ArchitecturePLAYBOOK Documentation

Release 1.4

Maikel Mardjan

Nov 19, 2020

Contents:

CHAPTER 1

Introduction

Smart people have been thinking on how to create IT architectures as long as there has been computers. Ideas come and go, however creating a good architectures can still be complex and time consuming. Especially when you try to invent the wheel for yourself. With this interactive playbook you can create your IT architecture better and faster. The focus of this architecture playbook is in on:

1. Knowledge reuse. Why reinvent the wheel again? It is far and more fun to create a better wheel for your organisation or IT project instead! Focus on the hard complex context specific issues. Use good open tools and knowledge for the easy 80%!
2. Easier creation of architecture documents and deliverables. This playbook has an extensive list of all(*) open tools available for creating your IT architecture or design. Using these open tools will speed up the process of creating your architecture deliverables and reduce your risks.
3. Quality improvement. By making use of content parts provided for various architecture deliverables you will lower your business risks. Complex business IT projects will fail. Now and in future. But if you make use of proven methods and tools developed from decades of IT architecture scientific research you will lower the risk for your project. Architecture will help to make your projects more successful in terms of costs, speed and profitability.

(*) If you miss a good open tool that is also usable for IT architecture creation, do not hesitate to contribute to this open publication!

This architecture playbook is divided in the commonly used architecture sections:

- Business
- Data
- Applications and of course
- Technology Infrastructure (TI)

This playbook is primarily created for on-line usage. But also ePUB and PDF versions are available.

1.1 Why this architecture playbook?

Creating a business IT architecture has many benefits. However creating a complete architecture is known to be:

- Difficult
- Time consuming

Of course many tools have been developed the last 30 years to make creating architecture (documents) easier. However many tools force you into a very tight harness without the needed flexibility. Complex problems need flexible solutions and tools should not slow you down in creating sketches, documents and typical architecture deliverables. A one size fits all tool for solving complex business IT problems does not exist. And since creating good architectures is knowledge intensive work that delivers real value for business you should be aware of vendor lock-ins and using methods and tools that do not share knowledge sharing. Open Source tools and open access documentation offer a default head start for reuse and improvement of knowledge work in IT architecture.

You can buy and read many many books on how to do architecture well. All books claim a magic method and promise ultimate success when followed. We love books and reading methods. E.g. TOGAF. However this playbook is different. This architecture playbook is not about describing how you should create your architecture. You are free to use every method you want. This architecture playbook is all about giving you direct usable content and tools that you can use and reuse for your architecture challenge.

So this playbook brings you ultimate freedom and flexibility. You can export the results from certain tools in any desired format. In this way you can still use your architecture enterprise tool(s) that you already have invested deeply in. So use the outcome of tools within this playbook in combination with the enterprise architecture tool that are mandatory within your organisation.

1.2 What is in this architecture playbook?

Many good books and courses already exist that cover the why and how of (Enterprise) IT architecture. So this book will not explain aspects and concepts of the very broad field of IT architecture. This playbook is targeted on providing practical tools for creating your IT architecture or design faster and better. So the focus is 100% on the real thing by using the following leading principles:

1. Tools that speed up creating your business IT Architecture.
2. Collections of principles, requirements, standards and reference architecture that speed up the creation of your architecture deliverable.
3. Collection of various Architecture OSS Tools that speed up creating your business IT architecture.
4. Collection of architecture QA tools to control and manage your IT projects.
5. Collection of reusable reference architectures, foundation architecture, architecture journals and more. Knowing where you can find what in an easy way will reduce your time. So you will have more time to spend on solving your problem in your organization.

1.3 Caution: This architecture playbook contains OPEN material only!

This architecture playbook is created to be open. This means that:

1. This architecture playbook is licensed using a liberal license (CC-BY-SA). This means that you can reuse remix, transform, and build upon the material in this architecture playbook for any purpose.

2. All tools mentioned within this architecture playbook are licensed under a OSS license and all documents referenced to is licensed under an CC license whenever possible.

This gives you the opportunity to reuse and mix content from this architecture playbook the way you want. This also means that all tools mentioned to in this architecture playbook are open and free to use without costs. We believe that sharing tools and knowledge, especially knowledge for making better architectures SHOULD be free. If you want to host this Architecture Playbook on your own company site, Intranet or just want to have more information on the OSS tools: Contact Us, or visit the github repository.

1.4 Notational Conventions

This architecture playbook uses the key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL”. This key words are to be interpreted as described in RFC 2119. Using these key words gives clarity and avoids confusion.

CHAPTER 2

Business Architecture

A Business Architecture defines the structure of your organization in terms of its governance structure, business processes, business services and products, business information and stakeholders. A business architecture outlines the relation to strategic goals towards a working system. So in short a business architecture can be regarded as a blueprint of the working of an organization. By using a business architecture your organisation can handle changes and problems with less cost and in less time since you know all the important dependencies, relation and information flows. Every successful organisation, small or large, SHOULD have a business architecture. A business architecture does not have to be fully descriptive, complete or created among a certain standard. Designing organisations and describing the working of an organisation is know to be complex and hard. However simple tools exist to create a meaningful business architecture for your organisation.

This section is created to speed up the process for creating a real business architecture.

You can use one of the following tools:

- Business architecture template(s)
- (Re)use business principles
- Modeling your business processes
- Defining your business products

2.1 Tools for creating a business architecture

To speed up the process of creating your business architecture you can make use of one of the following tools:

- Collection of many nice design principles used by companies (180+ examples).
- Archi. Archi™ GUI tool for creating a business architecture using the ArchiMate modelling™ language. The Archi tool is targeted toward all levels of Architects. The tool is MIT Licensed, so it provides a low cost solution to users who are looking for a free, open ArchiMate modeling tool.
- Causal Loop Diagram. Tool to make easy causal loop diagrams in your browser. A causal loop diagram (CLD) is a causal diagram that aids in visualizing how different variables in a system are interrelated. This FOSS tool

has an nice animation to presents effects also! The original (OSS) source can be found here. But the version on NOComplexity.com is a (GPL) fork with some extra features (source code).

- Camunda Modeler. Camunda Modeler is an OSS desktop application for editing BPMN process diagrams(2.0) and DMN decision tables. Business Process Model and Notation (BPMN) is the global standard for process modeling and one of the most important components of successful Business-IT-Alignment. BPMN is an open standard. But there are not many good OSS available packages available. Camunda Modeler is an OSS BPMN solution and is part of an open source platform for workflow and business process management. So when you use the Camunda Suite you can also use the execution engine for your processes you have modeled.
- Protégé. Protégé is an OSS web or desktop application that can be used for building business ontologies.
- DrawIO. A online FOSS program to create diagrams. Its a very advanced, but also very simple to use program to create all types of (business) diagrams. UML, BPMN, ArchiMate, swimlane diagrams and many more. If you like to create good diagrams fast, think of using this program. Since its FOSS the source can be found on https://github.com/jgraph/drawio if you want to host it yourself. There is also a desktop version (electron build) of draw.io. Check https://github.com/jgraph/drawio-desktop.

2.2 Business Principles

Having solid business principles is key for a successful company. Small or large. Having good and solid business principles is key for developing a good architecture within solid time and cost constraints. Business principles SHOULD be defined and agreed upon within a solid process with all key business stakeholders involved. The list below with principles below can give you a head start, since these principles are collected from various successful businesses.

2.2.1 Solution space

Statement
Never try to use technical solutions to a non-technical problem.
Rationale
Of course some IT and technology can help to solve problems, but technology and IT is never enough. A non technical activity, process, behaviour change, etc is always needed.
Implications
Do not focus on technology and IT solution only when solving problems.

2.2.2 Start simple

Statement
Start simple.
Rationale
Start simple. Build a high quality solution to a real problem for a cohesive group of people. If you solve one problem really well, then you can move on to the next problem (one simple approach at a time) instead of trying to tackle several things at once and, as a result, not really solving anything. Product development is about earning the right to build the next thing.
Implications
Complexity will come later when having invest lots of time and money and simple adjustments have more impact. Fixes will be more complex when products are mature anyway due to backward compatibility requirements.

2.2.3 Create a MVP fast

Statement
If your MVP takes a year to build. . . it's not an MVP.
Rationale
Creating a MVP should take not more than 1 month.
Implications
A created MVP is not ready for sale, but ready to learn from for later stages.

2.2.4 First, make it easy. Then make it fast.

Statement
First, make it easy. Then make it fast. Then make it pretty.
Rationale
When launching a new product cost are high. Making a good performing product is complex and expensive. So when working on a MVP, try to make a product easy to use and easy to create. Focus on UX.
Implications
Difficult back-end performance and scaling issues are handled later. This can increase development cost if performance is never accounted for in a design.

2.2.5 Use Open Data, Open Standards, Open Source, and Open Innovation

Statement
Use Open Data, Open Standards, Open Source, and Open Innovation
Rationale
Too often in international development, scarce, public resources are spent investing in code, tools, and innovations that are either locked away behind proprietary, fee-based firewalls, or created in a bespoke way for use in sector-specific silos. This principle: Use Open Data, Open Standards, Open Source, and Open Innovation provides a framework to consider an "open" approach to technology-enabled international development.
Implications

- Adopt and expand existing open standards.
- Open data and functionalities and expose them in documented APIs (Application Programming Interfaces) where use by a larger community is possible.
- Invest in software as a public good.
- Develop software to be open source by default with the code made available in public repositories and supported through developer communities.

