University of Oviedo

Universidad de Oviedo

EN
English

School of Computer Science

Architecture techniques

SOFTWARE
ARCHITECTURE

School of Computer Science

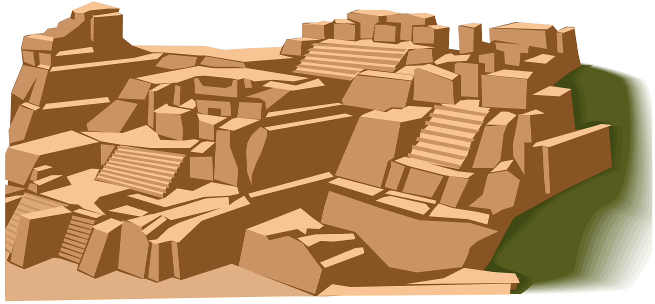
2024-25

Jose E. Labra Gayo

University of Oviedo

Software architect

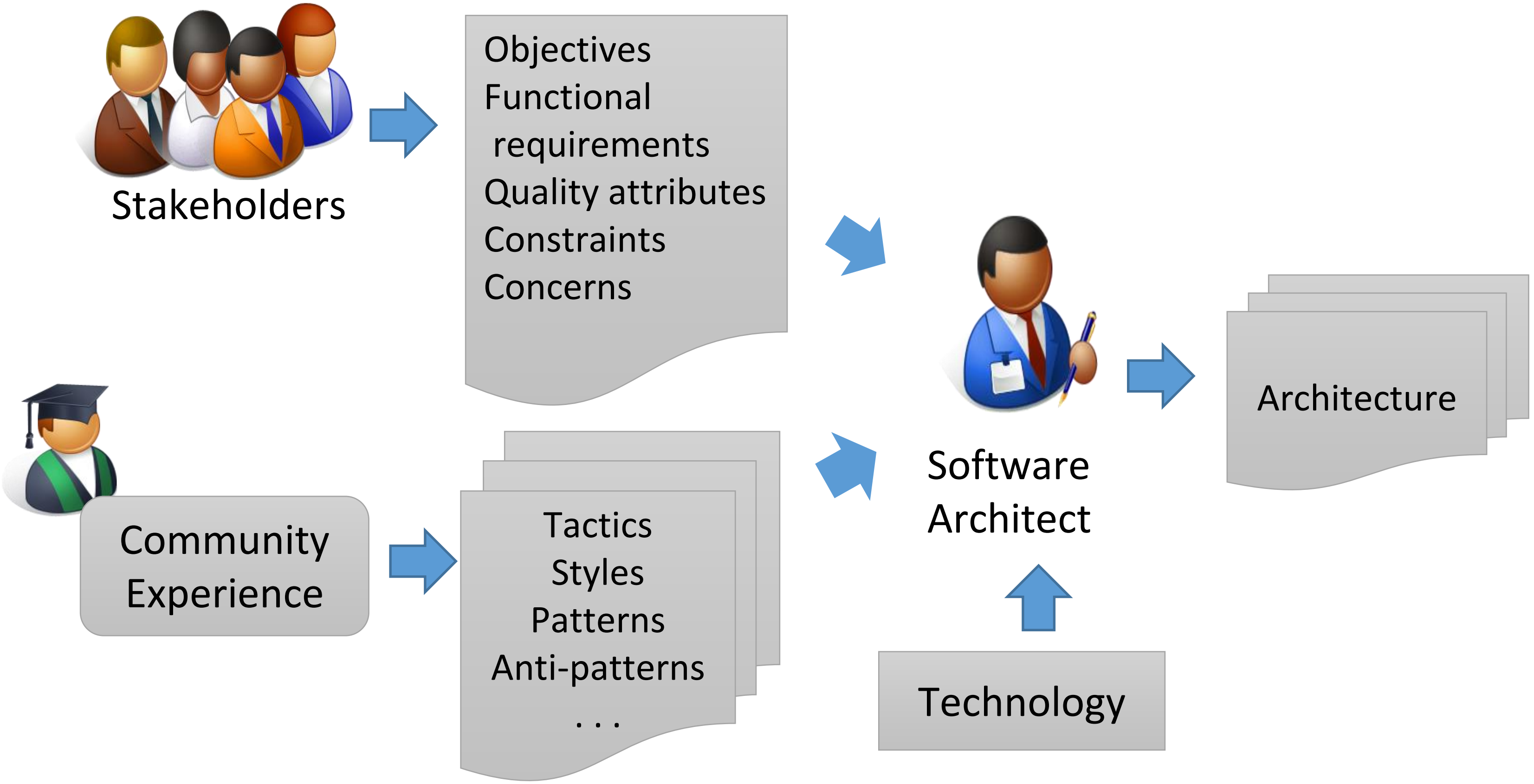
- Discipline evolves
- Architect must be aware of
  - New development techniques
  - Styles and patterns
    - Best tool = experience (*no silver bullet*)
      - Self experience
      - Experience from community



