WIT 2016 ITA Module

Lecture Group #1 - Part 4
Why Architecture Styles & Patterns?

The need for Reference Blueprints

A Reference Blueprint provides advice on how to address a situation, using proven & recurrent "Patterns" of Design that have been experimented with in similar situations.

It lists and explains the forces or constraints that can be safely exercised on the proposed Architecture.

It conceptualizes the positive & negative implications of the approach considered when designing and implementing the suggested blueprint Solution Design.

About NOT re-inventing the wheel

An application architecture is an instantiation of an architectural style for a certain system. The components and connections may be decomposed into architectures themselves using architecture patterns.

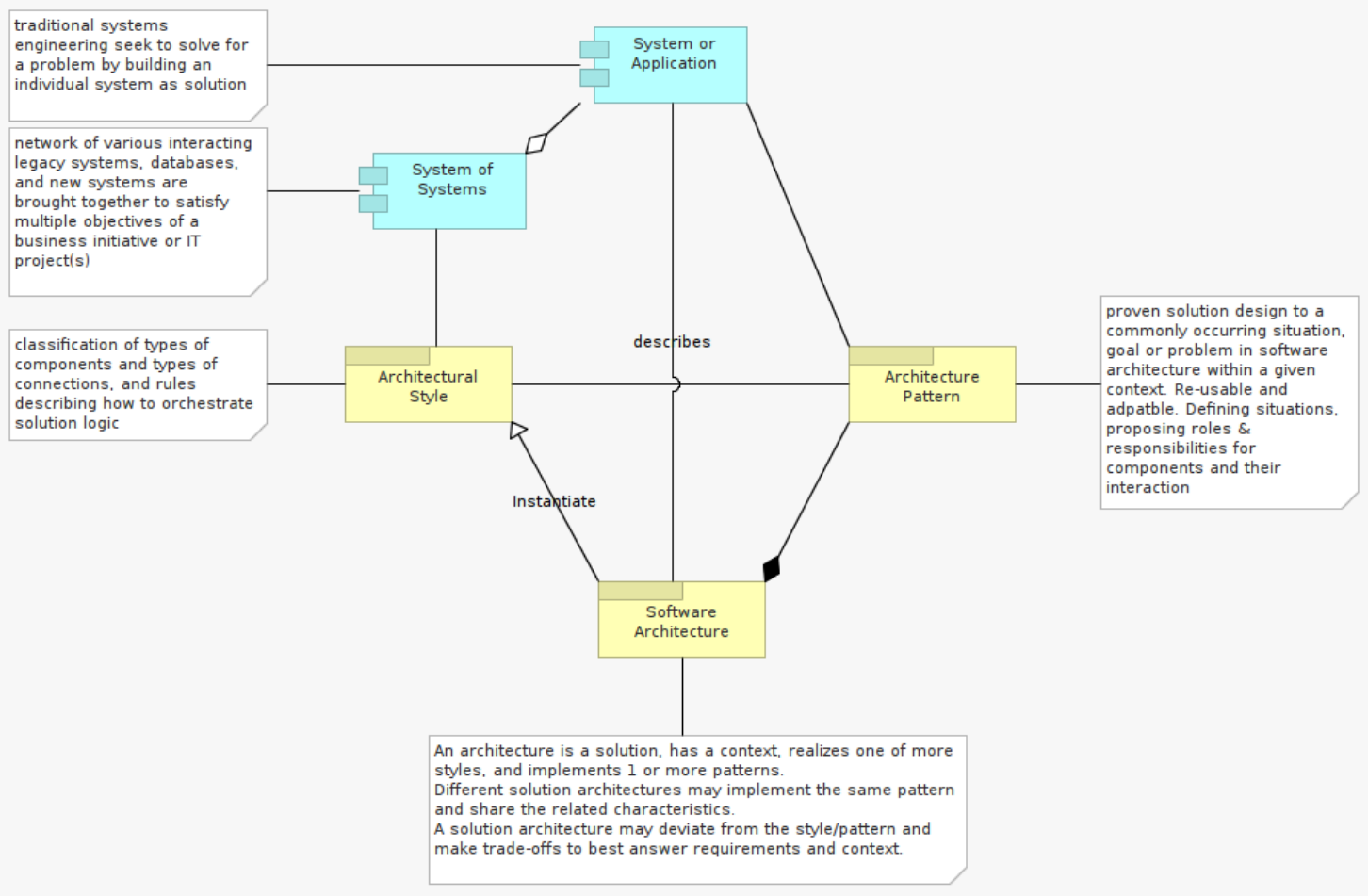
An architecture style is a family of architecture patterns, defining types of components and types of connections, and rules describing how to combine them.