2.2.6 Strategic focus

Statement

Investment decisions are driven by business requirements

Rationale

Core government needs and priorities should be the primary drivers for investment. Investment decisions should be defined by the agency's vision and strategic plans as well as the requirements of the business. These should also take into account whole-of-government strategic guidance. A business-led and business outcome-oriented architecture is more successful in meeting strategic goals, responding to changing needs and serving consumer expectations. Government service requirements will define any required technological support.

Implications

Agencies need to align with whole-of-government strategic direction. • Agency's strategic plans need to align with whole-of-government strategic direction. • Investment decisions should be made in accordance with the agency's vision and strategic plan. • Changes to processes, applications and technology should be made in response to an approved business initiative. • Design of business solutions will need to be aligned with, and traceable to strategic goals and outcomes. • Services, processes and applications will need to be designed from the perspective of the service user. • Building or redevelopment of applications and solutions will be undertaken only after business processes have been analysed, simplified or otherwise redesigned as appropriate. • Applications are delivered in a collaborative partnership with the business owners to enable solutions to meet user-defined requirements for functionality, service levels, cost and delivery timing.

2.2.7 Make things open: it makes things better

Statement

Make things open: it makes things better

Rationale

We should share what we're doing whenever we can. With colleagues, with users, with the world. Share code, share designs, share ideas, share intentions, share failures. The more eyes there are on a service the better it gets — howlers are spotted, better alternatives are pointed out, the bar is raised. Much of what we're doing is only possible because of open source code and the generosity of the web design community. We should pay that back.

Implications

2.2.8 Maximise Benefit to the Enterprise

Statement

Information management decisions are made to provide maximum benefit to the enterprise as a whole.

Rationale

This principle embodies "service above self ". Decisions made from an enterprise-wide perspective have greater long-term value than decisions made from any particular organisational perspective. Maximum return on investment requires information management decisions to adhere to enterprise-wide drivers and priorities. No Organisation Unit will detract from the benefit of the whole. However, this principle will not preclude any Organisation Unit from getting its job done.

Implications

Achieving maximum enterprise-wide benefit will require changes in the way we plan and manage information. Technology alone will not bring about this change. Some organisations may have to concede their own preferences for the greater benefit of the entire enterprise. Application development priorities must be established by the entire

enterprise for the entire enterprise. Applications components should be shared across organisational boundaries. Information management initiatives should be conducted in accordance with the enterprise plan. Individual organisations should pursue information management initiatives which conform to the blueprints and priorities established by the enterprise. We will change the plan as we need to.

2.2.9 Maximize Benefit to the complete enterprise

Statement

Information management decisions are made to provide maximum benefit to the enterprise as a whole.

Rationale

Decisions made from an enterprise-wide perspective have greater long-term value than decisions made from any particular organizational perspective. Maximum return on investment requires information management decisions to adhere to enterprise-wide drivers and priorities. No minority group will detract from the benefit of the whole. However, this principle will not preclude any minority group from getting its job done.

Implications

Application development priorities must be established by the entire enterprise for the entire enterprise. Strong enterprise governance is required. Information services should be shared across organizational boundaries. Information management initiatives should be conducted in accordance with the enterprise plan. Individual organizations should pursue information management initiatives which conform to the blueprints and priorities established by the enterprise. We will change the plan as we need to.

2.2.10 Reliability

Statement

Information and information systems are reliable, accurate, relevant and timely

Rationale

The take-up and use of lower cost channels will depend on users of services trusting the ability of the organization to provide reliable, accurate, relevant and timely information to consumers.

Implications

Good processes create good data. Processes will need to be the focus of ongoing continuous improvement (which in turn will improve reliability, accuracy, relevancy and timeliness). Our organization needs to deliver information which customers can rely upon.

2.2.11 Reuse and Improve

Statement

Reuse and Improve

Rationale

As the use of information and communications technologies in international development has matured, so too has a base of methods, standards, software, platforms, and other technology tools. Yet too often we see scarce resources being invested to develop new tools when instead existing tools could be adapted and improved. This principle: Reuse and Improve highlights ways that adaptation and improvement can lead to higher quality resources available to the wider community of international development practitioners.

Implications

- Use, modify and extend existing tools, platforms, and frameworks when possible.
- Develop in modular ways favoring approaches that are interoperable over those that are monolithic by design.

2.2.12 Reuse before Buy, Buy before Build

Statement
Prior to acquiring new assets, the company will reuse applicable existing information and technology assets. If no existing internal asset is available for reuse, the company prefers to acquire, by purchasing or licensing, applicable externally available assets. The company least preferred option is to custom build a new asset.
Rationale

- Reusing IT assets (for example, IT systems or data) that are already available is often the simplest, quickest, and least expensive solution, assuming that the IT assets in question sufficiently fit the intended purpose.
- It is less expensive to buy standard IT solutions than to custom build them, as long as they are not adapted and maintenance is left to the product supplier.
- Many authoritative data sources make their data products available (or offer data acquisition / generation services), reducing the company's need to generate such data itself.
- Custom development of IT assets is often very expensive to sustain.

Implications

- When functionality is required, existing IT assets in the organization must be evaluated and used first, unless they do not exist and/or are a significant mismatch to the required functionality.
- To ensure that IT assets are being reused as much as possible, business areas must be prepared to adapt to existing solutions that provide adequate functionality, particularly in situations where the accountable governance body does not deem that business area's practices to be required to be different from industry standard practices.
- The company will prefer COTS products and particularly those that are configurable. Some products are so configurable that there is little difference between extensive configuration and custom development. The company y must clearly understand when configuration equates to custom development (that is, the level of configuration is so high that the COTS solution is essentially the same as custom development). In these cases, the scenario will change from buy to build.
- Agreements or licenses to use data may have legal implications and legal consultation should be part of the process of deciding to use a new data source.

2.2.13 Routine Tasks are Automated Where Appropriate

Statement
Routine tasks that can be automated are automated, where the benefit justifies the cost.
Rationale

- Routine tasks require relatively little specific knowledge and can be automated fairly easily.
- Automated tasks are more cost efficient and timeefficient, and less errorprone, than manual tasks.
- Employee capacity requirements can be optimized, freeing them up to focus on more complex activities.

Implications

- The knowledge required to perform certain tasks is analyzed and embedded in an IT system when it can be easily formalized.
- Nonroutine tasks may not be automated.
- Individual performers will need to be able to automate their own tasks. Business areas should integrate automated work flows, where one business unit receives another business unit's automated output as its input.

2.2.14 Give before receiving

Statement
Give before receiving
Rationale
Giving is the only way to establish a real relationship and a lasting connection. Focus solely on what you can get out of the connection and you will never make meaningful, mutually beneficial connections and a sustainable business
Implications
Invest time and money in all stakeholder relations.

2.2.15 Information Management is Everybody's Business

Statement
All organisations in the enterprise participate in information management decisions needed to accomplish business objectives.
Rationale
Information users are the key stakeholders, or customers, in the application of technology to address a business need. In order to ensure information management is aligned with the business, all organisations in the enterprise must be involved in all aspects of the information environment. The business experts from across the enterprise and the technical staff responsible for developing and sustaining the information environment need to come together as a team to jointly dene the goals and objectives of IT.
Implications

To operate as a team, every stakeholder, or customer, will need to accept responsibility for developing the information environment. Commitment of resources will be required to implement this principle.

2.2.16 IT Responsibility

Statement

The IT organisation is responsible and accountable for owning and implementing all IT processes and infrastructure that enable solutions to meet business-defined requirements for functionality, service levels, cost, and delivery timing. Decisions should always align back to the requirement of the Business.

Rationale

Effectively align expectations with business requirements and our overall capabilities so that all projects are cost-effective and can be completed in a timely manner. Efficient and effective solutions should have reasonable costs and clear benefits relative to the business proposition.

Implications

The IT function must define processes to manage business expectations and priorities. Projects must follow an established process to reduce costs and to ensure the project has a timely completion. Data, information, and technology should be integrated to provide quality solutions and to maximise results.

2.2.17 Ease-of-Use

Statement

Applications are easy to use for end-users and administrators.

Rationale

The more a user has to understand the underlying technology, the less productive that user is. Avoid mistakes due to difficult comprehension interaction with a system. Most of the knowledge required to operate one system will be similar to others. Using an application should be as intuitive as driving a different car.

Implications

- The underlying technology is transparent to users.
- Training is kept to a minimum, and the risk of using a system improperly is low.
- Default (de-facto) GUI's are used for interacting with the system.
- No large user manual is needed.

2.2.18 Be Collaborative

Statement

Be Collaborative

Rationale

The saying: "If you want to go fast, go alone. If you want to go far, go together." is attributed to an African proverb, but could easily be a mantra for technology-enabled development projects. The principle: Be Collaborative suggests

strategies for leveraging and contributing to a broader commons of resource, action, and knowledge to extend the impact of development interventions.

Implications

- Engage diverse expertise across disciplines and industries at all stages.
- Work across sector silos to create coordinated and more holistic approaches.
- Document work, results, processes and best practices and share them widely.
- Publish materials under a Creative Commons license by default, with strong rationale if another licensing approach is taken.

2.2.19 Business continuity

Statement

Business continuity of Corporate activities must be maintained, despite IT interruptions.

