IT Project Guidance

Deliverables to Consider

## Description

Guidance on key and supporting IT deliverables expected by stakeholders.

## Synopsis

Projects with an IT component are unlikely to meet expectations without a clear idea of what is expected to be delivered.

This document lists deliverables to consider, along with a supporting information as to why they should be considered.

Contents

[Description 1](#_Toc145233623)

[Synopsis 1](#_Toc145233624)

[Introduction 4](#_Toc145233625)

[Background 4](#_Toc145233626)

[Problem 4](#_Toc145233627)

[Objective 5](#_Toc145233628)

[Scope 5](#_Toc145233629)

[Applicability 6](#_Toc145233630)

[Procurement & System Types 6](#_Toc145233631)

[Legacy Approaches to System Development 7](#_Toc145233632)

[Responsibility of Delivery 7](#_Toc145233633)

[Delivery Type 7](#_Toc145233634)

[Completeness 8](#_Toc145233635)

[Relationship to Requirements 8](#_Toc145233636)

[Deliverables 8](#_Toc145233637)

[Overview 8](#_Toc145233638)

[Types 8](#_Toc145233639)

[Overview - Sequence 9](#_Toc145233640)

[Governance Reports 9](#_Toc145233641)

[Context 9](#_Toc145233642)

[Direction & Approach 11](#_Toc145233643)

[State & Delivery Progress Reporting 11](#_Toc145233644)

[Operations Reporting 12](#_Toc145233645)

[Services 13](#_Toc145233646)

[Pipeline Service 14](#_Toc145233647)

[Information Service 14](#_Toc145233648)

[Service Environments 15](#_Toc145233649)

[Environment Types 15](#_Toc145233650)

[Environment Components 16](#_Toc145233651)

[Environment Configuration 17](#_Toc145233652)

[Discoverability 17](#_Toc145233653)

[Domain Name 17](#_Toc145233654)

[DNS Records 17](#_Toc145233655)

[Email Configuration 17](#_Toc145233656)

[Service Integration 18](#_Toc145233657)

[Service Discoverability and User Support Integration 18](#_Toc145233658)

[Public Facing Organisation Website 18](#_Toc145233659)

[Private Corporate Website 18](#_Toc145233660)

[Other (Optional) Websites 19](#_Toc145233661)

[Service Dependency Integration 19](#_Toc145233662)

[Service Text & Media 20](#_Toc145233663)

[Organisation Website Text & Media 21](#_Toc145233664)

[Service Terms & Conditions and Disclosures 21](#_Toc145233665)

[Information Service Copy & Media 22](#_Toc145233666)

[Service Copy Translations 22](#_Toc145233667)

[Search Engine Optimisation 22](#_Toc145233668)

[Service Data 23](#_Toc145233669)

[New System Data & Users 23](#_Toc145233670)

[Migrating Legacy System Data 23](#_Toc145233671)

[Migrating Legacy System Users 24](#_Toc145233672)

[Service Acceptance Criteria 25](#_Toc145233673)

[Service Documentation 26](#_Toc145233674)

[Service 26](#_Toc145233675)

[Default Case: Information Services 26](#_Toc145233676)

[Deliverables based on System Type 26](#_Toc145233677)

[Sequence of Deliverables 26](#_Toc145233678)

[Deliverable Components 26](#_Toc145233679)

[Logical Systems 27](#_Toc145233680)

[Documentation 27](#_Toc145233681)

[General Condition 29](#_Toc145233682)

[Variations 29](#_Toc145233683)

[CAB Approval of Deliverables & Go-Live RFC 29](#_Toc145233684)

[Appendices 30](#_Toc145233685)

[Appendix A - Document 30](#_Toc145233686)

[Tables 30](#_Toc145233687)

[Figures 30](#_Toc145233688)

[References 30](#_Toc145233689)

[[1] 30](#_Toc145233690)

[OWASP Foundation, “Home Page,” [Online]. Available: https://owasp.org. 30](#_Toc145233691)

[Review Distribution 30](#_Toc145233692)

[Audience 31](#_Toc145233693)

[Structure Conventions 31](#_Toc145233694)

[Diagram Conventions 31](#_Toc145233695)

[Terms 31](#_Toc145233696)

# Introduction

IT Project Delivery Management is a specialised subtype of general Project Management, with a distinct set of expected deliverables.

## Background

Market scarcity leads to a relatively common condition where IT projects are being delivered via generalist project managers with limited prior exposure to successful IT delivery processes.

## Problem

Delivery Managers are reliant on a reactive, ad hoc, process of incomplete discovery of expected deliverables – rather than having a clear understanding of objectives, expected deliverables and planned processes to deliver them. This leads to unnecessary remedial costs, while significantly increasing the risk of the project failing to deliver some or all the expected deliverables within budget and/or schedule.

# Objective

The purpose of this document is to provide managers a list of the various deliverables that will be required by the end of the project, along with a high-level description of its purpose and importance – hence why it is considered a deliverable.

# Scope

The scope of this document includes describing the types of Deliverables to consider delivering.

It is not within this document’s scope to describe how to deliver them according to ISO recommended delivery methodologies.

# Applicability

## Procurement & System Types

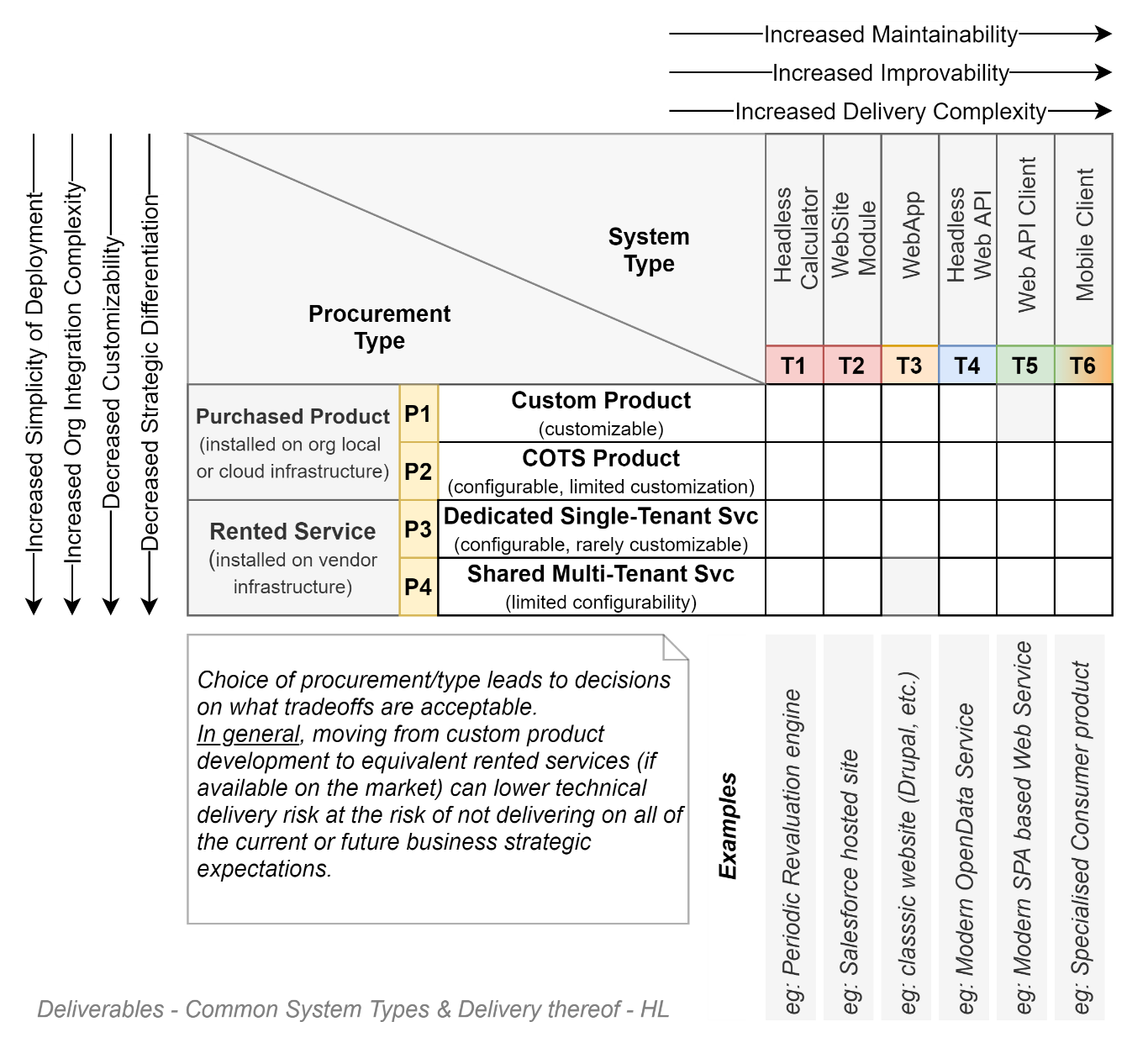


Figure 1: Common System Types and their Delivery

Rather than trying to cover all types of IT projects, this document lists the deliverables for the most common types commissioned within organisations.

The final system type is – at a high level -- determined by the organisation’s procurement preferences (buy versus rent), hosting preferences (on-prem versus cloud), technological capability (low to medium) – as per the diagram above.

The less capability an organisation has, the more it sees customisation a risk, preferring to stick to buying mature custom products, even if the product is strategically poorly aligned[[1]](#footnote-1). The more digitally mature an organisation, the more it can take advantage of customization abilities to deliver outcomes that are better strategic fits.

### Legacy Approaches to System Development

A category that is almost strictly limited to large organisations with a long legacy of IT involvement, but with a medium to low level of capability is the requirement for headless “patch” solutions (Type 1 in the diagram above). They often come in the form of ETL jobs, chron triggered scripts to move data extracts (CSVs) & files around. Wherever possible legacy systems of this type should be converted to type IV systems, potentially with a Type V interface as needed. No new non-trivial project should use this approach as it adds non-negligeable risk to an organisation. Some of these risks are listed below.

