is extraction. Extraction copies data from a data source on a periodic basis. Each different source type has its own extraction methods.

* Databases: Data is extracted via remote SQL queries, usually an ODBC connection.
* Documents: Data is extracted via local directory paths or remote URLs, usually a HTTP/S connection.
* Websites: Data is extracted via remote URLs, usually a HTTP/S connection.
* Log Files: Data is extracted via local directory paths or remote URLs, usually a HTTP/S connection.
* Email: Data is extracted through an email gateway, usually a POP or IMAP connection.

Most ETL applications require a series of control files to be written which control the extraction (as well as the transformation and load).

### Transformation

The second step in the ETL process is transformation.

Transformation iterates through the extracted data and applies a series of predefined rules to each row of data. Most ETL tools have hundreds of transformation functions. For example;

* **Adding arbitrary content**: Arbitrary columns, with pre-set values, can be added to the consolidation database during the extraction process. This is most often used to add descriptive details alongside extracted information.
* **Removing arbitrary content**: Any column that has been extracted from the data source can be explicitly deleted from the load. If your extraction query contains a wildcard (\*), then you may wish to delete extraneous columns.
* **Modifying the content of the cell**: Any column that has been extracted from the data source can be modified and transformed into any style you require. Different applications may do this in different ways, but commonly use regular expressions.
* **Splitting a cell into two**: Your ETL system needs to be able to split apart a data based on a given delimiter. The delimiter can either be a string or a regular expression.
* **Joining cells together**: Your ETL system needs to be able to join multiple columns together from the source data into a single column in the Data Warehouse. The join should be separated by an optional delimiter.
* **Dropping an entire row from insertion**: For data validation purposes you must be able to drop entire rows from the load based on the value, or transformed value, of a given column. This ensures that your data warehouse (or Data Mart) is in a stable and consistent state at all times.

During this stage the ETL tool performs all the validation, integrity and standardisation checks.

* **Data validation:** Often, applications do not perform sufficient validation checks when accepting user input. Off-the-shelf software is particularly prone to validation errors. The extraction process can validate the data and raise a flag if it fails. Optionally, it may also automatically correct and insert the invalid record into the data warehouse.
* **Data integrity**: Organisations which have large datasets and multiple database systems often have problems with data integrity. Minor corruptions and data errors can develop slowly over time, which can cause large reporting errors, and even corrupt an entire data system. Many older applications are prone to these integrity problems. The extraction process should be able to automatically identify most of these problems and should alert the administrators to these errors.
* **Data standardisation**: With data from multiple sources all stored within the consolidation database, it is important to be able to specify a data convention that can apply across every system. During the extraction process it is important to identify similar and duplicated data across all available data sources and ensure it is transformed into a standard form. Common standardisation include;
  + Gender: Male / Female => M / F etc.
  + Case Sensitivity: university => University
  + Addresses: Line 1 / Line 2 => Company / Street

All erroneous records should be flagged for administrator or DBA attention, to be addressed as early as possible. Erroneous records should always be resolved in the master data source to ensure stable and reliable data. These erroneous records can also identify logical flaws in the design of the business intelligence system, and are important to address early in the development cycle.

### Load

The final step in the ETL process is to populate the transformed records into the Data Warehouse. As each row is inserted into the consolidation database, the data warehouse triggers are fired.

Depending on the DBMS and business intelligence application used, these triggers can perform additional data validation and the incremental checks.

Depending on the complexity of the triggers and the indexes on the database tables, this process can be very time-consuming. Depending on the database design, it can be faster to drop all the indices on the tables for the load and recreate them after the ETL process.

### Historical ETL

In principle, a business intelligence system should contain an historical log of all changes made within the original data sources. Depending on organisational requirements, it is very important to retroactively insert historical data. Note that this is a very time consuming process, and can take weeks or months to complete a full historical regeneration, when dealing with large amounts of data.

There may also be issues with the completeness of the historical data. Not all data sources will store historical data or may have incomplete or simplistic historical data. For example;

* A record may store the fact that it is no longer current, e.g. cancelled, but not when.
* A record may store the last modified by and modified time, but not the actual changes.
* A record may store the changes in an audit log file, which can be difficult to process and extract from. Especially if they use natural language as part of the record. e.g. Person Name has Changed To X From Y.

Ideally the changes of a record will be stored in a series of incremental records.

## Reporting

There are three critical elements required for a business intelligence system to be useful to an organisations decision making capability.

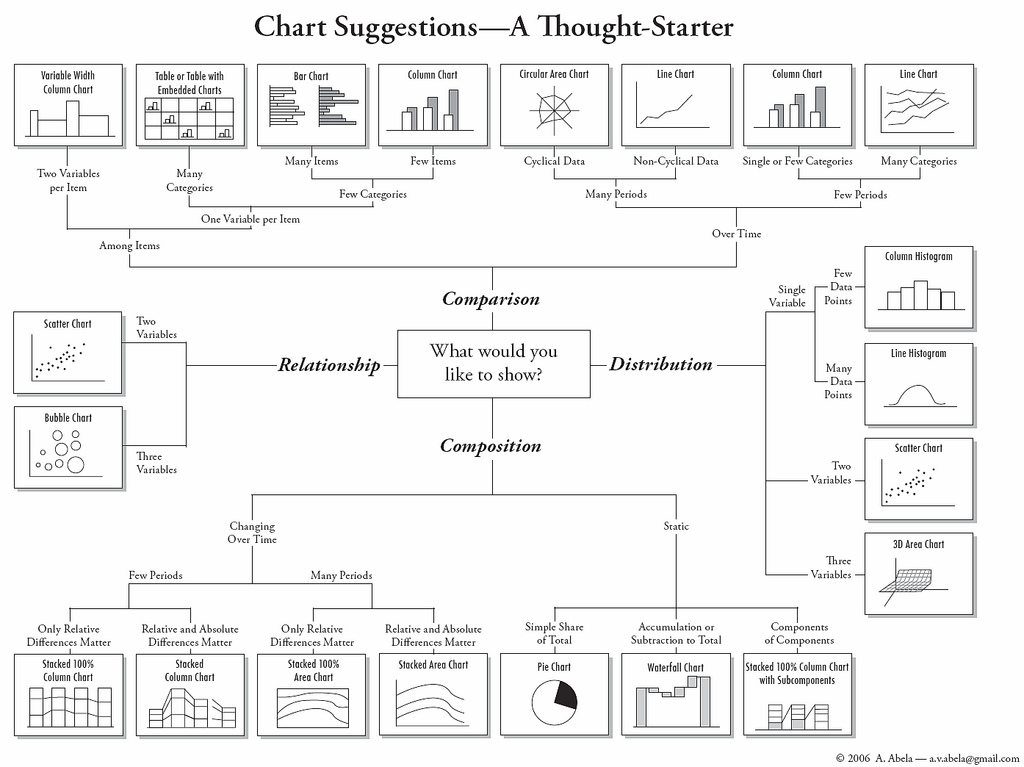
1. The data sources that supply the data to the business intelligence system must be appropriate to the business requirements. Irrelevant information can complicate the business intelligence system and in the worst case provide misleading information.
2. The input data must be cleansed and corrected. Incorrect or misleading data will corrupt the results of any reporting.
3. Most importantly, the person writing the report must understand the data, what it means and how to extract that information.

