IF3141
SISTEM INFORMASI
Solution-related Achitecture

Semester I 2022/2023

1

What is Architecture?

- Anything that we can classify as a system has an architecture, including buildings, living organisms (including humans), organisations, complex enterprise IT systems, software applications and individual components of those.
- ISO/IEC 42010:20111 'Systems and software engineering – Architecture description' standard:
  - 'The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.'
- The objective of anyone working on architecture is to determine the appropriate way in which to represent this architecture description; and this is often in the form of documentation containing an appropriate set of models.

2

Architecture is more than just structure

- It is worth re-reading the ISO 42010 definition again, particularly the last part; **'… and the principles guiding its design and evolution'**.
  - This emphasises that an architecture includes aspects such as requirements and constraints (in particular architecture principles) that inform what that architecture needs to be to satisfy key concerns of major stakeholders.
- While the ISO 42010 standard is specifically aimed at systems and software engineering, the definitions contained are used consistently to refer to related types of architecture.

3

Architecture Patterns

- Architecture structures are typically models and there are some fundamental patterns that apply in various contexts
- Four patterns: modular architecture, loosely coupled architecture, tiered/layered architecture, hierarchical architecture
- **Modular architecture**
  - Structured programming languages and tools introduced the ability to write software library modules containing routines that dealt with such issues that could be shared and invoked by multiple executables, although these would often be statically linked into a single executable during compilation and linking; this meant that a change to one of these modules would require that all executables that used it would need to be recompiled.

4

Architecture Patterns

- **Loosely coupled architecture**
  - Coupling is a term used to refer to the level of dependency between two software components, although the term applies equally to hardware components. The earlier modules described above tended to have a high level of dependency, in other words they were tightly coupled.
  - 'Coupling and cohesion' are often used in the same phrase. As we have seen, coupling relates to how dependent modules are on each other. Cohesion looks at how closely modular elements need to operate or be changed together; where this is the case, then there is high cohesion (and probably tight coupling). It is less problematic if these modules are co-located within the same coarser-grained component, which means that they will be deployed and maintained together.

5

Architecture Patterns

Tightly coupled modules	Loosely coupled components
Changing one module requires that others need to be updated.	Components can be updated or even replaced without significantly impacting on others.
Dependencies typically exist in both directions on any pair of modules; otherwise known as cyclic or circular dependency.	Dependency is typically one way, or acyclic. Where cyclic dependency does exist this can be broken by employing a suitable design pattern, such as the observer pattern.
One module is dependent on how the other is implemented, for example by programming language.	Components[1] are not dependent on any technical aspect of the other.

6

Architecture Patterns

- **Tiered/layered architecture**
  - There are different kinds of functional elements in any solution or software architecture based upon its responsibilities; for example, those that represent and manipulate data as compared with those that handle the system's interface. These can be organised into distinct layers or tiers of modules with similar responsibilities.
- **Hierarchical architecture**
  - It is preferable in layered architectures the higher layer is dependent on the services provided by the layer below, but not vice versa. This provides flexibility in that components at one layer can be servicing multiple clients at the layer above; for example, application components sharing common infrastructure services (and their underlying components).

7

Communication and Interoperation Patterns

- **Point-to-Point**
  - Point-to-Point tends to be tightly coupled, as the components are specifically linked logically or physically and they need to share common protocols, data formats and so forth. Problems arise when multiple point-to-points are required to connect a number of nodes, as the number of links required potentially increase exponentially when new components are added.

8

Communication and Interoperation Patterns

- **Hub-and-Spoke**
  - Where a significant number of components need to interoperate, this can be set up as a set of point-to-point connections between each component and a single hub component. This component primarily exists to provide the means of connecting any component with any other. In doing so, the hub is potentially de-coupling each component pair to some extent and it can achieve this, first, by acting as an introduction broker or discovery agent, dynamically linking client to server component pairs on demand.
  - The hub can also offer other brokering services such as determining whether the client has security access to the server or by performing translations of communication protocols or data structures. It can even take over some of the logic processing and manage complex distributed transactions, at which point it has become a processing server component in its own right.

9

Communication and Interoperation Patterns

- **Service-oriented Architecture**
  - The service-oriented model is fundamentally about how components can be defined by the services they require as a service consumer (or client) and provided as a service provider (server). These components are typically as de-coupled as possible in all aspects.
  - This model also applies to how we organise our business functions and it is the model that allows organisations to outsource certain functions, where they are defined, and interoperate according to the services they provide rather than according to how they perform the service. If done properly, it should not be evident to a service consumer whether the service is provided by an internal or an outsourced provider.