* Use single thread, non-OO legacy languages increasing development costs,
* Are rarely if ever developed with associated Test Harnesses to QA the logic.
* Require atypical infrastructure that can be expensive to run compared to current lighter runtime cloud-based environments,
* Are difficult to test due to being headless, with no User Interface,
* Lead to non-trivial security process errors due to using data to test, rather than relying on Unit Tests to test the logic,
* Are costly to integrate due to being developed in architectures that are either have no API interface, or if they do are not Web based APIs in alignment with 99.99% of the rest of current development effort in this organisation and the world in general.
* Are largely unsupportable,
* Are difficult to maintain without extensive logging, therefore downtimes can be significant.

## Responsibility of Delivery

Depending on the procurement process used, the responsibility of delivering the listed Deliverables is balanced differently between suppliers and this organisation.

Custom and COTS product packages will lead to more responsibility lying with the project, as opposed to Rented SaaS services, which will handle more of the tasks and development of the deliverables – but also allow for significantly less customisation.

## Delivery Type

Not all deliverables are visible to Governance or User Stakeholders, while remaining required by delivery Stakeholders. They are referred to here as Task deliverables.

Some Deliverables are required to improve Reporting and Governance but are not part of a System’s Certification and Accreditation process. These deliverables are referred to as Unverified Deliverables.

Some Deliverables will be required to have been sighted for Certification to be issued to a Product or Service. These deliverables are referred to as Validated deliverables and must be delivered by the project by its first release to production and thereafter kept up to date.

## Completeness

Not all system types – notably non-web based simple calculator modules -- will require all deliverables.

The appendices contain examples of the list of Deliverables to consider for some of the above Procurement/System type rubric.

## Relationship to Requirements

In mature organisations, when working on well managed projects to deliver well-understood and defined systems, the Deliverables specified within this document would already be defined as Delivery Work Items, viewable in the systems contracted Requirements.

This is rarely the case, for reasons covered in a related document (“Project Delivery Risks to Consider & Mitigate”).

This document is intended to mitigate that risk by providing Managers an ability to better communicate with delivery stakeholders and determine whether these deliverables are already on the Work Item register or need adding to ensure they are undertaken and completed.

# Deliverables

## Types

At a high level, the deliverables expected from an IT Project can be summarised as fitting within the following categories:

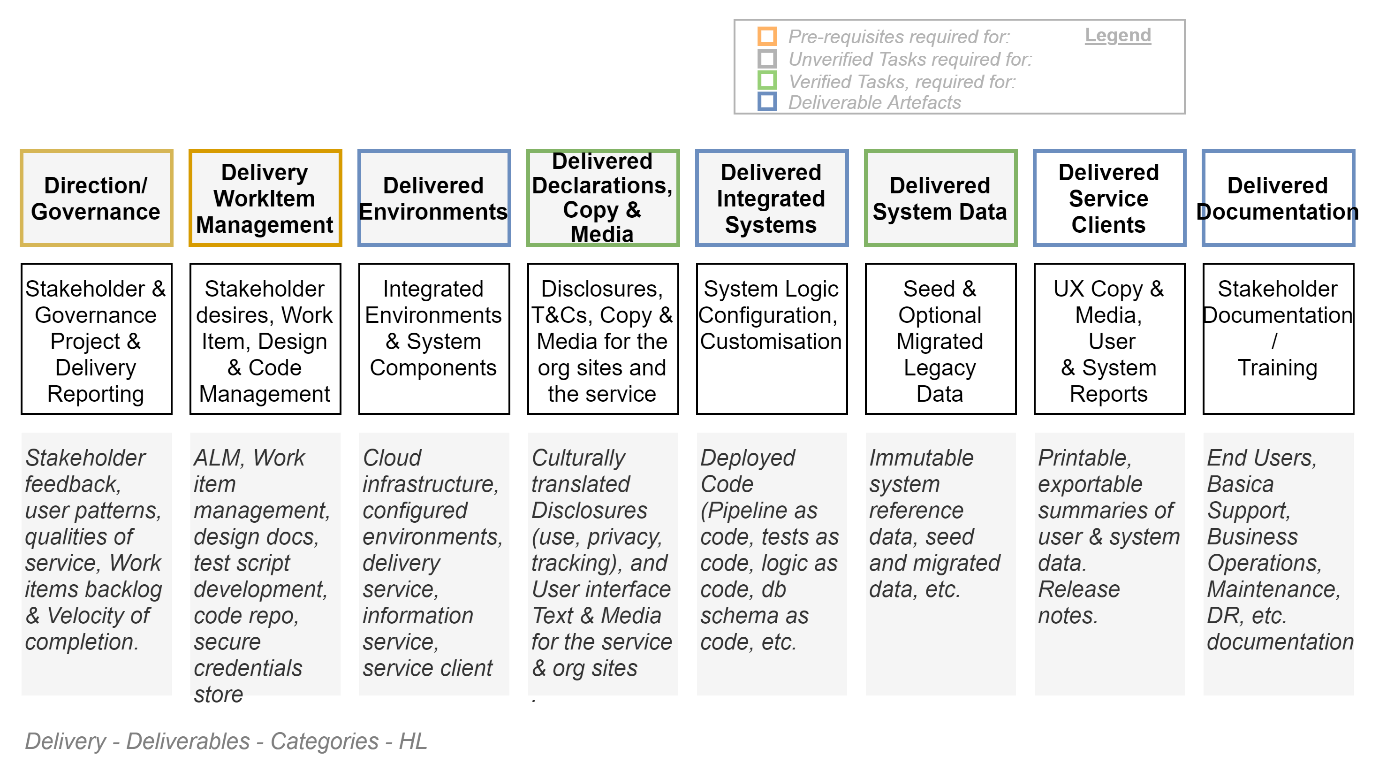


Figure 2: Deliverable Categories - HL

Each of the above categories is expanded on below.

## Sequence

Due to the general complexity of Service design, and the inter-dependency of deliverables, the order in which they described is arbitrary. The order chosen in this document is approximately as the diagram shown above, starting with expected Governance reporting artefacts and concluding with documentation.

## Governance Reporting

and delivery management reports managing the ensuing Decisions is an ongoing/living project deliverable.

### Context

Context Aspects to consider summarising to Governance boards include the following:

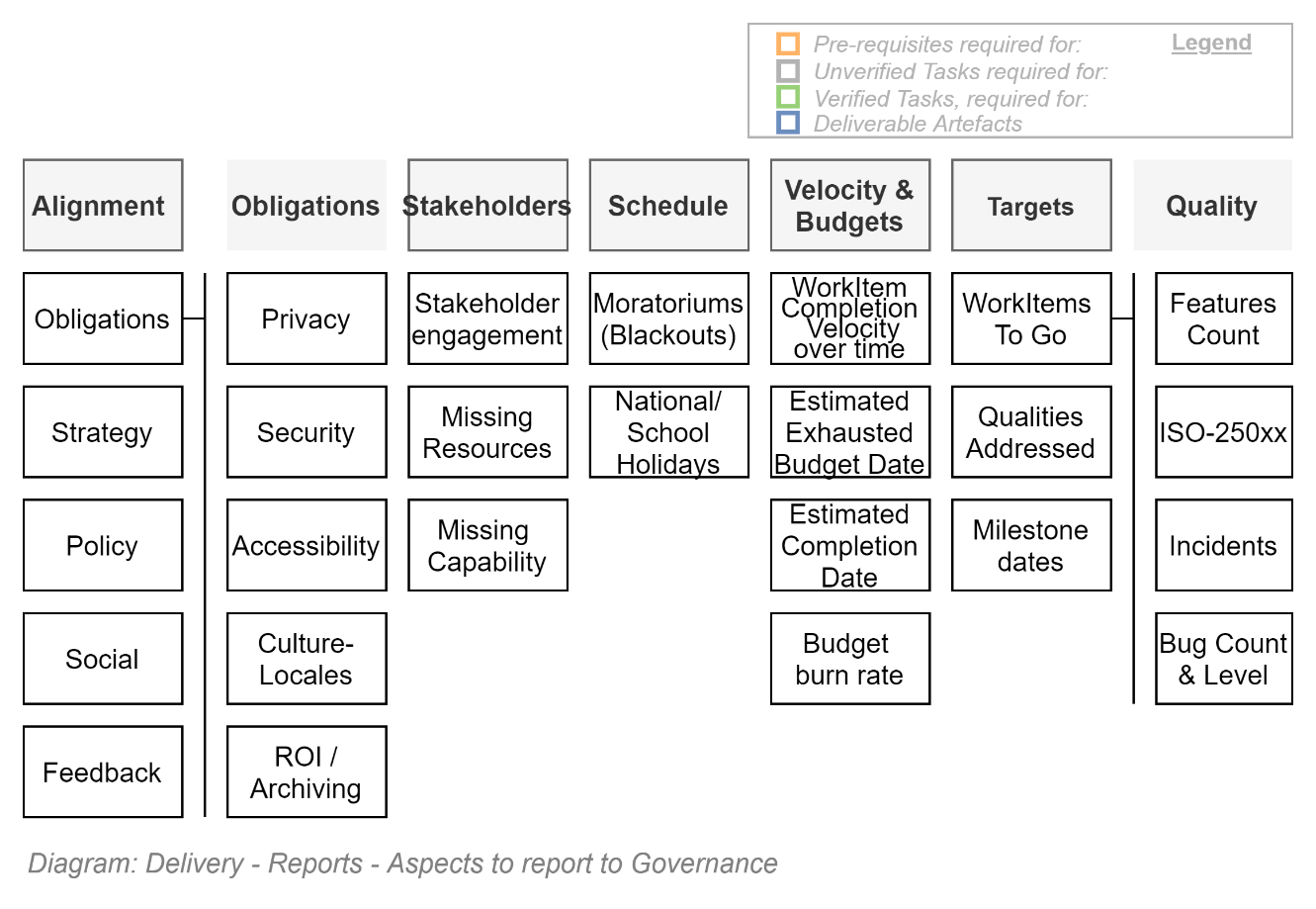


Figure 3: Governance Aspects

Note that the above rubric ensures the following is communicated:

* **Context:** ensuring all still concur the project remains aligned to larger strategies & other projects,
* **Expectations:** ensures all understand both the Functional and Quality Outcomes Desired by business & other stakeholders,
* **Constraints:**
  + **Guidance Framework:** ensuring stakeholders reuse past learnings & best practices,
  + **Decisions** that have been made to go against the frameworks, maybe to meet
  + **Obligations** (Laws, Regulations, Agreements & Policies) that must be met either way,
* **Resources** available to meet the above Expectations and Obligations (Time & Schedules, Capital and Capabilities)
* **Risks** in the form of missing resources, capabilities,
* **Metrics** in the form of Service Quality and Usage data, including user Feedback.
* **Issues:** Issue resolution Effort & Progress reports

Traditional resources developed to communicate the above include the following:

* Governance
* Progress
* Operations

### Direction & Approach

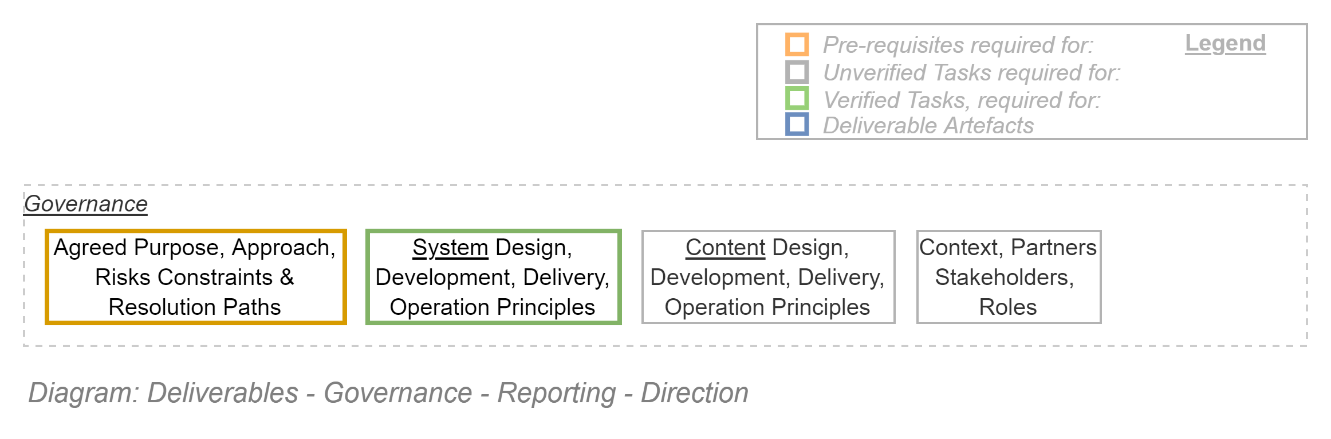


Figure 4: Governance Reporting

At the start of projects, it is important to establish guidelines that allow remove friction and allow autonomy of stakeholders to follow *agreed* best practices.

The agreements to obtain and encode include:

* An agreement on the purpose of the service
* An agreement on who the service is for,
* An agreement on the quality to which the service has to adhere,
* An agreement on our obligations that need to be met along with the desired features,
* An agreement on Guiding Principles for defining architecture,
* An agreement on how to resolve differences of opinion as to how best to follow Guiding Principles, and come to a decision,
* An agreement on processes to develop and subsequently maintain:
  + The delivery pipeline that runs everything,
  + The system being delivered to end users
  + The content that needs development and periodic refreshment to remain current.

A wiki or other form of electronic documentation is recommended to be developed at the beginning of projects, such that:

* Stakeholder buy-in is obtained to ensure agreement and continuing engagement,
* Delivery stakeholders have unrestricted access to the information on which to base their autonomous delivery effort,

### State & Delivery Progress Reporting

Many of the categories that require reporting no are constantly in flux and changing states.

They are best summarised using the following deliverables:

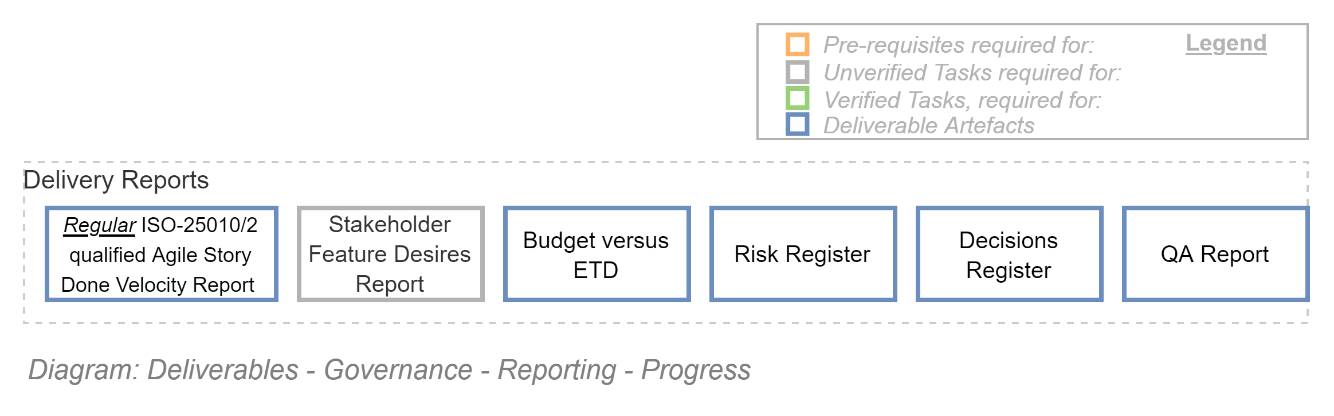


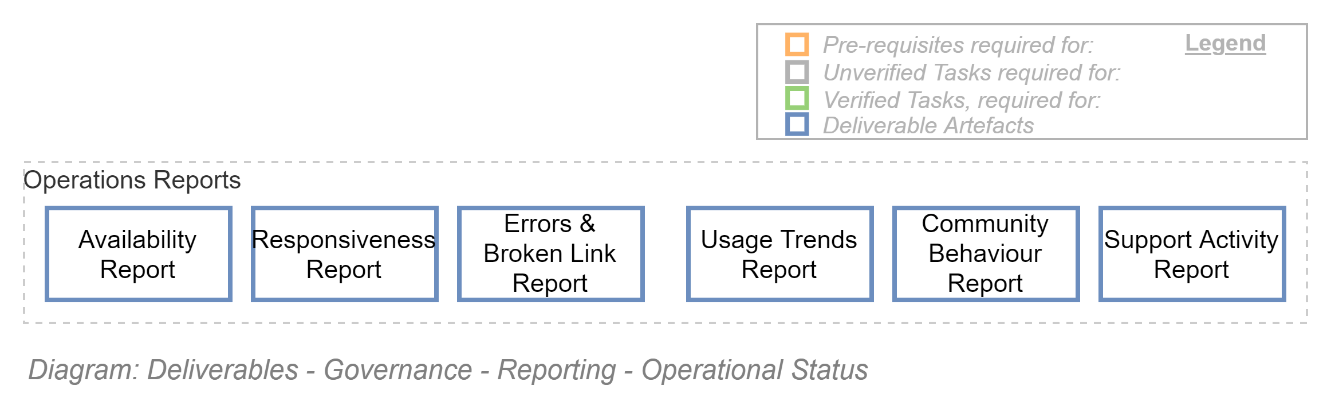
Figure 5: Governance Reporting - Progress

* Velocity of effort gives an indication as to how effectively the project team is able to work towards meet expectations. Velocity can be used to extrapolate an Estimated Time to Deliver (ETD).
* evolving stakeholder & user feedback should be considered to adjust the extrapolated ETD.
* Comparing the Budget to the ETD provides a means of estimating the quality of the final product, and whether additional budget is required,

***Tip:*** *Current best practice is to develop a live dashboard within the project’s Application Lifecycle Management (ALM) service suite which graphically summarises current remaining issues, work items, current velocity of progress through the work items. Mature ALM services make it easy to print the information for distribution,*

### Operations Reporting

When the system is deployed, the following deliverables summarise different perspectives of the service’s qualities and value:



* Availability speaks to QoS and whether service levels are being met
* Responsiveness speaks to how long people have to wait for service
* Errors & Broken Links speak to maintenance of the services resources
* Usage Trends speaks to whether the service is being used, falling out favour, worth further investment, etc.
* Community behaviour indicates whether the system is adding reputational risk to the organisation,
* Support activity, compared to usage trends, indicates how difficult the system is to use and whether it needs improving to release its value.