Rationale

Hardware failures, natural disasters, and lack of data integrity must not interrupt business activities.

Implications

- Recoverability, redundancy, and maintenance must be approached at inception. - Applications must be assessed regarding criticality and impact on the company's mission to determine which continuity level is required and which corresponding recovery plan must be implemented. - A business continuity plan must be present/developed.

2.2.20 Business Principle

Statement

These architectural principles will apply to all organisational units within the enterprise.

Rationale

The only way the University will be able to provide a consistent and measurable level of appropriately robust, reliable, sustainable services and quality information to decision-makers, is if all stakeholders abide by the University's overarching principles for its technology, information and business architectures.

Implications

This fundamental principle will ensure inclusion, consistency, fairness and continual alignment to the business. Without this the management of our technologies, information and business processes would be quickly undermined. Business Partners engaging with the business will work to find accommodation between interested parties around any conflicts with a principle relevant to the proposal.

2.3 Business architecture templates

Creating a business architecture means doing real business research. However for a quality business architecture it make sense to make use of a draft template. From a template you can easily add, remove or change subjects that need special attention within your context. Also since architecture documents always should be created for a clear

business goal and to be used by different stakeholders, all quality documents SHOULD contain some default reusable text blocks.

- Basic business architecture template
- Stakeholder template
- Business Architecture. EU template/document that describes the product, service strategy, organizational, functional, process, information, and geographic aspects of the business architecture.
- Business Architecture Document Template. Typical Togaf(tm) like business architecture template.

2.4 Using business viewpoints

Viewpoints can provide benefit to address specific concerns for certain stakeholders. Below a list of most used viewpoint within a business architecture:

- Motivation view: describes what the business achieves for itself and its stakeholders (direct and indirect value).
- Capability view: describes how the business delivers direct and indirect value in response to the challenges of the environment.
- Activity view: describes the day-to-day behaviour of the business.
- Responsibility view: captures the relationships between individuals and organizations in terms of responsibilities and commitments. These relationships and organizational interfaces may be represented as business services.

2.5 Help for creating a business architecture

- Help Guide for creating a business architecture when dealing with SOA/Integration. (CC License)

CHAPTER 3

Data Architecture

Every business, small or large SHOULD have a data architecture. In the core a data architecture gives the overview and insights into the only one real value of your IT: Information. A data architecture gives overviews, visuals and describes e.g.:

- What data is used where and how.
- Who owns what data.
- How is information created from data sources.
- Internal and external data sources used.
- Privacy & Security aspects of data (so be sure to have an data owner)
- What structures, objects and relations are the core of your information model?

3.1 Tools for creating a data architecture

To speed up the process for creating a data architecture you SHOULD use tools. Below is a selection of tools that will speed up the process of creating your data architecture. Be aware that data modelling and database design are two very different activities. The emphase for a data architecture SHOULD be on the conceptual and logical data models. In general physical data models are related to the sql or nosql storage engine used, in combination with scalability(performance) and security requirements.

- *Data Principles*.(See the data principles section in this book.) Using data principles saves you time and cost. Especially in the long term. Selecting data principles you need in your project of a solid collection gives you a head start.
- Protégé. Protégé is a free, open source ontology editor and knowledge-base framework. This open-source ontology editor and framework can be used for for building intelligent systems. The tool is developed by the Stanford Center for Biomedical Informatics Research and has a large and active community of users and developers.

- Archi. Archi™ GUI tool for creating an architecture, using the ArchiMate modelling™ language. Since Archi is targeted to all architecture aspects, this tool is usuable for creating conceptual, logical and physical data models too.
- WWW SQL Designer. This tool allows you to draw and create database schemas (E-R diagrams) directly in browser. A physical data model (sql) can also be imported and adjusted visually.
- MySQL Workbench. MySQL Workbench enables a DBA, developer, or data architect to visually design, model, generate, and manage databases. It includes everything a data modeler needs for creating complex ER models, forward and reverse engineering, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort.
- JSON Schema. Nice short book that gives you help with creating your JSON Schema.
- DrawIO. This online FOSS program can be used to create all thinkable data diagrams. It's a very advanced, but still a simple to use program. The source can be found on https://github.com/jgraph/drawio if you want to host it yourself.

CHAPTER 4

Data Principles

Since IT systems are always working on data. You MUST define some solid data principles. These principles will help you design the system and avoid complexity.

Below are some examples.

4.1 Timely

Statement: Data is made available as quickly as necessary to preserve the value of the data.

4.2 Machine processable

Statement: Data is reasonably structured to allow automated processing.

4.3 Primary data

Statement: Data is as collected at the source, with the highest possible level of granularity, not in aggregate or modified forms. Rationale

Statement: Data is available to the widest range of users for the widest range of purposes. Rationale

CHAPTER 5

Application Architecture

The application architecture describes the application components of a system. Part of the application architecture is providing high level insights in the software building blocks, services and micro services that are part of the application landscape.

5.1 Tools for creating an application architecture

Many OSS tools for creating an application are targeted on creating software code. Although an IT architect should be able to code, most of the time architects are more concerned with designing building blocks, interfaces and implementation constrains.

Using the following tools for creating an application architecture saves time and increases the quality of your application architecture:

- UMLet. UMLet is an open-source UML tool with a simple user interface: draw UML diagrams fast, export diagrams to eps, pdf, jpg, svg, and clipboard, share diagrams using Eclipse, and create new, custom UML elements.
- Papyrus Modeling environment. Papyrus is an industrial-grade open source Model-Based Engineering tool. Papyrus is the base platform for several industrial modelling tools. Papyrus can be used as graphical modelling tool, but aims to support MDA (Model Driven Architecture) completely.
- Archi. Since Archi(tm) is a real TOGAF based architecture and desing tool, Archi is a good solution for creation of an enterprise architecture. With Archi you can use the Archimate(tm) modeling language. Since Archimate is a language for Enterprise Architecture modeling, it is not suited and designed for creating more detailed overviews and designs.
- Open ModelSphere. Open ModelSphere is a powerful data, process and UML modeling tool.
- Modelio. Modelio is a modelling environment, supporting a wide range of models and diagrams, and providing model assistance and consistency checking features. Modelio can also generate Java code.

Creating an application architecture will lead to creating internal and external interfaces. APIs form the connecting glue between modern applications. Creating good interfaces is a MUST for every good architecture. Below some open tools that can help to speed up this step:

- API Blueprint. API Blueprint is all about the design-first philosophy. API Blueprint itself is OSS and has a growing base of OSS tools based on the spec. API Blueprint supports the complete chain for interface development (design, test, create etc).
- Swagger based tools. Swagger is a simple yet powerful representation of your RESTful API. Swagger has a large ecosystem of OSS tools that assist in creating, testing and documenting APIs.
- RAML based tools. RESTful API Modeling Language (RAML) makes it easy to manage the whole API lifecycle from design to sharing. The RAML specification is designed for design APIs better and faster.

5.2 Application Architecture Templates

To speed up the process of creating an application architecture you SHOULD make use of one of the templates below.

- Architecture description template for use with ISO/IEC/IEEE 42010:2011. (MSWord template or PDF)
- SAD (Software Architecture Document)
- Software Engineering Institute's(SEI) Architecture template (MSWord)
- Solutions Architecture Template (U.S. Government, HUD) (MSWord)

CHAPTER 6

Software Architecture

Architecture SHOULD be simple and have a clear purpose. Good software architecture methods are hard to find. Often it is too high level (TOGAF based with archimate diagrams). Often with no or very little value for software developers and managers. Or software architecture diagrams are a random collection of code-snippets and UML diagrams. Often only created since creating documentation was requested.

When you create an application, its architecture must do two things:

- Provide an easy way to communicate to **ALL** stakeholders.
- Enable various stakeholders to see different levels of granularity.
- Make sure that even you understand the working and key dependencies a few years later!

6.1 The C4 model for visualising software architecture

C4 stands for context, containers, components, and code . It is a great way to create a good software architecture. And it is an open method. A full training can be found on: https://s3.amazonaws.com/static.codingthearchitecture.com/presentations/bcsspa2019-the-lost-art-of-software-design.pdf

The C4 model was created as a way to help software development teams describe and communicate software architecture, both during up-front design sessions and when retrospectively documenting an existing codebase. It's a way to create maps of your code, at various levels of detail, in the same way you would use something like Google Maps to zoom in and out of an area you are interested in.

By using the C4 model for software architecture, you can capture just enough architecture to communicate to stakeholders and enable team understanding without having to define details that will change as your architecture changes.

Source and more info on: https://c4model.com/

6.2 arc42 software architectures

- arc42 is based on practical experience of many systems in various domains, from information and web systems, real-time and embedded to business intelligence and data warehouses.
- arc42 provides a template for documentation and communication of software and system architectures.
- arc42 supports arbitrary technologies and tools.
- arc42 is completely process-agnostic, and especially well-suited for lean and agile development approaches.
- arc42 is open-source and can be used free of charge, in commercial and private situations.

All arc42 templates can be found on: https://github.com/arc42/arc42-template Or check the arc42 download page on: https://arc42.org/download

6.3 The Bounded Context Canvas

Great FOSS tool that helps with your software design. Source code and instruction on: https://github.com/ddd-crew/bounded-context-canvas

The Bounded Context Canvas is a collaborative tool for designing and documenting the design of a single bounded context.