Any user who is expected to report on data from the business intelligence system must be given training on the usage of the business intelligence software, and just as importantly on the available data and structure within the data marts. By training a user in the data structure and content, as well as providing reference materials (such as a simplified data dictionary) users can create reports effectively and accurately.

In general, business intelligence systems provide three primary types of reporting capability.

1. Ad-hoc reporting: This type of report is written to answer a single business requirement. Normally it does not need to be run again, and can be discarded after use.
2. Scheduled Reporting: This type of report is written to meet a general business requirement. Once written is run on a scheduled basis where users can view the output. Users who do not have permission to create their own reports can often access the pre-run scheduled reports. It also allows complicated and processor intensive reports to be run during non-peak times.
3. Dashboards: Not all reports from a business intelligence system are comprehensive analytical reports. A report can be as simple as a single number or percentage to represent a Key Performance Indicator (KPI) for the organisation. By integrating the KPI reports from a business intelligence system with a digital dashboard managers can rapid and real time statistics about an organisation.

Most reports display aggregated information to represent an organisational fact. In many cases, BI tools have the ability to “drill down” into sub-reports to view detailed information regarding each data point. This functionality allows users to look at an organisation level of the data and see trends and statistics and to identify a data point of interest and further interrogate the information.



### Data access control

Data security and access control is critical in any data-rich environment. Sensitive and personal information should not be available to users who do not have a need to know.

Users should have their access to the data and stored reports restricted based on the business requirements and trust level. At the simplest level, an Access Control List (ACL) will list all users with permission to access a particular dataset. Restrictions can be implemented at the database, table or column level. Fine grained restrictions allow for a lot of control, however more administration required to maintain them.

# Technical Excellence

‘Make everything as simple as possible, but not simpler.’

Albert Einstein (paraphrased), 1933

## Business intelligence: creating information from data

### The phases of agile business intelligence

Designing a business intelligence system can be a complicated and time-consuming process. There are many factors, both technical and organisational, to consider and it is not always possible to resolve all of these in a timely manner.

In general, there are four major areas to designing and creating a business intelligence system;

1. Analysis and reverse engineering of data sources
2. Design of the consolidation data warehouse and data marts
3. Extraction, transformation and load of the data
4. Business level reporting on the data

In a traditional data warehouse environment these phases would be treated as sequential steps, with a major release of business reporting requirements at the end.

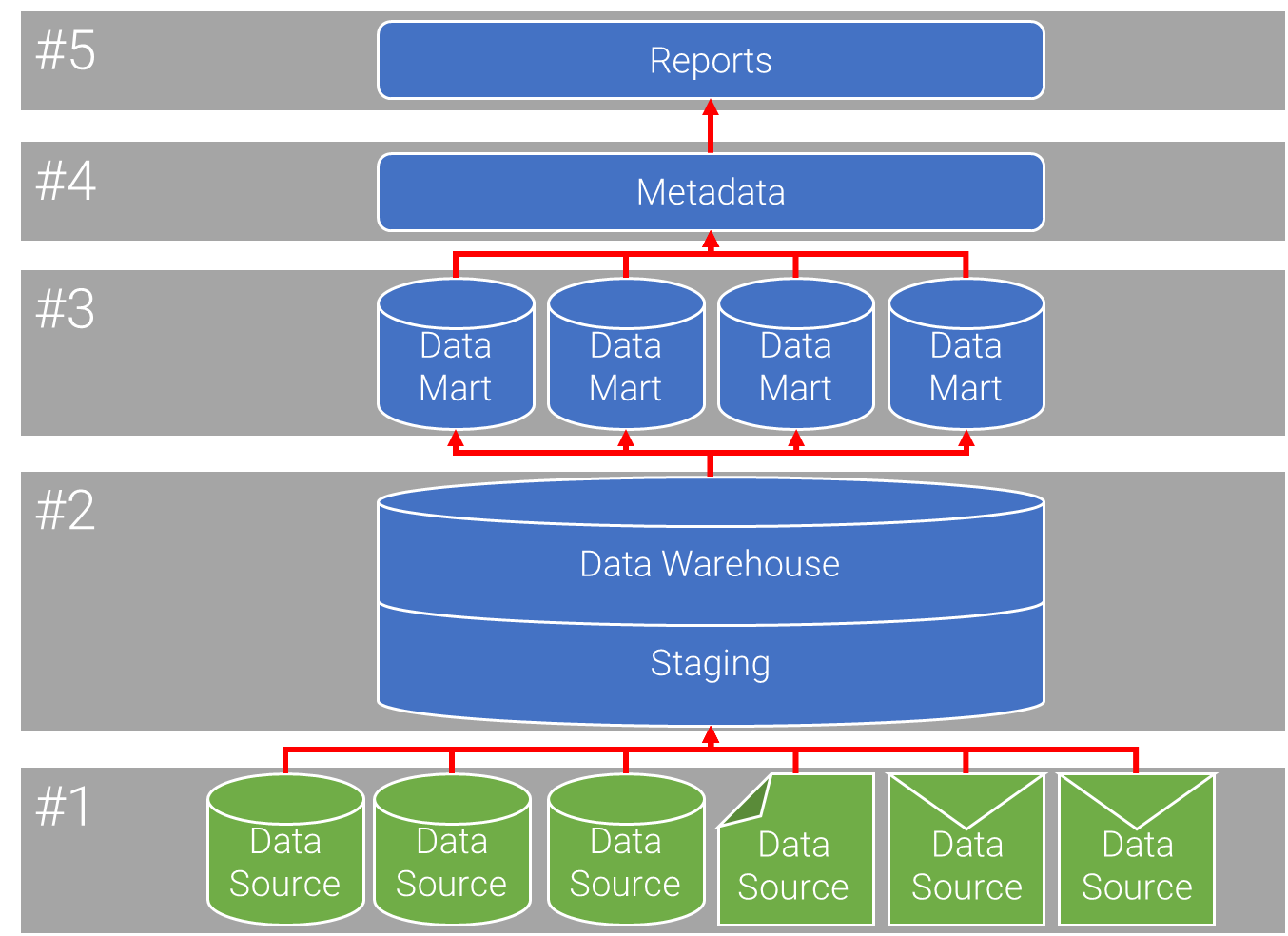


Figure : Steps in a Generic Waterfall Approach

But this isn’t the only way of designing and building a data warehouse. Using an iterative approach, where you rapidly iterate through these phases, you can have a production business intelligence system up and running in a short period of time.