Styles and Patterns help develop the "memory" of the Enterprise.

Choosing an architecture style/pattern requires understanding of the particular problem, objective/motivation of the systen

.. patterns inherently reflect the context in which they may be suitable.

Styles and Patterns in relation to Solution Design



Example: To help building the solution

An architecture style help to answer fundamental questions about the solution design:

- How are components classified?
- How do components interact?
- How responsive is the architecture?
- Is there a logical flow to the solution?

(...)

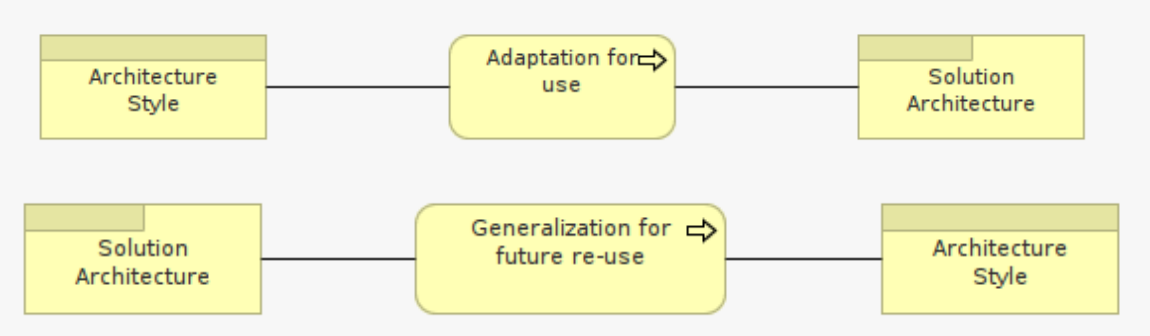
Example: To help operating the solution

An architecture style help to answer fundamental questions about the operation or change in the solution when in production:

- Does the architecture scale?
- How do you deploy and operate the architecture?
- Can you change the architecture without risk of breaking it?
- Is the solution extensible?

(...)

Architecture Style Definition



Architecture Style Definition

An Architectural Style is an abstraction of multiple solution architectures that have been designed and successfully deployed to address the same types of business problems.

An Architecture Style:

- incorporates the knowledge & best practices gained from those implementations into a "reference architecture".
- details architectural information in a common format such that solutions can be repeatedly designed and deployed in a consistent, high-quality, supportable fashion.
- describes the major foundational components such as architecture building blocks for an end-to-end Solution Architecture.
- describes a pattern of structural organization for a family of Systems types.
- specifies a fundamental behavior for data life cycle.

Purpose

The purpose of an Architectural Style (also known as reference blueprint) is:

- to provide a framework for scope identification, gap assessment, and risk assessment to develop a road map to design and implement a solution.
- to reduce the associated costs and risk for developing a solution from scratch.
- to document a set of organizational principles followed by an IT Solution as a whole, a System, or piece of a System.
- to define a vocabulary of components and connector types, and a set of constraints on how they can be combined.
- to help describing the constraints of the environment in which the solution will exist.

Make use of an Architectural Style when...

You want to find help and inspiration on how other Architects have addressed the Quality Properties of their Design (i.e. Concurrency, Distribution).

You want to strengthen a common design vocabulary throughout the Project Team.

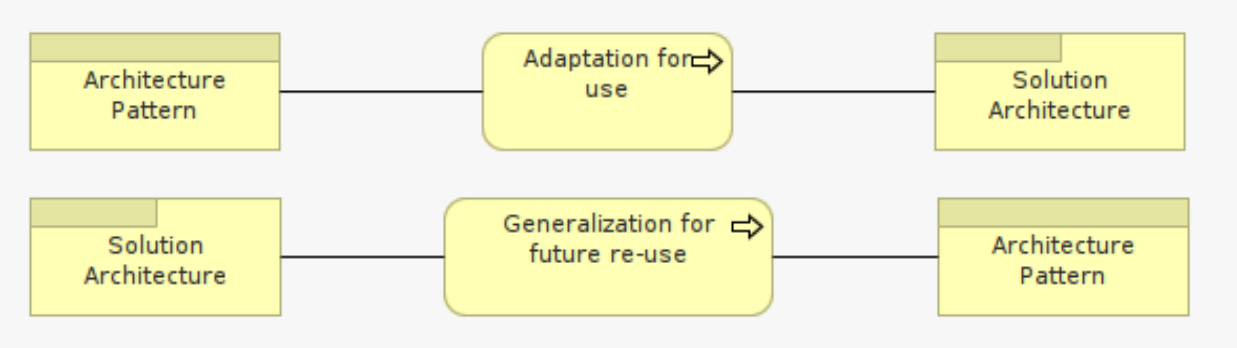
You want some level of certainty about ballpark cost - fast - for a given Solution Concept in PLANNING stage.