A bounded context is a sub-system in a software architecture aligned to a part of your domain.

The canvas guides you through the process of designing a bounded context by requiring you to consider and make choices about the key elements of its design, from naming to responsibilities, to its public interface and dependencies.

CHAPTER 7

Software Development

Software development is a process that can make or break your product quality. This section oulines some guidelines that can serve as input for your software development process. Smart guidelines that are transparent improves software.

7.1 Version control

Why would you use version control software and hosting (such as GitHub)?

- Easier to collaborate Version control makes it easier to work on the same code simultaneously, while everyone still has a well defined version of the software (in contrast to a google-docs or shared file system type of system). Moreover, version control hosting websites such as Github provide way to communicate in a more structed way, such as in code reviews, about commits and about issues.
- Reproducibility By using version control, you never lose previous versions of the software. This also gives you a log of changes and allows you to understand what happened.
- Backup Version control is usually pushed to an external a shared server, which immediately provides a backup.
- Integration Version control software and host makes it more easy to integrate with other software that support modern software development, such as testing (continuous integration ,automatically run tests, build documentation, check code style, integration with bug-tracker, code review infrastructure, comment on code).

It is highly recommended to start using version control on day one of the project.

7.2 Choose one branching model

A branching model describes how the project deals with different versions of the codebase, like releases and various development versions, and how to accept code contributions. Make the choice explicit in the contribution guidelines, and link to documentation on how to get started with it. Our default choice is GitHub flow branching model

GitHub flow is a very simple and sane branching model. It supports collaboration and is based on pull requests, therefore relies heavily on GitHub. The Pro Git book describes in detail the workflow of collaboration on the project

with use of git branches, forks and GitHub in Contributing to a Project chapter. Other more complicated models could be used if necessary, but you should strive for simplicity.

Learning a new branching model should not stand in the way of contributions.

7.3 Repositories should be public

Use SHOULD embrace an open architecture model and way of working. This means a open code repository.

A public code repository has several benefits:

- It makes your code findable. So it can be improved by anyone.
- It shows your code to world, allowing (re)use and enables.
- It enables collaboration.

Unless code cannot be open (e.g. when working with commercial partners, or when there are competitiveness issues) it should be in a public online repository. In case the code uses data that cannot be open, an engineer should try to keep sensitive parts outside of the main codebase.

7.4 Code Quality

There are several ways to improve software quality and find bugs quickly and easily. By following a set of conventions, code will look 'cleaner' and be more understandable. It will also help spot syntax errors and other errors early, without having to run or compile all the time.

7.5 Coding style

A coding style gives guidance on those parts of programming that are irrelevant to the compiler or interpreter. For instance, what do you call your variables? do you use spaces or tabs for indentation? Where do you put comments? .

There are open style guides for all programming languages. Choose an open standard and make clear what you use.

7.6 Software quality improvement tools

There are several web services that analyse code and make the quality of the code visible.

Code quality analysis services are web applications which have the following features:

- Automatically analyse your code after a Github push
- Usually free for open source projects
- Most supports multiple programming languages, but not every language will have the same level of features
- Grade or score for the quality of all of the code in the repository
- List of issues with the code, grouped by severity
- Drill down to location of issue
- Default list of checks which the service provider finds the best practice
- Can be configured to make the list of checks more strict or relaxed

- Can be configured to ignore files or extensions
- Can read configuration file from repository
- Tracks issues over time and send alerts when quality deteriorates
- Optionally reports on code coverage generated by a CI build

7.7 Software Releases

Releases are a way to mark or point to a particular milestone in software development. This is useful for users and collaborators, e.g. I found a bug running version x. For publications that refer to software, refering to a specific release enhances the reproducability.

Apache foundation describes their release policy.

Release cycles will depend on the project specifics, but in general we encourage quick agile development: release early and often Semantic versioning.

Releases are identified by a version number. Semantic Versioning (semver.org) is the most accepted and used way to add numbers to software versions. It is a way of communicating impact of changes in the software on users.

A version number consists of three numbers: major, minor, and patch, separated by a dot: 2.0.0. After some changes to the code, you would do a new release, and increment the version number. Increment the:

- MAJOR version when you make incompatible API changes,
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes.

7.8 API Design Guides

APIs are important and good API design guides help you.

- An open HTTP API guide can be found on: https://geemus.gitbooks.io/http-api-design/content/en/
- Microsoft REST API Guidelines: https://github.com/Microsoft/api-guidelines/blob/master/Guidelines.md

CHAPTER 8

Technology Infrastructure (TI) Architecture

A Technology Infrastructure (TI) Architecture is all about the hard IT. Hosting, capacity, connectivity, monitoring and operations. Having a flexible TI architecture serves all business processes.

Creating a simple overview of your systems is key for understanding. Of course you can use the great Archi to create Archimate TI views. But Archimate is not designed for the needs to get a good understandable overview of your TI components. Simple is better, so think of using the following tools for visualisation of your TI Architecture:

- diagrams.net General easy to use drawing software. Diagrams.net is open source, online, desktop and container deployable diagramming software. It has many default shapes and templates to create a quick system overview or network topology view quick.
- Diagrams. Diagrams lets you draw the cloud system architecture in Python code. Also easy to integrate with Jupyternotebooks. Diagrams was created to program your Cloud system documentation. So it support shaped for many Cloud Providers.

When creating a Technology Infrastructure Architecture you SHOULD calculate network usage. This to prevent performance issues and availability issues. Think of calculating:

- Network usage per user.
- IOPS usage per user and peak usage. This to check if your TI hosting provider meets your requirements.
- Network bandwidth peak usage.
- Average and peak network usage for some typical use cases. Since you do not live in a perfect world you will be forced to use assumptions. But document it and improve the calculations when major changes or incidents occur.

For systems that are not using PaaS Cloud Services a lot of attention, time and effort is needed for the TI part of the system. Below some TI templates that will make with creating the technical solution.

TI Templates:

- Technical Architecture (MSWORD). Template used at US Florida Department of Transportation.
- High-Level Technical Design (MSWord). HLD Template used at US Centers for Medicare & Medicaid Services.
- System Design Document (MSWord). Source US CMS.

- Service Design & Transition (MSWord). Source EU FitSM program.
- IT Service Capacity Plan (MSWord). Source EU FitSM program.

8.1 Abstraction Tools

To be flexible and remain portable open interfaces you SHOULD think of using abstraction layers. Abstraction layers on TI level can make your implementation easier. But the trade-off is of course that using an extra layer adds another dependency.

Good TI abstraction Tools to be used are:

- Apache Libcloud. Apache Libcloud is a Python library which hides differences between different cloud provider APIs and allows you to manage different cloud resources through a unified and easy to use API. Great documentation which offers standardization over the fuzzy Cloud Terms are included, check: https://libcloud.readthedocs.io/en/latest/index.html

CHAPTER 9

Quality and Risk Management

Quality is and will always the number one aspects. This section provides various tools and templates for managing your business IT quality aspects.

9.1 Tools for Architecture QA processes

- *Architecture checklists*.
- *Architecture Documentation Checklist*
- *Use proven IT standards for requirements*
- Non Functional Requirements (NFR list).
- *Overview of ISO 25010*

9.2 Tools for a better IT Architecture

An architecture checklist helps in the governance process. Architecture checklists can become long, complex and time consuming in usage. However the aim with this architecture checklist is that it will help you and all your stakeholders involved in a simple way when you are dealing with architecture quality and risk aspects.

This architecture checklist is composed out of some critical questions that all relate back to the main goal of doing architecture in the first place.

9.3 Simple architecture checklist questions

- Is your main goal covered and reached with this architecture?
- Does the architecture address operability?
- Does the architecture address the following quality attributes:

  - performance
  - availability
  - maintainability
  - modifiability
  - security
  - privacy
  - testability
  - operability
  - flexibility

- Does the architecture address and use principles? E.g. business principles.
- Can implementation risks easily be derived out of your architecture deliverables?
- Are architecture reviews done in a structured way? Architecture reviews SHOULD be performed to increase quality, control costs and reduce risks.
- Is it clear what assumptions are used for your architecture? (Take explicit and implicit assumptions into account!)

9.4 Architecture Documentation Checklist

When creating business IT documentation (e.g. IT architectures/designs) following these rules will help for producing sound documentation:

- Documentation should be written from the point of view of the reader, not the writer.
- IT Documentation should be organized for ease of reference, not ease of reading. A document may be read from cover to cover at most once, and probably never. But a document is likely to be referenced hundreds or thousands of times.
- Avoid repetition. Each kind of information should be recorded in exactly one place. This makes documentation easier to use and much easier to change as it evolves. It also avoids confusion.
- Avoid unintentional ambiguity. In some sense, the point of architecture is to be ambiguous. However unplanned ambiguity is when documentation can be interpreted in more than one way, and at least one of those ways is incorrect. A well-defined notation with precise semantics goes a long way toward eliminating whole classes of linguistic ambiguity from a document. This is one area where architecture description languages help a great deal, but using a formal language isn't always necessary.
- Standardize your documentation. Use always the same templates for the same documents within your organization. Even better: Make use of de-facto standardizations that also make sense for people outside your organization, since you will be working with many people outside your organization during the life cycle of your product. A standard organization offers many benefits. It helps the reader navigate the document and find specific information quickly.
- Record rationale. So document decisions made. Recording rationale will save you enormous time in the long run, although it requires discipline to record in the heat of the moment. It will prevent the same discussions over and over again and everyone knows why the chosen path is taken.
- Keep it current. Documentation that is out of date, does not reflect truth, and does not obey its own rules for form and internal consistency will not be used. Documentation that is kept current and accurate will be used.