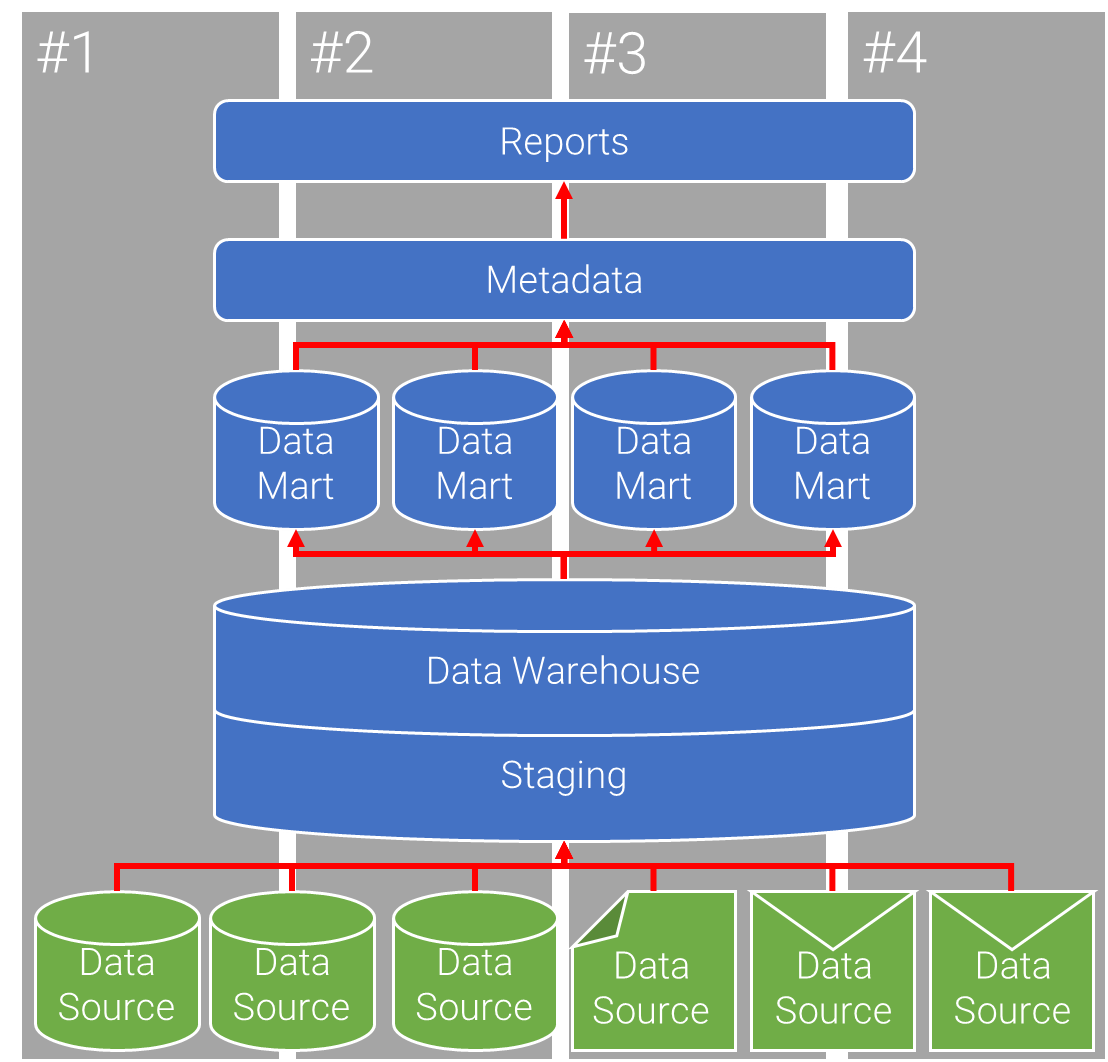


Figure : Steps in a Generic Agile Approach

It is good to think of business intelligence as a process, not a product. One of the most important lessons regarding data warehousing, and business intelligence in general is that it is never complete. Even with a non-agile approach, the new data sources need to be analysed and added, custom reports regularly need to be written, data validation and integrity checks need to be performed and standard database maintenance and tuning needs to be performed.

Fundamentally, the outcome of an agile business intelligence approach is no different from any other approach. Turning raw data into valuable information. For the sake of clarity, this course will use the terms data and information convey different meanings.

1. Data is the raw output of any database, website, log files or other data source.
2. Information is the processed and refined version of the data for human usage.

### What is your unit of work?

The question then becomes, what is your cadence or unit of work? The answer depends on how your organisational demands, but could be;

* **A requirement or request**: where each requirement triggers a prioritisation, development and deployment cycle. Not that there can be a major variance in effort between requirements depending on whether it is a simple report or requires modification to the data warehouse.
* **A report only**: like above, but limited to reporting only – data warehouse customisation is where each new report triggers a separate development process.
* **A data mart**: where you focus on developing and releasing specific data marts according to a determined priority.
* **A data source**: where you focus on a single data source and take it through all four areas of development. This is the slowest, but most predictable, unit of work.

Regardless of the cadence, you will start an agile business intelligence approach like you would any other – by identifying the information requirements of your organisation.

## Incremental data acquisition

### Information management

Like all business intelligence development, agile requires good data to start with. This means that without good information management throughout your organisation, the results from the business intelligence system are untrustworthy and can, in extreme cases, harm your organisation. Keep in mind the following concepts;

* Ensure your data is consistent throughout its entire life-cycle, especially if you have multiple databases that track separate business tasks.
* Keep all records up to date, and validate against external sources as often as possible. A good example of this is address validation, customer addresses can be validated against your countries central postcode database
* Prioritise your information, and treat important information accordingly
* Train your staff in information management policies – A good worker who documents and accurately records their work is better than a great worker who doesn't.

Many large organisations still use Excel or Access to manage their data, especially if it is from non-core business area. With cloud based information systems incredibly cheap to buy or have custom built, there should be no excuse not get these business areas to upgrade.

A business intelligence system can help identify issues in your information management policies in two ways.

* During the business analysis phases it can identify business areas with weak information management systems.
* Once operational, reports can be compared with the expected results, and any anomalies can imply quality issues with the source data.

### Data acquisition

Data acquisition can be one of the hardest parts of a business intelligence process. Business areas do not want to share raw data, external data sources are slow and you have very little control over them, and you don't have access to competitor’s data. This comes down to three types of data.

1. Data you own
2. Data you can access
3. Data you can infer.

Data you own is the easiest form, these are data sources that belong to you organisation. Depending on the organisational structure, you may need to put cross-department data sharing policies in place, but that should be one of the first steps in a business intelligence process anyway.

Data you can access is more problematic. This includes public and government data, purchased data, and research data. Most of this data comes attached with licence restrictions and these must be considered carefully before inserting them into your business intelligence system. However some of the public and government data can be the most useful, as you can start to compare external factors (such as weather, geo-demographic, crime, interest rates, etc) against your internal data sources. This in turn can lead to some startling discoveries.

Data you can infer is the least accurate data, but often very useful. This includes information such as competitor prices, competitor income and expense, customer profiles, future trends, geo-demographic breakdown of your target audience, etc. By inferring this data you can act on the information and compare the inferred outcome against real, known data and over time start to gauge its accuracy. This is really good for identifying trends, advertising campaigns, and customer purchasing decisions.

