A general model of software architecture design derived from five industrial approaches

By Christine Hofmesiter et al.

Presented by Group 2 and Group 3.

Agenda

- Introduction.
- Walkthrough of five methods.
  - Attribute Driven Design (ADD) Method, developed at the SEI
  - Siemens' 4 Views (S4V) method, developed at Siemens Corporate Research
  - the Rational Unified Process 4 + 1 views (RUP 4 + 1) developed and commercialized by Rational Software (IBM)
  - Business Architecture Process and Organization (BAPO), developed by Philips Research
  - Architectural Separation of Concerns (ASC) (Ran, 2000) developed at Nokia Research.
- Presentation of general method.
- Comparison using the general method.
- Analysis of architectural design methods.
- Conclusion and questions.

The overall idea of the article

- Create a general software architecture design approach based on five industrial software architecture design methods.
- Provide a framework for comparison of software architecture design methods.

Methodology used

- Analyzed five industrial software architecture design methods.
- Extracted the communalities and general software architecture design approaches.
- Compared the artifacts they use or recommend.

Result

- The five methods have a lot in common that can be combined into a general model of architecture design.

Are we surprised?

Are we surprised?

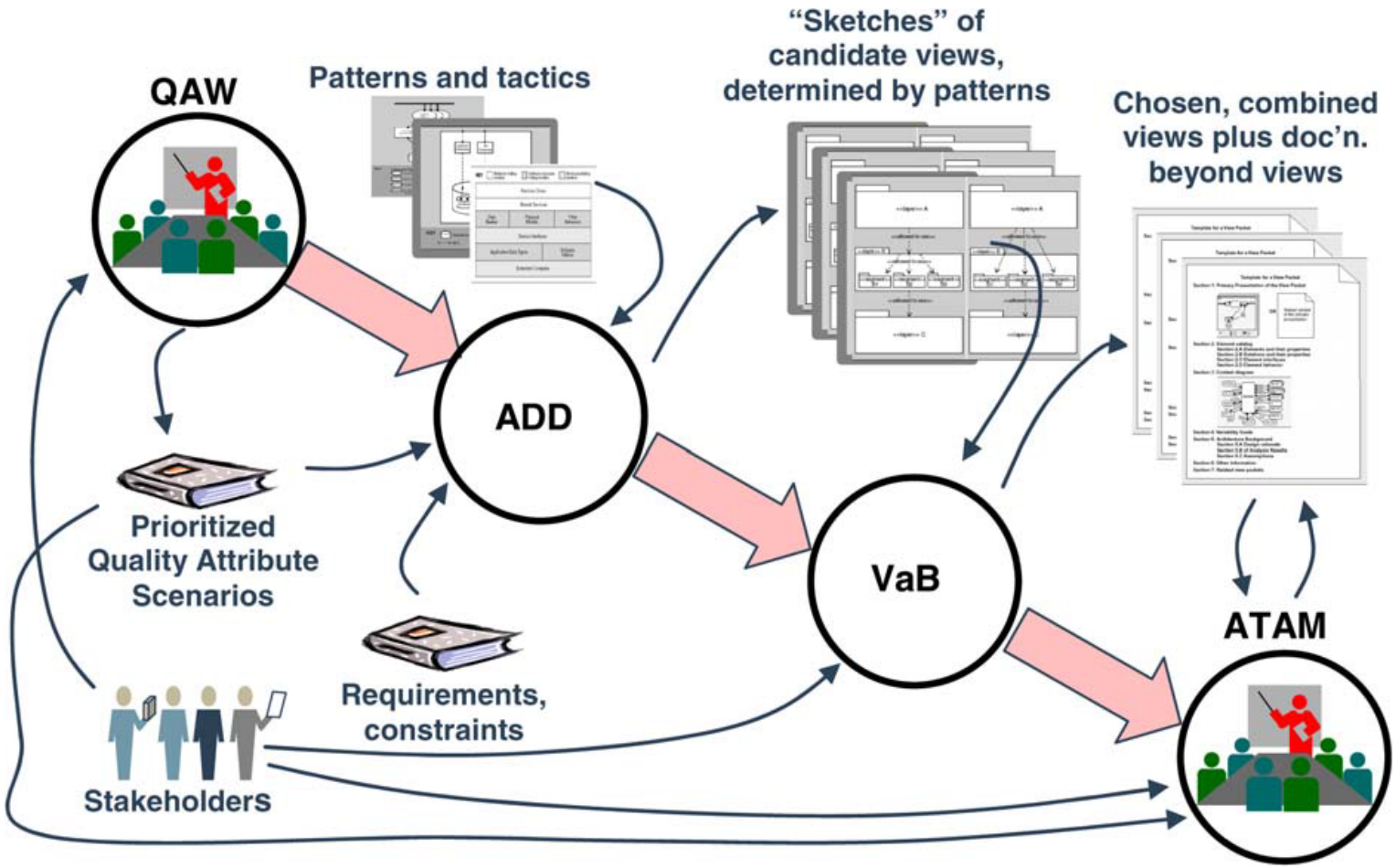
All software architecture design methods must have much in common as they deal with the same basic problems:

- Maintaining intellectual control over the design of software systems that require involvement of and negotiation among multiple stakeholders.
- The systems are often developed by large, distributed teams over extended periods of time.
- Must address multiple possibly conflicting goals and concerns.
- Must be maintained for a long period of time.

Attribute Driven Design (ADD) - Overview

- It was developed at the SEI.
- Approach to defining software architectures by basing the design process on the architecture's quality attribute requirements.
- In ADD, architectural design follows a recursive decomposition process where, at each stage in the decomposition, architectural tactics and patterns are chosen to satisfy a set of quality attribute scenarios.
- The architecture designed represents the high-level design choices documented as containers for functionality and interactions among the containers using views.
- The views that are most commonly used are one or more of the following: a module decomposition view, a concurrency view, and a deployment view.

ADD – relation to other activities.



Ref: Hofmeister, C., et al., A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software 80(1): 106-126 (2007).