10

Types of Architectures

- Many related architecture disciplines have evolved over the last 20 years, each focusing on different forms of systems.
- Three levels of granularity:
  - Enterprise Architecture;
  - Solution Architecture;
  - Software Architecture.

11

Types – Enterprise Architecture

- For many organisations, a significant development has been the adoption of enterprise architecture (EA) as typically the highest level and most holistic form of architecture developed and evolved about the organisation (the enterprise) over a strategic timeframe.
- Organisations are fundamentally systems with similar features:
  - a context or environment within which they exist;
  - a mission, the reason why they exist;
  - a set of products and/or services that they provide to a customer base;
  - an organisational structure with individuals fulfilling roles;
  - a set of processes that they execute to conduct their operating model;
  - information and data that they process and maintain about their organization (and all the above) for either their own purposes or to meet legal and compliance requirements.

12

Types – Enterprise Architecture

- **Strategy versus Tactics**
  - Most important of all, any organisation should have:
    - vision of what their future context or environment is likely to be and what they want to be in the future to remain relevant to that future
    - the need therefore to evolve through changes to meet the challenges presented by their constantly changing context or environment
    - a strategy to achieve that over a period of time.
  - Enterprise architecture is primarily concerned with these highest-level, long-term, strategic issues. The individual changes conducted to support this strategy can usually be seen as shorter-term, tactical changes, which include some business as usual activities intended to support the strategy

13

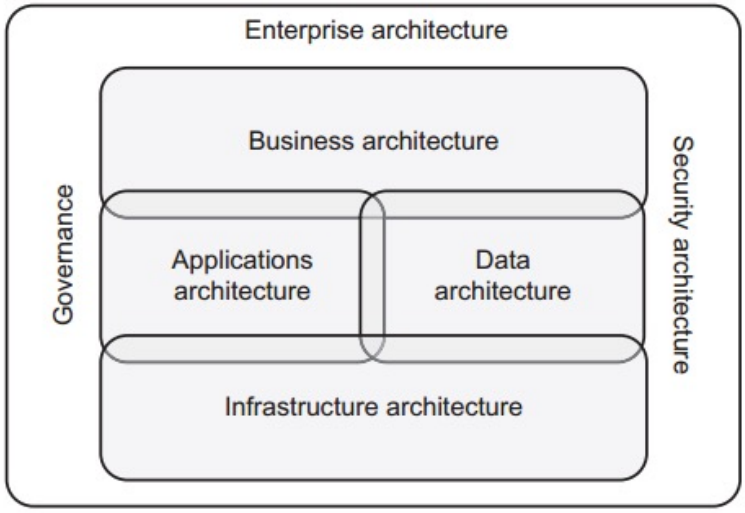
Types – Enterprise Architecture

- **Architecture domains**
  - Enterprise architecture is usually broken down into distinct architecture domains. EA frameworks relevant to most business or government enterprises typically use the following four domains:
    - Business architecture;
    - Data and/or information architecture;
    - Application(s) architecture;
    - Infrastructure or technology architecture.
  - These domains each contain a subset of the overall EA; they are interdependent and there will be overlaps of interest between them

14

Types – Enterprise Architecture

- **Architecture domains**



15

Types – Enterprise Architecture

- **EA processes, content models, organisation and frameworks**
  - Enterprise architecture function needs to determine:
    - **Organisation**: How it is to be organised internally into roles and responsibilities as well as how it will fit within the organisation as a whole.
    - **Process**: What activities the EA function is going to perform and when.
    - **Content**: What information about the enterprise will be recorded, how it is classified, organised and cross-referenced.
  - There are many frameworks available that organisations may adopt and/or adapt to fit their needs: TOGAF, Zachman Framework for EA, FEAF, DODAF, and MODAF
- **EA governance**
  - An important function of EA is to put in place principles and policies that constrain any change in activities to ensure that the strategic objectives are met.

16

Types – Enterprise Architecture

- **Architecture Principles**
  - Architecture principles are high-level constraints that are not directly actionable but that represent underlying rules and guidelines that drive the architecture.
  - 'Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission' (TOGAF 9.1)
  - Architecture principles fill the gap between high-level strategic intentions and concrete design'

17

Types – Enterprise Architecture

- **Architecture Principles**
  - Principles can be defined that relate to any domain or aspect; for example:
    - 'business units are autonomous', is clearly a business domain principle relating to the organisational structure;
    - 'routine tasks are automated', relates to business and application domains;
    - 'primary business processes are not disturbed by implementation of change' relates to all domains;
    - 'applications have a common look and feel', relates to application design;
    - 'only in response to Business needs are changes to IT systems made', is actually an application and technology domain-focused principle.