### Reverse engineering

Reverse engineering is, in the context of data warehousing, the process of discovering the design principles and structure of a source system through analysis of its structure, function and operation. During the analysis of the potential data sources, third party applications may need to be reverse engineered in order to understand the structure and availability of data within the application. This can be the only way to access your data from some vendors.

The process involves extracting the metadata on all the available relations and if possible the relationships from the data source, and building a data model and data dictionary from that. Most E.R. modellers provide some reverse engineering tools.

This can take significant effort which, when you are attempting to release incrementally, may be very time consuming. There are three things you must remember:

1. This is the cost of building a business intelligence system and not everything is going to be easy. Agile approaches won’t make this much faster.
2. Make sure the right people are involved in the reverse engineering. This is a perfect opportunity for pair programming techniques (high risk / high complexity).
3. Stop when you’ve found the data you need. This is not a “get everything” situation. You can always come back later.

It should be noted that, in many countries, even if an artefact or process is protected by trade secrets, reverse-engineering the artefact or process is lawful as long as it is obtained legitimately.

### Identifying potential data sources

Before a business intelligence project can be started, potential data sources need to be identified. A data source is any object which contains business information.

Initially the data sources will be the main transactional and communication systems used within the organisation. Consider the following as potential sources;

* Human Resource Databases
* Client/Customer Relationship Management (CRM) Systems
* Content Management Systems (CMS)
* Bespoke Application Databases
* Corporate Emails
* Reference Websites
* Wikis and Forums
* Documents and Spreadsheets
* Source Code Repositories
* Log Files

When identifying data sources, it is important to understand the types of data sources that are available. Each type has a unique method of extraction and translation. Databases are the most common source type, and unless otherwise mentioned are the default source type discussed during this course.

|  |  |  |
| --- | --- | --- |
| **Source** | **Analysis** | **Connection** |
| **Databases** | The designer needs to be able to analyse the database schema and content. To analyse a DBMS system access to the database schema is needed. In preference access should be via the schema diagram such as an E.R. Diagram. If this is unavailable, schema information can be extracted from a database connection such as ODBC/JDBC.  It may be impossible to extract the schema if there is only restricted access to the database. Delimited text extracts of the database should be made available that can be analysed. The analysis, at this point, is a highly manual process and will involve some guesswork.  If there is no success with any of these, then, as a final resort, the vendor should be able to provide the extraction queries. | SQL via ODBC, JDBC, or Remote TCP/IP |
| **Spreadsheets** | A good business intelligence system needs to be able to extract information from any available document. A lot of financial information, and other business specific information, is stored within spreadsheets.  In general information from spreadsheets can be extracted as if it were a single database table. However, be aware that there may be referential integrity issues to address. | URL via HTTP(S) or Local path |
| **Documents** | Like spreadsheets, a lot of domain specific knowledge is stored within standard office documents, and can be very powerful to an organisation if collected.  Word documents store structured information in the file header, including author and modification dates. Depending on the format, content within these files can be separated by header for storage. | URL via HTTP(S) or Local path |
| **Websites** | A lot of useful material is stored in websites, including research and reference material.  Internal websites are usually database-driven, and a business intelligence system would normally access the database directly. External websites can be very useful to extract data from, and can be used to improve the scope of available data for business intelligence and reporting.  Extracting from websites can be difficult as a website will normally contain a large quantity of extraneous information, such as advertising and navigation elements. The extraction tool should be smart enough to follow links to extract the entire site if required. | URL via HTTP(S) |

|  |  |  |
| --- | --- | --- |
| **Log Files** | Corporate servers produce a lot of statistical data. Statistical information is usually stored in simple log files which can be processed and integrated into the business intelligence system.  This can be highly useful to correlate staff and clients with website and server usage. | URL via HTTP(S) or Local path |
| **Email** | A highly under-utilised aspect of data warehousing is the ability to extract from and report on corporate emails. There can be privacy issues with extracting from personal email automatically, but addresses such as sales@company.com, info@company.com and contact@company.com can contain a lot of very good information regarding who is emailing you, how often they email and what they are asking about.  A lot of information can be extracted from the email headers including who sent the email, where it came from, when, how it was routed to your servers and even information about their mail client.  Processing the body of the email can be difficult, with freeform text processing still in its infancy. However some information can be mined from it. It is important to make sure you strip all HTML and images from the email for maximum efficiency. Although, as with the attachments, can be inserted into the business intelligence application separately. | POP or IMAP(S) |

### Creating business cases for data sources

Because analysis and integration of foreign data sources into a business intelligence system is a time consuming and complicated process, it is recommended that you write a short business case for each data source. This can help prioritise data source analysis and extraction, as well as identify problem areas for each source. An additional advantage is that it can help identify redundant systems and data.

This is written once, the first time a data source is touched, and referred to for each subsequent iteration or requirement that uses that data source.

I ask the following 10 questions:

1. What quantity of data is being dealt with?
2. How frequently does this data change?
3. When does this data change?
4. How much of this data is duplicated elsewhere?
5. How much of this data is obsolete or irrelevant?
6. How is this data used?
7. What reports currently use this data, and are they satisfactory?
8. What access is available to the data, and how can we extract it?
9. Is external or third party permission required before extracting this data, and how much will that cost?
10. Does the organisation have the skills in-house to perform the analysis and extraction or will we have to out-source?

### Incrementally building a data dictionary

A data dictionary is a set of metadata that contains definitions and representations of data elements. In the context of business intelligence, multiple data dictionaries are needed, one for each of the data sources, data marts and the consolidation database.

Agile teams should include updating the data dictionary as part of their definition of done. Additionally, if a team is using a Kanban board, this should be a named state (or defined as part of documentation). It is important to understand who is using the data dictionary – is it purely technical or will end users access it – and write it accordingly.

A good data dictionary should include both semantic and representational definitions for data elements. The semantic components focus on creating precise meaning of data elements. Representation definitions include how data elements are stored in a computer structure such as an integer, string or date format. For a data warehouse a data dictionary should also contain a definition of how historical data is stored, and extracted from the data sources. This is critically important to understand, when developing a good business intelligence system.

## Incremental design and enterprise architecture

### Feature driven development (FDD) for business intelligence

FDD defines five basic activities that break down complex systems and requirements into staged components suitable for incremental development. These activities are:

1. **Develop Overall Model:** Define the context, and create a high-level scoping document. This occurs at the beginning of the project and is regularly updated.
   1. High level scoping of the business intelligence requirements
   2. Create domain walkthroughs for each modelling area (e.g. the new data marts)
   3. Peer review of each model
   4. Merge into complete system model
2. **Build Feature List:** Decompose the model into subject areas, and within each subject area, a set of features to be delivered. This occurs at the beginning of the project and is regularly updated.
   1. Split domain into subject areas (e.g. Finance)
   2. Separate subject areas into business activities (e.g. Annual Report)
   3. Separate business activities into individual features (e.g. an attribute or report)
3. **Plan by Feature**: Prioritise, and assign, each feature to a development team. This occurs throughout the project at an iteration level.
   1. Assign features to data models
   2. Assign data models to developers (or development teams)
4. **Design by Feature**: Select, and design, the set of features to deliver in the current iteration. This occurs throughout the project at an iteration level.
   1. Create a set of features to be developed within the iteration
   2. Build sequence diagrams for each feature (or use the Kanban flow)
   3. Refine model for each feature
   4. Inspect and review the design
5. **Build by Feature**: Build, test and review each feature. This occurs within each iteration.
   1. Develop the code for each feature and class
   2. Unit test
   3. Promote to build

Each activity is comprehensively discussed and peer-reviewed to ensure clarity and agreement between all stakeholders.