Architect

School of Computer Science

Role of software architect



Stakeholders

Objectives
Functional
requirements
Quality attributes
Constraints
Concerns

Community
Experience

Tactics
Styles
Patterns
Anti-patterns
. . .

Software
Architect

Technology

Architecture

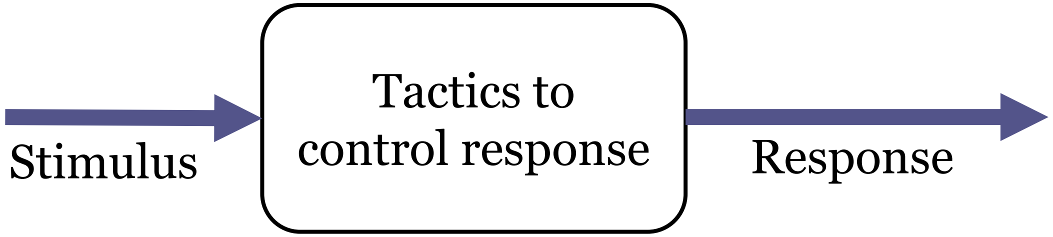
Tactics

Design techniques to achieve a response to some quality attributes

Tactics focus on a single quality attribute response

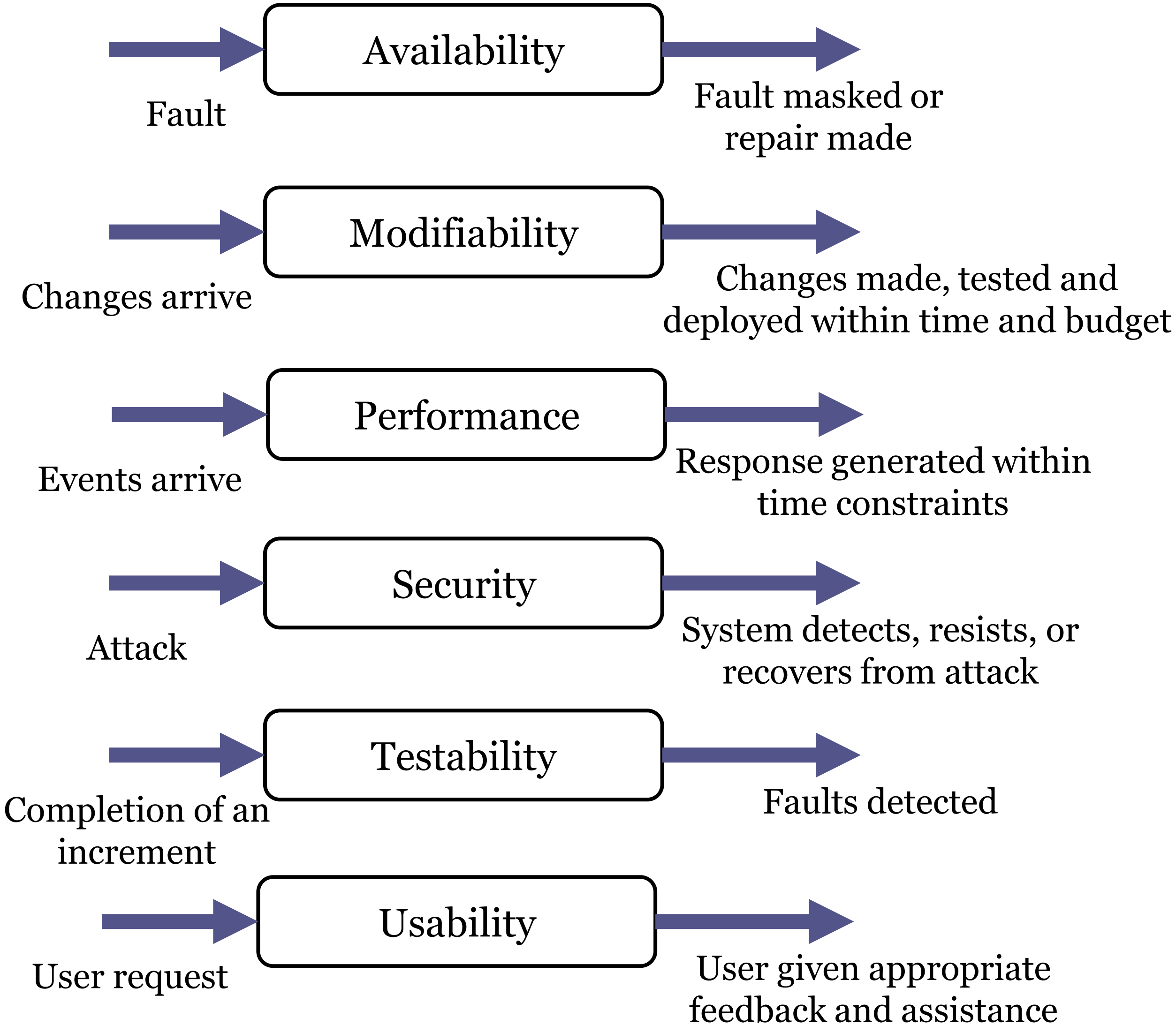
- They may compromise other quality attributes