18

Types – Enterprise Architecture

- **Policies**
  - Policies are needed as a more actionable implementation of the architecture principles. These may include:
    - policies that mandate that certain legislation, standards and industry regulations be complied with
    - policies on how the change activities will be run
    - policies on supplier engagement
    - technical policy that limits the choice of technologies to be used to help with standardization
    - development standards
- **Governance activities**
  - It is the role of a specific authority (perhaps the EA function) to ensure that the policies are complied with through governance activities

19

Types – Solution Architecture

- If EA is for long-term, strategic architecture then solution architecture is a shorter-term, tactical form of architecture. EA guides evolutionary changes to the whole enterprise over a period of time, whereas many of those individual changes will require the oversight of a solution architecture.
- **Holistic solution**
  - Solution architecture should be distinguished from the conventional design activities. Solution architecture will be a high-level description of the building blocks of the total, holistic solution architecture and how they interface and interoperate to meet the requirements.
- **Driving and controlling design**
  - This outlines the work for more detailed design and implementation which then needs to be overseen or governed by the solution architect; if any need to change any of the requirements or building blocks is identified, then they are best placed to analyse the impact across the architecture and resolve any issues.

20

Types – Solution Architecture

- **Architecture domains within Solution Architecture**
  - In most organisations, any solution should be regarded as a business solution.
  - There is a consideration how domains describe layers of a hierarchical architecture where each layer provides services to the layer above

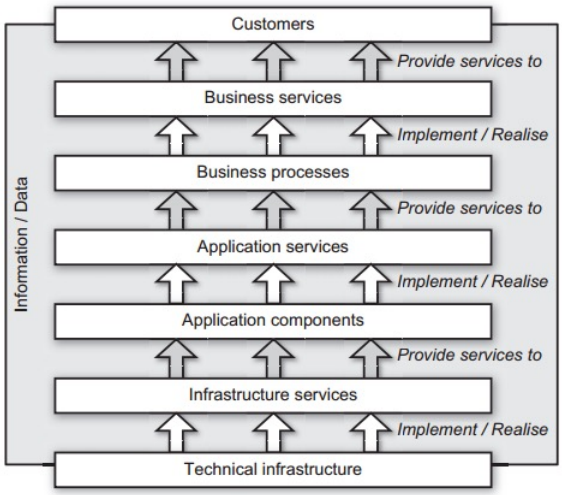
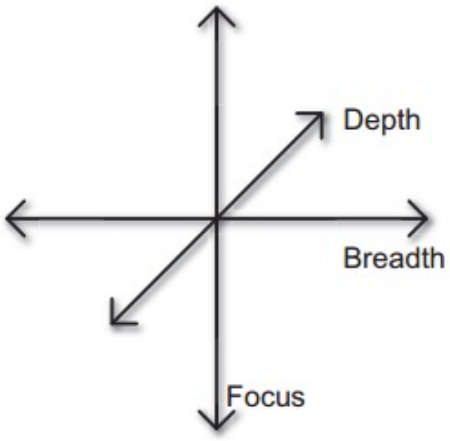


Figure 10.2 Hierarchical domain services

21

Types – Solution Architecture

- **Solution scope**
  - Any solution needs its scope to be defined and there are three dimensions of scope to be considered: focus, breadth, depth.



22

Types – Solution Architecture

- **Focus**
  - Solution may be focused on changing aspects of infrastructure arch, whose reqs are driven by app and data arch reliance on infrastructure services, thus, business arch may not figure in this solution.
- **Breadth**
  - Within each of the domains that are in focus, a solution is concerned with sets of architecture elements. Examples include business processes and people's roles within the business arch as well as new app software and infrastructure components and changes to the services being offered and depended on at each layer. The broader context of a solution arch is EA.
- **Depth**
  - There comes a point where the arch description stops and detailed design takes over; this is the depth. There are various ways of defining arch levels; the perspectives of the Zachman Framework are worth considering as a classification.