ADD - relation to other activities.

- The Quality Attribute´Workshop (QAW) helps in understanding the problem by eliciting quality attribute requirements in the form of quality attribute scenarios.
- The Views and Beyond (VaB) approach documents a software architecture using a number of views based on stakeholders' needs.
- The Architecture Tradeoff Analysis Method (ATAM) provides detailed guidance on analyzing the design and getting early feedback on risks.

Quality Attribute Requirements

Quality Attribute	Architecture Requirement
Performance	Application performance must provide sub-four second response times for 90% of requests.
Security	All communications must be authenticated and encrypted using certificates.
Resource Management	The server component must run on a low end office-based server with 512MB memory.
Usability	The user interface component must run in an Internet browser to support remote users.
Availability	The system must run 24x7x365, with overall availability of 0.99.
Reliability	No message loss is allowed, and all message delivery outcomes must be known with 30 seconds
Scalability	The application must be able to handle a peak load of 500 concurrent users during the enrollment period.
Modifiability	The architecture must support a phased migration from the current Forth Generation Language (4GL) version to a .NET systems technology solution.

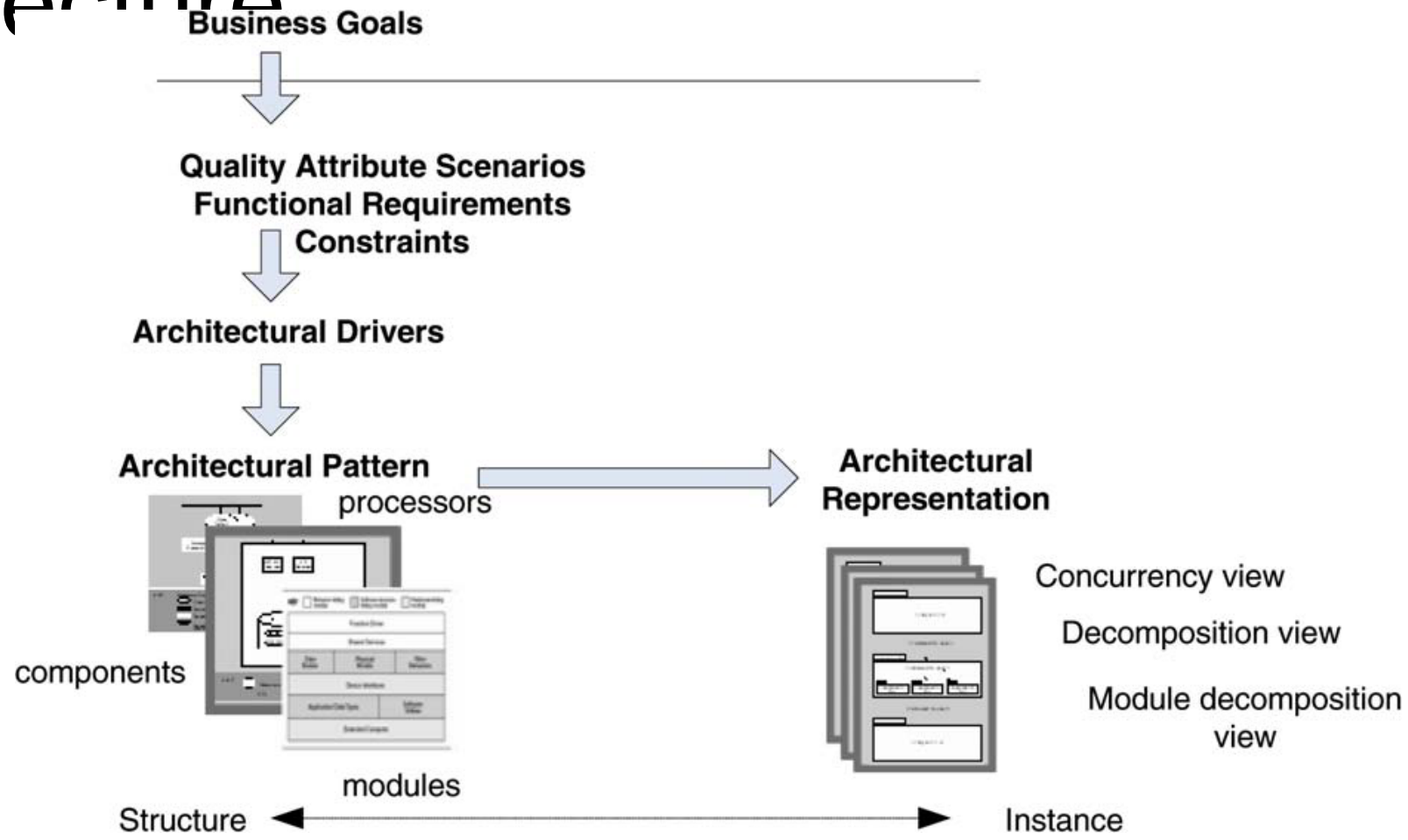
Ref: M. Ali Babar, Software Architecture Process, Lecture notes week 3, Lecture 4.

Constraints

Constraint	Architecture Requirement
Business	The technology must run as a plug-in for MS BizTalk, as we want to sell this to Microsoft.
Development	The system must be written in Java so that we can use existing development staff.
Schedule	The first version of this product must be delivered within six months.
Business	We want to work closely with and get more development funding from *MegaHugeTech Corp*, so we need to use their technology in our application.

Ref: M. Ali Babar, Software Architecture Process, Lecture notes week 3, Lecture 4.

ADD recursively designing the architecture



Ref: Hofmeister, C., et al., A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software 80(1): 106-126 (2007).

ADD - recursively designing the architecture

- Choose the module to decompose – Begin from the whole system
- Refine the module according to these steps:
  - Choose the architectural drivers from the set of quality scenarios and functional requirements.
  - Choose an architectural pattern that satisfies the architectural drivers. Identify child modules required to implement the tactics.
  - Instantiate modules and allocate functionality from the use cases and represent the results using multiple views.
  - Define interfaces of the child modules and document relevant information about constraints on the types of modules.
  - Verify and refine use cases and quality scenarios. This step verifies that nothing important was forgotten and prepares the child modules for further decomposition or implementation
- Repeat the steps above for every module that needs further decomposition.

