ISISTAN
Instituto Superior de Ingeniería de Software Tandil

A Roadmap of Architecture-centric Design Techniques

J. Andrés Díaz-Pace
adiaz@exa.unicen.edu.ar
PUC-Rio, May/21/2014

ISISTAN Research Institute
Facultad de Cs. Exactas, UNICEN
CONICET

UNICEN CONICET

Outline

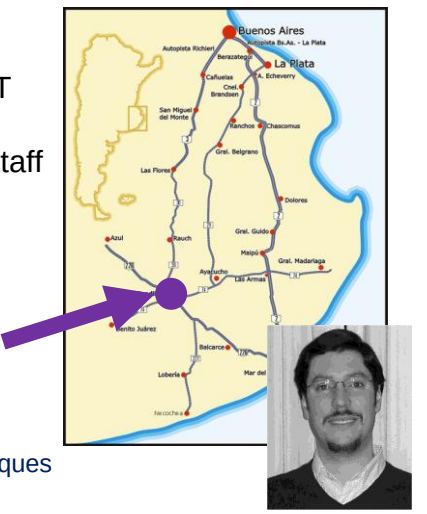
- Software architecture
  - Focus on quality attributes
  - Main building blocks
  - The architecture lifecycle
- Example: Modifiability
  - Scenarios, tactics, module views
- Some architectural methods
  - QAW, ADD, V&B, ATAM
- Conclusions and perspectives

Software Engineering Institute
Carnegie Mellon

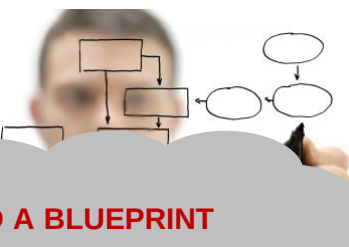
About me

- Adjunct professor at UNICEN University (Tandil, Argentina)
- Adjunct researcher of CONICET

- Former member of Technical Staff of SEI (2007-2010)
  - ATAM Evaluator Certificate.
  - Software Architecture Professional Certificate.
- PhD. Computer Sciences - UNICEN (2004)
- Research interests
  - Design driven by quality attributes
  - Conformance/reconstruction techniques
  - Assistive tools for software design
  - Object-oriented frameworks

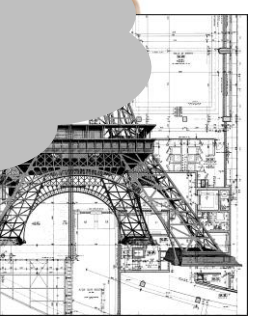


Importance of architectures

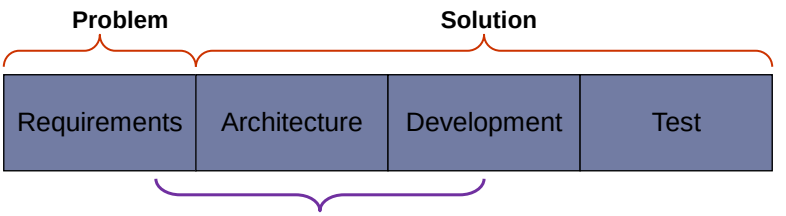


DO WE NEED A BLUEPRINT FOR CONSTRUCTING SOFTWARE SYSTEMS?

- Plan b
- A mana
  deal with compl
- Communication with stakeholders
- Prescriptions for the implementation

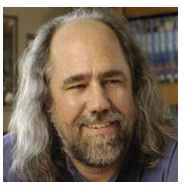


The architecture discipline



*All architecture is design but not all design is architecture. Architecture represents the **significant design decisions** that shape a system, where **significant is measured by cost of change**.*