## Services

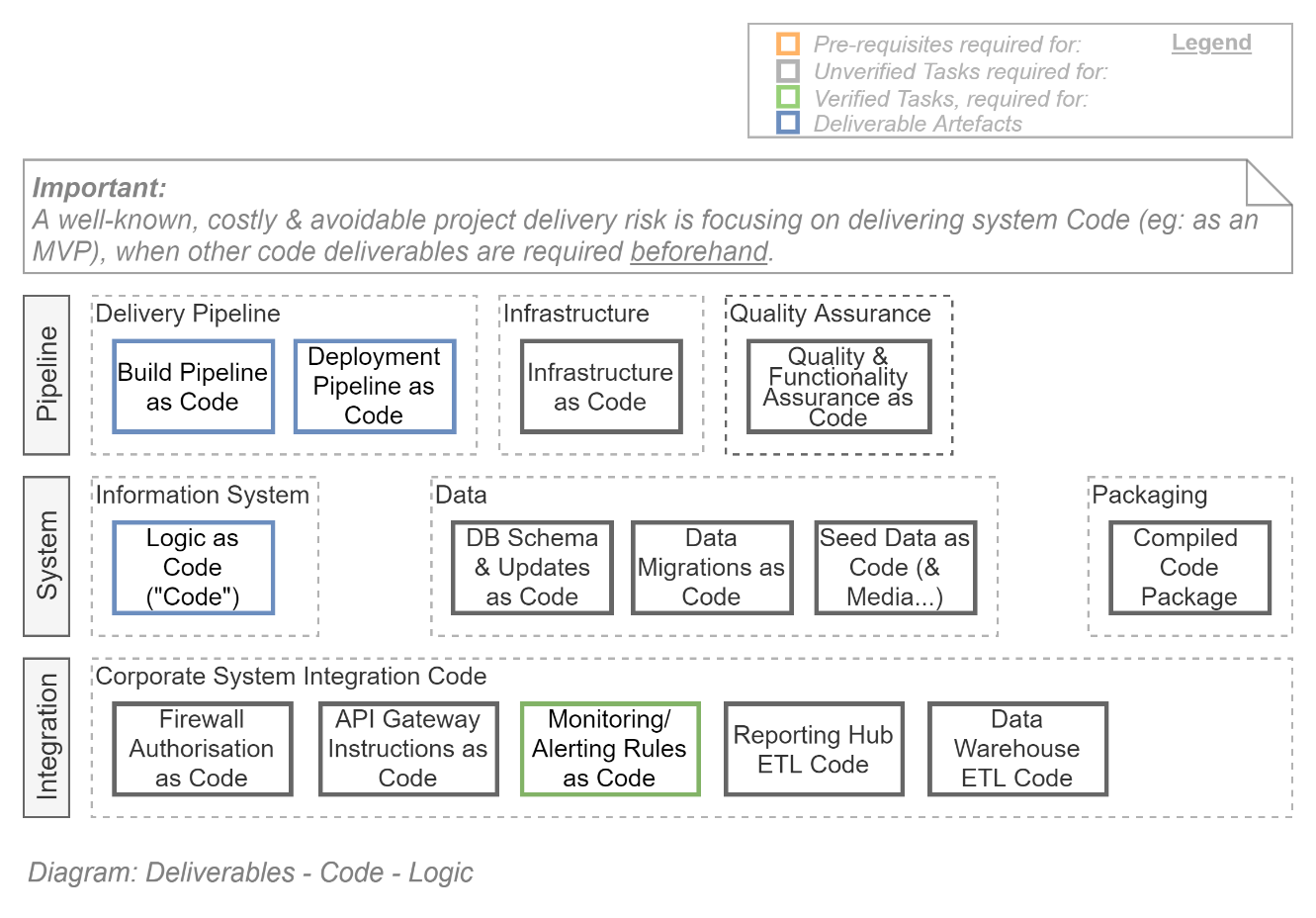


Figure 6: Code

Possibly the most impactful and prevalent IT delivery management error involves thinking there is only one system – the business information service -- to deliver, concentrating solely on its features and its intended go-live gate, working back to everything else as needed.

This is a critical delivery error.

Even if only one system were to be delivered, it isn’t the business service that needs to be delivered. Instead, it is the *pipeline* service that needs completion, so that it in turn (and not the delivery team) can deploy the business service whenever needed.

The above diagram emphasises that:

* there are several systems that need effort invested (pipeline, business service, other services), and
* the pipeline is the first system that needs effort applied.

*Tip:   
Instead of participating in a legacy cottage industry approach to produce a single* Product *by a specific date, participate in a modern industry approach of creating a current, automated robot* Service *first, which in turns manages the production and deploying of iteratively improving copies of the* Product*s.*

*We’re not in the business of developing products – we’re now in the business of developing business context specific deployment and QA testing robots.*

Delivering a single instance of a product, with no ability to rapidly maintain or even improve it, makes it a *liability*, not an *asset*.

### Pipeline Service

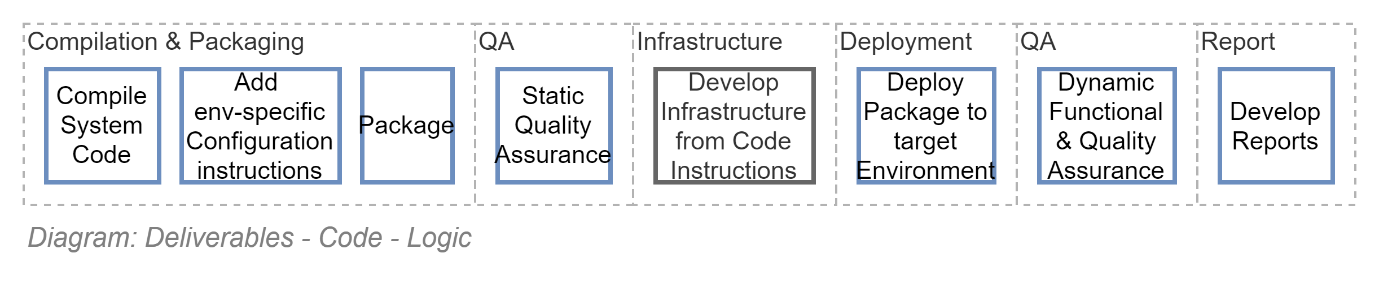


Figure 7: Deliverables - Code Logic

An automated pipeline service performs multiple duties, liberate other delivery stakeholders for other tasks, as shown above.

Although the pipeline must be started first, it is unrealistic to expect that stakeholders will have the patience to wait for its completion before seeing a first version of the product they are funding. Therefore, once the pipeline starts to work, one can start managing the development of the expected service. From then on, both services are worked on in tandem.

The investment of effort in a second system than the business expected system will not go to waste: the more complete the automation of the pipeline – the more QA it performs on the final product, etc. -- the less cost it takes to incrementally improve the information service over its service lifetime.

*Note:  
A common industry error is to think that Minimum Viable Product (MVP) implies to concentrate solely on the product, damn everything else. It doesn’t. MVP means that the minimum deliverables to successfully iterate and improve the system include a minimal pipeline being developed first.*

The Pipeline is not the only deliverable that should be begun before concentrating on the service itself. The rest of the deliverables listed below emphasis the getting started on other parallel prerequisites first: environment, data, media, automated QA.

### Information Service

Notice in this document’s index that the Service itself is listed near the bottom of this document’s list of deliverables as it can only be delivered after several other deliverables are addressed first, ni the following sections:

* Service Environments
* Service Integration
* Service Text & Media
* Service Data

## Service Environments

Environments are a combination of Components and Component Configuration, developed using Infrastructure as Code instructions.



Figure 8: Deliverables - Environments

### Environment Types

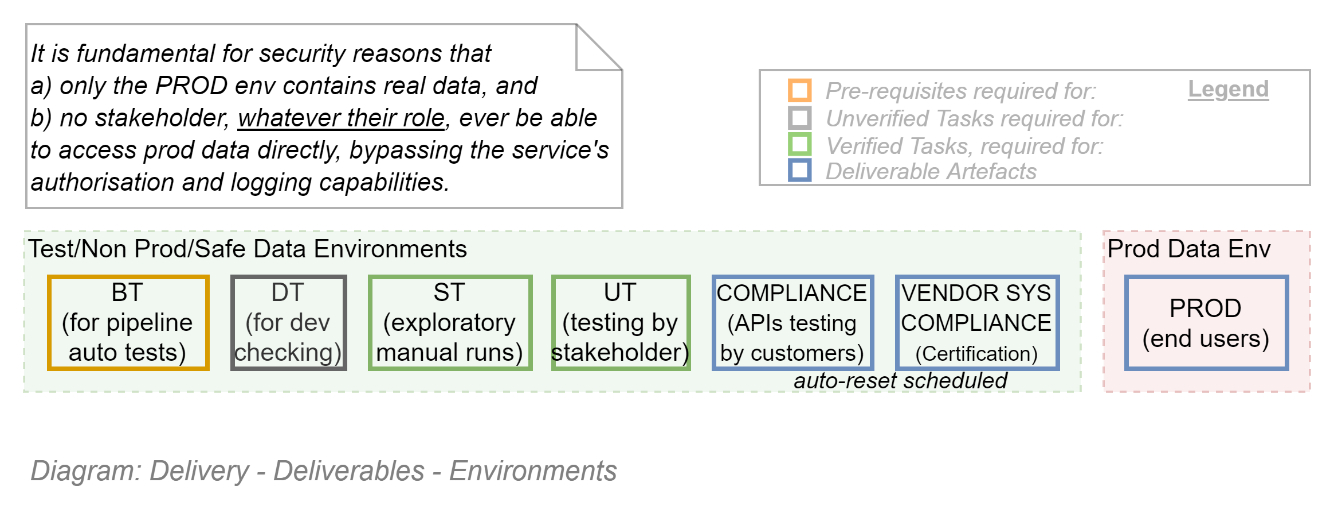


Figure 9: Deliverables - Environments - Types

Services are run on virtual or physical components, running within distinct environments. New logic code is progressed through different environments, each with a wider set of stakeholder access, ranging from an initial DT environment to which only the developers have access, to ST – to which test analysts have access – to UT when business stakeholders finally get to see the product with some fake data[[2]](#footnote-2).

### Environment Components

The exact configuration of the components required in an environment is highly dependent on the type of service being delivered. Refer to the project’s SAD for the required components.

Common components that are used in the vast majority of projects are listed below:

* Networking Services
* Web Service
* Caching Service
* Storage Services
  + Secure Credential Storage
  + Media/Blob Storage
  + Relational (SQL) Database Storage
  + Document (NoSQL) Database Storage
* Messaging Service (SMTP/Email)
* Less common:
  + Rule engine service
  + Workflow engine service
* Deprecated/no longer best practice but may be required in some cases:
  + Proxy servers

*Important:  
Although specific to each project, it is important that each Environment have identical infrastructure. Having a different set of components per environment is an indicator to managers that the service is probably being deployed manually and therefore incurring both technical debt and delivery risk.*