Siemens' 4 views (S4V) - Overview

- Developed at Siemens Corporate Research.
- This is based on best architecture practices for industrial systems.
- The four views are conceptual, execution, module and code architecture views.
- These four views separate different engineering concerns, thus reducing the complexity of the architecture design task.

S4V – The four views

- **Conceptual view**: The primary engineering concerns in this view are to address how the system fulfills the requirements. The functional requirements are a central concern.
- **Module view**: Modules are organized into two orthogonal structures: decomposition and layers. The decomposition structure captures how the system is logically decomposed into subsystems and modules. A module can be assigned to a layer, which then constrains its dependencies on other modules. The primary concerns of this view are to minimize dependencies between modules, maximize reuse of modules, and support testing.
- **Execution architecture view**: This view describes the system's structure in terms of its runtime platform elements. Runtime properties of the system, such as performance, safety, and replication are also addressed here.
- **Code architecture view**: This is concerned with the organization of the software artifacts. The engineering concerns of this view are to make support product versions and releases, minimize effort for product upgrades, minimize build time, and support integration and testing.

These views are developed in the context of a recurring Global Analysis activity.

For Global Analysis, the architect identifies and analyzes factors, explores the key architectural issues or challenges, then develops design strategies for solving these issues.

S4V – Factors that influence the architecture

- Factors that influence the architecture are organized into three categories:
  i. organizational
  ii. technological
  iii. product factors.
- These factors are analyzed in order to determine which factors conflict, what are their relative priorities, how flexible and stable is each factor, what is the impact of a change in the factor, and what are strategies for reducing that impact.
- From these factors the key architectural issues or challenges are identified.
- The next step is to propose design strategies to solve the issues, and to apply the design strategies to one or more of the views.
- During design, the design decisions are evaluated, particularly for conflicts and unexpected interactions. This evaluation is ongoing. Thus Global Analysis activities are interleaved with view design activities, and the design activities of each view are also interleaved.

S4V – Periodic architecture evaluation

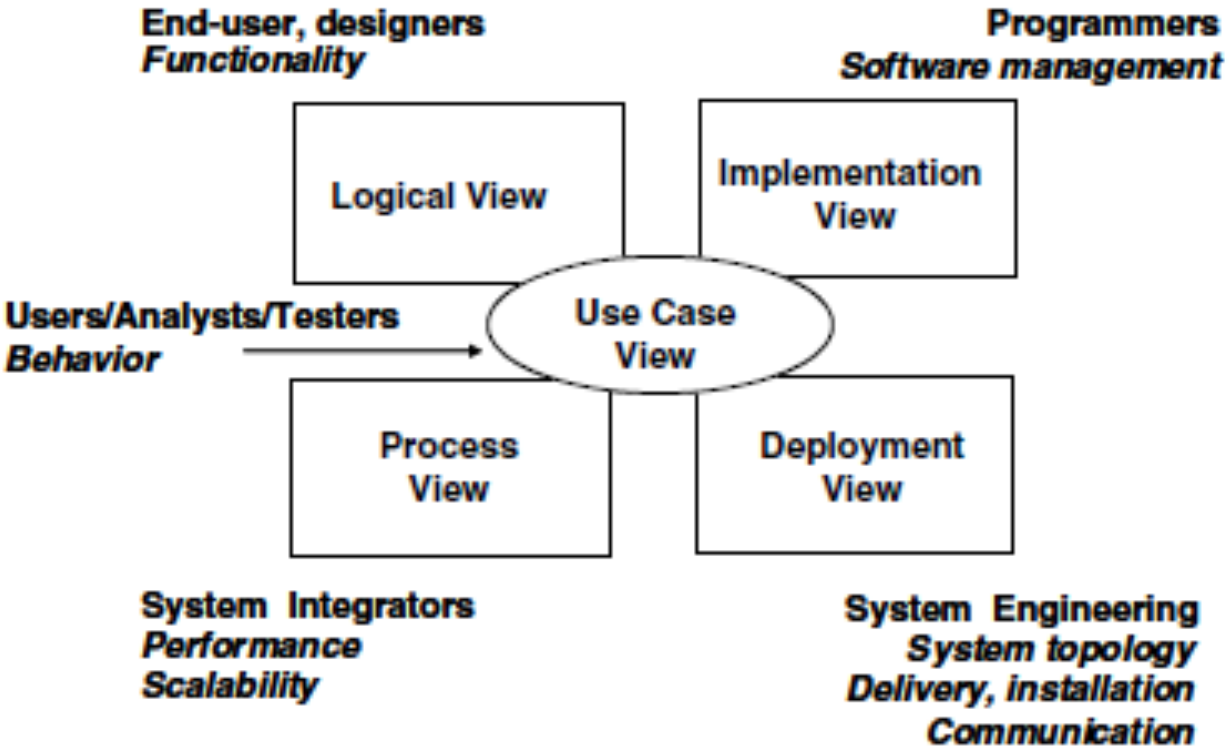
- In contrast to the ongoing evaluation that is part of the design process, periodic architecture evaluation is done in order to answer a specific question, such as cost prediction, risk assessment, or some specific comparison or tradeoff.
- This typically involves other stakeholders in addition to the architect.
- Global Analysis provides inputs to this kind of architecture evaluation, for example: business drivers, quality attributes, architectural approaches, risks, tradeoffs, and architectural approaches.

Rational Unified Process (RUP) 4+1 views - Overview

- The Rational Unified Process® (RUP®) is a software development process developed and commercialized by Rational Software, now IBM.
- For RUP "software architecture encompasses the set of significant decisions about the organization of a software system:
    - selection of the structural elements and their interfaces by which a system is composed,
    - behavior as specified in collaborations among those elements,
    - composition of these structural and behavioral elements into larger subsystem,
    - architectural style that guides this organization.