### Agile estimation

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 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.

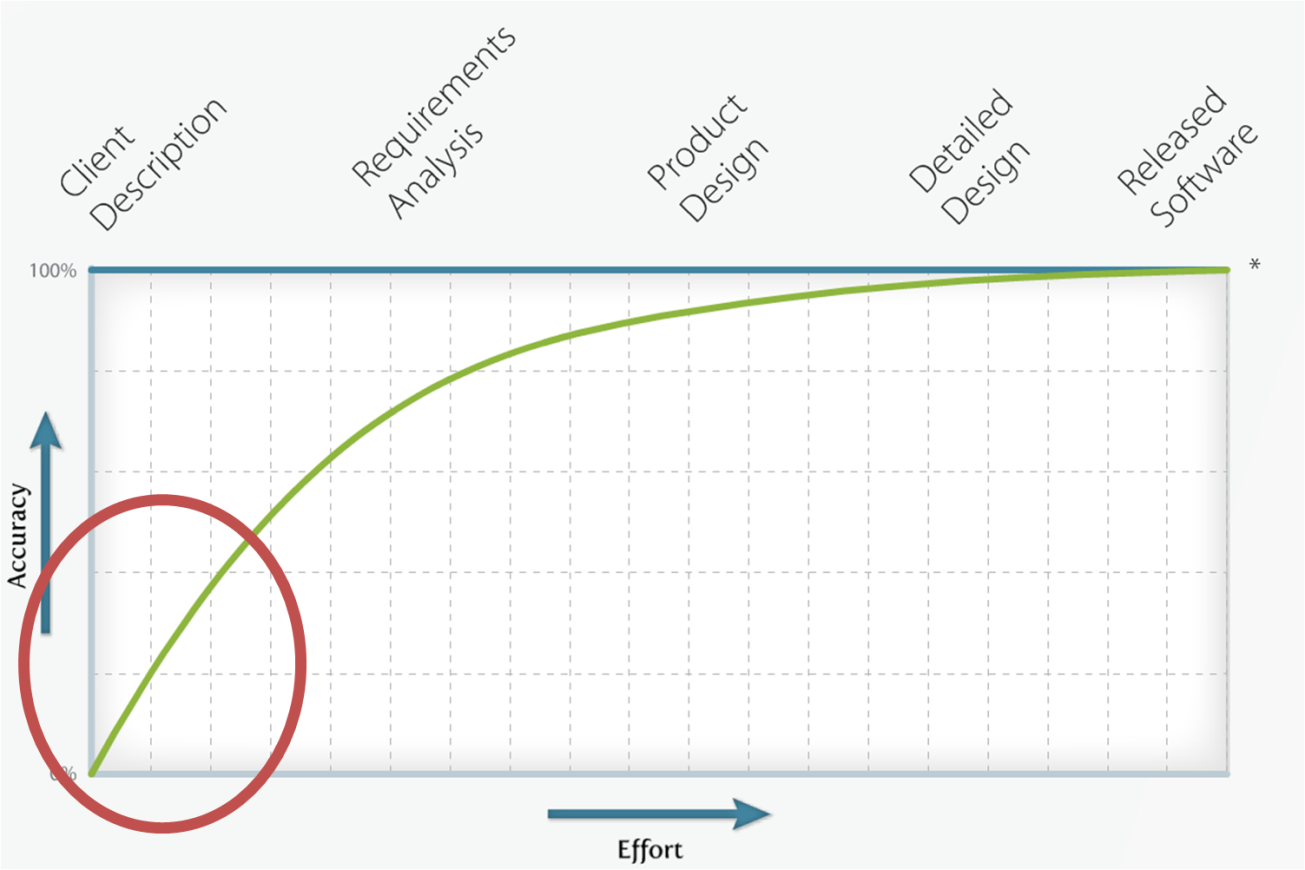


Figure : estimate Accuracy over Time

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.

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:** An estimation game, using a special set of cards (based on the modified Fibonacci series above), that promotes collaboration and consensus building during the estimation process. The rules are simple.
   1. 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.
   2. Each person lays a card face down representing their estimate of the task, and then simultaneously turns their cards over.
   3. People with high estimates and low estimates are given an opportunity to discuss their estimate before anyone else can speak.
   4. Repeat the estimation process until a consensus is reached.



Figure : PLanning Poker Cards

Converting an estimated cost into estimated time is very simple. Apply 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*

e.g.

*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*

### Planning process

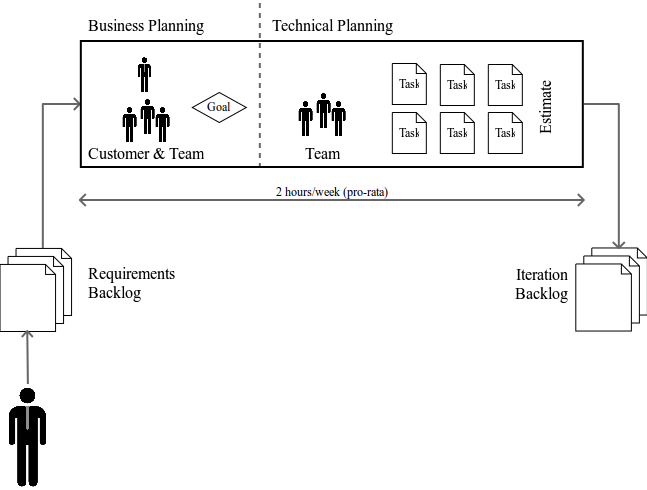


Figure : Iteration Planning

Whether using an iterative process, like Scrum, or continuous process, like Kanban, features and requirements need to be assessed, designed and planned. The iteration planning meeting is run before the start of each Iteration, and allows the customer and developers to discuss the requirements and work required for the next release. This step in the agile process focuses on determining the target scope of delivery for an iteration, and defines the iteration backlog. The iteration planning meeting should be no more than 8 hours long (4 weeks pro rata).

The product owner/customer representative and team facilitator are responsible for updating the product backlog in preparation for this meeting. 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 Iterations.

#### Part 1 – business specifications

The first part of the iteration planning meeting aims to convert features from the backlog into a realistic goal for this iteration. The customer representative is a part of this process and sets the priority of the tasks. 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. This part of the meeting should take no more than ¼ of the time.

Ensure that a copy of the product backlog has been distributed prior to the meeting to provide the project team with time to consider solution options for discussion during the workshop, and prepare clarification questions for the customer representative.

#### Part 2 – technical specifications

The second part of the iteration planning meeting is technical, and usually without the customer representative. This is the solution decomposition and estimating step in the planning process and aims to estimate the effort for all features in the release and (optionally) write the test cases.

