HITACHI
Inspire the Next

Introduction to

VANTAGE

A Delivery Methodology for Pentaho Based Analytics Solutions

Pentaho Consulting Services

February 2018

1 OVERVIEW 3

2 INTRODUCING THE VANTAGE METHODOLOGY 4

3 CLIENT ENGAGEMENT MODEL 5

4 THE 5 KEY STAGES OF VANTAGE 8

4.1 PLAN 10
4.2 DISCOVERY 14
4.3 DESIGN 20
4.4 EXECUTE 24
4.5 ENABLE 29

5 TOOLS FOR SUCCESS 32

5.1 AGILE PROJECT DELIVERY 32
5.2 SCRUM 37
5.3 DEVOPS 44
5.4 PROJECT ENVIRONMENTS 47
5.5 VERSION CONTROL 49
5.6 TESTING STRATEGY 50
5.7 EVOLUTION OF DB-SCHEMAS 53
5.8 SOLUTION DOCUMENTATION 54
5.9 PENTAHO BEST PRACTICES 54

6 SUPPORT DOCUMENTATION 58

6.1 GENERAL DOCUMENTATION 58
6.2 PENTAHO SERVICES 59

1 Overview

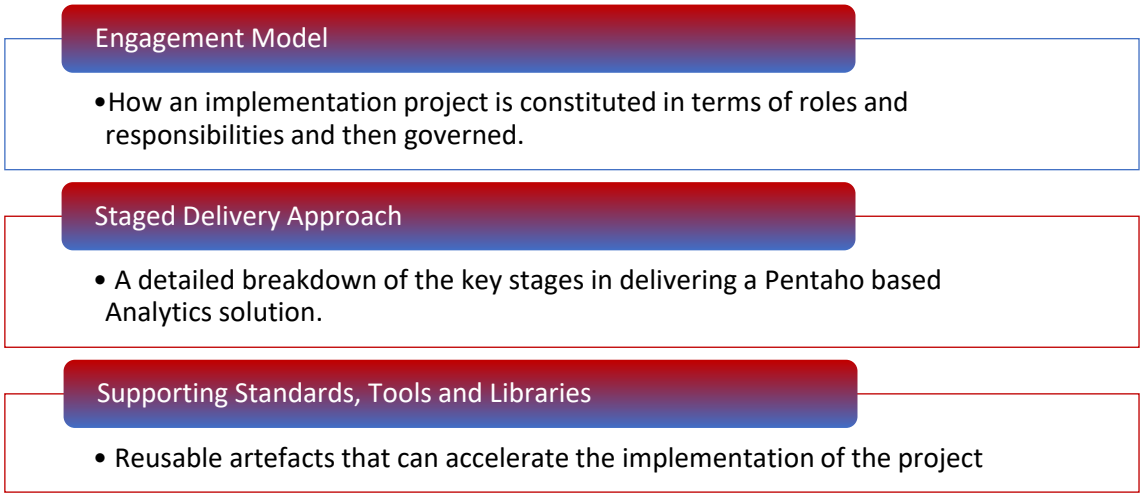
This guide describes VANTAGE, a project implementation methodology that has been developed by Hitachi Vantara Consulting Services. It is based on the team's extensive experience of successfully delivering Big Data and IoT solutions using the Pentaho Enterprise product at a wide variety of commercial and Governmental organisations.

VANTAGE encapsulates this extensive experience and best practice know-how into a single proven approach for delivering these types of projects in a consistent and repeatable manner. It is designed for use by Hitachi's own staff and those of authorized Partner organizations.

This document provides a single authoritative reference guide to VANTAGE with the objective of providing the reader with the following:

1. A proven approach to de-risking the implementation of a Pentaho based solution and driving towards a successful project outcome
2. Repeatable processes and best practices for accelerating the implementation and time to value
3. Improved quality through more consistent implementation delivery across multiple teams and team members
4. As an introduction to the tools and techniques used in our services engagements that help us manage and optimise the project delivery process

This guide comprises of three key themes:

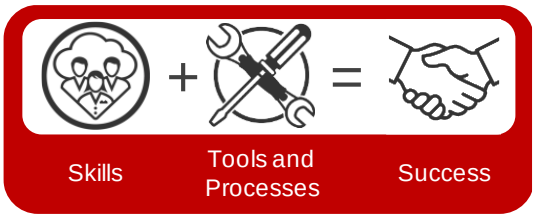


2 Introducing the VANTAGE Methodology

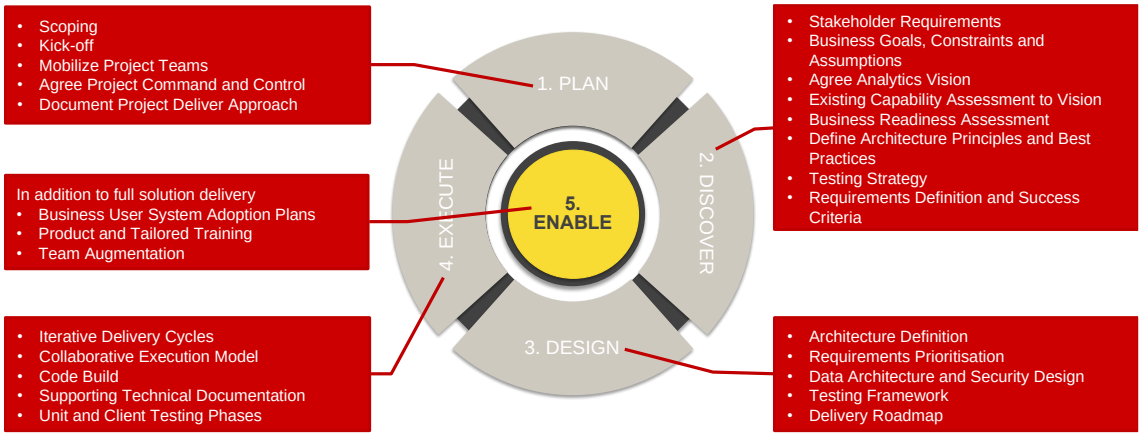
The Hitachi Vantara Analytics Consulting Services practice has successfully delivered Pentaho based solutions across a wide range of organizations and industries, from Financial Services, through to Telco, Retail, Manufacturing and others.

During the delivery of these projects, it became clear that there was a need to capture the best practice and know-how gained to enable all Pentaho practitioners to benefit from this previous experience; enabling them to deliver new projects more rapidly and with consistently higher levels of quality.

VANTAGE captures this prior extensive project experience and distils it into a single best practice approach to project delivery. Having a consistent proven approach to bringing together the right skills, tools and best practice reusable processes accelerates the time to taken to realize the business benefits of the analytics solution.



VANTAGE can be best described as a hybrid methodology. It uses an Agile-based approach to Application Development fused within a sequential five key stage approach to the overall Project Delivery.



Using an Agile for Application Development has many benefits, key amongst these are improved customer satisfaction with the delivered outcome. Agile encourages greater stakeholder engagement within a controlled process that ensures greater alignment between expectations and the actual outcome throughout the delivery process.

The remainder of this guide provides a detailed description of the five stages and introduces the reader to the best practices and tools we have found to be useful in accelerating the delivery of a high-quality analytics solution.

3 Client Engagement Model

As already discussed above, a successful approach to project delivery requires the ability to bring together tools, processes and implementation skills in terms of project staffing.

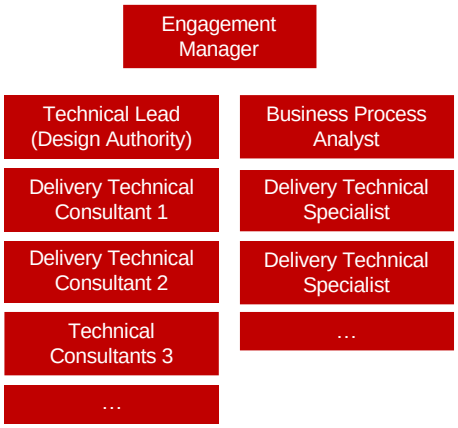
VANTAGE recommends a best practice approach to constituting a project team in terms of structure, roles and responsibilities. It assumes that a project team will have a hybrid structure comprised of staff from Hitachi Vantara's own consulting team, from its authorised partners and from a client's own organisation. This latter group may in turn be comprised of a blend of staff working onshore and offshore that are part of a subcontract and / or outsourced arrangement. It also assumes that a client may wish to have their own organisation enabled to deliver future analytics projects, subject to understanding the trade-offs between speed of delivery and the need to enable newly trained staff.

Based on these core assumptions, VANTAGE has the power to accommodate the majority of client deployment scenarios; and makes it even more imperative to have one common understanding and approach to the way a project is delivered.

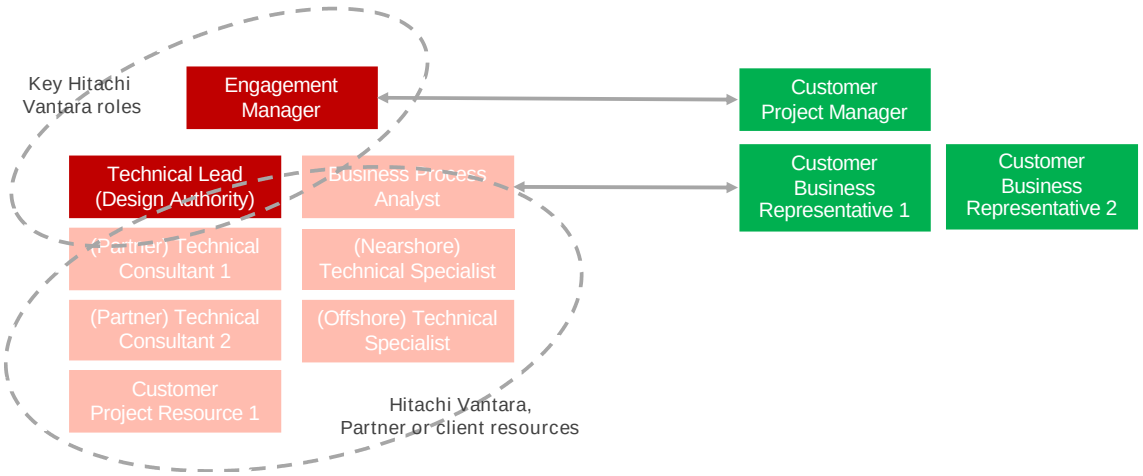
It is however worth noting that VANTAGE is not designed to support the delivery of monolithic managed services (outsourcing) contracts.

Project Team Structure

VANTAGE assumes the following generic team structure for delivering projects.



As discussed above, team roles may be staffed from a variety of organisations, but we expect that the organisation responsible for the successful outcome and completion of the project will take the key project leadership roles of Engagement Manager and Technical Lead (Design Authority).



The following section details the specific roles and responsibilities of the project team.

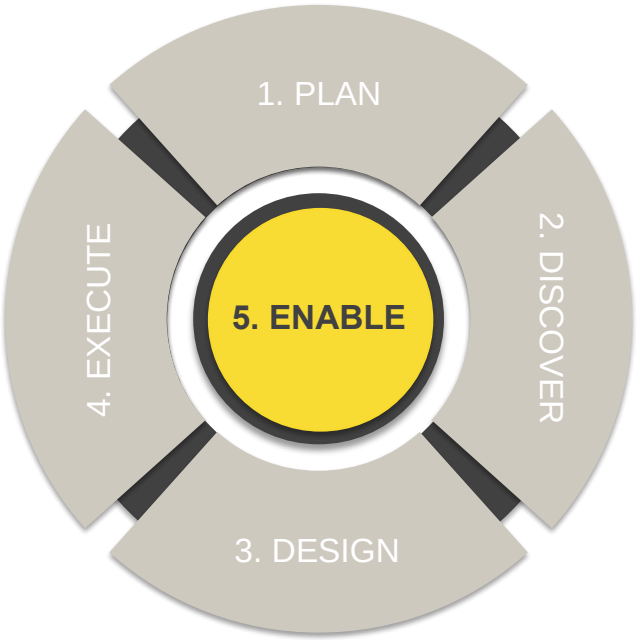
Roles and Responsibilities

Role	Activities
Engagement Manager	• Owns the entire services engagement lifecycle; from assignment through to successful delivery, sign off and billing with no customer disputes • Leads one or more delivery teams concurrently, depending on seniority and project size • Manages all internal and external stakeholders to ensure confidence in the service delivery process and is proactive resolving any issues • Produces and maintains all relevant project documentation: project plans, budgets, risk registers, change orders and status reports • Is accountable for accurate project revenue forecasting and margin (where possible) • Relentless focus on customer value and driving benefit realization • In new territories, leads development of the services business by working with regional sales leadership, and growing local delivery capability of regional services team and services partner eco-system • In established territories or projects, assists the services sales and contracting process and identifies incremental revenue opportunities, working with commercial teams to progress
Technical Lead	• Own the technical engagement lifecycle; from initial assignment through to successful delivery • Responsible for end-to-end technical design of solution • Understand, produce and maintain all relevant project design documentation • Must be able to establish credible advisor relationships with customers and colleagues alike on all technical matters relating to the company's products
Business Process Analyst	• Responsible for working with customers in a consultative environment to understand and capture overall business aspiration and specific requirements • Will support customer on Business Value analysis for project evaluation. • Guides customer towards best practice based on blueprints to help manage expectations and maximize chances of full benefit realization • Helps close any gaps in understanding between the business needs and technical capabilities • Is responsible to run workshops, create presentations and process documents to achieve objectives • Works with business representatives to capture detailed requirements as Agile stories with sufficient clarity to minimize rework • Can also work as SCRUM master to augment Engagement Management capacity

Role	Activities
Technical Consultants	• Knowledgeable in Pentaho products and responsible for technical implementation of all aspects of the solution to an expected level of quality. • Able to raise any escalation in case of any deviation or blocking in the project delivery
Technical Specialist	• As per Technical Consultant but with deep expertise in a specific aspect of product implementation.
Customer Project Manager	• Responsible for all client activities and ensuring appropriate client resources are made available • Internal management project status reporting • Tracks budget spend and key point for approval of paperwork
Customer Lead Architect	• Provides input to the project team on the current architecture and any future architecture states the team needs to be made aware of • Responsible for approving the technical solution
Customer Business Representative	• Key stakeholders and business users • Product Owner • Ensure client's requirements are correctly captured, prioritized and in line with the business case • Help provide user testing resources to validate the product • Provide sign off of the solution delivered

4 The 5 Key Stages of VANTAGE

VANTAGE comprises of five key stages of delivery. Each stage builds upon the other to provide a holistic approach to solution delivery.