  Software architecture also involves: usage, functionality, performance, resilience, reuse, comprehensibility, economic and technology constraints and tradeoffs, and aesthetic concerns."
- RUP defines an architectural design method, using the concept of 4 + 1 view (RUP 4 + 1):
  The four views that are used to describe the design are: logical view, process view, implementation view and deployment view, and using a use-case view (+1) to relate the design to the context and goals.

RUP 4+1 views - Overview

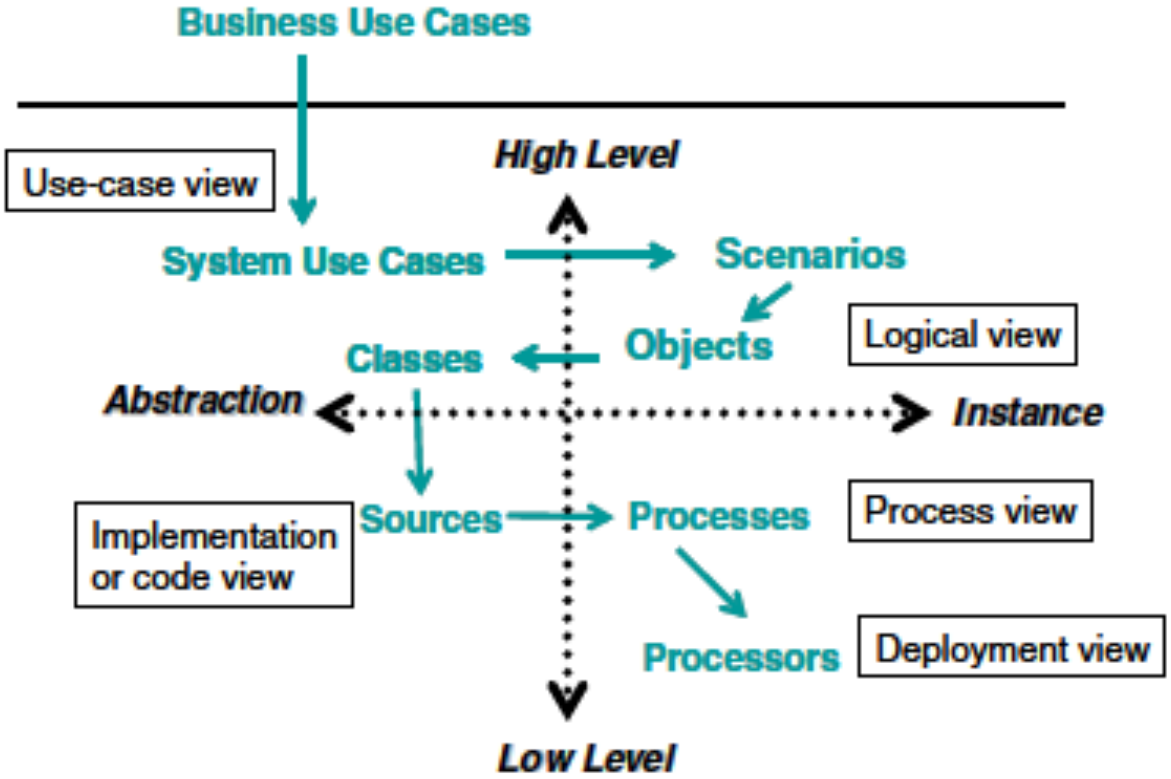


Ref: Hofmeister, C., et al., A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software 80(1): 106-126 (2007).

RUP 4+1 views –Architectural design

In RUP, architectural design is spread over several iterations in an elaboration phase, iteratively populating the 4 views, driven by architecturally significant use cases, non-functional requirements in the supplementary specification, and risks. Each iteration results in an executable architectural prototype, which is used to validate the architectural design.

RUP 4+1 views – Architectural design



Ref: Hofmeister, C., et al., A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software 80(1): 106-126 (2007).