- Review documentation for fitness of purpose. Only the intended users of a document will be able to tell if it contains the right information presented in right way.

9.5 IT standards

Requirement : Business Process Models must be based on BPMN 2.0

Description : A standard Business Process Model and Notation (BPMN) will provide businesses with the capability of understanding their internal business procedures in a graphical notation and will give organizations the ability to communicate these procedures in a standard manner. Furthermore, the graphical notation will facilitate the understanding of the performance collaborations and business transactions between the organizations. See: http://www.bpmn.org/

Requirement : Digital Signature Standard (DSS)

Description : This Standard defines methods for digital signature generation that can be used for the protection of binary data (commonly called a message), and for the verification and validation of those digital signatures. Three techniques are approved.

1. The Digital Signature Algorithm (DSA) is specified in this Standard. The specification includes criteria for the generation of domain parameters, for the generation of public and private key pairs, and for the generation and verification of digital signatures.
2. The RSA digital signature algorithm is specified in American National Standard (ANS) X9.31 and Public Key Cryptography Standard (PKCS) #1. FIPS 186-4 approves the use of implementations of either or both of these standards and specifies additional requirements.
3. The Elliptic Curve Digital Signature Algorithm (ECDSA) is specified in ANS X9.62. FIPS 186-4 approves the use of ECDSA and specifies additional requirements. Recommended elliptic curves for Federal Government use are provided herein. See: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

Requirement : FIPS PUB 198-1:The Keyed-Hash Message Authentication Code (HMAC)

Description : Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that provide such integrity checks based on a secret key are usually called message authentication codes (MACs). Typically, message authentication codes are used between two parties that share a secret key in order to authenticate information transmitted between these parties. This Standard defines a MAC that uses a cryptographic hash function in conjunction with a secret key. This mechanism is called HMAC [HMAC]. HMAC shall use an Approved cryptographic hash function [FIPS180-3]. HMAC uses the secret key for the calculation and verification of the MACs. See:http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

Requirement : Implementation must comply to the 'Do Not Track Compliance Policy'

Description : The website (DNS) is compliant with the privacy-friendly Do Not Track (DNT) Policy of the EFF.org organization. Reference: https://www.eff.org/dnt-policy W3C reference:http://www.w3.org/TR/tracking-dnt/

Requirement : JSON-LD

Description : JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases such as CouchDB and MongoDB. See: http://json-ld.org/

Requirement : Payment Card Industry (PCI) DSS v3.1

Description : The design/implementation must be compliant with the Payment Card Industry (PCI) Data Security Standard version 3.1. PCI DSS provides a baseline of technical and operational requirements designed to protect account data. PCI DSS applies to all entities involved in payment card processing — including merchants, processors, acquirers, issuers, and service providers. PCI DSS also applies to all other entities that

store, process or transmit cardholder data (CHD) and/or sensitive authentication data (SAD). Detailed spec on: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf

Requirement : Secure Hash Standard (SHS)- FIPS PUB 180

Description : This Standard specifies secure hash algorithms, SHA -1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256. All of the algorithms are iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. These algorithms enable the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest. The digests are used to detect whether messages have been changed since the digests were generated.See: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

Requirement : Semantic Versioning

Description : All version numbering must match Semver 2.0. Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format. See: http://semver.org/

Requirement : Web Hypertext Application Technology Working Group (WHATWG)

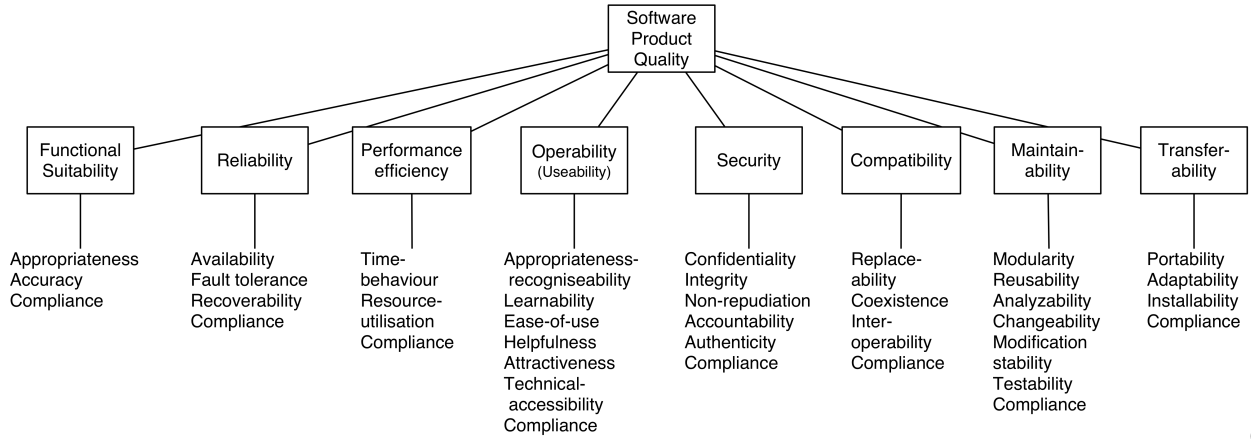
Description : The Web Hypertext Application Technology Working Group https://whatwg.org/ is a growing community of people interested in evolving the Web. It focuses primarily on the development of HTML and APIs needed for Web applications. The WHATWG was founded by individuals of Apple, the Mozilla Foundation, and Opera Software in 2004, after a W3C workshop. Apple, Mozilla and Opera were becoming increasingly concerned about the W3C's direction with XHTML, lack of interest in HTML and apparent disregard for the needs of real-world authors. So, in response, these organisations set out with a mission to address these concerns and the Web Hypertext Application Technology Working Group was born. See: https://whatwg.org/

9.6 Collections of Business IT standards and policies

Creating a policies should be done smart. So reuse existing open standard and policies documentation saves time and improves the quality of your context specific standard.

- NASA Technical Standards
- Principles and guidelines for how to write code and work together at Our Machinery

9.7 Short overview of ISO 25010



Overview

ISO Qualitiy Standard(25010)

The material of ISO is unfortunate not open. But since quality matters and ISO 25010 is used heavily for managing quality aspects within business IT systems a short overview.

9.7.1 Functionality

Functionality: A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

- Functional suitability: Degree to which a product or system provides functions that meet stated and implied needs when used underspecified conditions.
- Functional completeness : Degree to which the set of functions covers all the specified tasks and user objectives.
- Functional correctness : Degree to which a product or system provides the correct results with the needed degree of precision.
- Functional appropriateness : Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

9.7.2 Reliability

Reliability: A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time. Attributes:

- Reliability:Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.
- Maturity : Degree to which a system, product or component meets needs for reliability under normal operation.
- Availability : Degree to which a system, product or component is operational and accessible when required for use.
- Fault tolerance : Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
- Recoverability : Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

9.7.3 Performance Efficiency

A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

- Performance efficiency: Performance relative to the amount of resources used under stated conditions.
- Time behaviour : Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
- Resource utilization : Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
- Capacity : Degree to which the maximum limits of a product or system parameter meet requirements.

9.7.4 Compatibility

Compatibility:Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

- Co-existence : Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
- Interoperability : Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

9.7.5 Usability

Usability:Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

- Appropriateness recognizability : Degree to which users can recognize whether a product or system is appropriate for their needs.
- Learnability : Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
- Operability : Degree to which a product or system has attributes that make it easy to operate and control.
- User error protection : Degree to which a system protects users against making errors.
- User interface aesthetics : Degree to which a user interface enables pleasing and satisfying interaction for the user.
- Accessibility : Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

9.7.6 Security

Security:Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

- Confidentiality : Degree to which a product or system ensures that data are accessible only to those authorized to have access.
- Integrity : Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

- Non-repudiation : Degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
- Accountability : Degree to which the actions of an entity can be traced uniquely to the entity.
- Confidentiality : Degree to which a product or system ensures that data are accessible only to those authorized to have access.
- Authenticity : Degree to which the identity of a subject or resource can be proved to be the one claimed.

9.7.7 Maintainability

- Maintainability:Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.
- Modularity : Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
- Reusability :Degree to which an asset can be used in more than one system, or in building other assets.
- Analysability : Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
- Modifiability : Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
- Testability : Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

9.7.8 Transferability

- Portability:Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another
- Adaptability : Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
- Installability : Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or un-installed in a specified environment.
- Replaceability : Degree to which a product can replace another specified software product for the same purpose in the same environment.

Note that when you do a critical review on ISO20510 you will find that the following subjects are not explicit mentioned within the ISO25010 standard:

- Functional requirements
- Compliance (e.g. with laws, standards) requirements
- Documentation, Support and Training requirements and of course:

Functional requirements and compliance can be covered under the category ‘Functionality’. Documentation, support and training is can be covered by the categories ‘Maintainability’ and ‘Usability (operability)’.

Process quality aspects like:

- Project Timing requirements
- Project Budget requirements

Are not within scope of the ISO product qualities, but process quality aspects are very important to create good architecture products.

CHAPTER 10

Architecture References

10.1 Architecture magazines

Architectuur en Governance magazine

Architecture & Governance Magazine is a publication of Troux Technologies.

