Risks & Mitigations

IT Project Design

## Synopsis

The success of a programme/project which contains an IT element depends on the timely delivery of deliverables expected and usable by different stakeholder groups.

These expectations are unlikely to be met without a clear idea of what risks are common to IT projects, so as to avoid them. This document lists and explains common risks to consider avoiding.

## Author

Sky Sigal, Solution Architect, SPA, ITC, MOE

# Document

## Table of Contents

[Synopsis i](#_Toc75771624)

[Author i](#_Toc75771625)

[1 Document 2](#_Toc75771626)

[1.1 Table of Contents 1](#_Toc75771627)

[1.2 Creation 1](#_Toc75771628)

[1.3 Distribution 1](#_Toc75771629)

[1.4 Conventions 2](#_Toc75771630)

[2 Background 3](#_Toc75771631)

[3 Design Risks 4](#_Toc75771632)

[3.1 System Domains 4](#_Toc75771633)

[3.2 User Domain 4](#_Toc75771634)

[3.3 Design Approaches 6](#_Toc75771635)

[3.4 Environment Driven Design 7](#_Toc75771636)

[3.5 User Group Driven Design 8](#_Toc75771637)

[3.6 Common System Design Logical Errors 10](#_Toc75771638)

[3.7 Development Approaches 10](#_Toc75771639)

[4 Appendices 12](#_Toc75771640)

[4.1 Terms 12](#_Toc75771641)

[4.2 Market References 12](#_Toc75771642)

## Creation

Date: 18/05/2021

## Distribution

Distribution/Review List:

* Amy Orr, Data Architect
* Brad Atte Le Crouche, Solution Architect
* Diane Simpson, Infrastructure Architect
* Michelle Nicolson,
* Dries Venter, Cloud Infrastructure Specialist
* Grayson Mitchell, Manager, BE&S ICT SPA
* Jeff Grove, Security Specialist, Contractor
* Sally Murrey, ICT Change Advisor
* Sean Torley, ICT Change & Transition
* Stuart McGrigor, Solution Architect
* Werner Van der Merwe, Automation Lead
* Yujia Huang, Security Specialist, Contractor

## Conventions

### Terms

The Appendices include a Glossary of Terms to assist understand this document.

### Diagrams

Where applicable, diagrams are developed using ISO-\*, Archimate, UML or appropriate industry standards and conventions.

# Background

# Design Risks

## System Domains

Systems are interactions between the following distinct domains:

* User Services Domain
* Security Services Domain
* Business Services Domain
* Technology Services Domain
* Infrastructure Services Domain

## User Domain

The User Domain is the source of confusion which is mitigated by clearly defining the difference between terms that are often confused and thought of as the same:

* People, Users, Identities, Names.

They are not the same.

IT Project – Design Risks & Mitigations

Synopsis

Background

Problem

Objective

Key Design Concepts

Overall

It’s about

* Items and Classification of them
  + People, their Grouping
  + Records and Media and their organisation into Containers
  + Operations and Workflows
  + Permissions and Roles
  + Metadata and Discoverability

People

People

Persons

There are two types of Persons – physical Persons (ie, you, me, other living creatures) and logical Persons (eg: legal Organisations).

Persons and Names

A Person does not have a single definitive Name. A Name is contextual. For example, a Chinese person working overseas will have a Chinese name he is known by in china, and a westernized name, used on passports and working papers, making it easier for co-workers to refer to the same person. Both are real and legal Names. No name is more ‘real’ than the other, so it’s not a question of one being the Primary Name, and the others being Aliases -- it will depend on context.

Even if his name can be pronounced by western co-workers, there is a Chinese character version of the name, and a western alphabet version of the name.

And then there are almost universal nicknames and aliases (“Billy” as opposed to “William”).

Western countries use GivenNames that are chosen, Middle names that are most often chosen to indicate some aspect of lineage, and Surnames which can – but not always – be the name of a family unit. The practice is certainly not universal. Sometimes lineage is encoded in the Surname (“Urs”’s children will get the name “Urson/Orson”) for example. Some Indonesian names are tremendously long, encoding lots of information into them.

Online, we can have different DisplayNames, for example, be known as “Little MissFit”.

Persons and Identities

The fact that a Person has multiple Name depending on context is an important foundation to better understand that a Person therefore has multiple Identities[[1]](#footnote-1).

Identities are not universal. They are contextual, tied to the group you have a role within.

But they are not the same as the Name. For example, in the Home Group, you are maybe known by two or more nicknames, as well as your real name.

In a technological context, a Person may have an identity in an external IdP. The token it provides to Persons who provide correct Credentials contains information that the Service can then map to an internal System User record (or create the User record it if it is the first time the Person is using the System).

System User

The System User is a system specific representation of a Person[[2]](#footnote-2). The System User is then provided a customizable internal system Identity.

At which point the System User will have at least two Identities assigned to it – one containing what it knows as to its external IdP’s identity, and the second containing an internal, customisable, identity (often referred to as the User’s Profile).