RUP 4+1 views – Architectural design

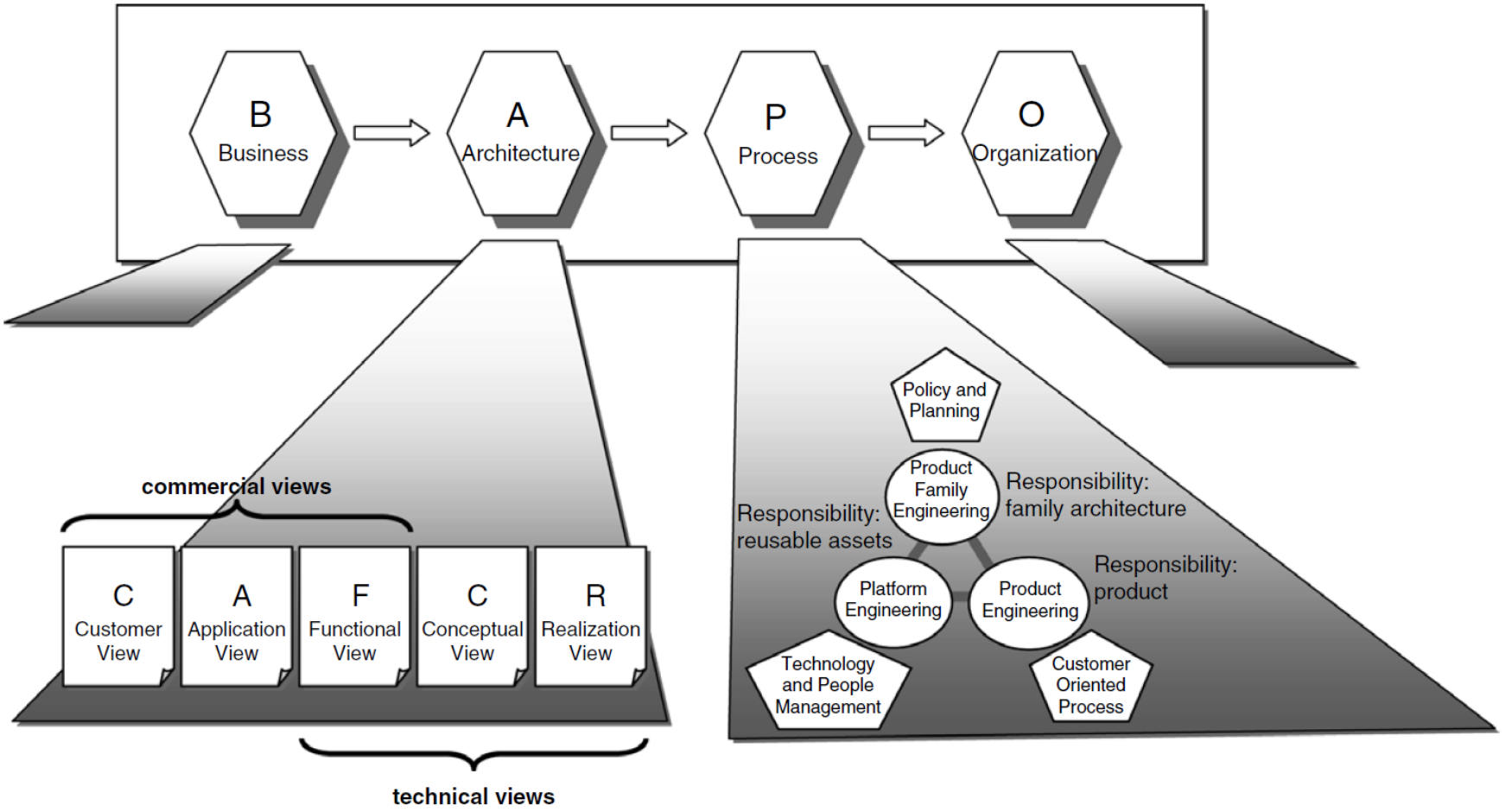
- Architectural design activities in RUP start with the following artifacts: a vision document, a use-case model (functional requirements) and supplementary specification (non-functional requirements, quality attributes).
- The three main groups of activities are:
  1. Define a Candidate Architecture.
     This usually starts with a use-case analysis, focusing on the use cases that are deemed architecturally significant, and with any reference architecture the organization may reuse. This activities leads to a first candidate architecture that can be prototyped and used to further reason with the architectural design, integrating more elements later on.
  2. Perform Architectural Synthesis
     Build an architectural proof-of-concept, and assess its viability, relative to functionality and to non-functional requirements.
  3. Refine the Architecture
     Identify design elements (classes, processes, etc.) and integrate them in the architectural prototype.
     Identify design mechanisms (e.g., architectural patterns and services), particular those that deal with concurrency, persistency, distribution.
     Review the architecture.

Business Architecture Process and Organization (BAPO)

- This approach was developed primarily by Philips Research.
- It is also known as the CAFCR approach.
- It aims at developing an architecture for software-intensive systems that fits optimally in the context of business, process, and organization.
- For the architecture, the five CAFCR views are defined: They are:

  Customer, Application, Functional, Conceptual, and Realization.
- These views bridge the gap between customer needs, wishes, and objectives on the one hand and technological realization on the other hand.

Business Architecture Process and Organization (BAPO)



Business Architecture Process and Organization (BAPO)

- In BAPO/CAFCR, the architect iteratively:

  (1) fills in information in one of the CAFCR views, possibly in the form of one of the suggested artifacts

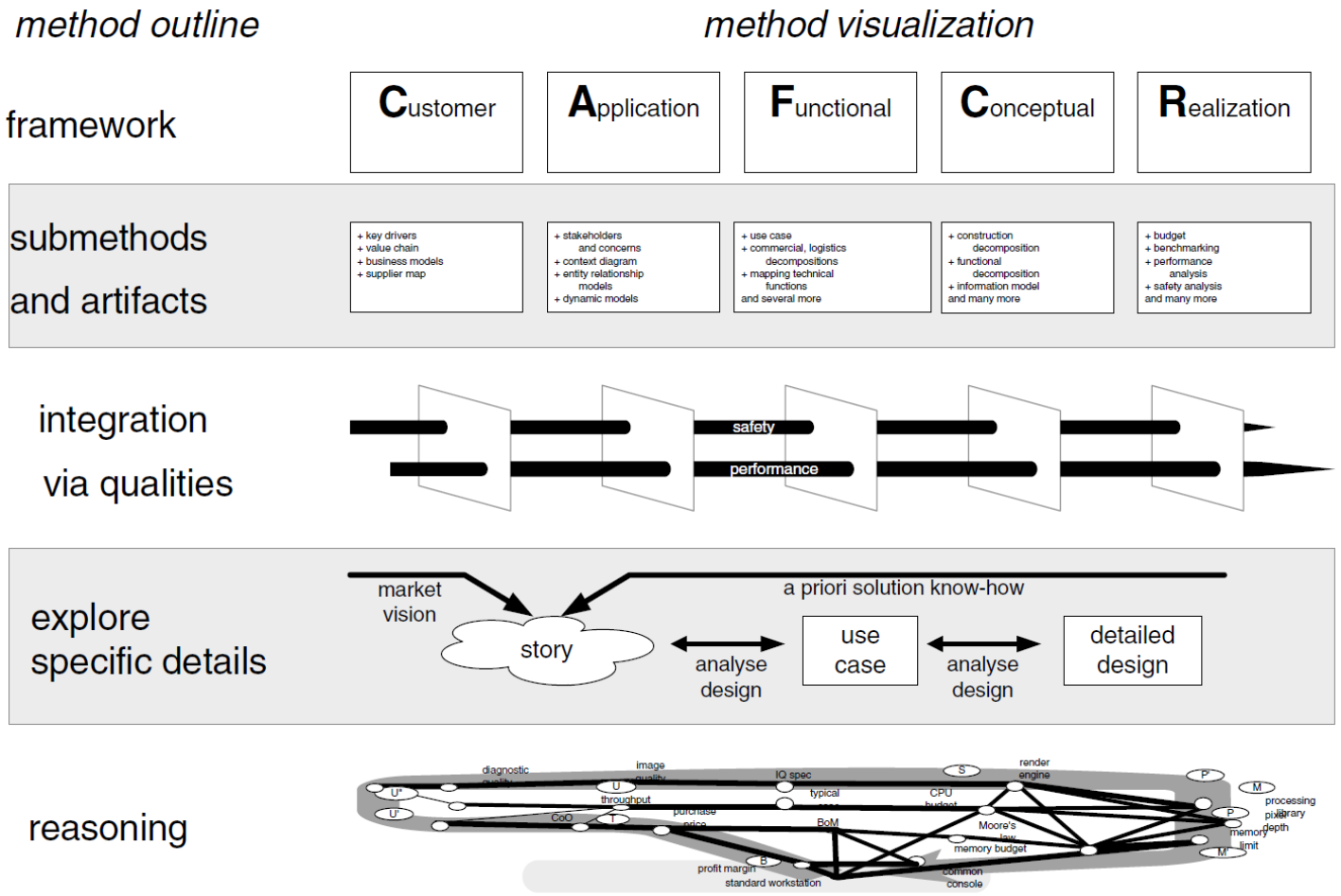
  (2) analyzes a particular quality attribute across the views to establish a link between the views and with the surrounding business, processes and organization.