Tactics are intended to control responses to stimuli



University of Oviedo

Tactics depend on QA



Availability

Fault

Fault masked or repair made

Modifiability

Changes arrive

Changes made, tested and deployed within time and budget

Performance

Events arrive

Response generated within time constraints

Security

Attack

System detects, resists, or recovers from attack

Testability

Completion of an increment

Faults detected

Usability

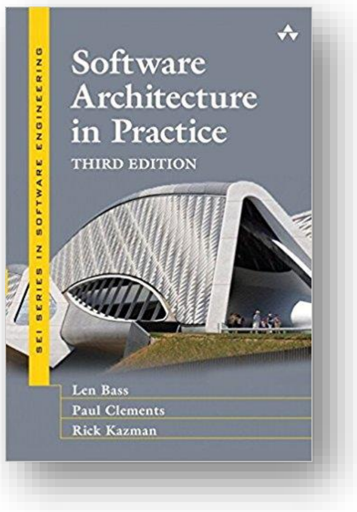
User request

User given appropriate feedback and assistance

School of Computer Science

Where can we find tactics?

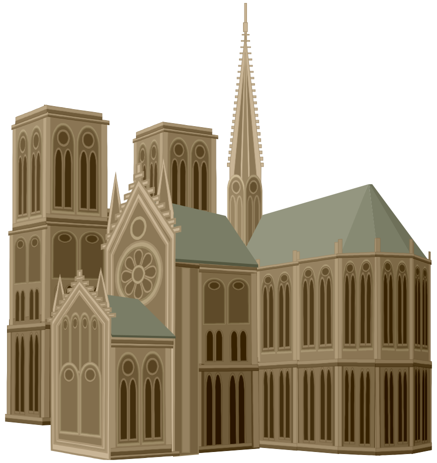
- Architect's own experience
- Documented experience from community
  - Books, conferences, blogs,...
- Tactics evolve with time and trends
  - Book "Software architecture in practice" has a list of tactics for quality attributes


SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
THIRD EDITION
Len Bass
Paul Clements
Rick Kazman

http://www.ece.ubc.ca/~matei/EECE417/BASS/ch05lev1sec1.html
https://www.cs.unb.ca/~wdu/cs6075w10/sa2.htm

Architectural styles

- Define the general shape of a system
  - They contain:
    - Elements: Components that carry out functionality
    - Constraints: define how to integrate elements
    - List of attributes:
      - Advantages/disadvantages of a style



Are there pure styles?

Pure styles = idealization

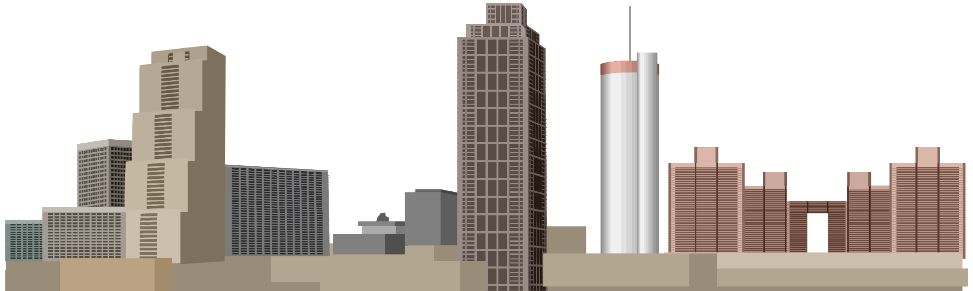
In practice, pure styles rarely appear

Usually, systems deviate from pure styles...