### Environment Configuration

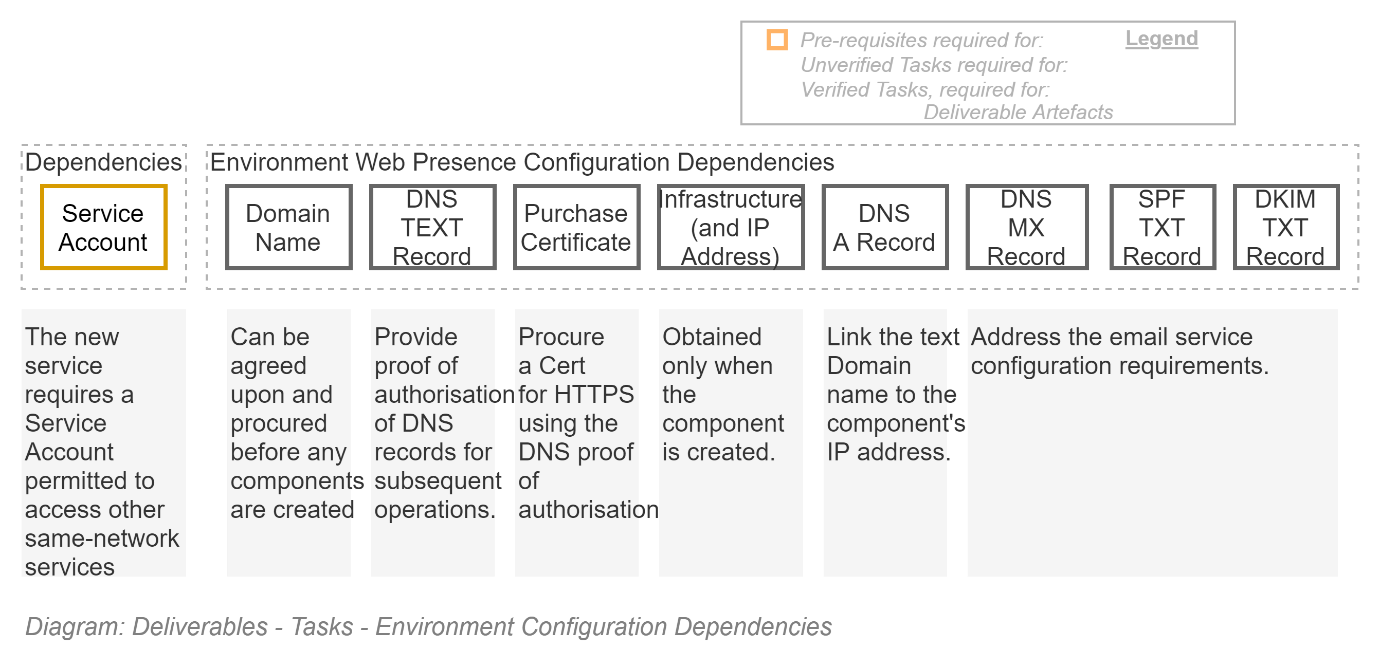


Figure 10: Deliverables – Environments - Configuration

In addition to creating infrastructure, it needs configuration before it can be used.

#### Discoverability

Services hosted on components within environments are given IP addresses (e.g.: 260.212.219.31).

IP Addresses are not easily discoverable by end users.

Users prefer using textual domain names to access services.

#### Domain Name

If the service will have its own domain (e.g.: *myservice*.com), a Domain Name must be purchased from a Domain Registrar. If the service is part of the many services that an organisation offers, the domain name does not need purchasing, and internal staff can provide a subdomain to the organisation’s domain (e.g.: *myservice*.mycorp.com).

#### DNS Records

For component to be Discoverable by end users by a textual domain name (e.g.: myservice.mycorp.com), the pair need to be associated within a Domain Name Service, by developing DNS A records (e.g.: myservice.mycorp.com == 260.212.219.31), as well recording aliases via TEXT Records (e.g.: *myservicealso.mycorp.com == myservice.mycorp.com*).

#### Email Configuration

Most services need to send out notifications to users by email. An MX record will probably be required at some point.

If the service uses a third-party email service, there is a strong risk the without proper configuration of SPF & DKIM records within the DNS, their email client will file the messages under JUNK.

## Service Integration

New Organisation Services are technically integrated with existing Organisation Services by a combination of means, for different purposes.

### Service Discoverability and User Support Integration

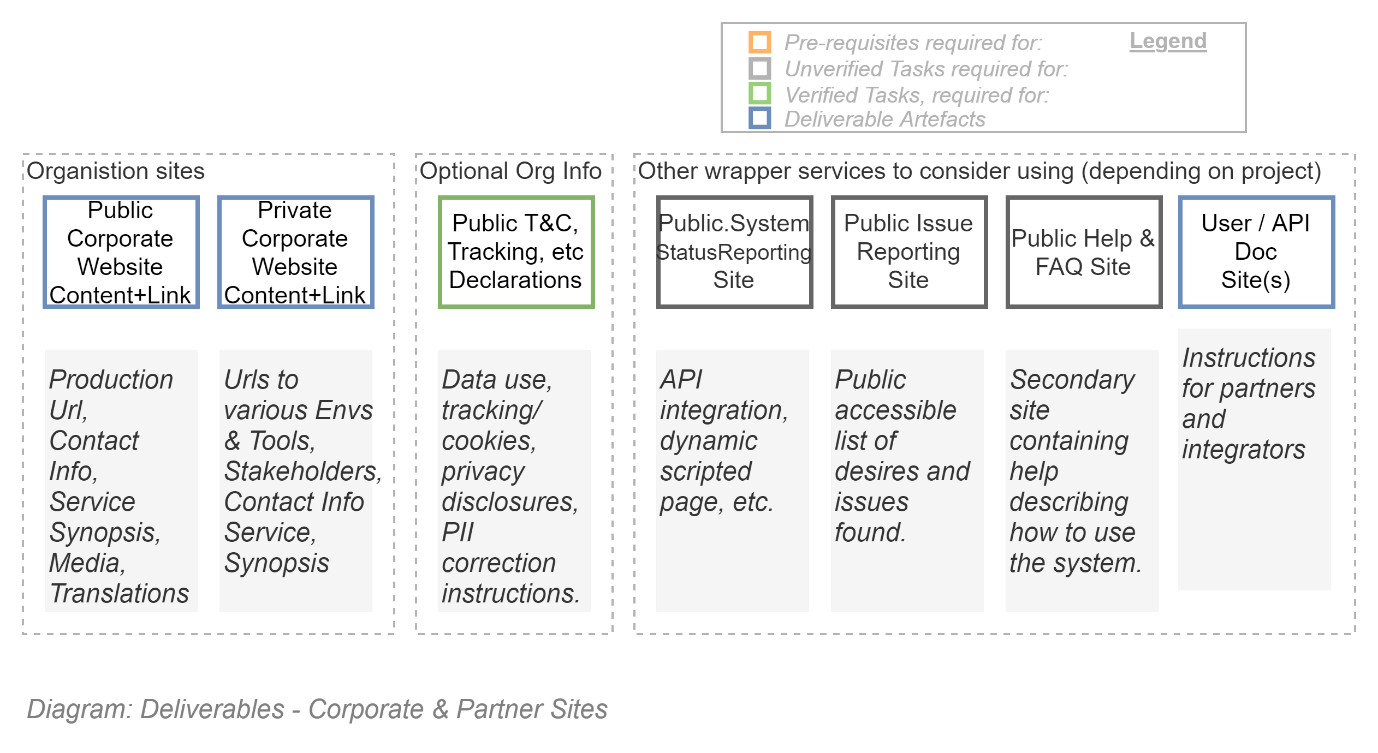


Figure 11: Deliverables - Discoverability - Corporate & Partner Sites

It serves no purpose to spend capital on developing a service that nobody can discover or finds difficult to understand.

An important part of developing services is integrating them to existing supporting corporate and partner sites that is already getting traffic, has SEO configured, and a convenient if simple and generic spaces to add contact information and very high-level documentation.

#### Public Facing Organisation Website

The public facing company website helps customers know about publicly available services. Or if the service is not publicly available, at least provides contact information for potential partners to find out more.

#### Private Corporate Website

The private corporate website (e.g.: Confluence or equivalent) helps corporate and select partner users know how what services are available, or if joining the project as a stakeholder, find out environments they can access, links to the tools the project is/was using (such as the Delivery ALM Project Space), etc.

#### Other (Optional) Websites

Other publicly available sites that should be considered to provide better service include the following:

* Public Status Site: so that users can find out why the service is unavailable, and when it will be back up,
* Public Issues Site: for users to post about encountered Issues that need fixing as well as Desired features.
* Public Help/FAQ site: to answer both any and common queries, decreasing training and support costs.
* API description sites.

The above services are generally specialist SaaS services that the organisation may or may not already have corporate subscriptions.

*Tip:  
For example, API documentation can be done by hand in a wiki of some kind – but there are specialist sites that offer an automated process of developing API documentation by scanning uploaded code, even providing code snippet usage examples, that can be transparently hooked up to the delivery pipeline. The process is easier, and the result is more comprehensive and current.   
Specialist services are available to automate the checking of the availability of sites and providing users information as to how long the site has been down and when it is expected to be back online.*

### Service Dependency Integration

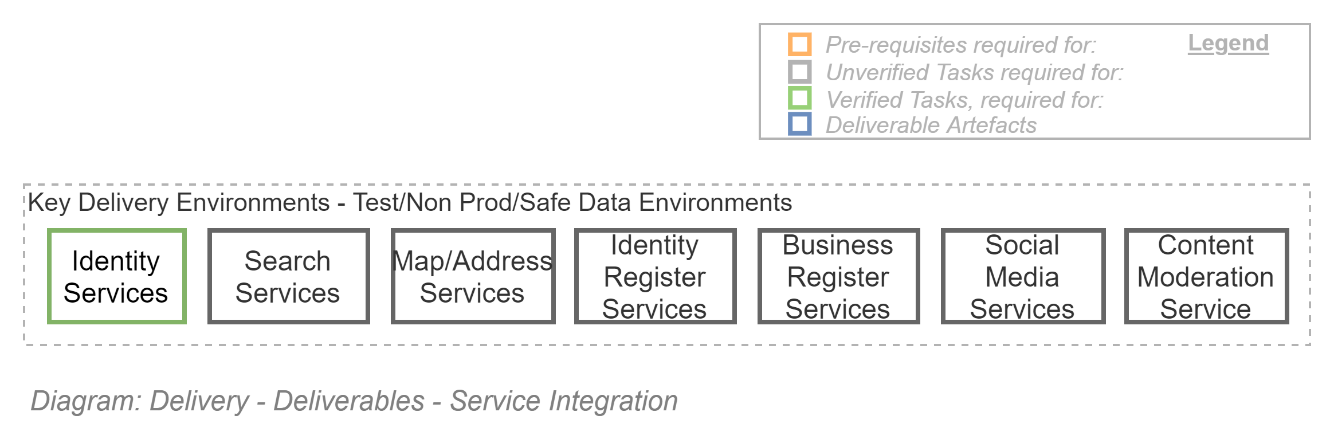


Figure 12: Deliverables - Service Integrations

Software is expensive to develop.

Hence the preference to re-use common central services which the organisation has already invested in, or can be readily subscribed to, when possible.

3rd party Services that are cost effective to integrate with, rather than build from scratch, include the following:

* Identity Management Services (i.e., Identity Provider (IdPs) Services).