- To counteract the tendency towards unfounded abstraction, specific details are often explored by story telling. The reasoning links valuable insights across the different views to each other.

- The architecture is complete when there is sufficient information to realize the system and the quality attribute analysis shows no discrepancies.

- At a larger scale, the processes described in the BAPO/CAFCR approach deal with the development of a product family in the context of other processes in a business.

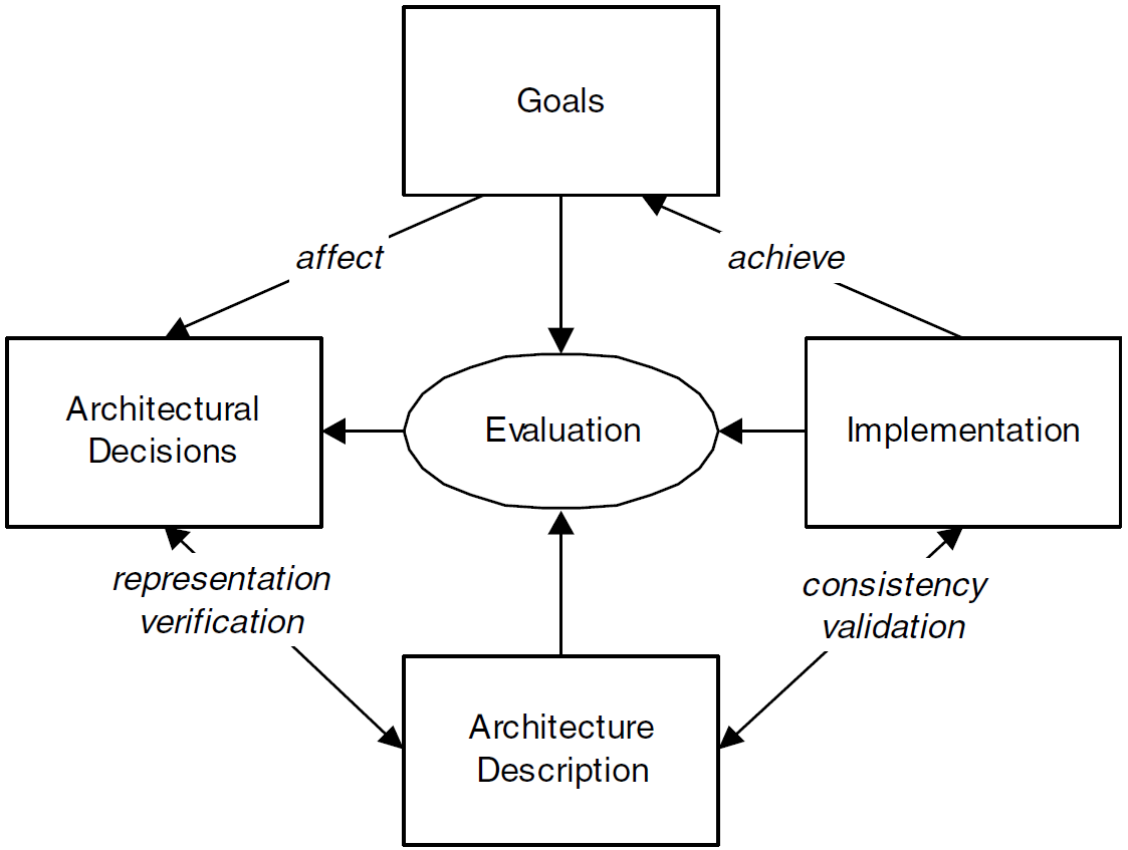
Business Architecture Process and Organization (BAPO)



Architectural Separation of Concerns

- Architectural separation of concerns (ASC) or ARES System of Concepts was developed primarily by Nokia.
- It is a conceptual framework based on separation of concerns to manage complexity of architecture design.
- Development goals affect architectural decisions. Architectural decisions are represented by architecture descriptions. Architecture descriptions are used to verify architectural decisions.
- Architecture description and implementation must be consistent. A validation process exists to determine consistency of architecture and implementation. Information regarding achievement of development goals is obtained once the implementation is available.

Architectural Separation of Concerns



Architectural Separation of Concerns

- Stakeholder goals need to be analyzed and the goals need to be refined into architecturally significant requirements (ASR) that state how to measure the achievement of the goal and indicate how the goal constrains software architecture.

- Software goes through a transformation cycle that consists of separate segments.
- During the design or development segment, the software is source code.

  During the build segment the software is object files and library archives.