...or combine several architectural styles

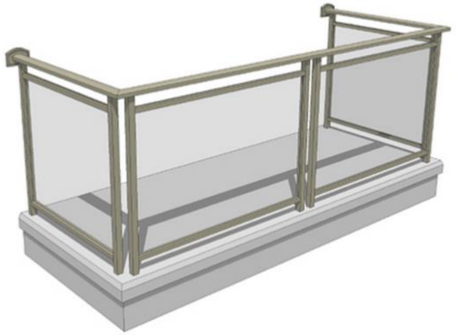
It is important to understand pure styles in order to:

- Understand pros and cons of a style
- Assess the consequences of a deviation from the style



Architectural pattern

- Reusable and general solution to some recurring problem that appears in a given context
  - Important parameter: **problem**
- 3 types:
  - Structural: Build time
    - Example: Layers
  - Runtime (behaviour)
    - Example: Pipes & filters
  - Deployment
    - Example: Load balancer



Pattern vs style

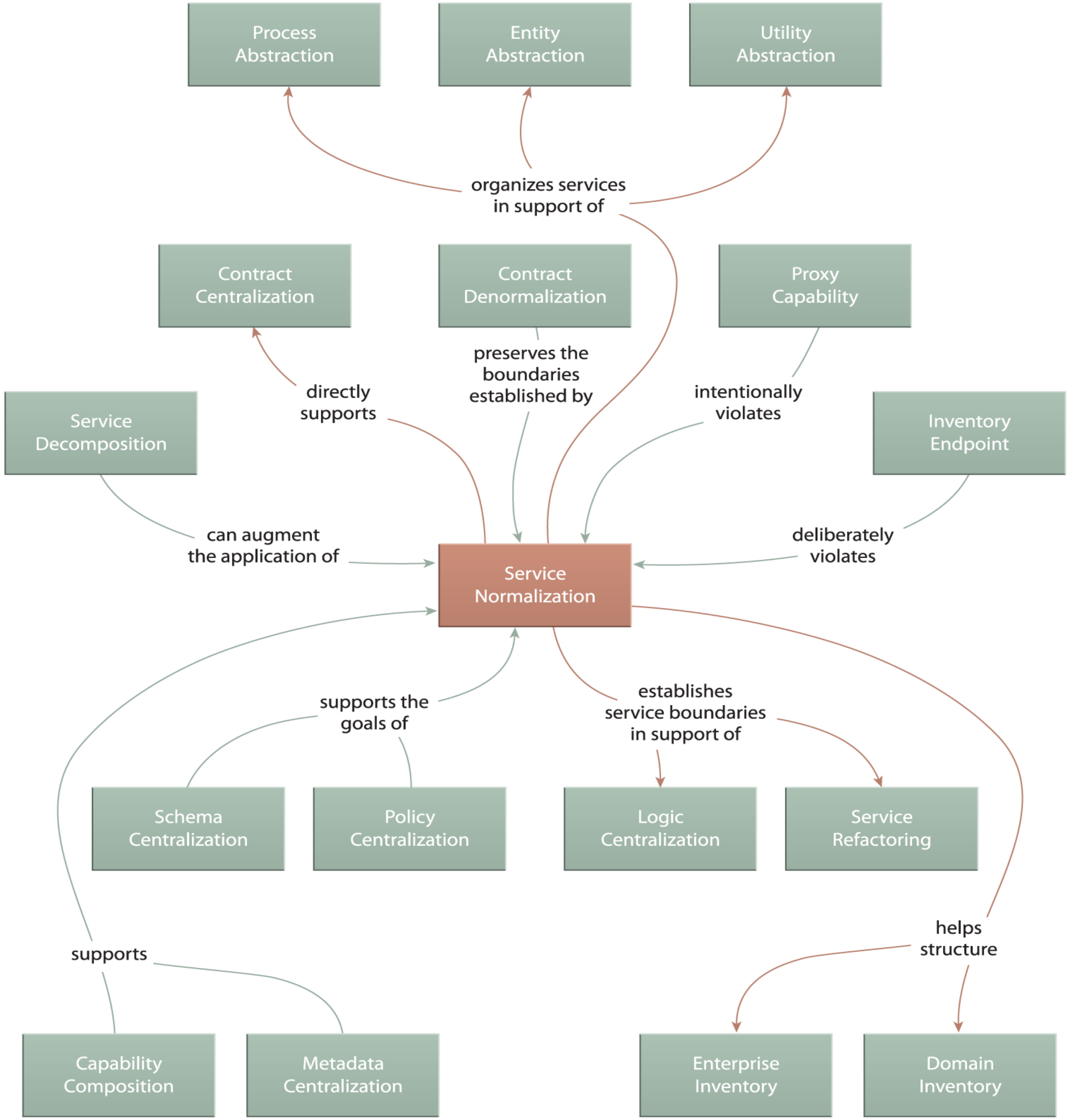
- Pattern = solution to a problem
  - Style = generic
    - Does not have to be associated with a problem
- Style defines general architecture of an application
  - Usually, an application has one style
    - ...but it can have several patterns
  - Patterns can appear at different scales
    - High level (architectural patterns)
    - Design (design patterns)
    - Implementation (idioms)
    - . . .