DoDAF V2.02 Journal

The DoDAF Journal is a community of interest based discussion board. The Journal includes descriptions of best practices.

Journal of Enterprise Architecture (AOGEA Journal)

The Journal of Enterprise Architecture (JEA) is published quarterly by the Association of Enterprise Architects. It is a peer-reviewed international quarterly publication for the Enterprise Architecture community. Standard magazine for all TOGAF certified architects...

Journal of Information Architecture

The Journal of Information Architecture is an international peer-reviewed scholarly journal. Its aim is to facilitate the systematic development of the scientific body of knowledge in the field of information architecture.

The Architecture Journal (Microsoft)

The Architecture Journal is an independent platform for free thinkers and practitioners of IT architecture. New editions are issued quarterly with articles designed to offer perspective

XR Magazine

XR Magazine is Dutch online platform and magazine for managers and architecten. (Language dutch).

10.2 Architecture Methods

Archimate

Archimate 1.0 version: The ArchiMate enterprise architecture modeling language offers an integrated architectural approach that describes and visualizes the different architecture domains and their underlying relations and dependencies. In a short time

ARIS (Architecture of Integrated Information Systems)

Long time a industry default standard. Not anymore however -) ARIS is now more a tool of Software AG's. ARIS Business Process Analysis Platform is ideal for organizations that want to document

Cloud Computing Patterns

Describing good solutions to reoccurring problems as patterns is a common practice in research and industry alike. While the development of cloud applications faces many new challenges

DYA

Primary Dutch EA method. Used and owned by Sogetti.

DYA Infrastructure Repository

DYA|Infrastructure brings business agility

EAM Pattern Catalog

The objective of the EAM Pattern Catalog is to complement existing Enterprise Architecture (EA) management frameworks

EAM Pattern Catalog

The objective of the EAM Pattern Catalog is to complement existing enterprise architecture (EA) management frameworks, which provide a holistic and generic view on the problem of EA management, and to provide additional detail and guidance needed to systematically establish EA management in a step-wise fashion within an enterprise.

FSAM (Federal Segment Architecture Methodology)

The Architecture and Infrastructure Committee released the Federal Segment Architecture Methodology (FSAM) v1.0 in December 2008. The FSAM features easy-to-use templates that expedite architecture development and maximize architecture use. The FSAM includes step by step guidance based on business-driven

FSAM (Federal Segment Architecture Methodology)

The Architecture and Infrastructure Committee released the Federal Segment Architecture Methodology (FSAM) v1.0 in December 2008. The FSAM features easy-to-use templates that expedite architecture development and maximize architecture use. The FSAM includes step by step guidance based on business-driven

GERAM (Generalised Enterprise Reference Architecture and Methodology)

The scope of GERAM encompasses all knowledge needed for enterprise engineering / integration. Thus GERAM is defined through a pragmatic approach providing a generalised framework for describing the components needed in all types of enterprise engineering/enterprise integration processes.

ITANA architecture library

ITANA library for architects. ITANA is focused on developing the skills, tools and a suite of resources to assist institutions with their enterprise, business and technical architectural needs. Very useful collection of documents, tools and more for architects!

TOGAF

TOGAF is a framework - a detailed method and a set of supporting tools - for developing an enterprise architecture.

10.3 Architecture organizations

Association of Enterprise Architects (AEA)

The Association of Enterprise Architects (AEA) is the definitive professional organization for Enterprise Architects. Our goals are to increase job opportunities for all members and increase their market value by advancing professional excellence

BIAN (Banking Industry Architecture Network)

The BIAN model is a Service Oriented Architecture with consistent service definitions, level of detail and boundaries. This makes it easier to choose and integrate commercially available products of different vendors.

Center for Enterprise Architecture (EA)

The purpose of the Center for Enterprise Architecture is to gather intellectual resources across Penn State to address open and important research concerns and questions that span the design

Distributed Management Task Force

DMTF's Systems Management Architecture for Server Hardware (SMASH) standard is a suite of specifications that deliver industry standard semantics, protocols and profiles to make data center resource management interoperable.

Enterprise Architecture Center of Excellence (EACOE)

The mission of the Enterprise Architecture Center of Excellence (EACOE) is to be the definitive source for all aspects of Enterprise Architecture

IASA (Global IT Architects Association)

Iasa is the premier association focused on the architecture profession through the advancement of best practices and education while delivering programs and services to IT architects of all levels around the world. Our mission is to make IT architecture the most recognized profession in the world.

ITAG – Information Technology Architecture Group

The MIT Enterprise Architecture Guide (EAG) documents MIT's architectural principles and goals, the current state of MIT's enterprise architecture, and a future state architectural vision. The EAG also includes information regarding the ITAG architecture review process. Since this document serves to inform developers about available enterprise tools and services, we expect the EAG will be useful to enterprise system developers across the institute.

NAF (Nederlands Architectuur Forum)

Dutch organization for promoting working with the IT architecture discipline. (EA driven)

NGI architectuur

Dutch department under NGI-NGN (Dutch non-profit IT organization for IT professionals). This architecture group works with couple with other Dutch EA groups when organizing meetings and creating workgroups.

Software Engineering Institute (SEI)

The Software Engineering Institute (SEI) is a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense (DoD). It is operated by Carnegie Mellon University. The SEI offers many (free) publication on all aspects of software architecture.

The Information Architecture Institute

The Information Architecture Institute is a 501(c)6 professional organization

The Jericho Forum

The Jericho Forum

The MOD research group

Research group on model-driven software engineering at SINTEF The MOD research group is part of the Department of Networked Systems and Services within the Division of Information and Communication Technology. This group is located in Oslo

The Open Data Institute (ODI)

The Open Data Institute works with companies and governments to build an open, trustworthy data ecosystem, where people can make better decisions using data and manage any harmful impacts. The ODI produces guides, standards and more to make the use of data simpler.

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 400 member organizations

Via Nova Architectura

Dutch EA platform. Community site for all Dutch Architects. Also one the main communication channel of other Dutch architecture communities like NAF, KNVI etc.

10.4 Architecture Patterns

Cloud computing patterns

CloudPatterns.org is a community site dedicated to documenting a master patterns catalog comprised of design patterns that capture and modularize technology-centric solutions distinct or relevant to modern-day cloud computing platforms and business-centric cloud technology architectures. Part of this catalog is comprised of compound patterns that tackle contemporary cloud delivery and deployment models (such as public cloud, IaaS, etc.) and decompose them into sets of co-existent patterns that establish core and optional feature sets provided by these environments.

Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications patterns

This site contains twenty-four design patterns and ten related guidance topics, this guide articulates the benefit of applying patterns by showing how each piece can fit into the big picture of cloud application architectures. It also discusses the benefits and considerations for each pattern. Most of the patterns have code samples or snippets that show how to implement the patterns using the features of Microsoft Azure. However the majority of topics described in this guide are equally relevant to all kinds of distributed systems, whether hosted on Azure or on other cloud platforms. Patterns can also be downloaded as ePUB or PDF (or ordered as hard copy book).

10.5 Cloud

Cloud computing patterns

CloudPatterns.org is a community site dedicated to documenting a master patterns catalog comprised of design patterns that capture and modularize technology-centric solutions distinct or relevant to modern-day cloud computing platforms and business-centric cloud technology architectures. Part of this catalog is comprised of compound patterns that tackle contemporary cloud delivery and deployment models (such as public cloud, IaaS, etc.) and decompose them into sets of co-existent patterns that establish core and optional feature sets provided by these environments.

Eucalyptus Cloud Reference Architectures

HPE Helion Eucalyptus, hereafter “Eucalyptus,” is an open source platform that allows you to build an Amazon Web Services (AW S)-compatible, on-premise cloud. It is designed to run on commodity hardware and provide an implementation of popular AWS-compatible services, such as EC2 (Elastic Compute Cloud) and Auto Scaling.

IBM Cloud Computing Reference Architecture 2.0

Currently adopted by the Open Group

NIST Cloud Computing Reference Architecture (Version 2)

NIST Cloud Computing Reference Architecture.

10.6 Example Architecture

Architecture of a reproducibility service

This architecture describes the relationship of a reproducibility service with other services from the context of scientific collaboration, publishing, and preservation. Together these services can be combined into a new system for transparent and reproducible scholarly publications.

10.7 Foundation Architectures

- The New European Interoperability Framework, https://ec.europa.eu/isa2/sites/isa/files/eif_brochure_final.pdf

EAM Pattern Catalog

The objective of the EAM Pattern Catalog is to complement existing enterprise architecture (EA) management frameworks, which provide a holistic and generic view on the problem of EA management, and to provide additional detail and guidance needed to systematically establish EA management in a step-wise fashion within an enterprise.

FSAM (Federal Segment Architecture Methodology)

The Architecture and Infrastructure Committee released the Federal Segment Architecture Methodology (FSAM) v1.0 in December 2008. The FSAM features easy-to-use templates that expedite architecture development and maximize architecture use. The FSAM includes step by step guidance based on business-driven

10.8 Guidelines

Google Site Reliability Engineering

This book is a collection of essays by one company, with a single common vision. Historically, companies have employed systems administrators to run complex computing systems.We apply the principles of computer science and engineering to the design and development of computing systems: generally, large distributed ones. Sometimes, our

task is writing the software for those systems alongside our product development counterparts; sometimes, our task is building all the additional pieces those systems need, like backups or load balancing, ideally so they can be reused across systems; and sometimes, our task is figuring out how to apply existing solutions to new problems.