Users, Groups, Roles, Permissions

It is a logical error to model users as members of a single group. Users can belong to multiple groups at the same time.

It is also a logical error to try to emulate the complexity of human interactions as Users within a Group, and relate Permissions to Users separately.

Users more often than not belong to multiple Groups at the same time.

Hierarchical Groups

Groups can be hierarchical. This is how most HR systems work. Inheriting permissions from parent Group Roles is an approach to distribute Permissions easily and quickly. As long as specific Permissions can be removed from a user if need be, and alternate Permissions added instead.

Roles are Group dependant.

Users have different Roles per each group. A consultant may be a Contributor to a project group, while being only Informed of another project, while being Accountable back at their own firm.

Roles are not Permissions

They are simply an easy way to assign Permissions to a User (according to their Role in a Group). But there are times that Roles are too unwieldy to correctly capture needs. For example, everybody in the Accounting Group could be assigned an Accountant role – giving them access to payment tools – but maybe new employees require the Permission to develop Medium or High value Purchase Orders should be removed from them. One can make up multiple Roles (Accountant/Trainee Accountant) but they usually don’t cover all desired permutations.

Location

Communication Channels

Groups and Organisations

Content

Overview

Records

Documents

Metadata

Operations

Permissions

Roles

## Design Approaches

### Protocols Driven Design

For portability reasons it is beneficial to provide externally accessible APIs that adhere to well-known or standards based Protocols.

For strategic and maintainability objectives it is a monumental error to develop a systems internals based on its external protocol obligations. This fact is even more crucial if the system is an external user facing service.

The reason it is a risk is that

* it locks your service offerings into an external consortium’s vision,
* removing the ability to differentiate your product from other offerings
* increases development time, effort and cost by artificially constraining development to protocols in the *internal* space -- where it is not needed – as well as in the external space, which is an accepted cost of doing business when chasing integration opportunities.
* inability to deliver on Vision if the protocol managers are unwilling to modify the protocol to address opportunities / risks the project has identifed.
* The product may be mortally damaged if the protocol is retired, leaving you with the worst of all worlds – internals that are hampered by an external consortium’s original vision AND the need to map the internals to meet the obligations of a new external protocol.

### Database Driven Design

As it is a risk to base internal logic entity design off of external API protocol obligations, it is a significant risk to develop a system off of its data design obligations.

### User Interface Driven Design

### Domain Driven Design

### Principle Driven Design

## Environment Driven Design

#### LAN based development

A strong financial incentive by software and hardware distributors remains to continue to sell LAN based architecture, environments, devices. But the era, from a user’s point of view, is over. IP6 over WAN, skipping IPv4 over LAN, is the future for all devices– certainly by the midpoint in the lifetime of any solution that is begun today.

Do not design solutions to be protected by a Firewalled LAN. Design them to be secure from an insecure device (corporate in-flight back-of-seat browser), over an insecure network (multiple public free wifi zones) in an insecure environment (eg: airport).

#### Desktop Development

Desktops have multiple advantages, including the option for eye height multiple screens, full size keyboards, power – yet market sales clearly indicate these advantages are trumped by mobility and the option to access and modify information anytime, anywhere[[3]](#footnote-3).

The market demand for mobility above all other features is currently driving hardware manufacturers into developing other emergent forms of computers (dual screen mobile phones, Augmented/Extended Reality headsup glasses, etc.). The future remains uncertain – but the past is certain: developing for desktop-bound computing first is a design error. The consensus is to develop adaptively for multi-format screens, ensuring Mobile phone interfaces are priorised over other formats – expecting to have to invest in adapting to other formats (AR/XR?) over the software’s service lifecycle.

#### Central Repository Development

Central repository information systems were the logical solution to managing information by a central agency who had the resources to provide the storage, systems, redundancy, development and maintenance resources needed. A key defining concept of this era was Storage.

Since, Storage is less of a defining constraint: memory is vastly cheaper -- to the point that most cellphones can now Store the education record repositories of large countries in less space than is used to hold 10 family photos. Redudency is addressed by having multiple phones/systems synced by P2P, between themselves as well as with cheap cloud storage.   
Search/Discoverability and Distributed Secure Storage is a design domain that is growing to take advantage of the lower hardware costs of such systems.

The key difference between the two design eras is the first was designed around Storage (DBs, stored procs, ETLs, etc.) and the second starts with agreement around exchange Protocols, leaving sorting out secure storage as a minor background necessity.

## User Group Driven Design

### Individual Consumer Focused Development

It is easier to develop for individual users for reasons including:

* The target audience is closer to what a developer is, therefore recognises,
* Being more easily recognised by developers, can be developed faster, without investigation and obtaining consensus from other stakeholders,
* Reduces, Delays or even Removes the need to develop capabilities thare are considered essential for Enterprise users: security, groups, hierarchy, roles, permissions, messaging, workflows, decision escalations, enterprise system integration.