Project foundations are established in the PLAN stage. A project kick-off with Engagement Manager, Technical Lead, Client PM and Lead Architect will ensure that all parties are aligned on the scope and that resource expectations can be met. Project transparency is a must for all participants, and the project logistics will aid this. A governance and support framework will be established, including communication plans, artefact production, status reporting and should they be needed, escalation procedures.

Once the key parameters of the project have been determined, project DISCOVERY takes place. The size and scale of the programme or project will dictate how long this stage takes. We will validate the goals and objectives of the project to ensure they are aligned to outcomes at a detailed level, further agreeing any major constraints or assumptions. Prototyping activities will be undertaken, and first iteration Product backlog generated.

The DESIGN phase brings together the goals, objectives, product backlog and technical landscape, to map out the data, systems and application architectures needed to deliver a solution. For complex engagements, an architectural gap analysis may be produced to help with prioritizing project deliverables and federating architecture delivery into separate delivery streams. In contrast, for small engagements, the design phase may be incorporated into a sprint 0.

At a point within the design phase, there will be a trigger to EXECUTE and begin implementing the architecture and content. Using our Agile+ approach, sprints will be initiated, leveraging our proven processes and best practices. In line QA processes and automated testing help ensure we deliver right first time and prevent incremental regression issues.

Key to any engagement is to ENABLE the client. This comes in many forms, from training (custom and product bespoke) to enabling a client's own technical resources by augmenting scrum teams. We will help assess the client's capability to take on new technology, highlighting risks and necessary actions.

It is important to realize that since our methodology aims to deliver quick wins, these phases often overlap, enabling us to execute in a flexible and adaptive way.

The following sections provide more detail on each phase.

4.1 Plan

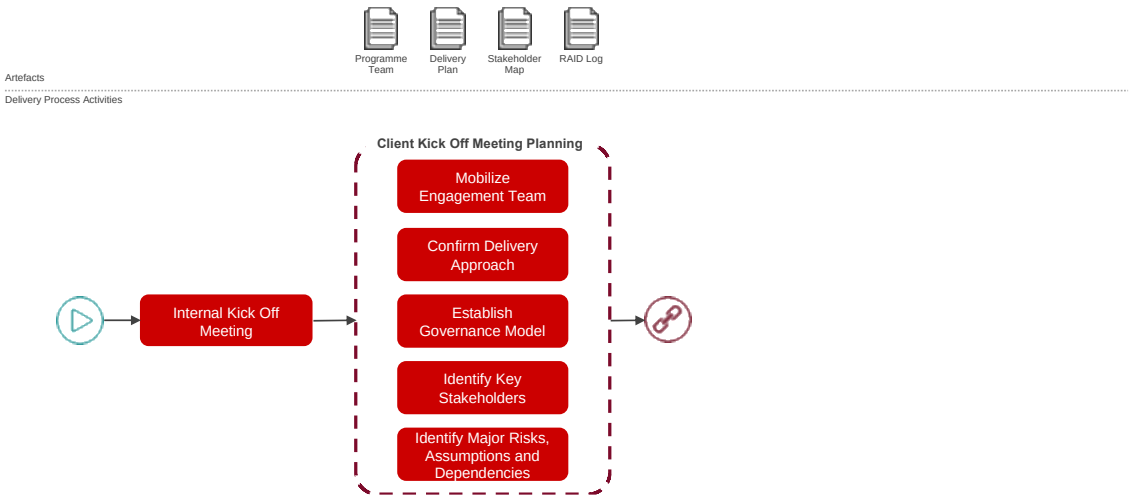
Project foundations are established in the PLAN stage. A project kick-off with Engagement Manager, Client PM and Lead Architect will ensure that all parties are aligned on the scope and that resource expectations can be met. Project transparency is a must for all participants, and the project logistics will aid this. A governance and support framework will be established, including communication plans, artefact production, status reporting and should they be needed, escalation procedures.

Key Activities

- Identify key stakeholders and business representatives
- Review and confirm scope
- Define and mobilize the project engagement team
- Agree the delivery and technical governance model
- Agree on main project cadence (weekly status calls / daily Scrum meetings)
- Define any constraining business or technical dependencies
- Plan, prepare and conduct project kick-off meeting

How is it done?

Below is the typical process flow we take when running the planning activities. Each process is then explained in detail.



Activity	Description
Internal Kick-off Meeting	Here the services team sync with the sales team to provide an accurate handover of the information gathered during the sales cycle. This allows us to ramp up quickly and not have to cover old ground with the client. The sales agents should take the services team through their MEDDICL classifications in order to capture the initial project objectives.
Client Kick-Off Meeting Planning	In order to prepare for the main client kick-off session, a planning meeting will first be held between the client PM and the project team. This meeting will be to ensure that the correct preparation activities are performed to maximize its effectiveness.
Mobilize Engagement Team	The combined client and delivery team. This should clearly set out the roles and respective responsibilities for each of the teams and members. This will help detail who from across the project team should be attending the client kick-off. At this point, the Engagement Manager will begin procedures for onboarding consultants and ensuring appropriate security checks have been completed. In addition, any partner paperwork will also be documented.
Confirm Delivery Approach	Whilst our VANTAGE delivery methodology is intended to be generic across multiple analytics projects, it may be necessary to adapt this to fit a particular client need, for example specific security sign off procedures. Addressing this now sets clear expectations across the extended team.
Establish Governance Model	Imperative to any quick and successful resolution of issues is to agree up front a governance model and escalation paths with terms of reference and team individuals forming it. This needs to cover both project and technical streams of activity. Our engagement model is designed to enable our teams to own the project delivery end to end, during this phase the addition of client specific representatives (such as a technical architect) can also be added.
Identify Stakeholders	Detail out stakeholders that need to be invited to the kick-off session. This should cover various areas of the client organisation, business representatives, delivery, architecture and operations. Often it helps to build a stakeholder map which will detail each stakeholder and the level of engagement they need for the project. Each stakeholder should be approached ahead of the session to prepare a view of their key requirements and priorities for the solution.
Major Risks, Assumptions and Dependencies	It is important to highlight early any actual or potential major impacts to the project to ensure that adequate contingency is put in place. At a minimum, expectations may need to be reset, for more serious items, it may even lead to the team standing down until resolutions are put in place. It is important to think of the future project health at this point and ensure resolution paths are implemented now and not by-passed potentially resulting in a failed project. It is important to properly assess items being recorded, for example, assumptions should be validated and dependencies should have clear lines to dependent actions and owners.

What is produced?

The following outputs are produced during the plan stage.

Artefact	Description
Programme Team	The complete team structure including Pentaho, client and 3rd party Systems Integrators, including appropriate governance models and escalation paths.
Delivery Plan	Documented delivery plan to ensure the approach is agreed and communicated. The governance model should also be documented here.
Stakeholder Map	Documents the stakeholder for the project and classifies their involvement
RAID Log	Living documented for Risks, Actions, Issues and Constraints

Who does what?

Each stage of the delivery methodology has a supporting RACI matrix. For those not familiar with these matrices, they are provided to provide clarity on responsibilities for the team members and provided a baseline for planning activities:

R-Responsible - has ownership of the activity

A-Accountable - makes sure the activity is executed

C-Consulted - provided input into the activity

I-Informed - is made aware of the activity outputs

The below RACI matrix maps activities and outputs to the project team.

Task	Activity or Output	Engagement Manager	Technical Lead	Business Process Analyst	Technical Consultants	Technical Specialist	Client PM	Business Representative	Product Owner
Internal Kick-off Meeting	A	AR	C	-	-	-	-	-	-
Client Kick-Off Meeting Planning	A	AR	C	C	-	-	R	-	-
Mobilize Engagement Team	A	AR	C	I	I	I	R	I	I
Confirm Delivery Approach	A	AR	C	-	-	-	R	-	-

Governance Model	A	A	C	-	-	-	R	-	-
Identify Stakeholders	A	A	C	C	-	-	R	C	C
Major Risks, Assumptions and Dependencies	A	AR	C	C	-	-	R	C	C
Programme Team	O	AR	C	-	-	-	R	-	-
Delivery Plan	O	AR	C	-	-	-	R	-	-
Stakeholder Map	O	AR	C	C	-	-	R	C	C
RAID Log	O	AR	C	C	-	-	R	I	I

When do things happen?

- During first 3-4 weeks after contract signature
- Project plan reviewed and updated
- Milestones / Drop-dead dates updated

4.2 Discovery

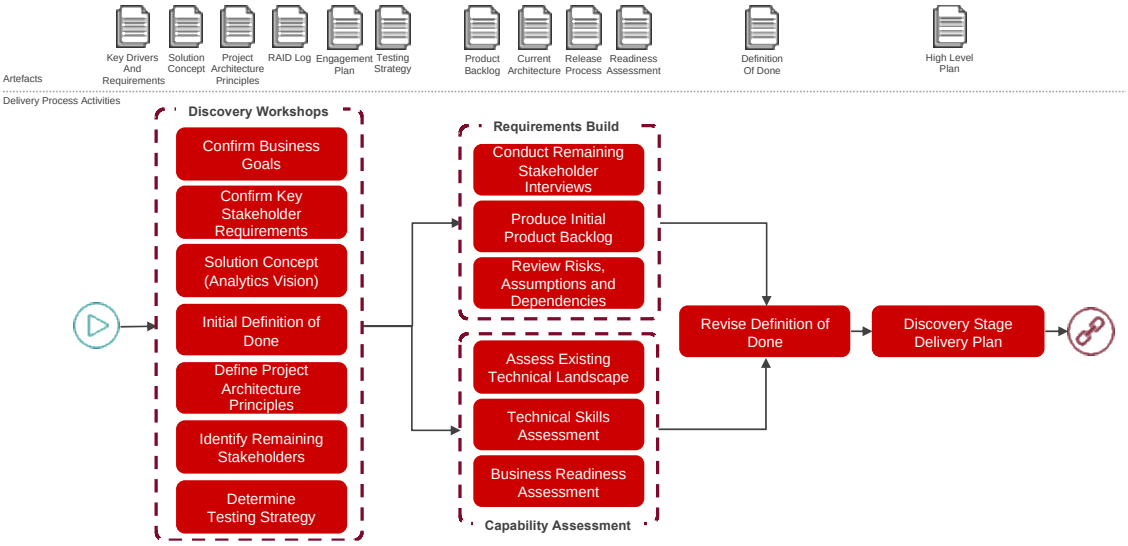
Once the key parameters of the project have been determined, project DISCOVERY takes place. The size and scale of the programme or project will dictate how long this stage takes. We will validate the goals and objectives of the project to ensure that they are aligned to outcomes at a detailed level, further agreeing any major constraints or assumptions. Prototyping activities will be undertaken, and first iteration Product backlog generated.

Key Activities

- Determine the client high level drivers, goals and objectives
- Confirm Key Stakeholder requirements and priorities
- Produce initial Product Backlog from all stakeholder requirements
- Define any constraining business or technical dependencies
- Provide initial solution concept and identify the Definition of Done and acceptance criteria
- Identify Architecture Principles that need to be adhered to
- Assess the organisations current technical and business landscape for the readiness to adapt to the new technology and processes
- Develop next iteration plan

How is it done?

Below is the typical process flow we take when running the planning activities. Each process is then explained in detail.



Activity	Description
Discovery Workshops	Once the planning steps have been executed, the main kick-off meeting can be planned. It is quite possible this meeting will actually be a workshop, with each stakeholder providing their input. The remainder of this section covers the activities to be completed within this phase.
Confirm Business Goals	During the sales process, the sales agent should have a clear view of the objectives of the project according to their MEDDICL assessment, the Engagement Manager should be in possession of these from the internal kick-off meeting. It is possible that the original objectives of the project are no longer in line with the objectives captured during the sales process. This stage provides an opportunity to re-affirm or evolve the original project goals.
Key Stakeholder Requirements	The purpose of the key Stakeholder Requirements is to identify the principle questions, issues, or concerns that must be addressed by the solution. Understanding stakeholders and their requirements allows the project to focus effort in areas that meet the needs of stakeholders. In the presence of the key stakeholders, over-arching priorities can also be agreed, which will help accelerate resolution of requirements escalations and prevent the project from being de-railed by lesser priorities.
Solution Concept	With all the key stakeholders partaking in this process, now is a great time for early validation on what the solution will look like. Once the key requirements have been understood, a quick sketch of the solution building blocks helps reinforce expectations and provide reassurance that the business problems are understood and will be addressed. This will then feed into the architecture design.
Initial Definition of Done	Always know what done looks like. At any stage within the project, the target and what needs to be done to get there should be clear. Again, this will help with correctly guiding effort to fulfil the project needs and avoid unnecessary distractions.