Pattern vs Style

- Styles, in general, are independent of each other
- A pattern can be related with other patterns
  - A pattern composed of several patterns
  - Interactions between patterns

Pattern languages and catalogs

- Pattern catalog
  - A set of patterns about a subject
  - It does not have to be exhaustive
- Pattern language
  - A full pattern catalog about some subject
  - Goal: document all the possibilities
  - They usually include relationships between patterns
    - Graphical map



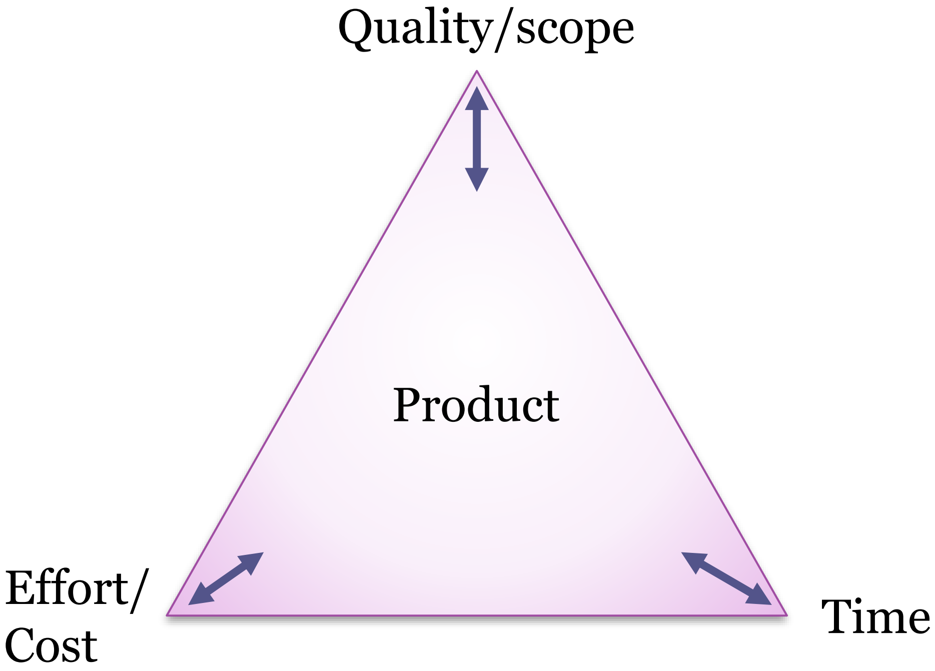
Copyright © Arcitura Education

Example of pattern language
Source: "SOA with REST" book

Build vs reuse

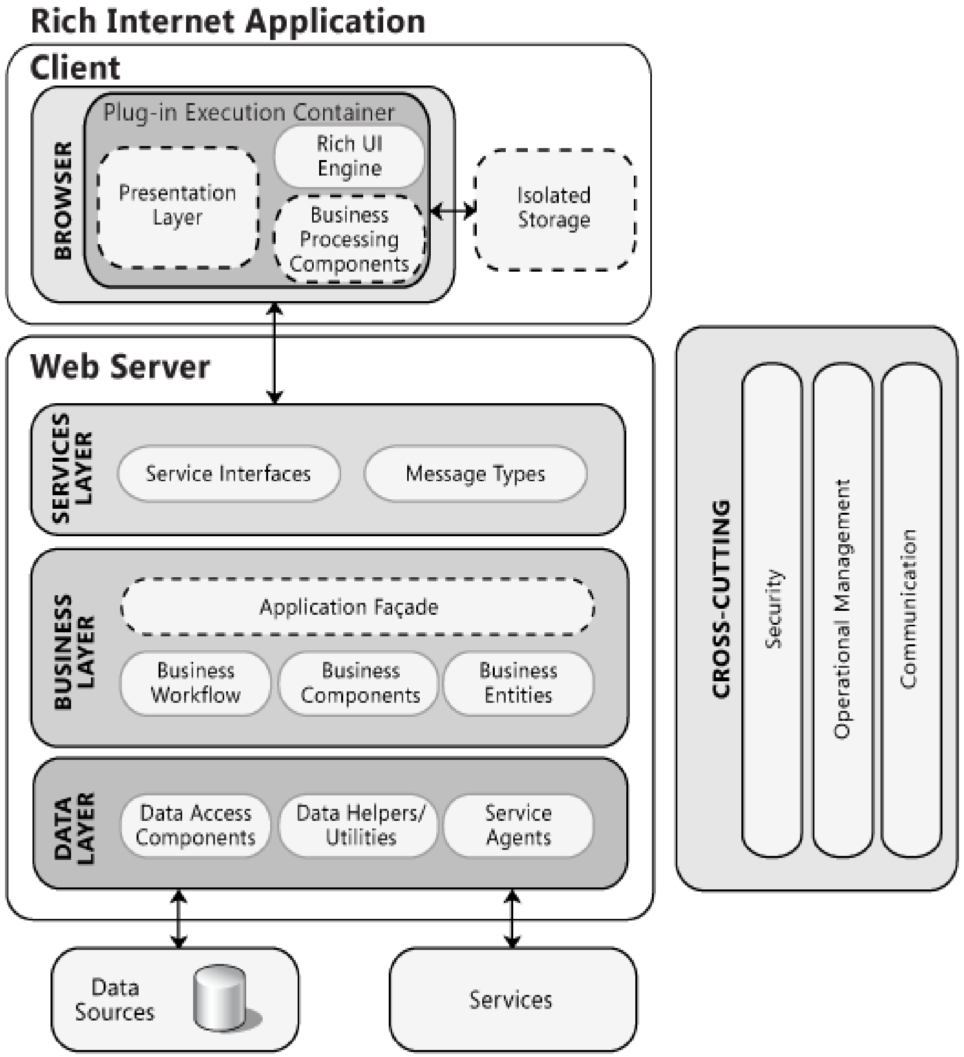
In some domains, reusing existing architectures may be more efficient

- Reference architectures
- Externally developed components



Reference architectures

- Blueprints that provide the overall structure for particular types of applications
  - They contain several patterns