Styles can be combined to provide a coherent decision-making framework for an Architect.

Example(s)

- Layered architecture style
- SOA: Service-Oriented architecture style
- EDA: Event-driven architecture style
- MicroServices architecture style

Architecture Pattern Definition



Architecture Pattern Definition

An architecture pattern is a concept that solves and delineates some essential cohesive elements of an application architecture at a component-level.

An architecture Pattern is:

...a general, reusable application component design to a commonly occurring situation, goal or problem in software architecture within a given context.

...similar to software design patterns but broader in scope.

...strictly described and adaptable.

Purpose

The purpose of an architecture Pattern is:

- to describe the required qualities of a solution (ex. performance, security, availability, etc.)
- to express some rules (or guidelines) of construction for application components in terms dependencies / decoupling, etc.
- to specifies their responsibilities and communication between them
- to express a fundamental structure for their organization.

Pattern vs Style/Frameworks

An Architecture Pattern is narrower in scope compared to an Architecture Style, or a Framework.

For example:

- Message Broker pattern vs SOA style
- Value Object pattern vs Layered Architecture style
- Back-end for Front-end pattern vs Microservices style

Architecture anti-patterns

Architecture anti-patterns are originally Architecture patterns that have been made irrelevant over time:

- by new hardware or software technologies
- because it didn't stretch to industrial production class scales
- ...turned-out to be a big mistake
- ...or simply not adopted at all (too difficult to implement, leading to over-engineering, etc.)

Development Frameworks

Frameworks Definition

A Development Framework is is a set of related components which you specialize, integrate and/or instantiate to implement an application or subsystem.

A Development Framework is / can be:

- a semi complete application containing dynamic and static components that can be customized to produce applications.

- a technology cradle helping to enforce an architecture style (using customizable, extensible component libraries)

- a set of development tools facilitating the construction of a solution in a pre-defined technology stack.

Purpose

Frameworks can facilitate the implementation of a solution conforming to a specific architecture Style and/or Patterns.

Frameworks are targeted for a particular application domain & consists of a set of components/modules/classes (abstract & concrete), whose instances:

- collaborate

- are intended to be extended, i.e. reused (abstract design)

- do not have to address a complete application domain (allowing for composition of frameworks)

Examples

- Drools BRE
- JBPM
- Eclipse
- Mule ESB
- J2EE/Spring
- Play/Grails/Rails
- Android
- Scoop, Hadoop, Hive, Kafka
- Docker/LXD

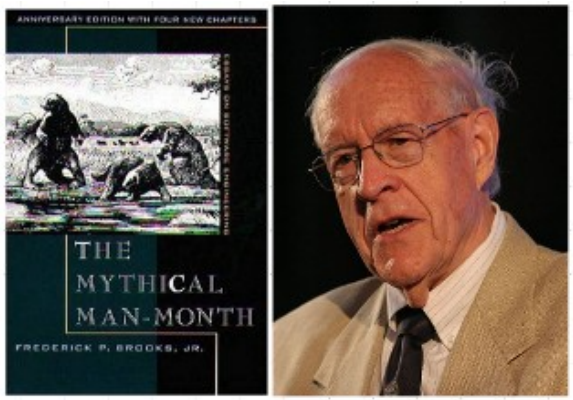
Choosing a Technology, a development Frameworks is not doing Architecture

Choosing a technology framework is not doing architecture.

Neither Styles nor Patterns or Frameworks are intended to become a substitute to the blend of experience, knowledge, and sound judgment an Architect displays.

The essence of architecture DECISIONS is to hear why & how an Architecture deviates from the Style it refers to, than how it clones it.

Again: “No silver bullet” (Brooks)



Example: Changing a claims application to a brand new framework may seem a perfect idea with cost under 1M$,

…but interoperability requirements means interface rewrite (protocols, contracts) bumping-up the cost to 3M$!