Define Project Architecture Principles	More often than not, an organisation will have their own set of architecture principles and/or design practices that need to be followed. These should be discussed with the client and an agreement reached on which ones the project should incorporate. These principles could affect the overall effort needed for the project, hence it is important to develop an early understanding of expected impacts.
Identifying Remaining Stakeholders	The Key Stakeholders will typically provide the directional information needed to develop a solution concept diagram. These will be higher level and for the expected outputs of the project. As always however, the devil is in the detail. It is important that the project team begins to collate a more detailed product backlog. In order to do this, determine with the key stakeholder and Client PM which other stakeholders should be consulted. This will likely include additional business users, source system owners, IT operations. 3rd parties (Hosting services, outsourced functions) and Security.
Requirements Build	The requirements build is in place to begin filling out the product backlog. During this phase, interviews will take place with all stakeholders to understand the shape and detail behind any requirements that need to be added to the product backlog. Depending on the level of technical transformation that the client is undertaking, these requirements may not be detailed. Since we adopt and agile approach to delivery, this is ok, and often to be expected whilst users become familiar with the data and technology.
Conduct Remaining Stakeholder Interviews	Each stakeholder should be interviewed by a skilled business analyst to extract what each stakeholder is looking for the solution to deliver. Requirements will be added to the backlog and conflicting/contradicting requirements will be highlighted to Key stakeholders and Product Owner for direction.
Produce Initial Project Backlog	The product backlog contains all the requirements raised by stakeholders. It should be built in an easily accessible and audited way. There are many standard tools for this. Section 5.2 provides more details.
Major Risks, Assumptions and Dependencies	It is important to highlight early any actual or potential major impacts to the project to ensure that adequate contingency is put in place. At a minimum, expectations may need to be reset, for more serious items, it may even lead to the team standing down until resolutions are put in place. It is important to think of the future project health at this point and ensure resolution paths are implemented now and not by-passed potentially resulting in a failed project.
Capability Assessment	Knowing the solution vision and constraining architecture principles provides us a needed view of the destination, assessing the existing technical and user capabilities provides us with the origin. This information will be essential in ensuring an efficient architecture roadmap and enablement plan.
Assess Existing Technical Landscape	For projects working alongside/evolving existing solutions, work with stakeholders to develop an existing architecture understanding, including data, applications, infrastructure and security. This information will be used within the design phase to build out the architectture roadmap. For projects that are delivering new capabilities that have no impact on existing systems, this step is not necessary.

	In addition, this stage will also assess the existing technical release plan in order to better understand the processes the project team will need to accommodate any associated impact on project estimation
Technical Skills Assessment	This step will allow us to assess the client's skill set to work with them on developing an enablement model. In this stage we assess the clients technical function, for example Pentaho and big data skills.
Business Readiness Assessment	Similar to the technical skills assessment, here we determine the level of comfort the business community have with the new technologies and processes that will be used. This stage is essential in ensuring the project team can adequately prepare the business for the changes ahead, which in turn will decide the success of the project – limited adoption is a failed project.
Revise Definition of Done	The project team should continuously assess the Definition of Done, so all parties are in agreement and the risk of project conflict is minimized. After the initial backlog, technical and business assessments have been completed represents a milestone to revisit this.
Discovery Stage Delivery Plan	At this point the project team should be able to revisit the existing plan and identify any revisions.

What is produced?

The following outputs are produced during the Discovery stage.

Artefact	Description
Key Drivers and Requirements	This document captures the driving business objectives and goals. This should reflect the original project business case developed by the business lead. The output here can often be translated into Project Epics (in Agile terminology)
Solution Concept	High level view of what the project is trying to achieve with first cut Solutions Concept Diagram
Project Architecture Principles	A list of the constraining architecture principles the client has given the project
RAID Log	Living document for Risks, Actions, Issues and Constraints
Engagement Plan	Plan detailing stakeholders and interview processes. Also covers the communication strategy for project stakeholders. For example, which stakeholder receive weekly or monthly status reports.
Product Backlog	The repository of requirements including clearly marked features for future releases
Current Architecture	Document detailing the current data, application and infrastructure architecture
Readiness Assessment	Assessment of the organisations ability to support the capabilities delivered by the new project
Definition of Done	Agreement on what the project outcomes should be. This document should also contain details of the expected project

	success criteria for the project including deliverables and documentation.
High Level Plan	Milestone plan and key dependencies with t-shirt size estimates to validate the project timelines and resource expectations.

Who does what?

The below RACI matrix maps activities and outputs to the project team.

Task	Activity or Output	Engagement Manager	Technical Lead	Business Process Analyst	Technical Consultants	Technical Specialist	Client PM	Business Representative	Product Owner
Discovery Workshops	A	AR	-	-	-	-	R	-	-
Confirm Business Goals	A	AR	C	C	-	-	R	C	C
Key Stakeholder Requirements	A	A	I	R	-	-	R	C	C
Solution Concept	A	A	R	C	I	I	C	C	C
Initial Definition of Done	A	AR	C	C	I	I	R	C	C
Define Project Architecture Principles	A	A	R	I	I	I	R	I	I
Identifying Remaining Stakeholders	A	A	C	R	-	-	R	C	C
Requirements Build	A	C	A	R	-	-	R	C	C
Conduct Remaining Stakeholder Interviews	A	A	C	R	-	-	R	C	C
Produce Initial Project Backlog	A	A	C	R	I	I	R	C	C
Major Risks, Assumptions and Dependencies	A	AR	C	C	-	-	R	I	I
Capability Assessment	A	AR	C	C	-	-	R	C	C
Assess Existing Technical Landscape	A	A	R	I	-	-	R	C	C
Technical Skills Assessment	A	A	R	I	-	-	R	C	C
Business Readiness Assessment	A	A	I	R	-	-	R	C	C
Revise Definition of Done	A	AR	C	C	I	I	R	C	C
Discovery Stage Delivery Plan	A	AR	C	C	I	I	C	C	C

Key Drivers and Requirements	O	A	C	R	-	-	I	I	I
Solution Concept	O	A	R	C	-	-	I	I	I
Project Architecture Principles	O	A	C	I	-	-	R	I	I
RAID Log	O	AR	C	C	-	-	C	C	C
Engagement Plan	O	AR	C	C	-	-	R	C	C
Product Backlog	O	C	A	R	-	-	C	C	C
Current Architecture	O	A	R	I	-	-	C	I	I
Release Process	O	A	R	I	I	I	R	-	-
Readiness Assessment	O	A	C	R	-	-	C	I	I
Definition of Done	O	AR	C	C	-	-	C	C	C
High Level Plan	O	AR	C	C	-	-	C	I	I

When do things happen?

Following the plan phase. Discovery may take several weeks to be able to schedule all interviews and paint a picture of what the solution needs to achieve.

For larger programmes of work and where a target solution is fairly well understood, it can be beneficial to begin design and development work before the Discovery phase is complete, in order to accelerate some early wins. For example, for a complex on-prem solution, an initial prototype can be quickly spun up on a cloud platform to provide the client with a sense of the solution and aid with evolving the product backlog.

4.3 Design

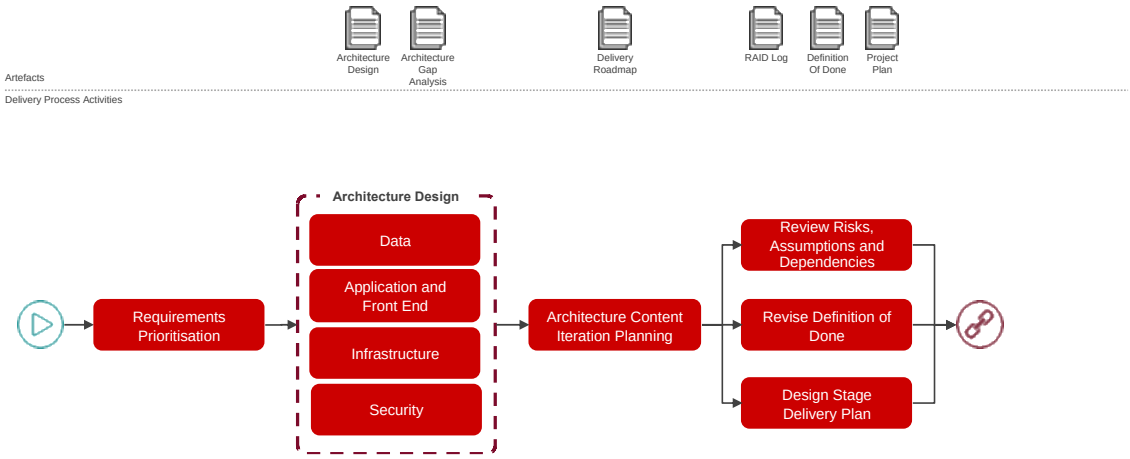
The DESIGN phase brings together the goals, objectives, product backlog and technical landscape, to map out the data, systems and application architectures needed to deliver a solution. For complex engagements, an architectural gap analysis may be produced from the current architecture to help with prioritizing project deliverables and federating architecture delivery into separate delivery streams. In contrast, for small engagements, the design phase may be incorporated into a sprint 0.

Key Activities

- Priorities the product backlog.
- Design the target architecture
- Develop a delivery roadmap
- Review Risks, Issues, Assumptions and Constraints
- Revise Definition of Done and High-Level Plan

How is it done?

Below is the typical process flow we take when running the planning activities. Each process is then explained in detail.



Activity	Description
Requirements Prioritisation	As the requirements list on the product backlog increases, it will become necessary to priorities them in order to focus the design effort and subsequent delivery increments. We do this to

	separate various activities and focus on delivering early wins and the key objectives of the project.
Architecture Design	The production of the baseline (where needed) to target architecture for the project, incorporating data, application, front end, infrastructure and security components. The below 4 sections are in the same document but are detailed separately to provide an itemized breakdown. The architecture design should be submitted to the client for review and sign off.
Data	The production of the conceptual data model and data flows. This will typically be at an entity level and show what source systems are used to feed the data solution. And data security and access policies will also be detailed as well as regulatory impacts.
Application and Front End	Any required application will be detailed in this section. This will cover the data service, application service layer and front-end client needs of the solution. Any required APIs will also be documented.
Infrastructure	Similarly, the Infrastructure design will cover the required solution servers, networks, backup and disaster recovery capabilities needed to support the application and data flows.
Security	Finally, the security architecture will define the authentication and authorisation design for the architecture as well as detailing out any auditing and logging solution.
Architecture Content Iteration Planning	Once the target architecture is being formed, the architecture components and data flows needed to build the solution will become clear, as will any dependencies between these building blocks. This is where the delivery roadmap planning begins to form. The project team will compare the design to the priorities of the project and begin to formulate a sequence for delivery, which may be sequential or in parallel where interdependencies do not exist.
Major Risks, Assumptions and Dependencies	It is important to highlight early any actual or potential major impacts to the project to ensure that adequate contingency is put in place. At a minimum, expectations may need to be reset, for more serious items, it may even lead to the team standing down until resolutions are put in place. It is important to think of the project health and ensure resolution paths are implemented now and not by-passed potentially resulting in a failed project.
Revise Definition of Done	The project team should continuously assess the Definition of Done, so all parties are in agreement and the risk of project conflict is minimized. The end of the design phase is the next critical milestone where any concerns should be addressed.
Project Plan	In conclusion of the design phase, a well-structured plan is delivered. This will incorporate the various architectural and content delivery work streams. Each stream is resourced according to the agreed delivery model.

What is produced?

The following outputs are produced during the plan stage.

Artefact	Description
Architecture Design	The principle design document for the end to end solution architecture covering data, applications, infrastructure and security.
Architecture Gap Analysis	Documents the differences between the current baseline and target architectures. This is then used to scope and resource the delivery plan.
Delivery Roadmap	High level view of the components and data content to be delivered as part of the overall solution, providing details of dependencies that influence the sequence and timing of delivery
RAID Log	Living document for Risks, Actions, Issues and Constraints
Definition of Done	Updated agreement on what the project outcomes should be.
Project Plan	The updated project plan covering all delivery streams, resources and timelines.

Who does what?

The below RACI matrix maps activities and outputs to the project team.

Task	Activity or Output	Engagement Manager	Technical Lead	Business Process Analyst	Technical Consultants	Technical Specialist	Client PM	Business Representative	Product Owner
Requirements Prioritisation	A	AR	C	C	-	-	R	C	C
Architecture Design	A	A	R	C	C	C	C	I	I
Data	A	A	R	C	C	C	C	I	I
Application and Front End	A	A	R	C	C	C	C	I	I
Infrastructure	A	A	R	C	C	C	C	I	I
Security	A	A	R	C	C	C	C	I	I
Architecture Content Iteration Planning	A	AR	C	C	C	C	C	I	I
Major Risks, Assumptions and Dependencies	A	AR	C	C	-	-	R	I	I
Revise Definition of Done	A	AR	C	C	I	I	R	C	C
Design Stage Delivery Plan	A	AR	C	C	I	I	C	C	C

Architecture Design	O	A	R	C	I	I	I	I	I
Architecture Gap Analysis	O	A	R	C	I	I	I	I	I
Delivery Roadmap	O	AR	C	C	I	I	C	I	I
RAID Log	O	AR	C	C	-	-	C	C	C
Definition of Done	O	AR	C	C	-	-	C	C	C
Project Plan	O	AR	C	C	-	-	C	I	I

When do things happen?

Once sufficient information from the Discovery stage has been collected, the Design phase can start in parallel, it cannot complete though until Discovery has completed. This approach enables the project team to parallelise activities and deliver in a more agile manor, focusing on early wins. For very complex projects, it may make more sense to complete the Discovery phase first. This is more waterfall in approach, moving to agile for the execution phase, the benefit here though is that the potential for technical debt is minimised. There is no hard and fast rule, the project team should discuss the potential for risk with the stakeholders who are funding the project.