Microsoft REST API Guidelines

The Microsoft REST API Guidelines, as a design principle, encourages application developers to have resources accessible to them via a RESTful HTTP interface. To provide the smoothest possible experience for developers on platforms following the Microsoft REST API Guidelines, REST APIs SHOULD follow consistent design guidelines to make using them easy and intuitive. cc-by-sa document created by Microsoft architects

10.9 Industry Architectures

Architecture of a reproducibility service

This architecture describes the relationship of a reproducibility service with other services from the context of scientific collaboration, publishing, and preservation. Together these services can be combined into a new system for transparent and reproducible scholarly publications.

BIAN (Banking Industry Architecture Network)

The BIAN model is a Service Oriented Architecture with consistent service definitions, level of detail and boundaries. This makes it easier to choose and integrate commercially available products of different vendors.

Distributed Management Task Force

DMTF's Systems Management Architecture for Server Hardware (SMASH) standard is a suite of specifications that deliver industry standard semantics, protocols and profiles to make data center resource management interoperable.

EURIDICE

EURIDICE is an EU funded project which deals with the development and implementation of new concepts in the area of intelligent Cargo.

EURIDICE

Overview of the EURIDICE architecture main concepts and components. (Link to all public architecture documents). EURIDICE Integrated Project Euridice is an Integrated Project funded by EU's Seventh Framework Programme ICT for Transport Area. The basic concept of Euridice is to build an information services platform centred on the individual cargo item and on its interaction with the surrounding environment and the user.

Rackspace Open Cloud reference architecture

Description of Rackspace cloud architectural configurations so you known how to use it for your business or personal project. (cc-by license)

10.10 Microservices

Designing microservices: Domain analysis

Part of set of articles. Good read before designing, using and building and running microservices architecture on Azure.

Microservices architecture style (Microsoft)

Nice overview for what, how and when to use microservices. And of course tailored also for Azure specific details.

Microsoft REST API Guidelines

The Microsoft REST API Guidelines, as a design principle, encourages application developers to have resources accessible to them via a RESTful HTTP interface. To provide the smoothest possible experience for developers on platforms following the Microsoft REST API Guidelines, REST APIs SHOULD follow consistent design guidelines to make using them easy and intuitive. cc-by-sa document created by Microsoft architects

10.11 Mobile

Mobile Security Reference Architecture

The Mobile Security Reference Architecture (MSRA) is a deliverable of the Digital Government Strategy (DGS). A key objective of the DGS is to procure and manage mobile devices, applications, and data in smart, secure, and affordable ways. The MSRA has been released by the Federal CIO Council and the Department of Homeland Security (DHS) to assist Federal Departments and Agencies (D/As) in the secure implementation of mobile solutions through their enterprise architectures

10.12 Principles

ITAG – Information Technology Architecture Group

The MIT Enterprise Architecture Guide (EAG) documents MIT's architectural principles and goals, the current state of MIT's enterprise architecture, and a future state architectural vision. The EAG also includes information regarding the ITAG architecture review process. Since this document serves to inform developers about available enterprise tools and services, we expect the EAG will be useful to enterprise system developers across the institute.

The Principles for Digital Development

The Principles for Digital Development find their roots in the efforts of individuals, development organizations, and donors alike who have called for a more concerted effort by donors and implementing partners to institutionalize lessons learned in the use of information and communication technologies (ICTs) in development projects.

U.S. Digital Services Playbook

U.S. Digital Services Playbook: The American people expect to interact with government through digital channels such as websites, email, and mobile applications. By building digital services that meet their needs, we can make the delivery of our policy and programs more effective.

10.13 Reference architectures

ATHENA Interoperability Framework (AIF)

The ATHENA Interoperability Framework (AIF) provides a compound framework and associated reference architecture for capturing the research elements and solutions to interoperability issues that address the problem in a holistic way by inter-relating relevant information from different perspectives of the enterprise.

Australian Government Architecture Reference Models

The AGA Reference Models provide a common language for Australian Government agencies so that their architectures can be described in a common and consistent manner. cc-by licensed material See also: http://www.finance.gov.au/policy-guides-procurement/australian-government-architecture-aga/aga-rm/

Cloud Computing Portability and Interoperability (Open Group)

Open Group document. This guide analyzes cloud computing portability and interoperability.

Cybersecurity Framework (NIST)

The Framework Core offers a way to take a high-level Security Framework.

Data Transfer Project

Architecture document of The Data Transfer Project. The DTP project is formed in 2017 to create an open-source, service-to-service data portability platform so that all individuals across the web could easily move their data between online service providers whenever they want. The contributors to the Data Transfer Project believe portability and interoperability are central to innovation.

Distributed Microservice Architecture with Docker

Master's Thesis focused on microservice architecture and Docker

DoDAF (Department of Defense Architecture Framework)

Reference architecture of the Department of Defense (US) The DoDAF Architecture Framework Version 2.02

EAM Pattern Catalog

The objective of the EAM Pattern Catalog is to complement existing Enterprise Architecture (EA) management frameworks

Eucalyptus Cloud Reference Architectures

HPE Helion Eucalyptus, hereafter "Eucalyptus," is an open source platform that allows you to build an Amazon Web Services (AW S)-compatible, on-premise cloud. It is designed to run on commodity hardware and provide an implementation of popular AWS-compatible services, such as EC2 (Elastic Compute Cloud) and Auto Scaling.

EURIDICE

EURIDICE is an EU funded project which deals with the development and implementation of new concepts in the area of intelligent Cargo.

European Interoperablity Reference Architecture(EIRA)

The European Interoperability Reference Architecture (EIRA©) is an architecture content metamodel defining the most salient architectural building blocks (ABBs) needed to build interoperable e-Government systems. The EIRA© provides a common terminology that can be used by people working for public administrations in various architecture and system development tasks. The EIRA© was created and is being maintained in the context of Action 2016.32 of the ISA^2 Programme. The EIRA uses (and extends) the ArchiMate language as a modelling notation and uses service orientation as an architectural style.

GEMMA (GEMeentelijk Model Architectuur)

(Dutch site) GEMMA 2.0 is een doorontwikkeling naar een architectuur die de gehele gemeentelijke informatievoorziening beschrijft, helpt bij het reduceren van de complexiteit van de informatievoorziening, bij het organiseren van samenwerkingsverbanden en het positioneren van functies in de cloud.

GERAM (Generalised Enterprise Reference Architecture and Methodology)

The scope of GERAM encompasses all knowledge needed for enterprise engineering / integration. Thus GERAM is defined through a pragmatic approach providing a generalised framework for describing the components needed in all types of enterprise engineering/enterprise integration processes.

IBM Cloud Computing Reference Architecture 2.0

Currently adopted by the Open Group

Interoperability Solutions for EU public administrations

Reference architecture documents for use and reuse developed within EU program. ISA's collaborative platform to find, reuse and share a wealth of ready-to-use interoperability solutions for eGovernment and best practices and discuss with your peers!

ITAG – Information Technology Architecture Group

The MIT Enterprise Architecture Guide (EAG) documents MIT's architectural principles and goals, the current state of MIT's enterprise architecture, and a future state architectural vision. The EAG also includes information regarding the ITAG architecture review process. Since this document serves to inform developers about available enterprise tools and services, we expect the EAG will be useful to enterprise system developers across the institute.

ITANA architecture library

ITANA library for architects. ITANA is focused on developing the skills, tools and a suite of resources to assist institutions with their enterprise, business and technical architectural needs. Very useful collection of documents, tools and more for architects!

ITSM Reference Architecture Framework

FitSM is a free and lightweight standards family aimed at facilitating service management in IT service provision, including federated scenarios.

Microservice Architecture Reference Architecture 2017 (RH)

Microservice REF architecture 2017 of RedHat. Tailored for Java/JBOS. But with good general principles. This reference architecture provides a thorough discussion on microservices, some of the factors that go into determining a client's needs, and cost to benefit parameters. After defining several potential modularity levels, this paper focuses on business-driven microservices and provides an implementation using JBoss EAP 7 (CC-BY-SA)

Microsoft Industry Reference Architecture for Banking (MIRA-B)

MIRA-B This 2012 Microsoft Industry Reference Architecture for Banking gives financial institutions a framework to ensure IT meets their strategic goals across channels and various customer needs.

Mobile Security Reference Architecture

The Mobile Security Reference Architecture (MSRA) is a deliverable of the Digital Government Strategy (DGS). A key objective of the DGS is to procure and manage mobile devices, applications, and data in smart, secure, and affordable ways. The MSRA has been released by the Federal CIO Council and the Department of Homeland Security (DHS) to assist Federal Departments and Agencies (D/As) in the secure implementation of mobile solutions through their enterprise architectures

NEXOF-RA

The overall ambition of NEXOF-RA is to deliver a Reference Architecture for the NESSI Open Service Framework (ranging from the infrastructure up to the interfaces with the end users) leveraging research in the area of service-based systems to consolidate and trigger innovation in service-oriented economies

NIH Enterprise Architecture Framework (National Institute of Health)

As a comprehensive framework the NIH Enterprise Architecture identifies how IT assets directly enable NIH

NIST Cloud Computing Reference Architecture (Version 2)

NIST Cloud Computing Reference Architecture.