Other common services that systems commonly rely on include:

* Search Services, to help where Google can’t see (into the system’s private pages),
* Map & Address Validation Services (to decrease data entry errors)
* AI based Content (Text & Media) Moderation Services (to mitigate & flag abusive comment within user feedback and public messaging)
* Identity Register Services (to decrease data entry errors by validating name spelling & DOBs)
* Business Register Services (Business Numbers (e.g.: BNs), etc.)
* Social Media Ratings/Feedback Services (Twitter/Instagram/blog posting)
* User Rating/Feedback Management Services (e.g.: Disqus, etc. to better understand what users have issue with or desire next)

## Service Text & Media

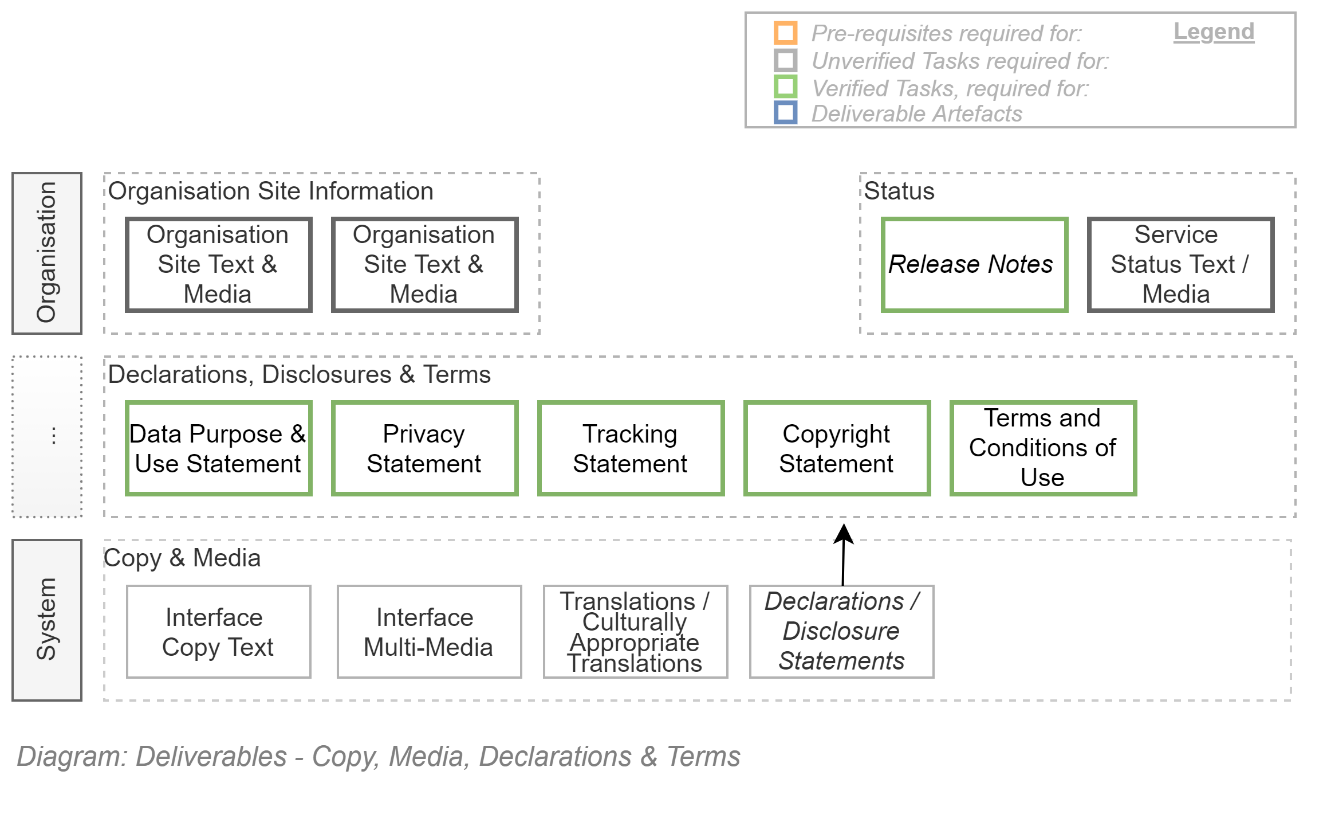


Figure 13: Deliverables - Copy, Media, Declarations & Terms

A service generally provides functionality to manipulate information – but its interface requires Copy Text and Media to put on the screens to make the service understandable and accessible. Therefore, less costly to operate.

There are several systems for which copy & media needs developing -- starting with the Organisation Sites.

### Organisation Website Text & Media

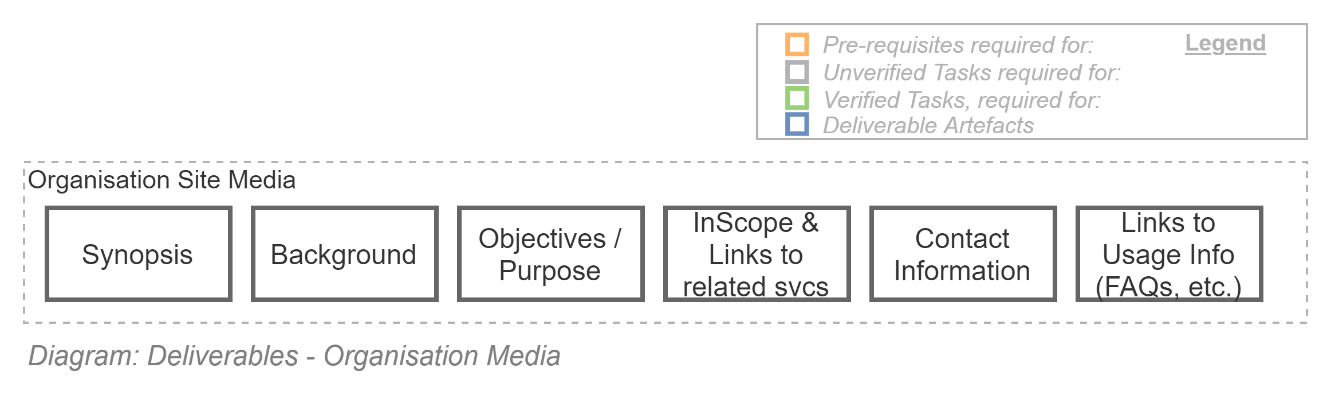


Figure 14: Deliverables - Media

In the previous section we outlined that Integration from Corporate sites to the service being provided is a deliverable. To complete the deliverable, the websites will require the creation of Copy & Media to drive users towards consuming the service.

Organisation website text & media usually is about providing a consumable brief of the service, including a Synopsis, Background, Objectives, Users & Stakeholders, Scope, Contact information.

Sometimes videos are commissioned (e.g.: sign language translations of text).

### Service Terms & Conditions and Disclosures

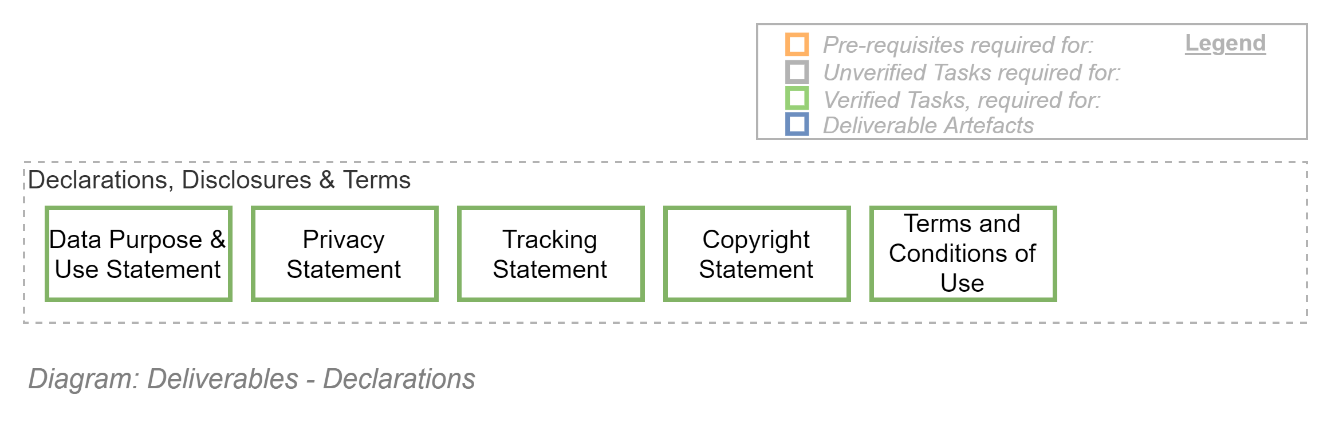


Figure 15: Deliverables - Declarations

It stands to reason users should not provide systems data without knowing what the data’s intended use, who it will be shared with, and how to correct errors.

It also stands to reason users should be able to use a system without being subjected to abusive behaviour by other users.

It also stands to reason that in return for the permission to use automation services provided by an organisation, users can be expected to abide by certain conditions (Rights versus Responsibilities).

It is common practice to provide these declarations and agreement terms for review when a user either accesses or joins an information service.

If the service is integrated and reliant on a organisation provided identity service, these disclosures and terms can be provided by the organisation’s common identity service.

If the service has been permitted to use another 3rd party consumer identity service (e.g.: Google, Microsoft Accounts), or manage user credentials internally to the solution[[3]](#footnote-3), then the project must develop these disclosures themselves, and incorporate them into the system.