4.4 Execute

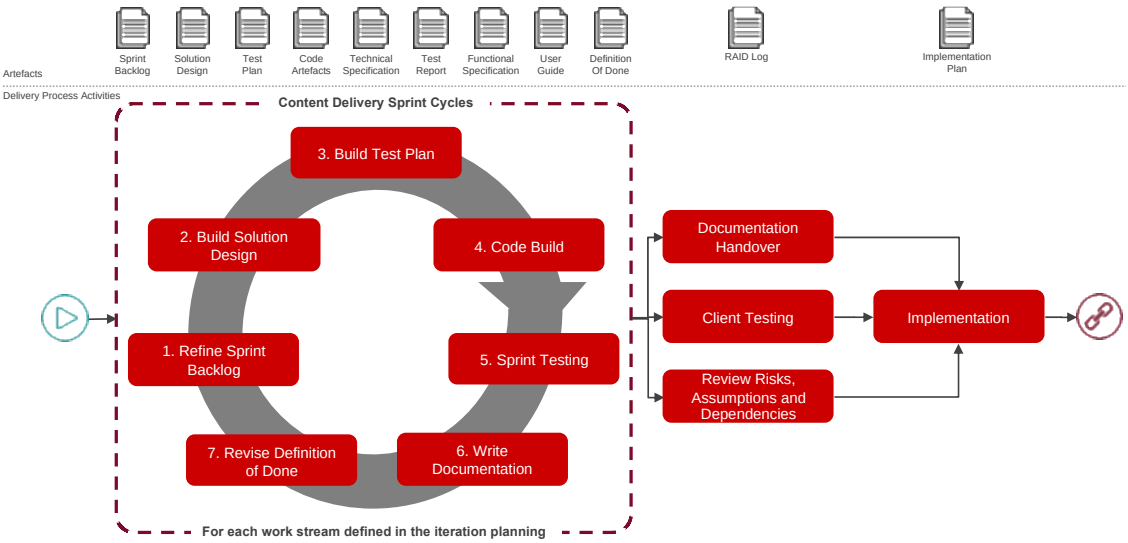
At a point during or at the end of the design phase, there will be a trigger to EXECUTE and begin implementing the architecture and content. Using our Agile approach, sprints will be initiated, leveraging our proven processes and best practices. In line QA processes and automated testing help ensure we deliver right first time and prevent incremental regression issues.

Key Activities

- Sprint activity to deliver the architecture and the content
- Parallelize work stream activities
- Documentation and handover
- Client Integration, UAT and Performance Testing
- Implementation

How is it done?

Below is the typical process flow we take when running the planning activities. Each process is then explained in detail.



Activity	Description
Content Delivery Sprint Cycles	Within the Design phase, Architecture and Content Iteration Planning breaks out the different work streams, basically enabling the effort to be structured into parallel and dependent elements. For each work stream an independent Content Delivery Sprint Cycle is started. This means that for a larger project there may be multiple sprint streams running in parallel, each delivering its own content. Our content delivery process leverages the Scrum methodology, a detailed overview can be found in sections 5.1 and 5.2.
Refine Sprint Backlog	With the sprint backlog from the Discovery stage as an input, the refining process ensures that we continue to define and priorities the sprint backlog.
Build Solution Design	The Design phase is focused on building out the architecture design for the overall project. The Solution Design provides a detailed overview of what the sprint cycle is delivering. For smaller engagements the architecture design may be sufficient, however for larger projects, it may be necessary to produce multiple designs to fulfil the needs of different audiences. For example, the architecture design will provide details of deployment architectures and data flows, the solution design will provide much more detail of aspects such as staging area folder construction, data mart designs and key business rules that are implemented.
Build Test Plan	As the sprint begins it is important to understand how the deliverables will be tested. Within our methodology we use a QA function to define and build out manual or automated tests as part of each sprint, this enables the development team to improve quality and minimize the introduction of regression errors. We aim to leverage the clients automated environment environment. The project team also need to agree how users will support the sprint cycle testing since our agile approach brings the users closer to the developers to reduce feedback times for changes and quickly steer the project back on course if needed.
Code Build	All coding effort should be fully documented within the actual code. Creating pages of detailed technical documents soon becomes difficult to maintain and burdensome, having the narrative in the data flows is far more helpful and easier to maintain. Available to the project team are a large number of best practices and coding standards to guarantee the quality and consistency of deliverables.
Sprint Testing	The sprint team QA function will be responsible for executing the sprint testing using the automated and manual tests developed in sprint. At the end of each sprint cycle, a test report is produced.
Write Documentation	Whilst we avoid generating excessive documentation, there are some key deliverables from each sprint phase. A technical and functional specification are produced. The technical specification provides operations with a list of technical deliverables whilst the functional specification provides users with a summary of what business functionality is being delivered. Where requested the project team can enhance the functional

	specification to create a user guide, which clearly benefits the rollout of the solutions, particularly where the business team may not have been involved in the sprint testing.
Revise Definition of Done	The project team should continuously assess the Definition of Done, so all parties are in agreement and the risk of project conflict is minimized. As the sprint teams begin to develop the solution, the team should always be aware of what they need to deliver and escalate should there be a misalignment.
Documentation Handover	Once the sprint cycle has completed the appropriate documentation should be handed over to the client teams. The technical governance board should review the solutions design. Operations should review and accept the technical specification, likewise the business users accept the functional specification and user guides. Any feedback should be captured in the sprint retrospective and added to the product backlog is necessary.
Client Testing	The approach to client testing will be agreed in the Delivery Plan. Typically, we will conduct UAT and Performance testing. For larger projects, Integration testing will be executed bringing together the multiple streams and any other project activities. In addition any required Penetration or Performance testing should be completed according to the non-functional requirements gathered during the Discovery and Design phases.
Review Risk, Assumptions and Constraints	As part of the execution phase, it is important to stay on top of any project impacts and impediments that prevent or delay development. Any significant impacts should be escalated to the Engagement Manager. At this stage all risks, issues, assumptions and constraints should be reviewed and that any agreed resolution path has been delivered.
Implementation	The implementation steps should be agreed as part of the Delivery Plan. Often this will be undertaken by the client operations team with our support. This must ensure that all relevant enablement activities have been conducted.

What is produced?

The following outputs are produced during the plan stage.

Artefact	Description
Sprint Backlog	Detailed list of the solution user stories
Solution Design	Detailed technical view of the solution
Test Plan	Plan for Unit Testing and User Acceptance testing
Code Artefacts	The developed code
Technical Specification	Detailed view of the technical specification delivered within sprint. Can also be used as the operational guide.
Test Report	End of sprint report
Functional Specification	Detailed view of the functional capabilities delivered within sprint
User Guide	Any user guides that need to be delivered to end users

Definition of Done	Updated agreement on what the project outcomes should be.
RAID Log	Living document for Risks, Actions, Issues and Constraints
Implementation Plan	Planned activities to implement the solution into the live environment.

Please see section 5.1 and 5.2 for a comprehensive overview of our agile methodology.

Who does what?

The below RACI matrix maps activities and outputs to the project team.

Task	Activity or Output	Engagement Manager	Technical Lead	Business Process Analyst	Technical Consultants	Technical Specialist	Client PM	Business Representative	Product Owner
Content Delivery Sprint Cycles	A	AR	C	C	C	C	R	C	C
Refine Sprint Backlog	A	A	C	R	C	C	C	C	A/R
Build Solution Design	A	C	A	C	R	-	I	I	I
Build Test Plan	A	C	C	A	R	C	I	I	I
Code Build	A	I	A	-	R	C	I	-	-
Sprint Testing	A	I	A	R	C	C	I	C	C
Write Documentation	A	I	A	I	R	C	I	-	-
Revise Definition of Done	A	AR	C	C	C	C	C	C	C
Documentation Handover	A	A	-	C	R	C	I	I	I
Client Testing	A	R	I	I	I	I	A	C	R
Review Risk, Assumptions and Constraints	A	AR	C	C	-	-	R	I	I
Implementation	A	AR	C	C	C	C	R	I	I
Sprint Backlog	O	A	C	R	C	C	C	C	C
Solution Design	O	C	A	C	R	I	I	I	I
Test Plan	O	C	C	A	R	I	I	I	I

Code Artefacts	O	I	A	-	R	C	-	-	-
Technical Specification	O	I	A	-	R	C	-	-	-
Test Report	O	I	A	I	R	C	I	I	I
Functional Specification	O	I	A	C	R	I	I	I	I
User Guide	O	I	I	A	R	C	I	I	I
Definition of Done	O	AR	C	C	-	-	C	C	C
RAID Log	O	AR	C	C	-	-	C	C	C
Implementation Plan	O	AR	C	C	-	-	C	I	I

When do things happen?

The execution stage can begin once the design phase has begun. Similar to the design phase not needing Discovery to complete before starting, so the execute phase does not need either the Discovery or Design phase to complete before beginning. Again, this is to enable the delivery function to start as soon as is sensible to enable early wins.

4.5 Enable

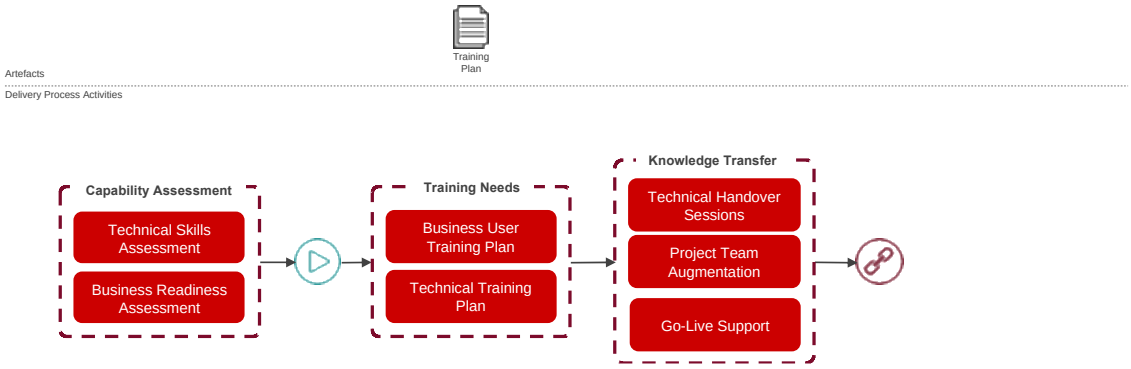
Key to any engagement is to ENABLE the client. This comes in many forms, from training (custom and product bespoke) to enabling a client's own technical resources by augmenting scrum teams. We will help assess the client's capability to take on new technology, highlighting risks and necessary actions.

Key Outputs

- Technical and business training plan
- Solution Handover sessions
- Post Implementation Support
- Celebrate Project Success

How is it done?

Below is the typical process flow we take when running the planning activities. Each process is then explained in detail.



Activity	Description
Training Needs	As part of the capability assessment, the existing skill sets of the technical and business resources are reviewed. Where there are gaps, the project team needs to work towards ensuring that the wider team is prepared. Our training team will work with the project team to plan out what training activities will be needed and align these to go-live milestones. This provides a well-timed ramp up for all client resources.
Business User Training Plan	Will focus on preparing business users for the new solutions to be launched. User training can be classed as either classroom

	or tailored. Class room training relates to our standard class training courses, whilst tailored aligns the training to the solution. With tailored training, our training team will work with the project team to build bespoke materials, this approach accelerates user adoption since the team are learning how to use the solution.
Technical Training Plan	For the technical teams there are a wide range of standard class courses that provide a strong introduction to the Pentaho toolset. In addition, our website contains details of many best practices for Pentaho deployments.
Knowledge Transfer	Knowledge Transfer is an important part of enablement, and we have a great deal of experience in conducting this. Principally, this enables us to ensure that clients are able to build, support and modify their own solutions should they wish.
Technical Handover Sessions	These sessions are designed to provide a detailed walkthrough of the solution design and code delivered, including best practices and design patterns adopted within the solution.
Project Team Augmentation	Augmentation refers to when a client's technical resources are seconded to the sprint teams delivering the solution or vice-versa. We will maintain the key role of design authority but encourage those clients who need to ramp up their technical resources to allocate some of their resources to the project team.
Go-Live Support	Once the solution goes live, we will provide post implementation support and where requested floorwalkers to help with user enablement. Our support team will then ensure that product Q&A is resolved promptly.

What is produced?

The following outputs are produced during the plan stage.

Artefact	Description
Training Plan	The detailed plan linked to project milestones. Contain details of committed training courses and attendees.

Who does what?

The below RACI matrix maps activities and outputs to the project team.

Task	Activity or Output	Engagement Manager	Technical Lead	Business Process Analyst	Technical Consultants	Technical Specialist	Client PM	Business Representative	Product Owner
Training Needs	A	A	I	R	-	-	R	C	C
Business User Training Plan	A	A	I	R	-	-	R	C	C
Technical Training Plan	A	A	C	R	-	-	R	C	C
Knowledge Transfer	A	A	R	I	C	C	R	-	-
Technical Handover Sessions	A	A	R	I	C	C	R	-	I
Project Team Augmentation	A	AR	-	-	-	-	R	-	-
Go-Live Support	A	AR	C	C	-	-	I	I	I
Training Plan	O	A	I	R	-	-	R	C	C

When do things happen?

Enablement should be discussed during the project kickoff and implemented through the project lifecycle.

5 Tools for Success

The VANTAGE Delivery Methodology is supported by multiple key processes and technical elements to accelerate a successful project delivery. This section discusses the tools to success and covers a range of topics going from Agile+ and Scrum, to Continuous Integration and the application of our extensive library of technical best practices.

5.1 Agile Project Delivery

The VANTAGE Delivery Methodology is profoundly supported by Agile project delivery principles to achieve a higher project success rate. But what defines an Agile project delivery, why do we need it and how does it accelerate project success? This section delves deeper into Agile Project Delivery topics.

Introduction

Agile is an umbrella term for a set of methods and practices based on Agile Manifesto that aims to develop better software by preferring

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

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

Collective experience of practitioners and industry research has shown that Agile methods produce a higher success rate than traditional methods like Waterfall. Often traditionally run projects go over budget and go beyond the committed delivery date. Moreover, the quality of delivered projects is rarely good. It has bugs and fixing takes more time. Result is unhappy customers and low ROI.

A quality, on time and stable delivery results in motivated and happy customers. These customers become our strongest promoters recommending our product and services.