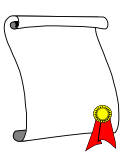
- Grady Booch



Eeles, IBM - 2009

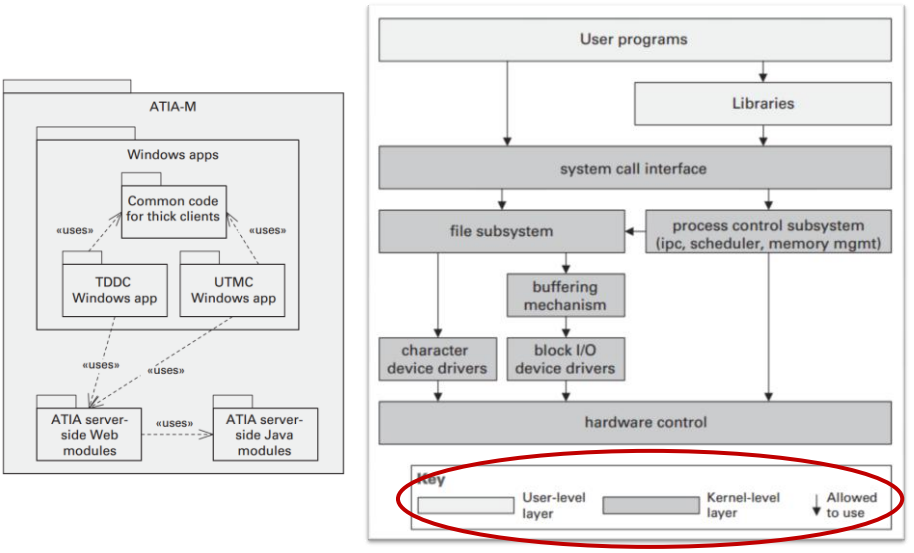
The role of quality attributes (NFRs)

- The problem is not just to get the functionality right …
- **QAs**: Properties of a software product through which the stakeholders judge the quality of the product
  - Performance
  - Security
  - **Modifiability**
  - Availability
  - Usability
  - …
  - *"Web-fiability"*
  - *Energy consumption*

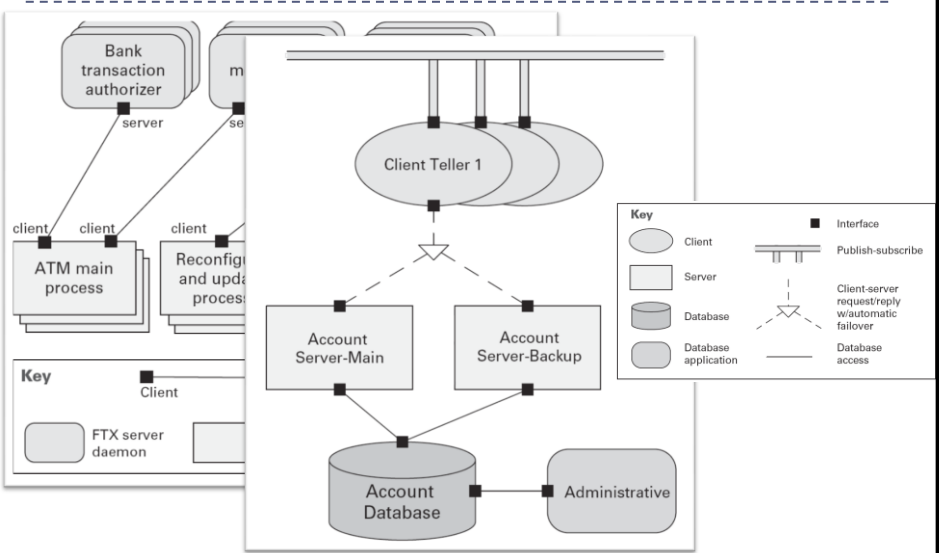


Entry condition: Understanding those quality-attribute requirements critical for the system goals