Due to the volume of users, Individual Consumer focused development usually leads to quickly addressing capabilities that allow

* Multi-language and multi-cultural user interfaces
* User Feedback
* User Invitation

Observations of Individual Consumer focused development include:

* The system will not have a robust architecture of permissions and role management to allow remote assistance.
* Collaboration when provided is on an “all or nothing basis” which requires a level of trust that an organsisation with a public reputation to protect cannot risk.

### Enterprise Focused Development

Whereas Individaul Consumer Development Development leads to systems that meet individual expectations, at the expense of capabilities required to work as individual roles within groups, to achieve common agreed outcomes.

Enterprise focused development usually emphasise good practices by providing:

* Capabilities to integrate and reuse organisation provided specialised services – such as IdPs, SIEMs, Reporting, Data warehousing.
* Capabilities to authorise access using a Permission based authorisation architectures for finer control (bundled as logical Roles for easier allocation) as opposed to only relying on a more primitive and limited Role based authorisation architecture which might be suitable in a Individual Consumer based design.

Enterprise focused Development is often poor at the following capabilities:

* Capability to work in partnership with collaborators & consultants from outside the organisation,
* Capability to assign roles & permissions per projects (usually limited to an organisation-wide HR defined role)
* Capability to assign roles & permissions per resource (as opposed to per organisation or per project),
* Capability to assign multiple identities to the same resource (as opposed to having only one official way to name a person irrespective of their cultural background, one language to enter it in, irrespective of the umlauts, macrons, etc. in their original name, one title throughout the same organisation, one desktop phone number, etc.
* Capability to collect feedback, as most groups don’t have the mandate to put up a project specific feedback center, nor the resources to dedicate a person to handling feedback long term -- and a central service may find it difficult to know where to forward messages to.
* Capability to meet Multi-language and multi-culture user interface objectices.

### Collaborative Group Focused Development

The “sweet spot” that takes the best of both Individual and Enterprise focused design approaches to provide the capabilities required to meet current service lifecycle needs[[4]](#footnote-4).

Collaborative Group based development recognises that

* all tasks are managed by a group which may start with just a single person as the groups owner,
* may be joined by a different person, inside or outside the enterprise, all of which may invite others,
* all of which may have introduced themselves first using unofficial names (nicknames, English replacement names (eg: “Ben” as an easier name for English speaking natives to use instead of mangling the person’s real name), etc.
* all of which can be assigned group specific roles.

The Collaborative Group First -- irrespective of whether it is an organisation defined Group or not – provides the necessary

* Capabilities to invite internal peers and external consultants to specific group roles in order to work collaboratively.
* Capabilities to share resources with collaborating agencies as well as public individual services consumer users.
* Capabilities to mirror legacy Enterprise system capabilities by providing means to nest Groups, inherit parent Permissions unless overridden, designate a top Group as an Organisation and/or Enterprise.

For all its positive capaibilites, there remains a downside to the Collaborative Group-First based Design: it is emergent, slow to make headway in the enterprise space against entrenched sales forces, hence for most objectives cannot be bought off the shelf. Depending on the capabilities of a project’s development team this may affect budget and timelines, positively or negatively.

## Common System Design Logical Errors

### Users, Groups

### Organisations, Partners, Franchises

### Permissions and Roles

### Permissions and Responsibilities

### Notifications, Disclosures and Versioned Obligations

### Diagnostics Logging and Auditing

### Physical versus Logical Deleting of Information

## Development Approaches

### Task Driven Development

### Demo Driven Development

### User Driven Development

### Test Driven Development

# Appendices

## Terms

## Market References

* Mobile First pays better:
  + “In 2019, mobile-first companies had a [combined](https://techcrunch.com/2020/01/15/app-stores-saw-record-204-billion-app-downloads-in-2019-consumer-spend-of-120-billion/) $544 billion valuation, 6.5x higher than those without a mobile focus.”
    - https://techcrunch.com/2021/06/26/this-week-in-apps-android-apps-on-windows-11-app-store-search-ads-hit-china-apple-argues-against-sideloading/

1. Mom still calls you “Dearie” at home, while your boss knows you “Mary” in his group’s meetings and those you manage may know you as “Miss Larkins” (and hopefully no other alias beyond your back). [↑](#footnote-ref-1)
2. A system User is specific to a single System – it cannot be a User in two different Systems (although it’s external Identity – from the IdP -- can be used to set up a different System User in a different System). [↑](#footnote-ref-2)
3. An era everybody would like to forget is the decade lost to the development SmartClients, when browsers were being marketed (by desktop OS developers) as not yet capable enough for practical business needs. Note that Mobile Applications are exactly the same thing -- SmartClients on mobile operating systems – hence our recommendation to stick to multi-screen browser based development. [↑](#footnote-ref-3)
4. Gen-XYZ, the intended public users of a service, as well as the internal new hires for any organisation that is not dying under the weight of an aging workforce clinging to what it knew in the past, work more dynamically, from multiple devices, from multiple locations, seeking input and validation from multiple sources. [↑](#footnote-ref-4)