23

Types – Solution Architecture

- **Architecture contract**
  - Roles and responsibilities should be defined before work begins. This is typically done in the form of an architecture contract. This can be considered as the architecture's terms of reference. The deliverables can be defined using the three dimensions of scope discussed above.
- **Solution architecture governance**
  - In terms of compliance, two issues are involved:
    - Compliance: the solution architect needs to be aware of the constraints placed on the solution by the EA function (if one exists) as well as any placed by the solution specific requirements.
    - Enforcement: the solution architect in turn creates constraints on the underlying detailed architectures and designs, for example the software architecture and design, where these need to comply with issues apparent in the holistic solution architecture.

24

Types – Software Architecture

- Software architecture is fundamentally concerned with the purpose and structure of software applications, including software components, that may be considered to deliver platform services within the Infrastructure architecture.
- If operating within a described solution architecture, then its purpose is typically to break down the solution architecture functional building blocks or components into more granular, decomposed and eventually more physically implementable software components where appropriate.
- The software architect incorporates more implementation-specific factors such as:
  - choice of development languages and tools;
  - structural constraints presented by platform technologies chosen
  - identifying and employing appropriate logical and physical design patterns that encourage robust software design and implementation.

25

Types – Software Architecture

- **Tiered software architecture**
  - Two-tier, client-server
  - Hierarchical client-server
  - Three-tiered software architecture
  - N-tiered software architecture

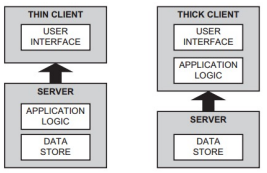


Figure 10.4 Client-server patterns

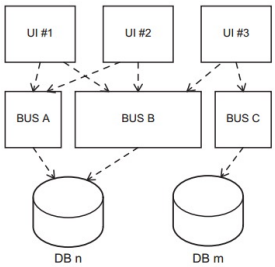


Figure 10.5 Three-tier software architecture with components shared across clients

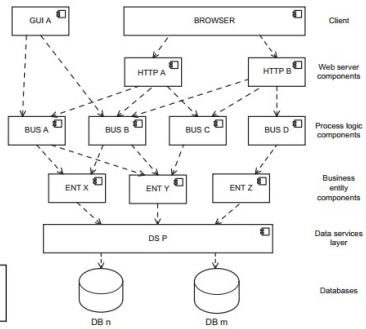


Figure 10.6 N-tiered software architecture

26

Types – Software Architecture

- **Component-based software architecture**
  - Characteristics:
    - Systems as assemblies of components
    - Functionality provided through component interactions
    - Component specification can typically be done at a 'black-box' level
    - Testing components

27

Stakeholders and Roles in Architecture

- Organisation owners/board
- External legal, standard and compliance regimes
- Senior business management
- Project, programme and portfolio management
- Operations managers
- Process workers and application users
- Suppliers
- Solutions developers

28

Architecture Management

- **Architecture Planning**
  - Strategic
    - EA should be a business-as-usual function, constantly helping organisations to manage strategic change. As well as maintaining the EA content, this function should be prioritising and planning tactical change activities, often communicated in the form of a 'road-map'.
  - Tactic
    - Enterprise and solution architects may be involved in a number of projects, such as a programme that delivers a number of changes as outlined in the EA road-map. Where there are a number of separate deliverables to be assembled to deliver a holistic solution, the programme plan can be regarded as a more detailed tactical road-map reflecting inter-project dependencies and synchronisation issues that need to be managed as risks.

29

Architecture Management

- **Architecture Governance**
  - Governance has been mentioned throughout this chapter. It is a key function of an architect, as it is a crucial part of architecture management. Architecture without governance is like requirements without testing.
  - Governance therefore needs to oversee all these aspects, particularly through validation and verification of both the final structure and the adherence to principles and other constraints imposed on its development.
  - Governance therefore requires:
    - Something to govern
    - Defined governance roles with the authority to govern
    - Governance processes

30

Architecture Management

- Something to govern, including:
  - an architecture contract that defines roles, responsibilities and authorities
  - the target system organisation or structures (components and their relationships) as described at an abstract level in the architecture description;
  - requirements that include applicable legislation, standards, principles and policies should form part of an architecture description.
- Defined governance roles with the authority to govern
  - Typically, there are levels of governance from corporate, through enterprise architecture to solutions and Specific architecture levels.

31

Architecture Management

- Governance processes, including:
  - establishing the architecture contract at the outset to outlines roles, responsibilities and authority, including governance roles;
  - defining what will be reviewed through the process (see 'something to govern' above), when and by whom;
  - conducting reviews;
  - reporting and acting on the outcomes of the reviews, which may include granting dispensations to those concerned allowing them to vary from the original description for valid reasons;
  - taking action to correct any discrepancies or non-conformities.

32