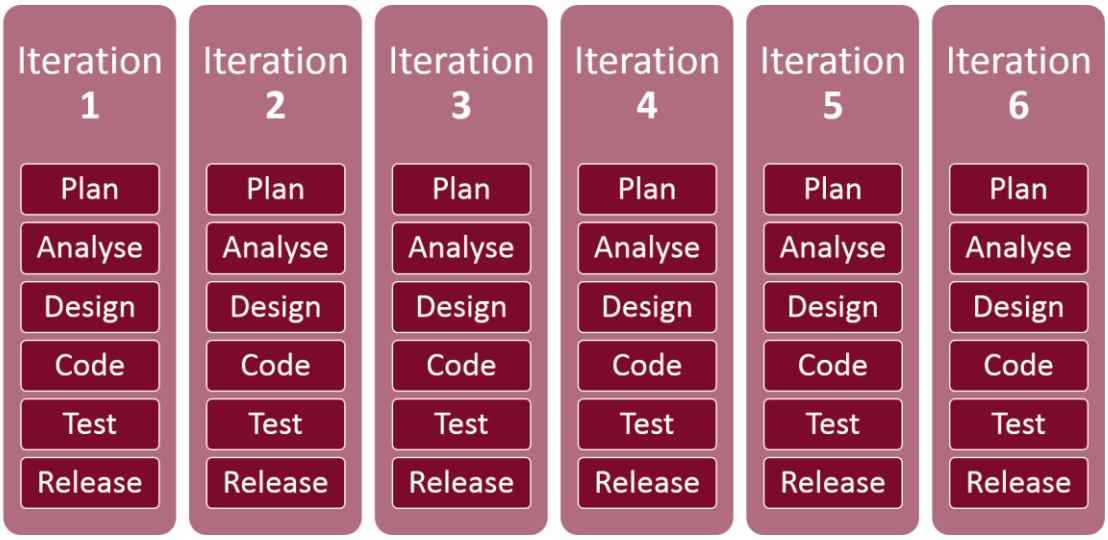
Agile has achieved this in a number of industries on a consistent basis. As with any methodology it’s not always rainbows and unicorns in the Agile world too, hence this guide also covers the challenges in setting up and implementing Agile projects.

Achieving Success

In a traditional delivery model, the entire project is planned beforehand and executed sequentially.

Our example project takes 3 months to finish and the customer gets to evaluate the outcomes only after the release stage. This evaluation feedback potentially uncovers incomplete or irrelevant requirements, unusable features and poor solution design. This not only results in more work in the form of another project, change requests or support tasks but also leads to a negative customer experience.

In contrast to this, Agile recommends iterations - short release cycles in tandem with feedback loops allowing for course correction early in the project. Typical release cycles are 2 to 4 weeks long, where the time length gets decided and agreed on before starting the project. For instance, the 3-month example project is now divided in 6 iterations of 2 weeks.



Every time an iteration is started, the scope for that iteration is planned before starting any work. Only that part of the full solution scope that the project team deems achievable in the iteration period (2 weeks) is agreed. This means that the time is fixed and that the scope is agreed to fit into the time box of 2 weeks.

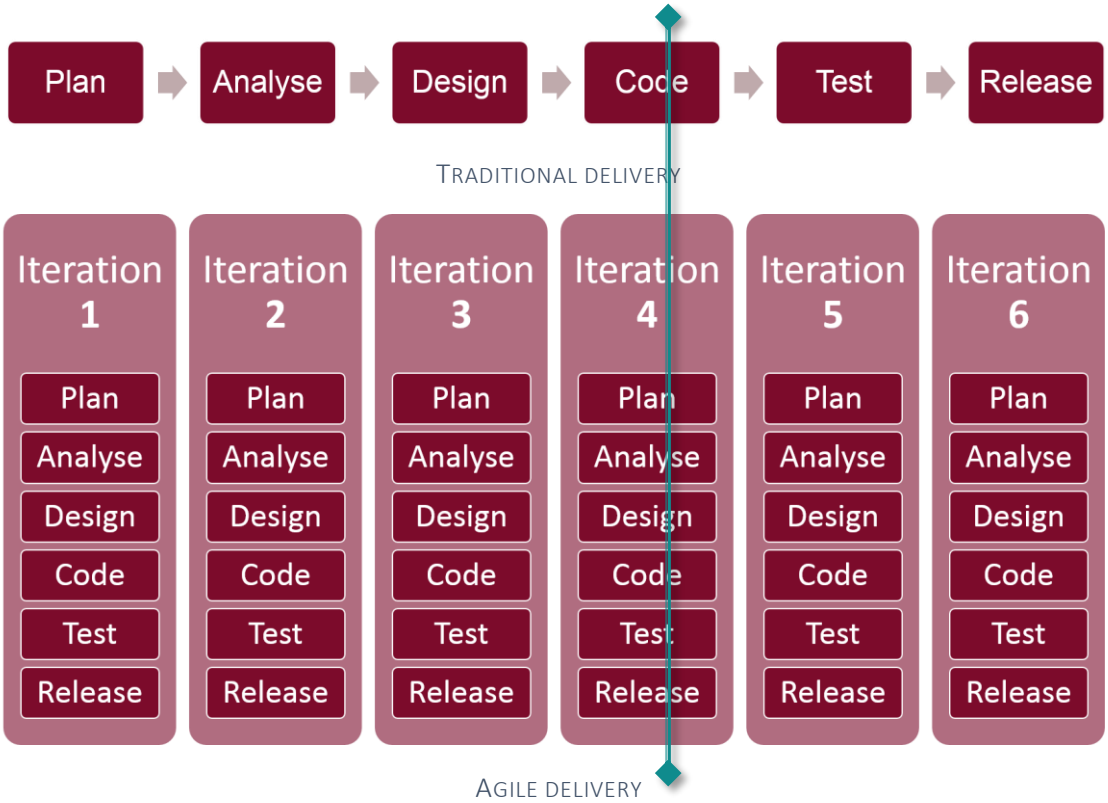
Planning is done by prioritizing items that would deliver the highest business value first. As planning is value driven, regular customer engagement comes naturally to the process. During planning, customer and project team decide which high value items could be delivered in the upcoming release considering design constrains and other dependencies. Higher valued items that could be released after two weeks are selected and lower valued items are deferred for following iterations.

An interaction release is an artefact that could be tested by the customer and delivers business value. Based on the outcomes of the test, customer shares feedback with the project team and reviews the scope for the next iteration. As feedback is shared so early the team gets to pivot early if deemed necessary. This is an improvement over the traditional model where response to feedback is only possible at the end of the project. Iterative reviews are instrumental in de-risking the delivery. More on delivery risk in the next section.

Agile as such achieves success by being iterative and adaptive i.e. adapting quickly to changing circumstances. Traditional model is more predictive and relies quite heavily on analysis and planning with micromanagement and change control to achieve project success.

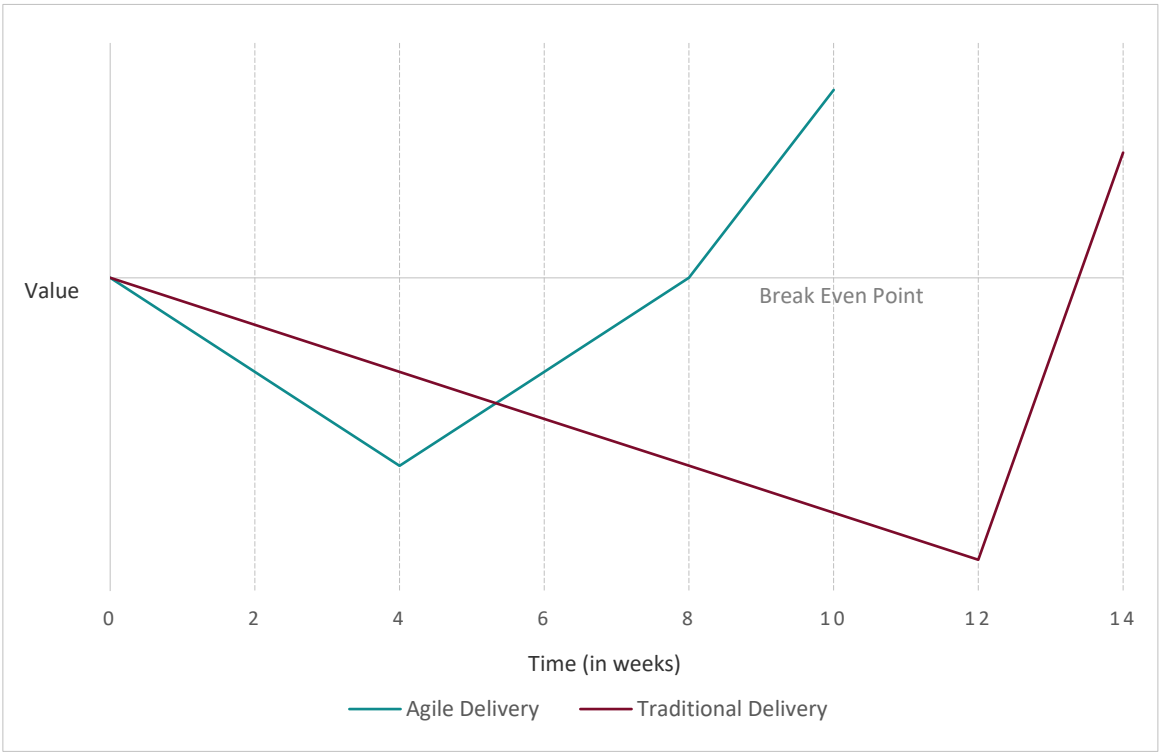
Reducing Delivery Risk

Consider the example project is running out of budget at around 2 months' time denoted by the green line in the following diagram.



In the traditional model, nothing has been delivered yet as some code has been done but the team is not sure if it works as it hasn't been tested and the solution is far from being released.

In an agile model, most valuable items are already delivered. The reality is, more than half the project is ready to be released or may have been already released to the customer at this time. The lower value deliverables are not complete, work on them may have been started but it's not done. This isn't an ideal situation but at least not a complete disaster as in the traditional case.



The chart above compares the traditional and agile models for project value delivered over time. In a traditional project the first few months are spent in requirements gathering, analysis and design phases. None of the functionalities have been delivered creating negative business value and exposing project to financial risk. There are bright chances that technology and competitors have moved ahead rendering the project requirements and design outdated by the time planning is finished.

Whereas in agile, planning sessions for every iteration tend to be short as delivery timelines are shorter. Emphasis being on delivering value every two weeks, the scope is smaller, and planning takes less time. Requirements are created just-in-time for an iterative delivery. The customer receives working functionality every two weeks resulting in lower financial risk due to a shorter release time and shorter payback period. The customer has more control over what should be scoped for next iteration based on market forces and available resources and technology.

In a vendor-partner environment where internal and external teams are extremely mobile moving between projects and customers, the agile model helps in reducing the impact of the mobility. Imagine changing teams in middle of traditional delivery project. On-boarding and knowledge transfer of a bigger unilateral scope adds more time and risk. While in agile the scope being small and relevant only for next two weeks makes an easier on-boarding. It also invites a different perspective to planning sessions whenever the team changes bringing new ideas and sometimes better solution paths. Teams are able to adapt without abandoning a plan.

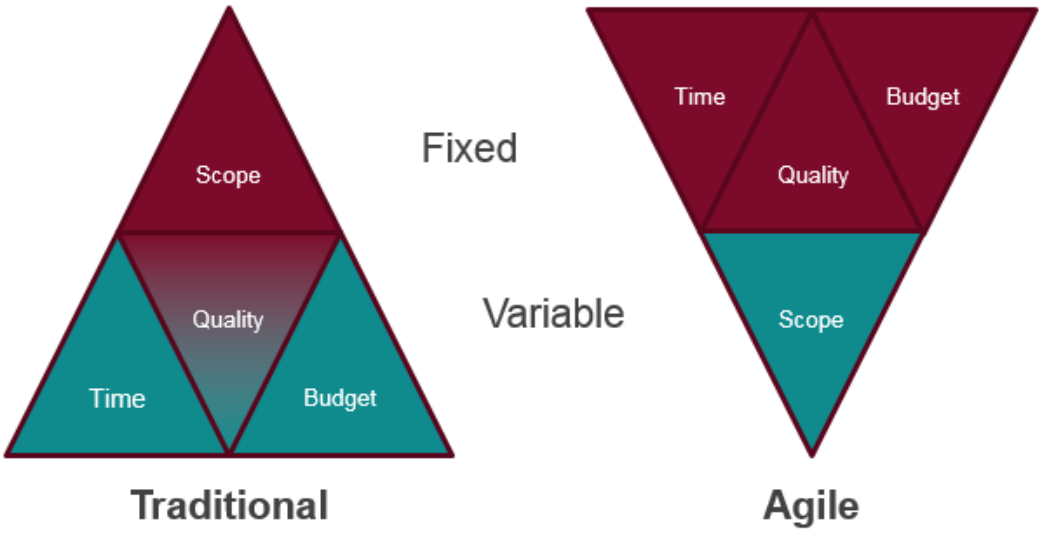
This helps in managing expectations and reducing surprises so that customers know what they are getting and roughly when they are getting it. The central idea is to make scope flexible by creating requirements just-in-time.

Flexible Scope

The traditional model for project delivery is driven by the primary goal on completing the requirements and features that make up the initial scope. Owing to this unilateral prioritization, customers unintentionally accept that the delivery time and budget will be variable. Halfway through the project if they realize project won't hit the committed delivery date the only variables they can play with are:

1. Time – accept a later release date
2. Budget – add more resources to the project which will increase the cost

The change in delivery plan and cost affects the quality acceptance criteria as its seldom revisited after the initial agreement.



Unlike the traditional projects, agile projects have a fixed time box and budget while the scope varies.