  - Can be the de-facto standard in some domains



Source: Microsoft Application Architecture Guide, 2nd Ed.

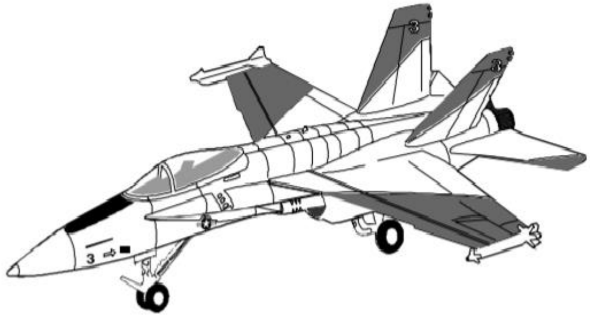
Domain Specific Software architecture

Combination of:

- Reference architecture for an application domain
- A library of components for that architecture
- A method of choosing and configuring components to work within an instance of the reference architecture

Specialized for a specific domain

Examples: ADAGE, MetaH



Externally developed components

- Technology stacks or families
  - MEAN (Mongo,Express,Angular,Node), LAMP (Linux,Apache,MySQL,PHP),...
- Products
  - COTS: Commercial Off The Self
  - FOSS: Free Open Source Software
    - Be careful with licenses
- Application frameworks
  - Partial implementation of a specific area of an application
  - Very popular for UIs
- Platforms
  - Complete infrastructure to build & run applications
    - Example: JEE, Google Cloud