### Information Service Copy & Media

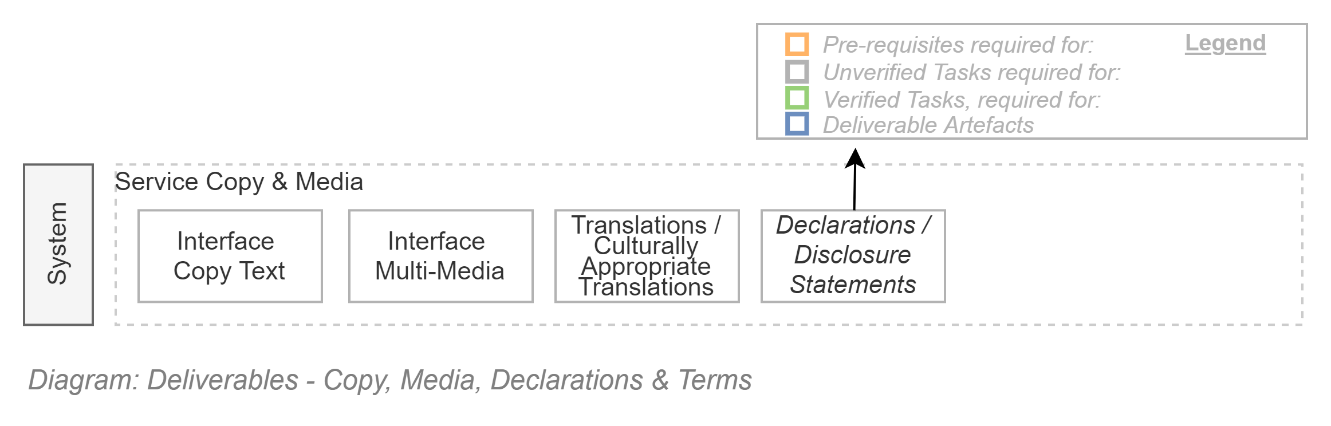


Figure 16: Deliverables - Copy, Media Declarations and Terms

The service itself will require Copy & Media in multiple languages to make its functionality accessible & understandable.

#### Service Copy Translations

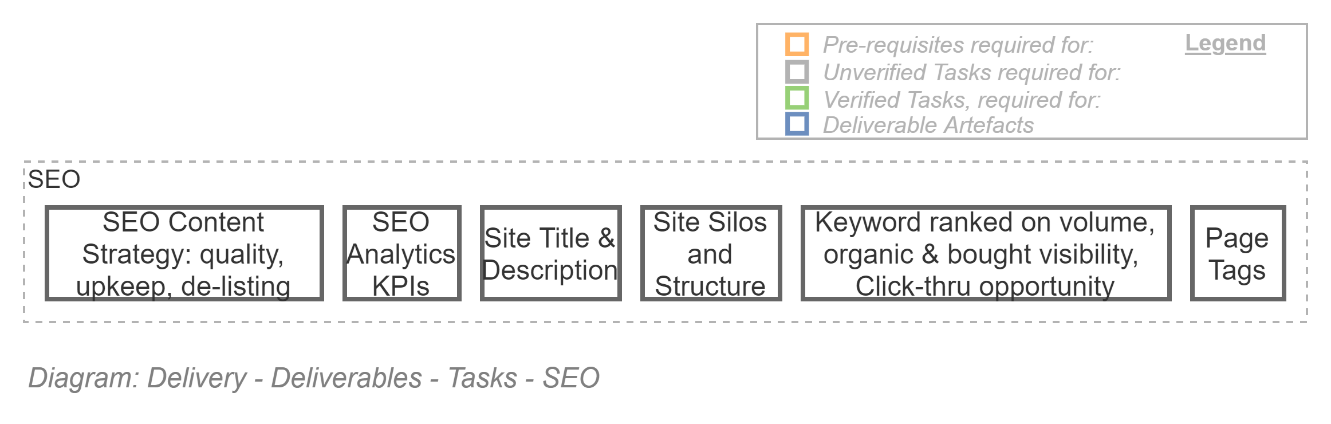
This organisation is obligated to meeting Usability obligations, including ensuring the service is available in the official national languages of the countries the service is available.

This means that all text (and media) is required to be translatable.

*Note:  
Organisation Software qualities has evolved over time in phases. Phases that the market has grown through include increasing expectations of Functionality, Usability, Accessibility, Security and – lately - Privacy. The latest phase is an expectation that software is available not only translated, but relevant, to different cultures. Cultural awareness is a current obligation of NZ Government agencies.*

*Note:*  
*Mature organisations depend on external SaaS services to manage and provide current text and media, culturally translated, so that updates and corrections can be applied without having to redeploy the whole service (embedded Language packs is no longer considered an optimal pattern in website development).*

#### Search Engine Optimisation

Search Engine Optimisation is a specialised design skillset to ensure data is automatically and correctly categorized when entered to be the later easily found by end users.

Search Engine Optimisation is undertaken for system data to be both discoverable (but not disclosed to unauthorised users) from the WWW as well as via the system’s internal search functionality if such exists.

It involves classifying publicly accessible data to be found by public search engines (Google, etc.), and classifying protected data to be found by an internal integrated 3rd party Search Service.

## Service Data

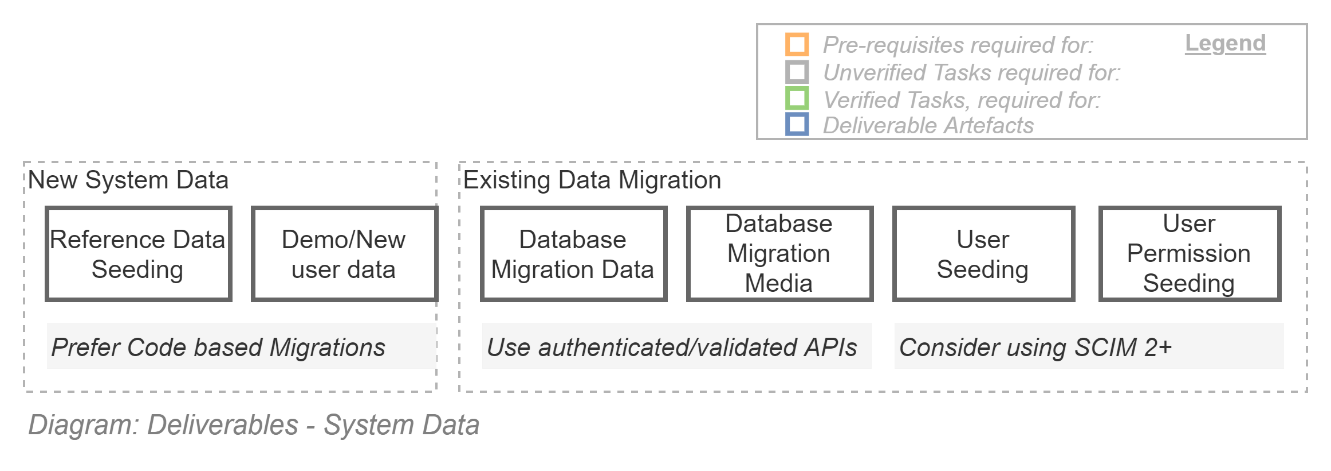


Figure 17: Deliverables - Systems Data

### New System Data & Users

New services only need seeding with immutable data, waiting for users to add mutable data.

Very rarely does a project think ahead to providing demo/training data for new users, but it’s a possibility as well.

New systems start off with no users, so don’t need seeding of such.

### Migrating Legacy System Data

Services that are replacing legacy services – which are legion in mature organisations – require migrating data from the legacy service to the new service.

Traditionally legacy data was migrated by ETL from legacy database to the new system’s database.

This approach is no longer acceptable for multiple reasons:

* it requires both systems to be in the same network (impractical in cloud environments), or
* an insecure opening of the legacy database to the WWW for a cloud service to access it, or
* the installation of an invokable agent on the legacy database (not always possible), or
* an impossible to secure dump of the data to SQL to reimport into staging tables and from there into the new service’s database,
* all data will be bypassing the new services validation capabilities, causing unexpected data bugs to resurface intermittently, for a long time.

Current best practices are to use ETL to extract the data from the legacy service, then invoke the new service’s APIs, which are presumably secured and validating data, rejecting unacceptable data, keeping the new system clean of it.

Note that migrated data is most often database records, but can include media of different types (e.g., images of driving licenses that were provided to back submitted data).

### Migrating Legacy System Users

Another form of data that needs migrating is Users.

Traditional approaches to migrating users involved much the same steps – and issues – as migrating system data.

Current best practices are similar to migrating system data – namely use the new systems authenticated, validating API endpoints to create new users, roles and Permissions.

*Note:  
If the new system provides a SCIM API interface, its use is recommended.*

## Service Acceptance Criteria

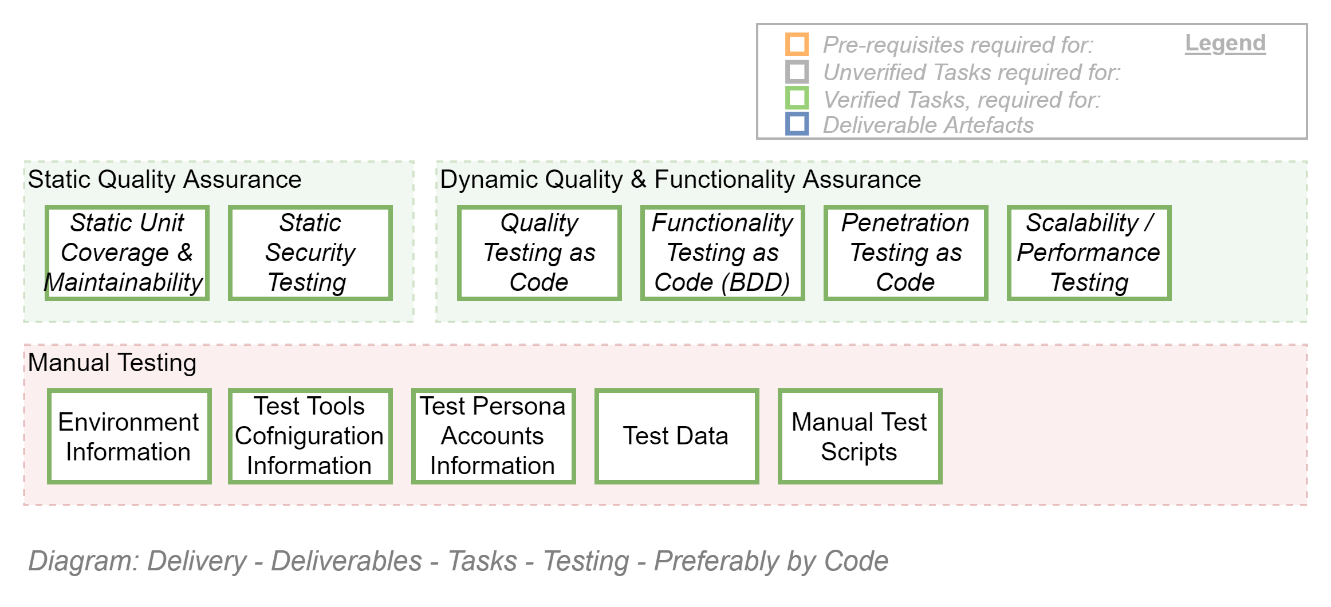


Figure 18: Deliverables - Tests

Services should not go live before they are meeting their Acceptance Criteria demonstrating they are fit for purpose.