When delivery dates are fixed, customer is aware that a working increment of the software will be delivered based on an agreed scope. This also helps in planning deliveries of other dependent systems in a multiple vendor environment.

Having fixed budget forces customer to decide what priority items based on business value have to be delivered for a successful outcome of the engagement.

Quality acceptance criteria is defined at the start of the project and can be maintained throughout by adapting to the changing priorities of deliverables.

As the project moves along and vision of the outcome becomes clearer new requirements can be included that would improve the usability and efficacy of the features. Less valuable features can be dropped or might not even get created during this exercise.

5.2 Scrum

Although the agile project delivery principles discussed in the previous section provide the general approach in how we deliver Analytics and IOT projects, the actual delivery processes and practicalities are based on the Scrum Agile methodology that we embed into the VANTAGE Delivery Methodology stages.

This section introduces the Scrum methodology and discusses its main elements, the project roles and responsibilities and highlights the importance of breaking down stories into functional tasks, in order to give the product owner the full flexibility in constructing a worthwhile workload for each project iteration.

Introduction

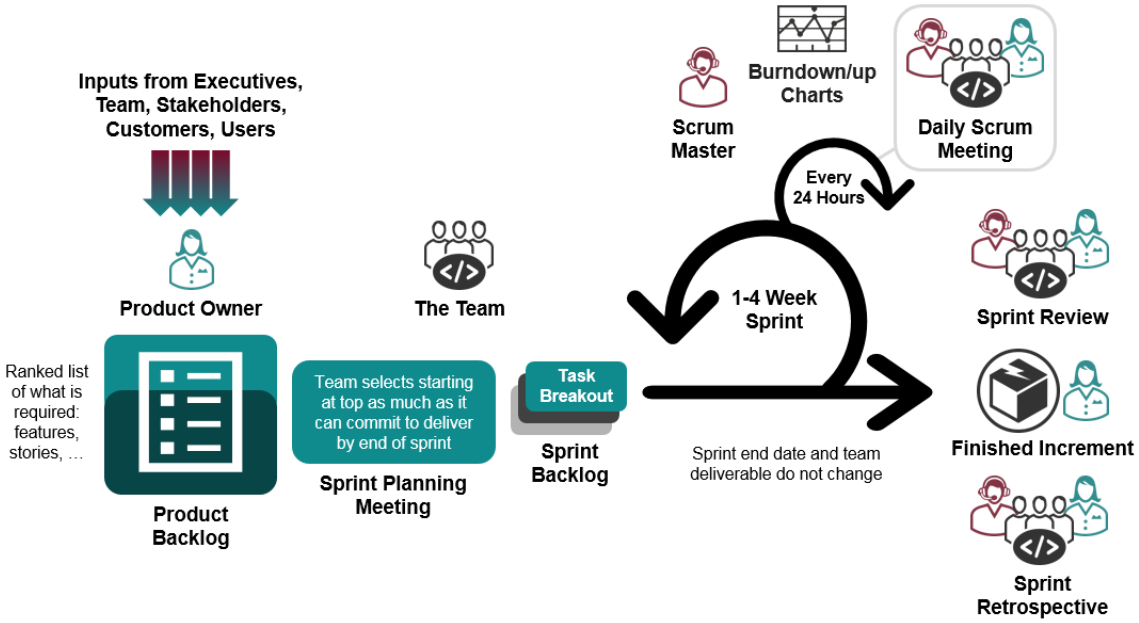
There are multiple agile processes that apply the manifesto in different ways. Scrum relies on planning, execution, and constant review. The main goals of Scrum are:

- Focus on top priorities
- Maintain transparency
- Fail fast

Scrum consists of repeating time-boxed iterations called sprints. A sprint can be 1-4 weeks long. Shorter sprints usually generate too much administrative overhead to be productive. Longer sprints water down the benefits of adapting an agile approach.

Using Scrum, development teams are able to focus on small, concrete tasks at hand during the sprint without getting bogged down by big-picture complexity. At the same time, the big picture is never out of sight.

Scrum Definitions



- Product - Functionality that a Product Owner has identified as important
- Product Owner - Writes product requirements into stories, creates releases
- Story - A brief statement of a product requirement or a customer business case
- Product Backlog - Defines a complete, ordered list of stories, commonly grouped into themes or epics
- Scrum Master - Facilitates Scrum by moving stories into sprints.
- Sprint - The basic unit of development in the Scrum process.
- Sprint Planning - Defines the stories being worked in the current sprint
- Sprint Backlog - Defines the detailed and granular tasks for the current sprint
- Daily Scrum Meeting - 15-minute standup meeting where each developer discusses what he accomplished yesterday, what he plans to accomplish today and any impediments or risks he faces
- Sprint Review - End of Sprint meeting with developers and stakeholders to present what was accomplished and receive feedback. Feeds back into the Product Backlog for future Sprints
- Sprint Retrospective - Final event of the Sprint, held after the Sprint Review meeting, that enables the Scrum Team to continuously improve their approach to their work

Story

A user story is a brief statement of a product requirement or a customer business case created by a Product Owner. Typically, stories are expressed in plain language to help the reader understand what the solution should accomplish. Stories represent a unit of work that can be resolved in one sprint. Stories that take longer than a sprint to complete should be broken into one or more stories and grouped into an epic. Several epics can be grouped into themes, which are the highest conceptual level of the Scrum

process. User stories focus on customer value and encourage lack of requirement specificity in order to foster a higher level of collaboration between the stakeholders and the team. A user story is a metaphor for the work being done, not a full description of the work. The actual work being done is fleshed out via collaboration revolving around the user story as development progresses.

User stories are an excellent way to capture the requirements into the product backlog. They not only capture what needs to be done, but also capture who will use it, how, and why. Likewise, shorter time frames and acceptance criteria keep up the time and result focus. The team's awareness, interaction, and collaboration help focus on software creation, which enables the teams to use lightweight processes and documentation.

A good user story uses the "INVEST" model:

- Independent – reduced dependencies = easier to plan
- Negotiable – details added via collaboration
- Valuable – provides value to the customer
- Estimable – too big or too vague = not estimable
- Small – can be done in less than a week by the team
- Testable – good acceptance criteria

Sprint

The sprint is the basic unit of development in the Scrum process. The Scrum Master creates a sprint and assigns user stories to it based on what the development team can complete in a finite time period.

A sprint can be of any length, but typically takes between one and four weeks to finish. The goal of each sprint is to produce a fully tested and releasable solution. For this to occur, the effort required to complete the stories must match the capacity of the development team.

The importance of the fact that it must be possible to deploy the solution to production at the completion of the print cannot be underestimated. This principle informs the way the development team must break down stories into tasks and requires the development team to define the project's branching, testing and documentation strategies. At the end of a sprint a story must be objectively done or not done. If it is accepted as done, it is in the code base and it can be deployed. If it is not, it is expected to not be in the code base, or at least not cause any side effects or errors in other features. It is the development team's responsibility to come up with the techniques and mechanisms to ensure that this is the case. At the end of each sprint the product owner must have the option to "ship it" to production.

Scrum Roles

- Development Team
    - Break down stories into tasks and estimate time needed during sprint planning.

    - Implement the solution, tests and documentation for each feature/story during the sprint.
    - Raise any impediments/blockers with the scrum master immediately.
    - Update own tasks with estimated time left every day.
    - Demonstrate completed tasks during sprint review.
- Technical Lead
    - Solution architecture and design
    - Testing strategy
    - Branching strategy
    - Deployment strategy
    - Documentation strategy
    - Infrastructure and tooling decisions
    - Project structure (repository layout etc.)
    - Ensure team observes project standards.
    - Provide training for new developers joining the team.
    - For the purposes of sprint planning / story development the TL is a regular developer.
- Scrum Master
    - Responsible for management of Scrum practices
    - Make sure things are running smoothly and processes are being followed
    - Scrum Master is the first level of escalation for impediments and blockers. The Scrum master is responsible for chasing solutions and unblocking the development team.
    - Run daily standup meeting, presenting team burn-down chart. Inquire about causes of delays early to identify and eliminate possible impediments.
- Product Owner
    - Liaise with stakeholders to identify features and priorities
    - Maintain and prioritize backlog of stories, also observing dependencies across stories.
    - Monitor mid-term project metrics to assessing overall project progress.
    - Present stories during sprint planning to development team.
    - Own stories: must be able to answer developer questions arising during planning and development. Must be able to make decisions regarding to scope and implementation details.

Sprint Cycle

A sprint cycle consists of 4 recurring phases typical for a Scrum-based workflow:

1. Sprint Planning
2. Development time with daily standup meetings
3. Sprint Review
4. Sprint Retrospective

5.2.1.1 Sprint Planning

The goals of Sprint Planning are to:

- Transform requirements and features into tangible tasks.

- Clarify any questions and close any gaps in understanding between product owner and development team.
- Estimate the effort needed to complete the tasks.
- Arrive at a set of tasks that the development team commits to deliver until the end of the sprint.

5.2.1.1.1 Phase 1: Story Presentation

Sprint Planning starts with the product owner presenting the development team with the stories to be estimated. The development team self-organizes into individuals and small groups picking these stories up and transforming them into tasks that they feel confident to implement. Flipcharts, drawing boards, etc. are instrumental in the discussions among the team members and product owner during planning.

The development team needs to be sure they understand the requirements completely and must discuss any edge-cases, error handling and acceptance criteria with the product owner to close any gaps in understanding.

5.2.1.1.2 Phase 2: Creating Tasks and Estimates

Then the developers go on to break the stories down into tasks and estimate the time needed for completion in hours. All tasks are expected to have the following properties:

1. A task must not be larger than 6 hours. It is assumed that it is hiding intangible complexity, and an effort must be made to break it down further to properly work it out.
2. A developer must be able to objectively demonstrate that a task is done on the integration environment. It must be tangible for the product owner.
3. A task is a vertical slice of functionality. Horizontal technical architecture layer slices are not accepted as tasks.

Rules 1 and 2 are easily observed and enforced. Rule number 3 is very important but is often violated until the development team gets used to breaking down stories into vertical slices. For example, if the story is to create a new OLAP cube with dimensions a,b,c,d and measures y and z from an existing DWH, developers might break it down as:

Task	Estimate
ETL to build the cube	6h
OLAP schema	2h
Deployment script	2h
Integration Test	2h
Document Dimensions and Measures in data dictionary	1h
Total	13h

The above tasks cannot be accepted, even though they might seem plausible estimates, and reflect the way the developers work. The above makes it impossible to track progress and make decisions on this story in practice.

A proper break down looks more like this:

Task	Estimate
Initial cube with measure y and dimension a	4h
Add dimension b	1h
Add dimension c	2h
Add dimension d	2h
Add measure z	4h
Integration Test	2h
Document Dimensions and Measures in data dictionary	1h
Total	16h

The initial estimate had ETL, Schema, and deployment script tasks. These all violate rule 2 and 3. These are not tangible to the product owner. The product owner wants to use an analytics cube. The new break down follows the following rationale: the initial cube is built with the simplest dimension and measure included. The ETL, schema and deployment script are necessary technically, but have no value to the product owner. Therefore, they are not mentioned. It takes 4h to create a simple but working cube, and make sure it works on all environments. Adding dimensions and measures to expand on that takes additional time. The integration test is a separate entity, as is the documentation, so they remain as separate tasks. The new estimate ends up being somewhat higher. It is often the case that a task will initially appear with the highest allowed time estimate and hide some complexity that should be spelled out. Once that effort is made, it turns out that the task takes more effort, but the break down is more realistic and it also enables the product owner to decide that the troublesome measure z may have to go. Its cost/benefit ratio may not be good enough. And if one of the tasks cannot be completed during the sprint (say, dimension c proved difficult or impossible to implement), the product owner has the option of shipping the cube to production without it.

5.2.1.1.3 Phase 3: Commitment

Once all tasks are estimated in hours. They are collected into a spreadsheet and held against the time available in the sprint. The total hours available is calculated as working days in sprint x team size x working hours a day. Assuming 7 working hours a day (depending on established obligations/meetings/expected distractions), a sprint length of 3 weeks = 15 days – 2 days for sprint planning and sprint review, and a

team size of 6 the total amount of working time is 13 x 6 x 7h = 546h. Any known time off is subtracted from that.

Usually the total number of tasks will exceed the available time. The product owner shifts tasks in and out of scope as per business requirements to achieve the most valuable outcome in the time available. At the end the team is asked to give a final assessment of whether they can commit to completing all tasks in scope. Any corrections to the estimates need to be made now. Any risks or unknowns must be addressed at this point. At the end of the sprint planning the team commits to completing the in-scope items.

Sprint planning should usually be a full day activity. Usually a considerable effort goes into understanding requirements correctly, breaking down the tasks do some research work in case new technologies or APIs are used etc. The reliability of the team commitment is very much determined by the effort put into sprint planning. Small teams working short sprints may be the exception.

After the planning, tasks committed to would be entered into an agile project management tool. The team must assign tasks and update the estimated time to complete each of the tasks. Producing the data for the burn-down chart.

5.2.1.2 Daily Stand Up

During the sprint, there is a single meeting the team must attend to. The daily stand up meeting. The goals of this meeting are to:

- Make sure the team updates their remaining time estimates for tasks.
- Address any reasons the burn-down is not making expected progress. This includes any impediments or blockers the developers are facing. This includes resolving technical difficulties individual developers might face by reaching out to domain experts or teaming members up to address an issue as a tag-team.

5.2.1.3 Sprint Review

The Sprint Review is similar to sprint planning in the sense that it is usually a half-day or full-day activity. Its purpose is for the development team to demonstrate completed functionality to the product owner, and the product owner deciding whether to accept or reject a story. If a task cannot be shown as completed on the integration environment, it is considered failed. The reasons are not important, if it could not be deployed to the integration environment for any reason at all, it cannot be shipped to production, and more work is necessary to make it happen.