The general flow of activity in this step is described in the following diagram:

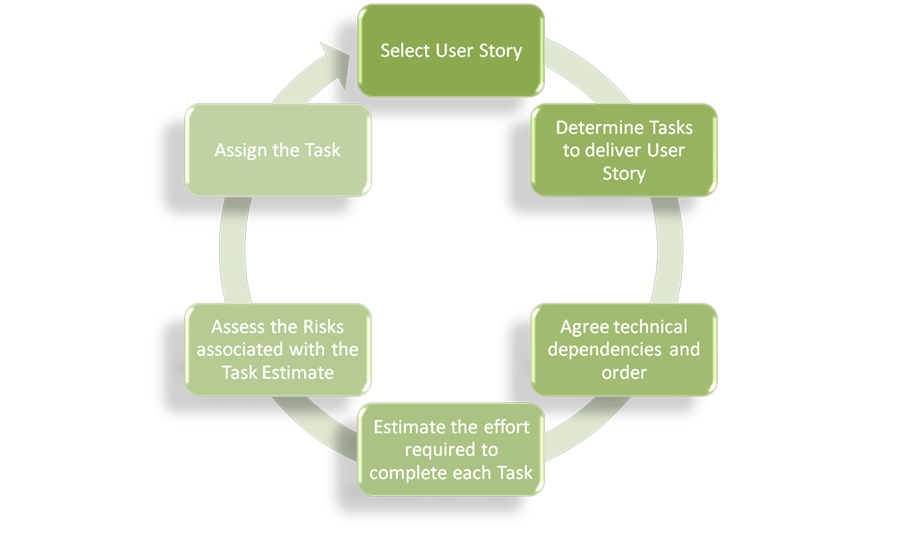


Figure : Design and Planning Workflow

As a guideline, large tasks should be broken into tasks preferably no longer than 1 day and tasks that involve waiting should be split into separate tasks. Research tasks should have a high estimate risk. This is done to enable accurate tracking and the calculation of velocity.

For complex user stories or those with a high number of interdependencies it may be necessary to split the task decomposition and estimating activities across 2 days and allow the team members an opportunity to consult with external parties on feasibility and obtain input into the estimating process.

Always remember that tasks can be created, but features can't.

## Constraining principles

To be effective in the long term, all incremental business intelligence design activities and processes need to be constrained. We can’t let BI teams do anything they want unchecked, while at the same time we can’t restrict their ability to deliver. To that end we create a set of ranked principles - rules that apply to all activities regardless of business requirements.

Your principles will differ depending on what is important to your organisation.

In general, principles will constrain a team in areas of quality, communication, staff engagement, security or branding. Be aware that each additional principle brings with it an increase in cost for every activity. For this reason, principles are given a ranking. This can use any prioritisation model, but I generally recommend MoSCoW.

* **M - Must Have**: non-negotiable principles that every activity must comply to me in order to be considered done.
* **S - Should Have**: teams need to comply with these principles or justify their non-compliance.
* **C - Could Have**: optional principles that act as guidelines at the discretion of the team.
* **W - Won’t Have**: negative principles that teams should avoid.

Beyond the agile values and principles there are common agile principles that are recommended. These are drawn from various agile frameworks including Extreme Programming (XP) and FDD.

### Technical principles

* **Develop by feature:** Functions are broken into the smallest possible part and developed in the order of priority.
* **Regular inspections**: Carried out to ensure good quality design and code.
* **Configuration management**: A log of all features and the source code that have been completed to date.
* **Regular builds**: Everyone must commit their changes (e.g. ETL code) every day. This ensures there is always a fully functional system.
* **Test the system**: Regularly and automatically.
* **Regular refactoring**: To reduce unnecessary dependencies between features.
* **Common code standards:** Across all aspects of the BI system. (e.g. documentation, naming conventions, whitespace)
* **An understandable system metaphor**: All technical components should be clearly named and those names should be understandable.
* **Work in priority order**: All developers should select the highest priority feature from the backlog when they finish any feature.

### Organisational principles

* **Pair Programming:** Where appropriate developers should pair on complex tasks; 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
* **Feature teams**: A small, dynamically formed team that designs and develops each feature.
* **Technical ownership**: Each technical component is assigned to a single owner who is responsible for the consistency, and performance of that function.
* **Listen to and understand the customer:** Key to agile is understanding between the team and the customers.
* **Transparency with the customer:** Customers can attend daily stand-ups, see the backlog in its current state, see the state of each task via a Kanban board or integrated dashboard and access a test version of the BI system

## Continuous integration, automated unit testing and continuous deployment

### Test driven development (TDD)

Test-driven development is a development methodology that requires ETL and report developers to create automated tests before writing the ETL code or function itself. If the tests pass, then the code is displaying correct behaviour. Where possible, these tests should be run automatically. Depending on the ETL tool, this may need to be automated in an external tool (validating the load) such as DBFit. Also where possible, these tests should be kept in a version control repository along with the ETL code.

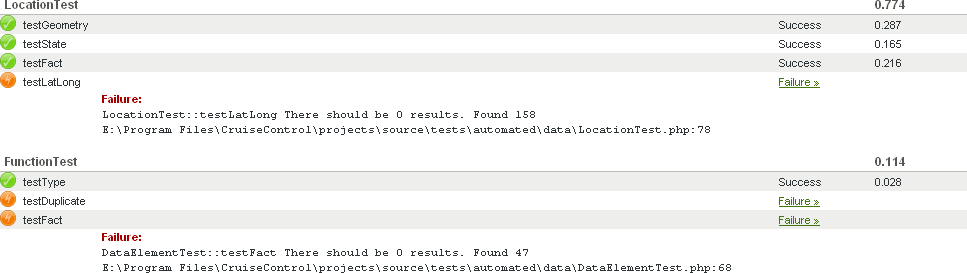


Figure : UNit Test Screenshot

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 representatives to make informed decisions about the project.

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) in a clone of the production environment
5. Clean up the code as required. (Refactor)