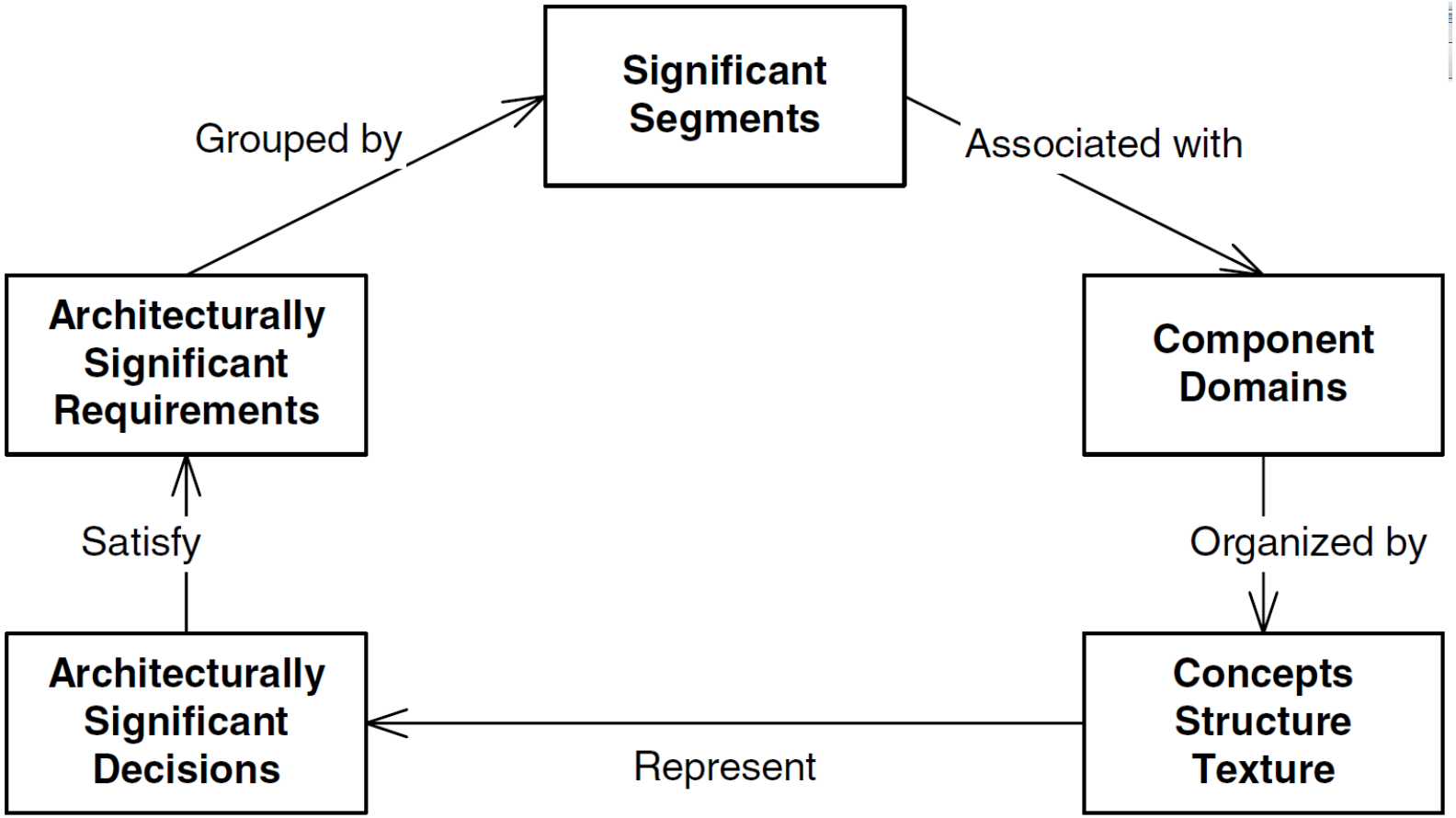
  During the upgrade segment the software consists of executable files.

  During the start-up segment the software is system state and groups of executable entities with their dependency structure.

  During the operation segment software is threads and objects.

- Architecturally significant requirements must be grouped so that requirements in different groups may be satisfied independently, while requirements within each group may interact and even conflict. This can be achieved if we group the requirements by the segment of software transformation cycle.

Architectural Separation of Concerns



Architectural Separation of Concerns

- ASC also pays special attention to design of texture (fine-grained or coarse grained).
- In ASC, the architect analyses design inputs and using a palette of techniques, produces and prioritizes ASR (architecturally significant requirements), groups ASR by segments of the software transformation cycle that they address.
- Implementation (write-time) design addresses the ASR concerned with the write-time segment.
- Design decisions make implementation technology choices, partition functional requirements between different architectural scopes of product portfolio, product family, or single product, establish portability layers for multiplatform products, allocate classes of functional requirements to different subsystems, and develop description of the API facilitating work division and outsourcing

Architectural Separation of Concerns

- Performance (run-time) design deals with run-time ASR addressing concurrency and protection, develops performance models and makes decisions regarding task and process partitions, scheduling policies, resource sharing and allocation.

- Finally, delivery/installation/upgrade design decisions address the ASR of the corresponding segments.

- Typical decisions address partitions into separately loadable/executable units, installation support, configuration data, upgrade/downgrade policies and mechanisms, management of shared components, external dependencies and compatibility requirements.