Each developer presents the stories/tasks they have been working on. The product owner formally accepts or rejects each story and task. If a task is rejected it is typically rolled over to the next sprint, but the product owner may just as well decide that other stories are more important. Since the work is available on the integration server, the product owner is welcome to personally try the implemented functionality.

After the sprint review a stable version of the solution is tagged in version control, and the product owner decides if and when this version should be deployed to UAT or production.

5.2.1.4 Sprint Retrospective

The Sprint Retrospective is the final event of the Sprint, held after the Sprint Review meeting and prior to the next Sprint Planning, that is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

This is at most a three-hour meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. All team members should attend.

During the Sprint Retrospective, the team discusses:

- What went well in the Sprint
- What could be improved
- What will we commit to improve in the next Sprint

The Scrum Master encourages the team to improve its development process and practices to make it more effective and to ultimately increase the solution quality.

By the end of the Sprint Retrospective, the team should have identified improvements that it will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation.

Tracking Progress

The burn-down chart, showing the daily development of estimated hours left of all tasks in a sprint is the most important short-term reporting tool. Any deviations from the linear burn-down must be investigated immediately, especially after the mid-point of a sprint. Sometimes work just takes longer, meaning that the estimates were not very good. Sometimes external dependencies cause unexpected delays or infrastructure is an issue. If the issues cannot be resolved and the sprint nears its end, the team must still strive to deliver a potentially shippable solution. That means that half-done or defective pieces that are not expected to be finished must be isolated from the stable code base.

Tool Support

The simplest tools are best when working with this approach. Often a simple Excel template is simple and sufficient, especially when collecting tasks during Sprint Planning. For tracking progress, and accepting/rejecting tasks, a scrum-specific tool is indispensable. The developers must be able to assign tasks in the team, and update the estimated time to completion on them, which in turn generates a burn-down chart for the sprint, individual stories, and individual developers.

5.3 DevOps

Using Scrum, the goal of each sprint is to produce a fully functioning piece of software that has been fully tested and is ready to be released into the final solution that is being developed. These short intervals are there to deliver results quickly and incrementally,

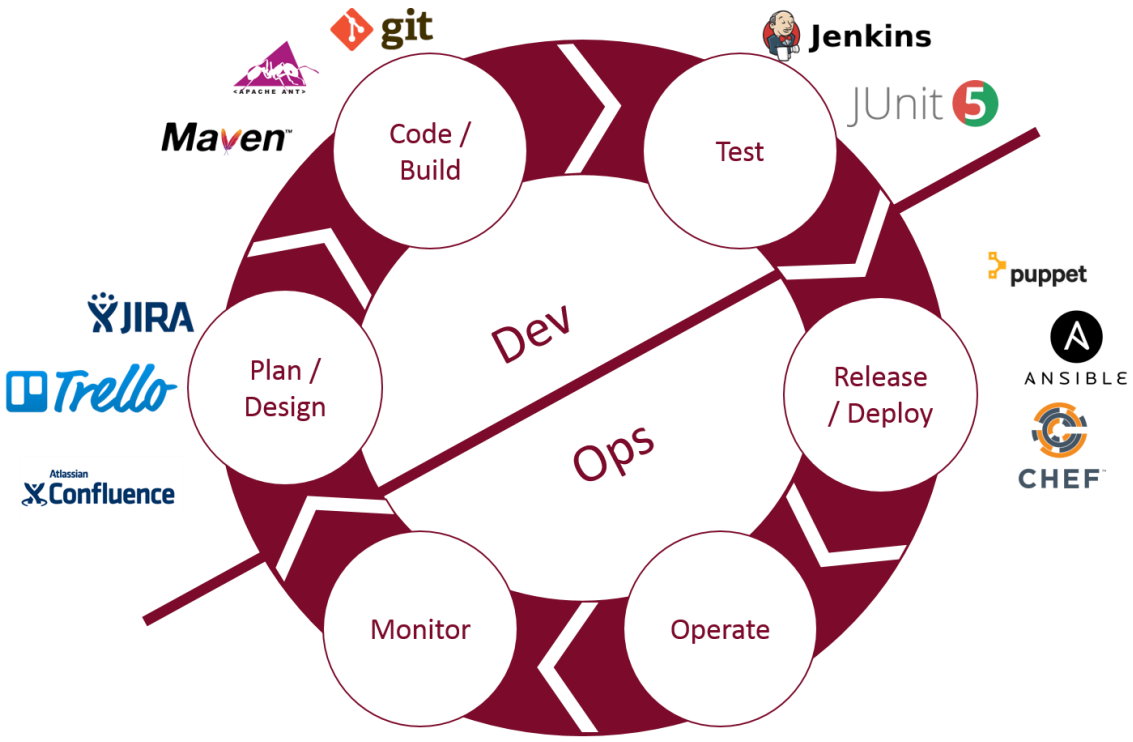
using frequent feedback loops to keep the project on the right track and to maximize the value of the solution.

Delivering on such promises requires more than having such an agile feedback loop at the functional level. In order to accelerate project success, the same continuous feedback and release process is needed at the code level of the solution providing us with the technical practices and guarantees that the new solution increment is properly working according to technical specifications and that we can reliably integrate the iteration into the production environment.

The set of practices intended to reduce the time between committing an increment to a solution and the increment being placed into production, while ensuring high quality is called DevOps and using agile and DevOps in tandem increases project success and productivity: Agile for planning and development, DevOps for testing, deployment and operations.

DevOps is referred to as a set of practices that are centered around communication, collaboration and integration between software development and IT operations teams and automating the processes between them. Its main objective is to provide guidance on creating an automated environment where iteratively building, testing and releasing a software solution can be faster and more reliable.

The DevOps lifecycle is demonstrated by the following diagram:



DevOps Maturity Phases

There are several phases in the process of adopting DevOps in the organization, which are in nature an evolution of each other.

5.3.1.1 Continuous Integration

Continuous Integration is often the first step of a DevOps implementation in an organization and is the most common level achieved. Continuous integration supports the agile project delivery methodology with a software development lifecycle that has great emphasis on test automation to check that the application is not broken whenever new iterations are integrated into the central solution repository.

Development teams need to continuously merge their changes back to the central solution repository as often as possible. This way, the team's changes can be validated by creating a solution build and running automated tests against the build. By doing so, the developer gets immediate feedback and can fix bugs easily if required and you avoid the integration hell that usually happens when people wait for release day to merge their changes into the central repository.

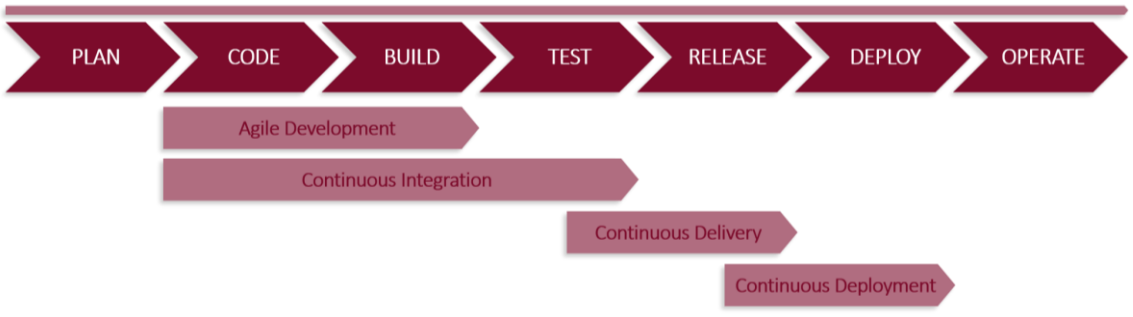
5.3.1.2 Continuous Delivery

Continuous Delivery builds on continuous integration to make sure that you can release new iterations to the customer quickly in a sustainable way. This means that on top of automating your testing, you can also automate your release process. As a result, you can deploy your application at any point of time without practically any human intervention up to the point of the actual environment deployment where manual intervention is still needed.

5.3.1.3 Continuous Deployment

Continuous Deployment goes one step further than continuous delivery by providing an end-to-end solution deployment into production without any human intervention. As a result, every solution increment that passes all stages of your software development lifecycle is automatically released to your end-users. This way, continuous deployment accelerates the feedback loop with your end-users by having a continuous release process

The different DevOps maturity phases and how they build on each other are graphically represented in the following diagram.



DevOps Benefits

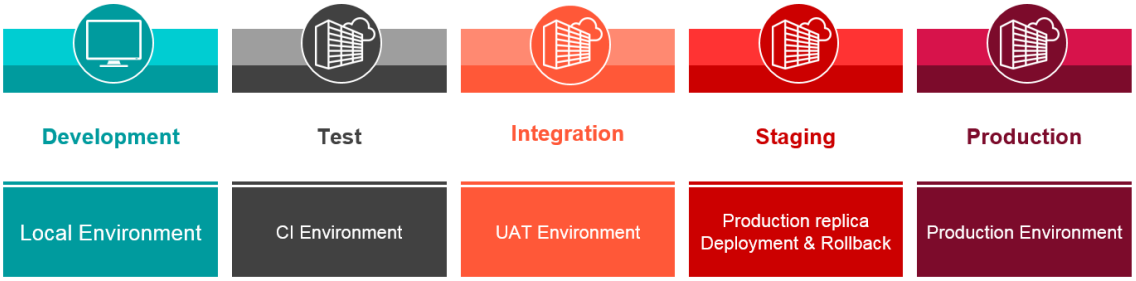
Before the usage of DevOps practices, software development and operations were composed of separate teams which were working in relative siloes, making the end-to-end process slower and very inefficient.

DevOps's practices try to address these challenges by establishing a collaborative and cross-functional ecosystem. Its main objective is to promote and improve the collaboration between all development and operations to jointly carry out the different stages of a software development lifecycle. Going from early stages as planning through delivery and automation of the delivery process to be able to:

- Improve quality of deployment procedures
- Increase the frequency and quality of deployments
- Decrease the amount and severity of failures with software releases
- Improve resolution procedures and mean time to recovery
- Improve solution quality
- Realize a faster time to market

5.4 Project Environments

Adopting the DevOps development and operations lifecycle for your project requires several project environments. At the beginning of a project it is necessary to get clarity on which environments are needed, and how they are implemented and hosted. There should always be at least isolated development environments for each developer, and a test environment with a continuous integration server running tests. The integration environment is required if the project requires user acceptance testing. The staging environment is required if the project team must provide a space for long-running acceptance testing and/or content-promotion and deployment support.



Development Environment

This is a shared-nothing environment local to each developer. Each developer has their own instances of databases, Hadoop clusters, BI servers etc. This is often achieved by running local instances of databases, or VMs for more complex setups, like a single-node Hadoop cluster in a VM.

Test Environment

This is an isolated environment where the continuous integration server runs the test suite. Again, the test environment has its own set of databases, Hadoop clusters, etc. The test environment is isolated, so the test suite can restore a 'clean-slate' state and load known data fixtures without interfering with anybody's work. It is crucial that building

this environment is a repeatable automated process, so the test server can be easily moved, or easily expanded to multiple worker machines using consistent configuration.

Integration Environment

This is an isolated environment where successful builds are automatically deployed to on a regular basis. This is the best place to experience the 'current state' of the solution. Its main purpose is to make sure that deployment works as expected and enable user acceptance testing. Like with the test environment, it is crucial to be able to replicate the integration environment from scratch using some variant of provisioning technology.

Staging Environment

This is an isolated environment that closely mimics the characteristics of the production environment. It is regularly provisioned with a dump of the state of the production environment. The staging environment's purpose is to closely simulate the deployment and rollback procedure. It is also sometimes used for longer user acceptance tests, as it tends to be stable for longer periods of time. If that use case is the dominant one this environment is often referred to as the QA Environment. Again, it is beneficial to be able to restore a staging environment from scratch.

Production Environment

This is the environment the solution runs on productively. Access to this area is usually heavily regulated. The developers of a solution are usually not accessing this area directly. Some knowledge about this environment is necessary though: hardware specifications (CPU, HD I/O throughput, RAM), network characteristics, availability of an internet connection as well as any exiting or enforced configuration options do inform architecture decisions in solution design.

Environment Configuration

When developing a project that will be running in various environments, it is not only important to separate the project code and configuration, but also to stick to a consistent solution for configuration management. Every environment that your project will be running on will have a dedicated configuration setup and it is recommended to keep this environment configuration in a version control system (checked in to environments/test, environments/staging for example.). The local configuration for developers is not checked in using the "ignore" feature of the version control tool, so every developer can have a differing one. It is helpful however to keep a default local configuration checked in so developers joining the team can create a copy of that for their local environment and get started quickly.

Switching between configurations can then be as easy as setting an environment variable pointing to the folder containing the desired environment configuration. If that environment variable is not set, the location of the local environment is assumed.

5.5 Version Control

Enterprise projects have multiple developers collaborating on a project. Collaboration requires a central repository of artifacts hosted in a version control system. Git offers good flexibility for version control as well as flexible hosting and branching strategies.

The Pro git book is a good source of information/training on how to effectively use git and is freely available in multiple languages: https://git-scm.com/book/en/v2

SourceTree is a good free multi-platform UI client for git:
https://www.atlassian.com/software/sourcetree/overview

Branching Strategy

A basic version control branching and merging strategy should be agreed on in the project. Branching enables implementing a new feature in an isolated area that allows you to prove that it works before it gets merged into the main code base. Branching also allows more than one developer to implement features in parallel without creating code conflicts and allows you to separate production ready code from untested code.

A simple initial approach is to just use feature branches for new stories, and merge into a common development branch once done. Once the project team is expected to support the solution after it goes into production, a more sophisticated approach is needed to coordinate hot-fixes, bug fixes in patch releases, as well as unreleased features in development.

There are many branching strategies out there. Take a look at the strategy known as GitFlow (https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow) and tailor it to the project at hand, simplifying based on the actual requirements of the project.