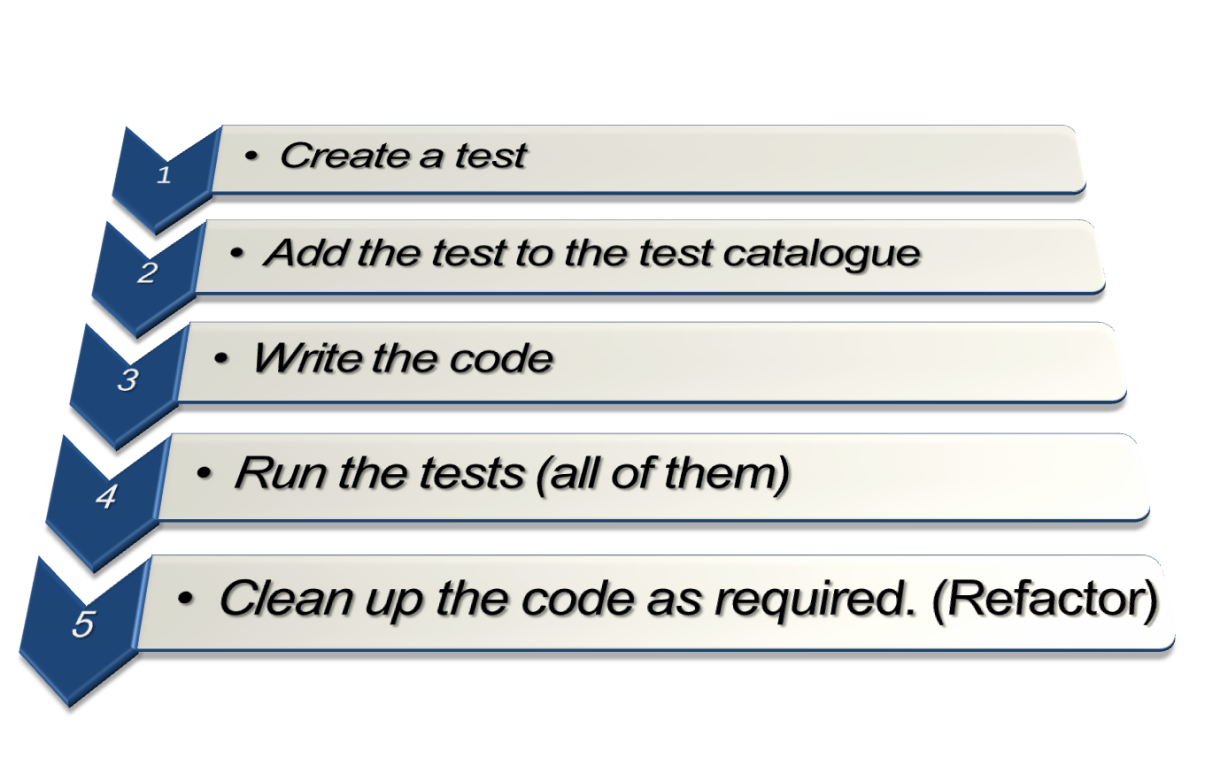


Figure : TDD Workflow

The tests should cover;

* Create tests for all ETL functions,
* Create tests for all parameter combinations and boundary cases,
* Create tests to examine the database for logical errors
* Report interface,
* Non-functional components,
* Performance

### Continuous integration

Continuous integration (CI) brings together the developed, yet distributed, functions and capabilities on a regular basis (usually several times per day). In addition, through the use of a central CI build server, business intelligence teams can automate significant portions of the quality control cycle. These include;

* **Automated unit testing**: Refer to TDD above.
* **Automated integration testings**: Refer to TDD above.
* **Automated regression testing**: Refer to TDD above.
* **Code coverage**: Depending on the ETL and associated unit testing tool, it is possible to calculate how much of the system is covered by unit tests.

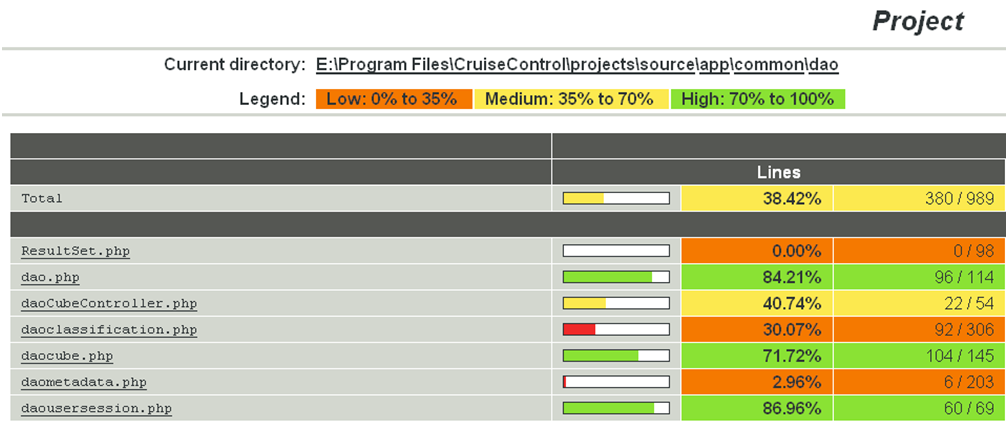


Figure : Code Coverage Screenshot

* **Performance profiling**: Automatically test the performance of the submitted change
* **Code standardisation**: Inspect the developed code for deviations from the internal code standard. (e.g. correct inline documentation (docblock), correct variable naming conventions, correct whitespacing conventions, complex code that may require refactoring and incomplete or unused functions)

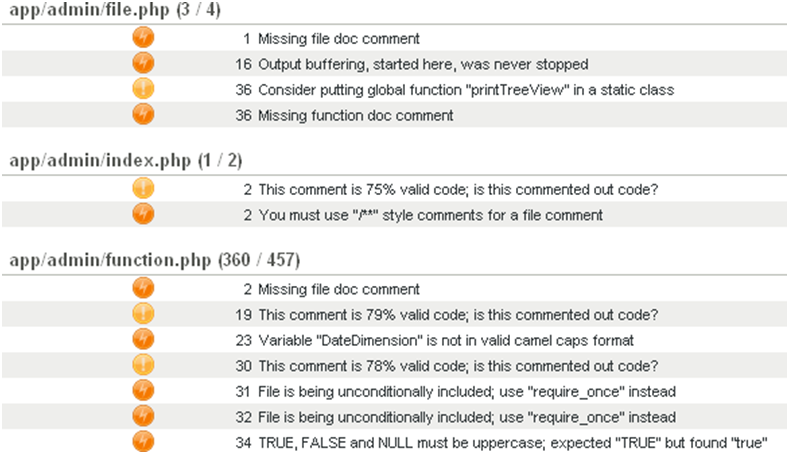


Figure : Code Standard Screenshot

* **Documentation generation**: The automation of system and user documentation from appropriate metadata from business intelligence systems.

### Continuous delivery

An extension to continuous integration is continuous delivery. This practice automates the promotion of passing ETL code and reports to the production environment (with automated rollback if required).

# Agile Project Planning

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

Winston Churchill, ~1960

Everything that has been covered in this course so far is focused in the technical delivery of a business intelligence system. Using Kanban, Value Stream Mapping and the related continuous development approaches is, in itself, a comprehensive delivery model. However, many organisations fund and resource business intelligence systems via formal projects.

In this context we can wrap all the technical practices and principles from the previous sections within an agile (in this instance Scrum) product or project framework.

## Beginning the process

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 and improve overall decision making and financial planning.

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

The outcomes from Project Initiation are:

* The team should be identified and brought together
* If not part of the team, identify and train the product owner / 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.

## Fixed time, cost and 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 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

### 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)

## Scrum as a business intelligence product and project framework

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. At a high-level, the Scrum framework is primarily team based, and defines associated roles, events, artefacts and rules. The three primary roles within Scrum are:

1. The product owner who represents the stakeholders – also known as the customer representative,
2. The scrum master or team facilitator who manages the team and the Scrum process
3. The team, about 7 people, who develop the system.