Traditionally, acceptance criteria were checked by dynamic and static testing.

Current best practice is to develop automation of criteria checking directly within a deployment pipeline service.

Developing quality & functionality checking automation:

* decreases the amount of time required to validate criteria are met,
* increases the amount of time available to define additional criteria checks,
* permits early definition and distribution to inform developers how to prove the criteria are met,
* dramatically decreasing the amount of test failures, and therefore schedule slippage,
* all of which allows more time to iteratively improving quality.

In addition, automated tests permit service quality to be quickly ascertained, even after key personnel depart -- which must be expected over the whole service’s lifetime.

Automated testing is by now a well-established practice. Tools, best practices and skilled resources exist to test functionality, interoperability, integration, security, code maintainability, performance.

*Note:  
A combination of In-house legacy capabilities and market scarcity ensure that many in-house tested projects believe the only option is to continue doing manual testing.*

Either approach require deliverables to be developed to meet expectations.

*Note:*  
*The cost benefit of automated integration is relatively simple to calculate: an automation specialist may cost 1.5 times a traditional manual tester, spent primarily during the main development phase. With the operational period of most services being 10 times or more the length of the development phase, and most full regression tests taking about a month to schedule, prepare for, and do, automation comes out cheaper from the moment the service is released more than every 2 years during the operations phase.*

*Note:  
It is best practice to iteratively develop Acceptance Criteria logic prior to commencing System Logic, defined lower down.*

## Service Documentation

*TODO {DIAGRAM} Explanation.*

## Service

Once everybody knows what targets to meet – the automated Quality Acceptance tests run by the delivery pipeline -- it’s relatively easy to manage the process of achieving those micro targets in an iterative manner until the constantly evolving business expectations are met.

### Default Case: Information Services

The vast majority of business-desired systems are Information Services – web services that provide a means for users to add, update, remove information from a central store.

The project’s architect makes a decision as to whether the system should be a standalone service (e.g.: a Custom, COTS or SaaS service) or a logical module within a larger framework (e.g.: Salesforce).

## Deliverables based on System Type

Which one of the 4 key system types (Custom, COTS, Single-Host SaaS, Multi-Host SaaS) is chosen by the architect will inform what deliverables will be expected.

*Tip:  
Whereas with Custom & COTS SaaS based products one has to do practically everything, one finds that many of the infrastructure and configuration tasks are already taken care of (that’s what one is paying for).*

## Sequence of Deliverables

The deliverables have some dependencies between them that in turn inform scheduling, as reflected in the diagram below.

### Deliverable Components

Many of the deliverables are developed from a collection of subcomponents. When determinable, the following diagram lists these components in a proposed order of completion.

#### Logical Systems

For example, most Systems are developed in the following order of things:



#### Documentation

Is less specific in terms of the order, but should include the following:



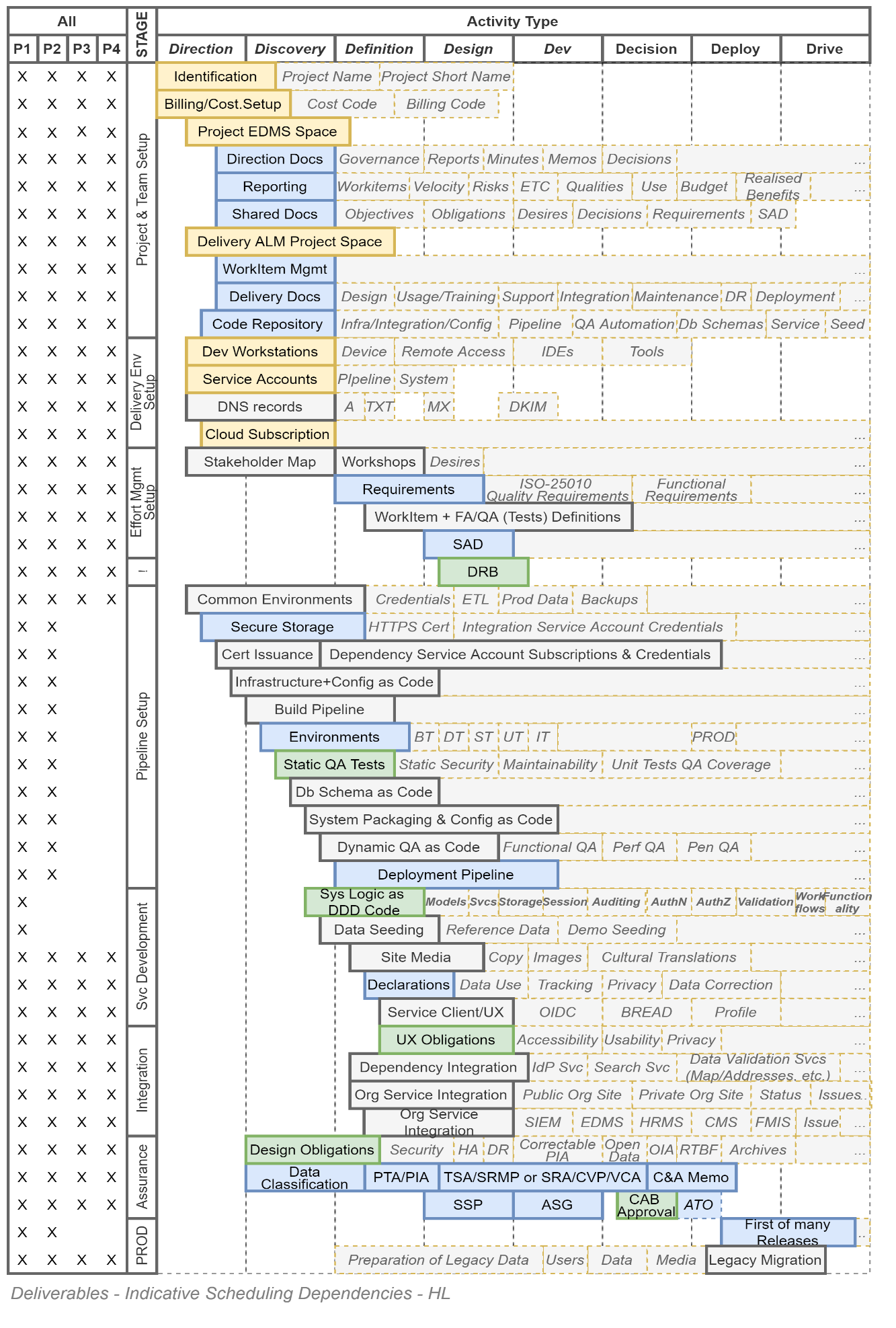


Figure 19: Deliverables - Task Scheduling

## General Condition

The general map of the dependencies is as follows:

%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22edgeStyle%3DorthogonalEdgeStyle%3Brounded%3D0%3BorthogonalLoop%3D1%3BjettySize%3Dauto%3Bhtml%3D1%3BfontSize%3D8%3BfontColor%3D%23666666%3B%22%20edge%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20relative%3D%221%22%20as%3D%22geometry%22%3E%3CmxPoint%20x%3D%22395.02857142857147%22%20y%3D%22980%22%20as%3D%22sourcePoint%22%2F%3E%3CmxPoint%20x%3D%22480%22%20y%3D%22980%22%20as%3D%22targetPoint%22%2F%3E%3C%2FmxGeometry%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

## Variations

The second type of system – the logical module within a larger framework (such as Salesforce) deserves special mention.

The fact of the matter is that the logical module is within a larger multi-tenant SaaS service (type IV) that has already obtained CAB approval for its Functionality, Security, Privacy, Supportability, Operability, Maintainability.

The only artefacts that are required to obtain CAB Approval for logical module are the module’s privacy artefacts and security artefacts.

# CAB Approval of Deliverables & Go-Live RFC

A CAB Approval process is described in a separate document: *IT Project Guidance – Accreditation*.

Appendices

Appendix A – Document

### Authors & Collaborators

Sky Sigal, Solution Architect

### Tables

**No table of figures entries found.**

### Figures

**No table of figures entries found.**

### References

|  |  |
| --- | --- |
| [1] | OWASP Foundation, “Home Page,” [Online]. Available: https://owasp.org. |

### Review Distribution

Refer to document storage auditing metadata.

|  |  |
| --- | --- |
| Identity | Notes |
| Amy Orr, Data Domain Architect |  |
| Diane Simpson, Infrastructure Architect |  |
| Stuart McGrigor, Solution Architect |  |
| Brad Atte Le Crouche, Solution Architect |  |
| Dries Venter, Cloud Infrastructure Specialist |  |
| Werner van der Merwe, Automation Lead |  |
| Yujia Huang, Security Specialist |  |
| Jeff Grove, Security Specialist |  |
| Sean Torley, ICT Change & Transition |  |
| Sally Murrey, ICT Change Advisor |  |
| Grayson Mitchell, Solution Architect |  |

### Audience

The audience is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure Conventions

Where possible, the document structure is guided by ISO-\* standards or best practice.

### Diagram Conventions

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (Archimate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Terms

Refer to the project specific Glossary.

1. Note that “mature products”, practically by definition, have been around a long time and most probably developed before Integration & Reuse was considered a Strategic requirement – therefore missing APIs. [↑](#footnote-ref-1)
2. It is absolutely essential from a security point of view that production data -- even if old & stale, obfuscated, or only partial -- is never accessible by stakeholders outside of the Production running environment. The reason is that within Prod, all operations (views, changes, deletes) are permanently recorded. Operation records of all non-prod environments are not kept – and direct access to production data in databases within those non-prod environments would allow bypassing auditing, defeating security objectives. [↑](#footnote-ref-2)
3. Note that it is considered a high risk for solutions to manage credentials within the solution, and would absolutely require prior acceptance from architectural & security governance boards. [↑](#footnote-ref-3)