A tutorial on multiple git workflows and branching strategies can be found at https://www.atlassian.com/git/tutorials/comparing-workflows

5.6 Testing Strategy

The DevOps section already introduced Continuous Integration as a software development lifecycle that has great emphasis on test automation to check that the application is not broken whenever new iterations are integrated into the central solution repository. Effectively running automated tests on your solution increment requires the project to adopt a testing strategy that meets the project requirements. The testing strategy defines the environments and framework needed to support your tests and the way these will be set up and configured. The strategy also defines the test process and the level of testing that will be done to meet acceptance criteria.

Testing Infrastructure

Testing should be automated as much as possible. To that end, there should be a test environment in place, running a continuous integration server. Currently Jenkins is the most popular open source continuous integration server. https://jenkins-ci.org/

Preparing a test environment is usually a non-trivial task, as test databases, test files, as well as Pentaho software itself would often need to be part of it together with the integration server. Even more complex infrastructure like a Hadoop cluster is often required. Therefore, it is important to stick to an automated solution for building and deploying a test environment. If the test environment needs to change, that change is documented in version control. If any experimental changes or misbehaving tests cause the test environment to become corrupted, the test environment is easily restored. At the same time the automated scripts building the test environments provide a complete and unambiguous list of requirements and prerequisites for the project to run and successfully pass the test suite, which is very valuable when moving to production.

Testing Automation Framework

Every project should use a testing automation framework that is comprised of a combination of practices and tools that are designed to help developers and QA professionals test more efficiently. The framework should provide a set of guidelines, rules and tools to guarantee coding happens according to standards, to support the team to write test cases in a consistent way, to support test-related activities like loading external resources, and to allow for re-use of other commonly used helper functionality. The test framework should also allow running parts of the test suite selectively: individual tests, or a group of tests.

In general, you need a testing automation framework to help make your test automation code more reusable, maintainable and stable.

Tests should be easy to write for all developers on the team. So simple scripting languages or Domain Specific Languages are typically preferable to frameworks in more sophisticated languages.

It is important that at least one team member is intimately familiar with the test framework. Writing and structuring tests efficiently can be a complex task, and it is important to follow best practices right from the start.

Java is always available on Pentaho project environments, so any test framework in a language that runs on the JVM is an option.

As an alternative approach it often makes sense to have ETL test other ETL. If for example a sub-transformation is expected to modify certain fields in a stream, it is relatively easy to embed it into a test transformation that verifies the output is as expected using the merge diff PDI step, and have the transformation fail if it is not. These kinds of test ETL are best included into the test suite by having the test just run the test ETL and verifying the exit code.

Test Suite Design

A good test suite consists of a set of shared data fixtures re-used by multiple tests, some helpers to load the data fixtures as well as clearing all test-fixtures and data to establish a known clean slate state.

In data projects a good test is structured around the following steps:

- Reset the solution database
- Load the data fixture appropriate for the test
- Run the portion of the solution to be tested (usually some ETL or solution command)
- Verify the expected state by many small assertions, each verifying a single aspect of the expected outcome.

Any code that is repeatedly needed for tests should be factored into easy-to-use helpers. For example, there should be helpers to reset the database, load a fixture and run ETL, so the test remains easy to read, and is not obscured by low level calls to database client tools or shells.

Levels of Testing

A data project can be tested at different levels. A transformation may be tested individually, or an entire DWH load may be tested by running it on a known set of source data and running a SQL or MDX query on the resulting reporting schemas. Other tests may be aiming at verifying agreed acceptance criteria, or performance considerations.

5.6.1.1 Unit Testing

In traditional software development the most natural way to test code is unit tests. Small pieces of code like classes or functions are tested in small scenarios. In data projects whose logic is mostly embedded in ETL processes, unit tests are usually not very effective. Testing individual ETL jobs and transformations is maintenance heavy: fixture data sets need to be prepared, and each feature and side effect of an ETL job must be verified. Since it is the nature of ETL to try and accomplish a lot of processing in a compressed space, testing for everything a piece of ETL does is cumbersome and error prone, and the effort needed to do it properly may very well outweigh the benefits.

Unit testing does make sense on utility ETL performing specific isolated tasks and re-usable sub-transformations. For most of ETL logic integration or feature testing is a better choice.

5.6.1.2 Integration Testing

In integration testing the aim is to verify that a logical subset of ETL works together as intended. Any complex logical step in an ETL process like, loading data into staging tables, building a derived business entity in a DWH, or building a star schema can be the subject of an integration test.

An integration test usually brings the data fixture into place, then runs the appropriate subset of ETL, and verifies many aspects of the outcome: the ETL must succeed, it must produce certain output data, it may have to produce some log files with expected content etc. This is the most appropriate approach for testing most ETL logic in data projects. Its main challenge is that a consistent realistic fixture of source data must be maintained to make the ETL process operate in a way that resembles production operation as closely as possible.

Integration tests should also be written around expected behavior during failure conditions. How does the ETL behave if files are missing or corrupted, or if databases are down or contain unexpected data, or an ETL job is run with unexpected parameters? Integration tests are written to verify sensible behavior, which is solution specific (retries, reconnects, log warnings, halts etc.)

5.6.1.3 Feature Testing

Often a data project will have a set of features or properties listed as part of the project's requirements list. When writing tests to cover those, usually they are integration tests in nature, these tests are often called feature tests, and they are useful in the acceptance phase of the project. These are often the tests the customer can most directly relate to, because they are phrased in terms agreed on during scope and requirement management discussions.

5.6.1.4 Regression Testing

Sometimes a bug is discovered that is worth keeping an eye on. Writing a test that specifically guards against a known bug reappearing is called regression testing. Regular integration and feature testing are usually a good guard already, but if the nature of the bug makes it hard to test in regular tests or depends on conditions that are hard to reproduce in regular tests, a regression test is a good way of making sure that the bug can't reappear unnoticed.

5.6.1.5 Stress Testing

Under which conditions does the solution break? This is an interesting aspect of every data project. Every solution breaks eventually, under the right circumstances. And it may break in many ways. Stress tests are designed to find out what the limits are, and make sure the solution does not regress beyond acceptable limits.

Stress tests establish unusual preconditions or unexpected situations before running parts of the solution, and then verify the solution behaves reasonably.

Stress tests are usually not scheduled to run as part of the regular test suite. They are often scheduled to run once overnight instead. As they tend to take more time to complete.

5.6.1.5.1 Examples for possible stress tests:

What happens if the main ETL job is executed concurrently? Test may try to start 5 of them as quickly as possible and verify that 4 of them fail gracefully immediately. That procedure would be repeated a couple of times. Any spurious failures in this test indicate some sort of race condition that needs to be taken care of.

What is the minimum amount of memory Pentaho Data Integration needs to complete a typical run? Start the solution with a conservative minimum like 1GB and see if and when it starts stalling. Any jumps in memory requirements will be identified and can be examined if this test is done.

If an ETL transformation is reading BLOB fields from a database, how big do they have to get for PDI to go out of memory?

5.6.1.6 Performance Testing

Performance is often an important aspect of data projects. It is worth running a realistic workload for the solution on a regular basis and recording the time it took to complete (the DI server takes care of that). If at any time the performance starts to deteriorate, it is often easier to find the underlying cause by looking at the commits around that time.

Performance tests are usually also not scheduled to run as part of the regular test suite. They are often scheduled to run once overnight instead, together with the stress tests.

5.7 Evolution of DB-Schemas

One aspect of data projects is that they generally create and evolve some sort of DB schema, often multiple DB schemas. These schemas are often hard to evolve and hard to keep in sync across environments.

There are several approaches to dealing with this issue. The simplest “solution” is to try and not evolve the schemas, but always create them from scratch, or as part of the ETL itself. After all the ETL could create some tables on the go, like staging tables for example.

As long as the solution is not in production, and the ETL user has sufficient permissions to create tables, this approach works. The developers simply maintain a set of SQL files that re-create the schema in its entirety from scratch when needed. The developers create a clean slate on their working environments using those files. And the test environment is using them as well before running tests.

This simple approach should be followed for as long as possible, but eventually there comes a time where the DB schema is used in production, and changes need to be done when deploying. In this case, it is still recommended to always re-create the entire schema on developer machines and test environments. However, during deployment to the staging environment the incremental approach is used. Either running hand crafted SQL or using a specialized tool to help with the task.

There are a couple of tools that allow managing incremental changes to databases. One good example being http://flywaydb.org/

Whether a tool is used, or manual scripts are run, the migration files from one release to another must be part of version control. And the final test is provisioning the staging area with the production DB, successfully deploying to staging changing the schema in the process and running the test suite on staging successfully.

5.8 Solution Documentation

Every data project needs documentation. It is preferable that the solution is self-documenting, by having help screens, or help pages that are created as part of the regular Scrum sprint cycle. The presence of those is also testable to some extent, so if help pages are expected to be in place, automated testing can be used to verify they are there.

Ideally, if the help pages are created using a convertible format, generating a full set of reference documentation can also be automated as part of CI, with the resulting document being a static webpage or a PDF document.

The technology chosen to write documentation should allow the team to version it as part of regular version control, work on it concurrently and the content should preferably be readable in a browser.

5.9 Pentaho Best Practices

Pentaho technical best practices gathered from hundreds of implementations and customer use cases offer the shortest path to success and ongoing value for our customers by providing guidance and support throughout the project lifecycle. From helping you being proactive about your non-functional solution requirements during the Discover phase, to accelerating your project setup and guiding your development, testing and deployment efforts during the Design and Execute phase.

As a result, the Pentaho best practices allow your final solution to not only deliver business value by fulfilling the stakeholder requirements, but doing so with a robust solution that is up to industry standards, that is performant and providing the operational insights needed to monitor and support the solution after go-live.

The Pentaho field-proven best practices library allows you to tackle common project challenges, including:

Project Setup and Lifecycle Management

- Challenges
  - Multiple developers will be collaborating on the solution, requiring development standards and a central repository of artifacts
  - The solution needs to integrate with the Customer's Version Control System (VCS)
  - The solution needs to be environment agnostic, requiring the separation of content and configuration
  - The quality of the solution needs to be assured before it hits production requiring different tests to be performed
- Best Practices
  - Lifecycle Management
  - Version Control System integration (Git)
  - Configuration management: separating content from configuration (managing configuration externally)
  - Naming standards
  - Logging and monitoring
  - Testing strategies
  - Creating a testable ETL solution

Solution Architecture

- Challenges
  - Solution needs to integrate with the customer's existing IT landscape
  - Solution hardware needs to be sized appropriately allowing for a performant solution
  - Solution needs to run in the cloud
  - Data model needs to support the reporting use case
  - Redundant data needs to be archived
  - Reporting performance needs to be optimal in order to promote usage of the solution
  - Solution needs to be embedded in a different application
  - Solution needs to restrict in terms of operations permissions and data access by user profile
- Best Practices

- Data modeling
- Sizing guide
- Mondrian cache management
- Pentaho and Amazon Web Services
- Content embedding guidelines
- Different security providers
- Data security
- Pentaho operation permissions

Platform Installation, Configuration and Management

- Challenges
  - Pentaho software needs to be correctly installed
  - Solution needs to be available at all times
  - Solution needs to be upgraded to the latest Pentaho version available
- Best Practices
  - Tomcat Memory settings
  - Pentaho Server High-Availability
  - Pentaho Software Upgrades

Development Guidelines

- Challenges
  - Solution needs to be developed for maximum performance
  - Solution needs to make use of reusable components to reduce development effort and maximize your solution maintainability
  - Final solution will be supported by a different team, so logging and monitoring should be put in place that allows for a supported solution
  - Errors can and will happen and need to be handled
  - Solution needs to be as self-healing as possible
  - Failing ETL should require a simple restart
  - The way numbers are presented on the screen affects the way they are consumed
- Best Practices
  - Metadata Injection

- ETL Error Handling
- ETL development guidelines
- ETL performance tuning
- UX guidelines
- OLAP design guidelines
- Modeling guidelines

6 Support Documentation

6.1 General Documentation

Customer Record - Contains information related with the customer:

- Business Core
- Business Model
- Stakeholders
- Plus and minus about customer environment

Project Document

Project Requirements Document

- Contains all details about the project requirements
- Contains references to the user stories per requirement

Project Plan

- Roles allocation to project
- Time planning
- Project Deviations

Solution Design

- Architecture scheme
- Model Design
- Use Cases diagrams
- Mockups (if CTools)

6.2 Pentaho Services

JUMPSTART YOUR SUCCESS

Pentaho Consulting Services offer pre-packaged programs and custom engagements that help you develop, design and implement your analytics projects quickly. Get your project up and running quickly with services, training and best practices that ensure a seamless implementation.

ACCELERATE YOUR TIME TO ROI

- Prototyping: Bring your data integration and analytics strategy to life with a custom-built prototype tailored to your requirements, using your data.
- Deployment Assurance Program: Eliminate project false starts with experts that collaborate with you to deploy best practice-based solutions.
- Training and Education: Develop your own Pentaho experts with our wide range of classroom, online and on-site training.
- Architecture Workshop for Embedding Analytics: Learn the best techniques for embedding Pentaho's platform in your application or service.

MINIMIZE RISKS AND ROADBLOCKS

- Solution Delivery: Outsource the implementation of your project to our reliable and skilled data integration and business analytics consultants.
- Migration and Upgrade Support Packages: Easily migrate from our community edition or a legacy BI solution, or upgrade to our latest release.
- Staff Augmentation: Leverage our team of skilled resources to supplement your project team.
- Engineering Services: Let Pentaho experts develop customized product features that easily plug into your business analytics solution.

MAXIMIZE YOUR INVESTMENT

- Health Checks: Periodically review your business analytics strategy and deployment to ensure optimal design and use of Pentaho functionality.
- Technical Account Manager: Get regular, on-going expert advice, guidance and problem resolution throughout your entire project lifecycle.
- Upgrade Services: Quickly deploy and operationalize new releases of Pentaho.