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.

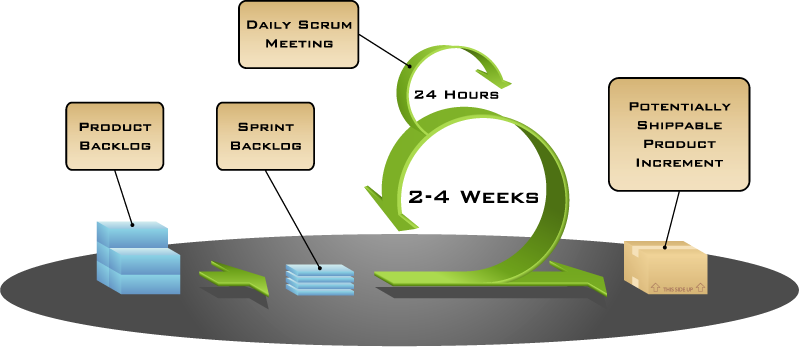


Figure : Scrum Framework

The requirements that form the basis of the project are collated into what is called a product 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 : 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 product 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/team facilitator and (ideally) the product owner/customer representative. At the 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 system, 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 product backlog and the process begins again. Burn rate is monitored to determine when funding will be exhausted.

Table : 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 product backlog. |
| Product backlog | The product 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 product 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. |

## Product backlog

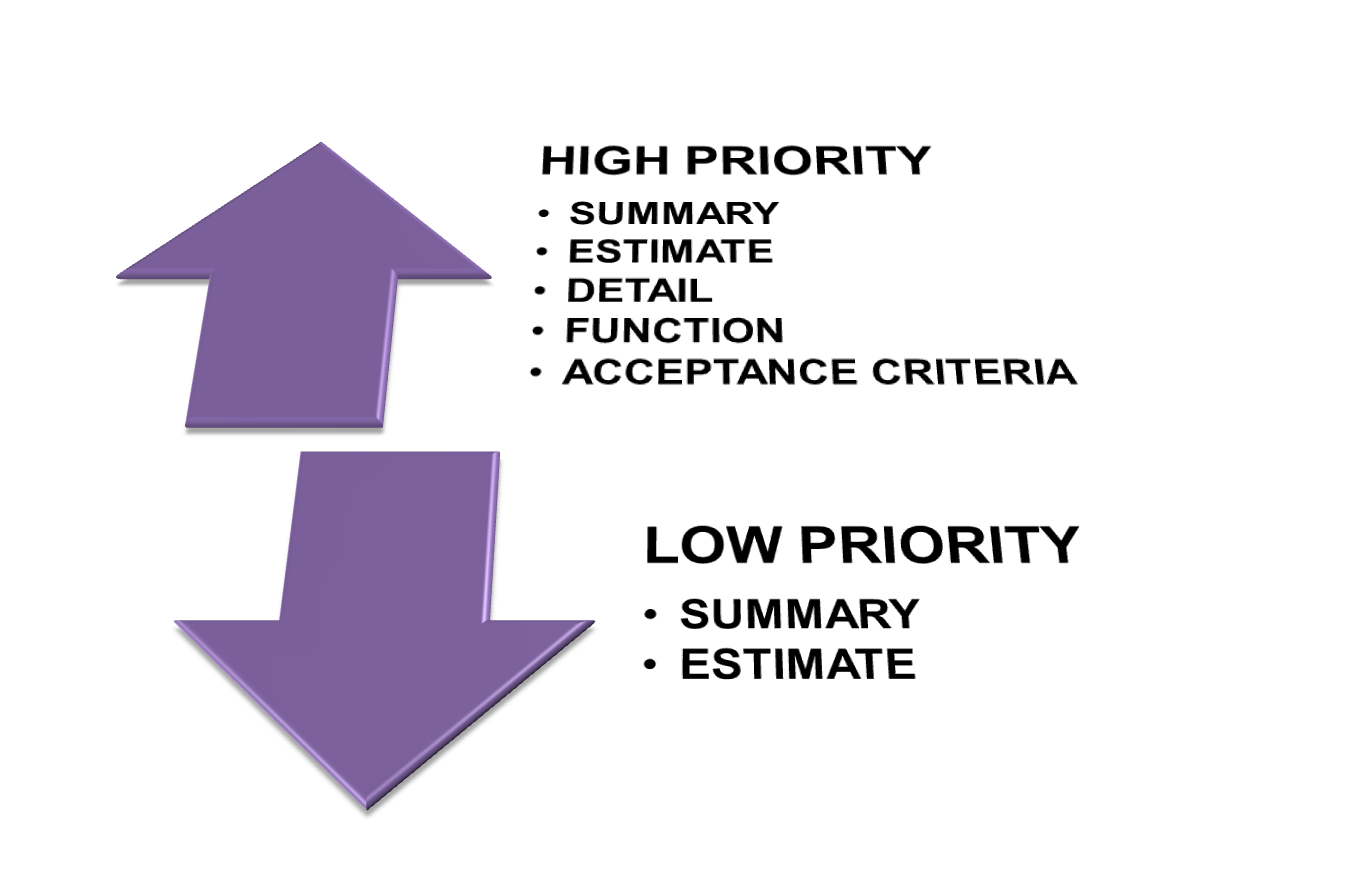


Figure : Product Backlog

The backlog contains all the user stories (features) for the product. The product owner or 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.

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.

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 : Key Story Elements

|  |  |
| --- | --- |
| Planning Element | Description |
| User story | The user story is a description of the business need, usually expressed as a feature. |
| 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 |
| 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 * 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. |

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.

## Kaizen (改善)

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. **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.

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 1-3 hours, 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.

## Product review

At regular intervals, the scrum master or 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.

## 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 scrum master or 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 scrum masters or team facilitators from multiple teams to answer the same 3 questions as before, but relating to their teams.

# References

## Books and 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
* Data Warehouse.com: http://www.datawarehouse.com/
* DM Review: http://www.dmreview.com/index.cfm

## Tools

* Trac: <http://trac.edgewall.org/>
* Watir: <http://watir.com>
* PostgreSQL: <http://www.postgresql.org>
* Pentaho: <http://www.pentaho.com/>
* Jasper Reports: <http://jasperforge.org>
* SAP BI Warehouse: <http://www.sap.com>
* Teradata: <http://www.teradata.com/>

1. Images with thanks from Jeff Patton: <http://www.agileproductdesign.com/> [↑](#footnote-ref-1)
2. Toyota Production System: Beyond Large-Scale Production, Ohno (1988). [↑](#footnote-ref-2)
3. Kanban: Successful Evolutionary Change for Your Technology Business, Anderson (2010). [↑](#footnote-ref-3)
4. Image thanks to: http://www.nexusis.com/solutions/cloud/cloud-consulting/ [↑](#footnote-ref-4)
5. Images thanks to: http://leadinganswers.typepad.com/leading\_answers/2011/09/pmi-acp-value-stream-mapping.html [↑](#footnote-ref-5)
6. Image thanks to: New Zealand Postal Group via Directing the Agile Organisation [↑](#footnote-ref-6)