NORA 3.0

Dutch Government Reference Architecture (version 3.0))

Open Security Architecture (OSA)

OSA distills the know-how of the security architecture community and provides readily usable patterns for your application. OSA shall be a free framework that is developed and owned by the community.

Oracle Enterprise Architecture Framework : Information Architecture Domain

The OEAF:Information Architecture. Oracle EA Information reference architecture. OEAF Domain consists of the following components: Data Realms, Capability Model

Oracle Reference Architecture Security Release 3.1

This document (2010) provides a reference architecture for designing an enterprise security framework. This framework supports the security needs of business solutions and helps to unify the disparate security resources commonly found in IT today. It offers security services that are critical to the integrity of modern distributed and service-oriented solutions, and beneficial to legacy systems as well.

Reference Architecture for Scalable Word Press Websites (on AWS)

Description of scalability and deployment options when using AWS for hosting of WorPress sites.

Reference Architecture Foundation for Service Oriented Architecture

This document specifies the OASIS Reference Architecture for Service Oriented Architecture. It follows from the concepts and relationships defined in the OASIS Reference Model for Service Oriented Architecture. While it remains abstract in nature

ROSA (Referentie Onderwijs Sector Architectuur)

(Dutch) Reference Architecture site for Education Sector.

SOA Reference Architecture

This specification presents a SOA Reference Architecture (SOA RA)

Software Assurance Maturity Model

The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization.

The Chromium Architecture (Google)

Complete architecture and design documentation of the Google Chromium Projects.

VMware Infrastructure Architecture Overview

VMware Infrastructure architecture Overview (PDF whitepaper).

VMware vSphere

VMware vSphere

10.14 Security architecture

Cybersecurity Framework (NIST)

The Framework Core offers a way to take a high-level Security Framework.

Mobile Security Reference Architecture

The Mobile Security Reference Architecture (MSRA) is a deliverable of the Digital Government Strategy (DGS). A key objective of the DGS is to procure and manage mobile devices, applications, and data in smart, secure, and affordable ways. The MSRA has been released by the Federal CIO Council and the Department of Homeland Security (DHS) to assist Federal Departments and Agencies (D/As) in the secure implementation of mobile solutions through their enterprise architectures

Open Security Architecture (OSA)

OSA distills the know-how of the security architecture community and provides readily usable patterns for your application. OSA shall be a free framework that is developed and owned by the community.

Open Web Application Security Project (OWASP)

The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software.

Oracle Reference Architecture Security Release 3.1

This document (2010) provides a reference architecture for designing an enterprise security framework. This framework supports the security needs of business solutions and helps to unify the disparate security resources commonly found in IT today. It offers security services that are critical to the integrity of modern distributed and service-oriented solutions, and beneficial to legacy systems as well.

Privacy Management Reference Model and Methodology (PMRM)

The Privacy Management Reference Model and Methodology (PMRM, pronounced "pim-rim") provides a model and a methodology for: · understanding and analyzing privacy policies and their privacy management requirements in defined use cases; and · selecting the technical services which must be implemented to support privacy controls. It is particularly relevant for use cases in which personal information (PI) flows across regulatory, policy, jurisdictional, and system boundaries.

SABSA (Sherwood Applied Business Security Architecture)

SABSA is a proven methodology for developing business-driven

Software Assurance Maturity Model

The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization.

10.15 Software Architecture

Distributed Microservice Architecture with Docker

Master's Thesis focused on microservice architecture and Docker

Microservice Architecture Reference Architecture 2017 (RH)

Microservice REF architecture 2017 of RedHat. Tailored for Java/JBOS. But with good general principles. This reference architecture provides a thorough discussion on microservices, some of the factors that go into determining a client's needs, and cost to benefit parameters. After defining several potential modularity levels, this paper focuses on business-driven microservices and provides an implementation using JBoss EAP 7 (CC-BY-SA)

10.16 Standards

Advancing Storage and Information Technology(SNIA)

SNIA is a not–for–profit global organization, made up of member companies spanning the global storage market. SNIA's mission is to lead the storage industry worldwide in developing and promoting standards, technologies,

Digital Signature Standard (DSS) (FIPS PUB 186 – 4)

This (US) Standard defines methods for digital signature generation that can be used for the protection of binary data (commonly called a message), and for the verification and validation of those digital signatures.

Distributed Management Task Force

DMTF's Systems Management Architecture for Server Hardware (SMASH) standard is a suite of specifications that deliver industry standard semantics, protocols and profiles to make data center resource management interoperable.

FIPS PUB 198-1:The Keyed-Hash Message Authentication Code (HMAC)

Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that provide such integrity checks based on a secret key are usually called message authentication codes (MACs). Typically, message authentication codes are used between two parties that share a secret key in order to authenticate information transmitted between these parties. This Standard defines a MAC that uses a cryptographic hash function in conjunction with a secret key. This mechanism is called HMAC [HMAC]. HMAC shall use an Approved cryptographic hash function [FIPS180-3]. HMAC uses the secret key for the calculation and verification of the MACs.

ITSM Reference Architecture Framework

FitSM is a free and lightweight standards family aimed at facilitating service management in IT service provision, including federated scenarios.

Semantic Versioning 2.0.0

In the world of software management there exists a dread place called "dependency hell." The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself, one day, in this pit of despair.

CHAPTER 11

NFR Capabilities

This appendix provides some a very few examples of NFRs. Having good NFRs is crucial for your solution. Asking users to provide NFRs is not the best way to success. More productive is to discuss NFRs with your users that you think are worth discussing. Some NFRs, e.g. for availability, MUST BE discussed since this has severe consequence for the solution and maintenance costs. Some NFRs e.g. for security MUST BE implemented and are not really suited for users discussions. Most of the time you will be confronted with bad NFRs for your architecture after realization. So you better do it right from the start.

Table 1: NFR Capabilities

Capability	Description	Tags
(network)Session lifetime is limited	Session lifetime is limited.	Security
A test specification for the system must be available	A test specification for the system must be available in order to perform test of the created system.	Documentation, NFR
Clock Identity Authentication and Authorization	Authentication refers to verifying the identity of the peer clock. Authorization, on the other hand, refers to verifying that the peer clock is permitted to play the role that it plays in the protocol. For example, some nodes may be permitted to be masters,while other nodes are only permitted to be slaves or TCs. Authentication is typically implemented by means of a cryptographic signature, allowing the verification of the identity of the sender. Authorization requires clocks to maintain a list of authorized clocks, or a "black list" of clocks that should be denied service or revoked.	Security
Data logging:Sensitive data is not logged in clear text by the application.	Sensitive data is not logged in clear text by the application.	NFR, Security
Database connections, passwords, keys, or other secrets are not stored in plain text.	Database connections, passwords, keys, or other secrets are not stored in plain text.	NFR, Security
Disaster Recovery	The solution will be configured to be split across the minimal two data centers where possible with failover from data center to data center in the event of a disaster, In addition, each data center needs to be able to run in a self sufficient manner should it become isolated from the other.	NFR
Documentation must be available in an open document format	All system documentation must be made available in open document format. System documentation is (not exhausted) operational manuals, code documentation, test specs and test reports, installation manuals.	Documentation, NFR
Encryption keys must be secured	Encryption keys must be secured.	Security
High Availability	All components should be configured in a high availability configuration to eliminate single points of failure, and minimize solution outages.	Availability
Maintainability	The system should allow for easy software upgrades with minimal outage. The outage should be restricted to no longer than one day, and allow for the use of a back up system for service continuity while the upgrade the taking place.	NFR
Maintainability	Any solution must be maintainable by the affected maintenance team, both initially and throughout its lifecycle. Unnecessary complexity in maintenance, such as by requiring additional / unusual skills or tools or having a complex solution design, adds risk to the solution's supportability and must be justified.	maintainability, NFR
Manageability	All solutions must be managed throughout their lifecycle, including startup, shutdown, backup, updates, security / permission changes, etc. Administrators and support personnel must be able to conduct such routine activities effectively in order to ensure that the solution does not incur excessive cost or experience unnecessary outages.	manageability, NFR
Minimize Footprint	Stack multiple components within single operating system instances where possible to minimize both the number of physical and virtual servers required to run the solution.	NFR
Privileged Accounts must not be used for	Privileged and super-user accounts (Administrator, root, etc.) must not be used for non-administrator activities. A secure mechanism to	Security

CHAPTER 12

Help

We encourage all information professionals (EA Architects, IT Architects, Information Architects,Application Architects, TI Architects,Business IT Consultants etc.) to help to improve this Architecture Playbook.

Join the team to:

- Review and correct the content. (Yes there are still too much typos)
- Add new content blocks
- Discuss the content so it gets better.

You can contribute using the following Github repository:

- https://github.com/nocomplexity/ArchitecturePlaybook

Licensing

When you submit text to which you hold the copyright, you agree to license it under:

- Creative Commons Attribution-ShareAlike 4.0 International License (“CC BY-SA”)

12.1 Contributors

The following people have contributed to this document to make it better:

[name] [OPTIONAL email] [Optional Organisation name]

Leif Andersen - www.modst.dk

Simon Reitinger

Sebastian Kempken

Pengcheng Ding

Alex Stapleton

Roberto Basile

Simon Reitinger

If you like your name stated here: This book is open source. Issues and pull requests are welcome. All contributors will be added to this list.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search