A general model for software architecture design

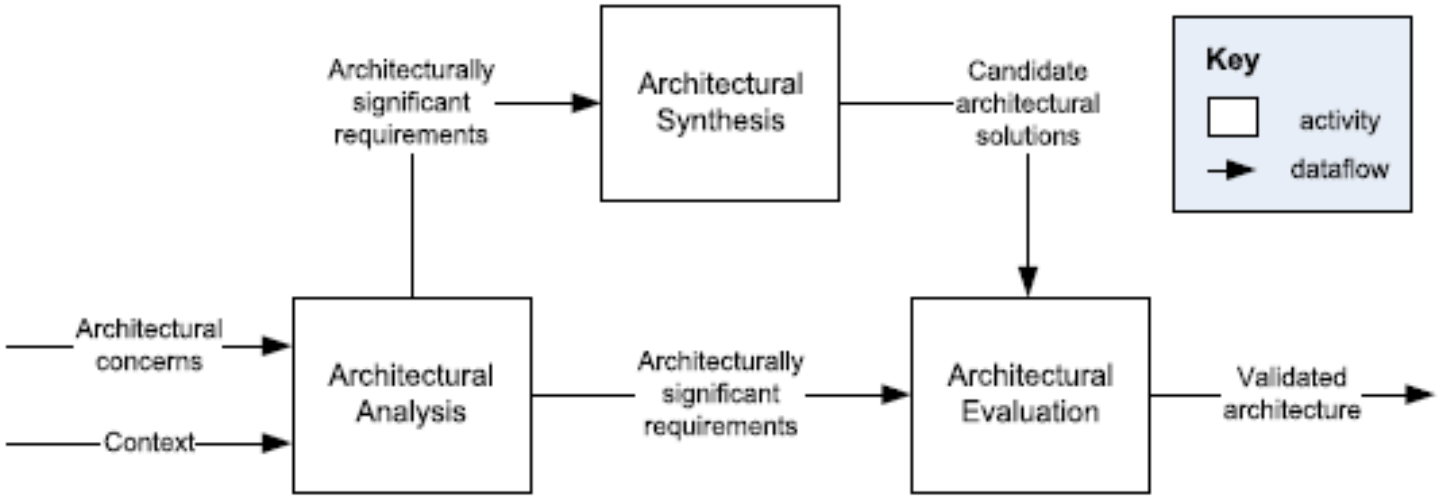
The key requirement of the general model was that it be general enough to fit the five architecture design methods, and provide a useful framework for comparing them.

- First part is about the main activities.
- Second part is a characterization of its workflow.

The main activities of the model, and their related artifacts

- **Architectural concerns**
  Concerns include system considerations such as performance, reliability, security, distribution, and evolvability. Most architectural concerns are expressed as requirements on the system.
- **Context**
  Context, determines the setting and circumstances of developmental, operational, political, and other influences upon that system. This includes things like business goals (e.g., buy vs. build), characteristics of the organization (e.g., skills of developers, development tools available), and the state of technology.
- **Architecturally significant requirements**
  An ASR is “a requirement upon a software system which influences its architecture”.
- **Architectural analysis**
  Define the problems the architecture must solve.
- **Candidate Architectural solutions**
  Present alternative solutions, and/or may be partial solutions (i.e., fragments of an architecture).
- **Architectural synthesis**
  This activity proposes architecture solutions to a set of ASRs, thus it moves from the problem to the solution space.
- **Validated Architecture**
  consists of those candidate architectural solutions that are consistent with the ASRs.
- **Architectural evaluation**
  ensures that the architectural design decisions made are the right ones.

Architectural design activities



Other inputs that are critical to the design process:

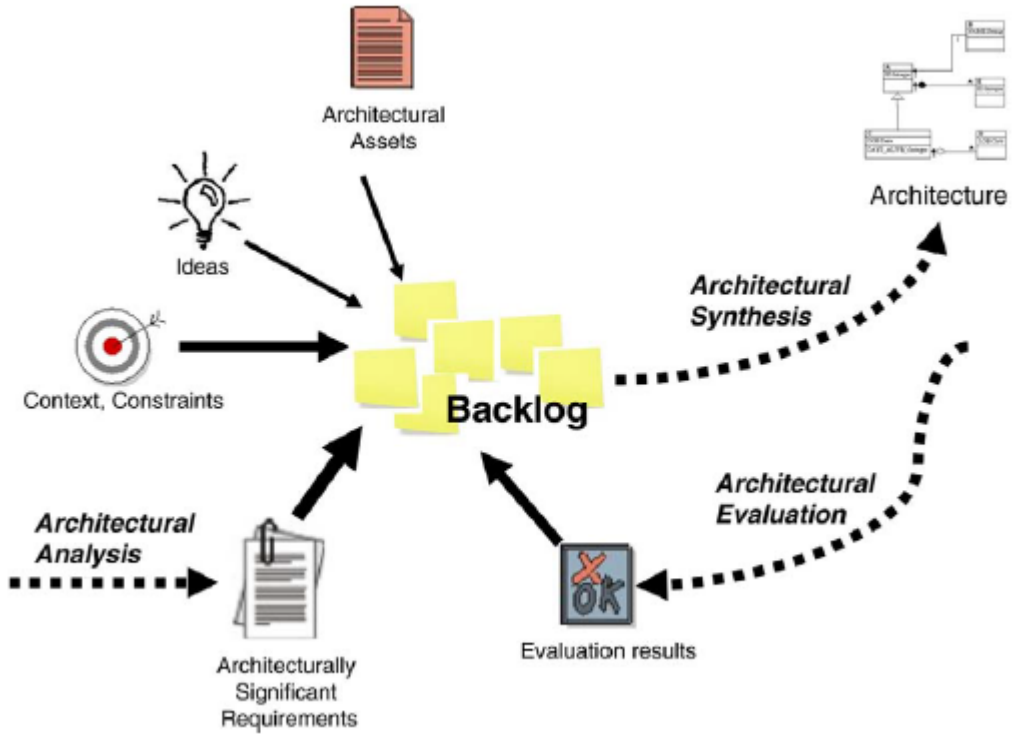
- Design knowledge
  Comes frome the architect.

- Analysis knowledge
  Is needed do define the problem and evaluate the solution.

- Realization knowledge
  In general the design must be evaluated using realization knowledge, in order to ensure that the system can be built.

Workflow and the concept of backlog

The backlog is used primarily because it is not possible to analyze, resolve, find solutions and evaluate the architecture for all architectural concerns simultaneously



The backlog is constantly fed by:

(a) selecting some architectural concern and/or ASR from architectural analysis,
(b) negative feedback in the form of issues or problems arising from architectural evaluation, and to a lesser extent,
(c) ideas from the architect's experience, discussions, readings, etc.