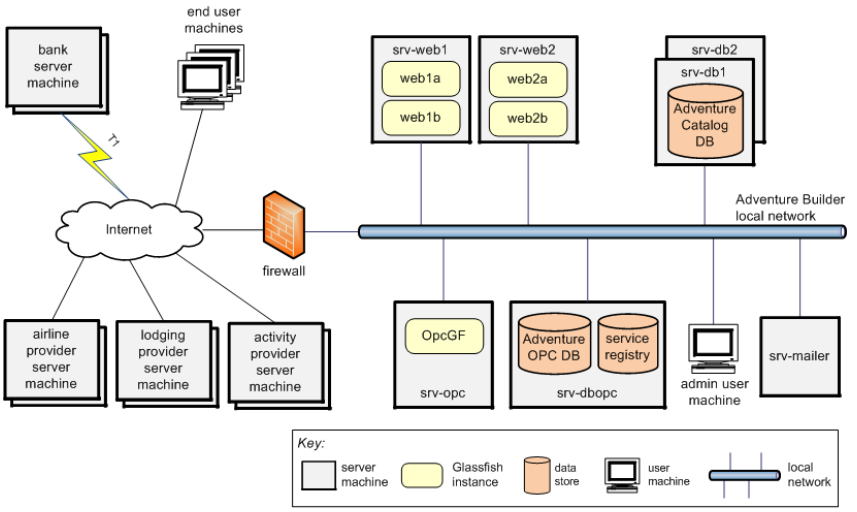
Architecture examples (blueprints) - 1



Architecture examples (blueprints) - 2

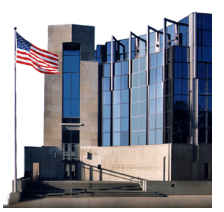


Architecture examples (blueprints) - 3



Architecture-centric engineering

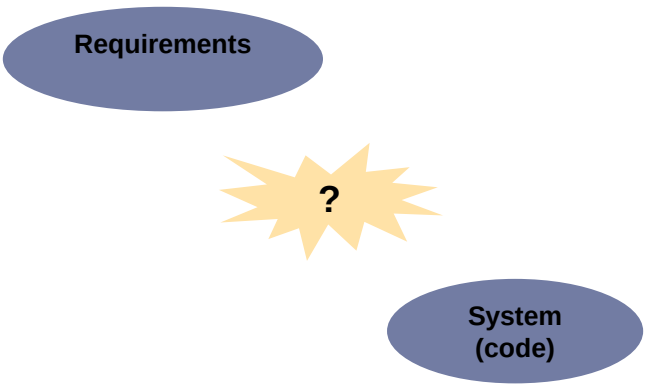
"Architecture-Centric Engineering (ACE) is the **discipline** of using architecture as the focal point for performing ongoing **analyses** to gain increasing levels of **confidence** that systems will support their **business goals**."
[ACE initiative, SEI]



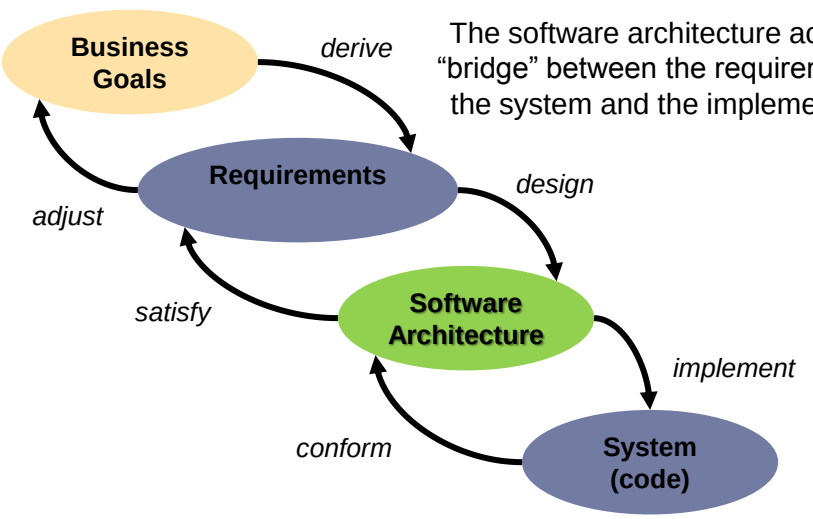
- An architecture of a system consists of:
  structures (elements, relationships)
  + content (responsibilities of the elements)
  + design decisions
- **QUALITY ATTRIBUTES as the leitmotif**

From requirements to implementation

There is a sizeable gap between requirements and code



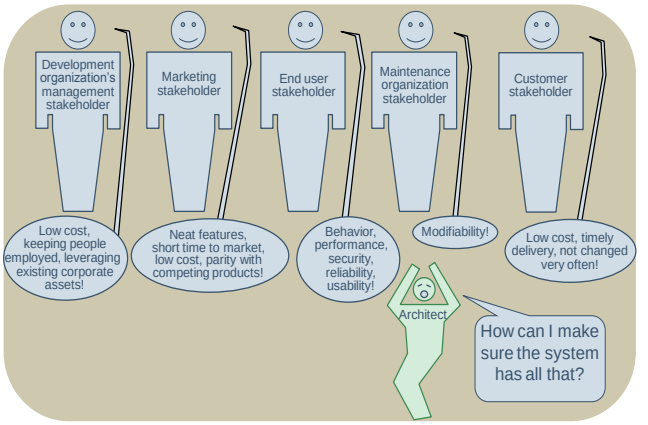
The architecture as a bridge



The software architecture acts as a "bridge" between the requirements of the system and the implementation

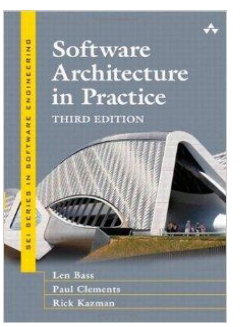
The influence of stakeholders

- Different **stakeholders** have different **quality-attribute concerns** (and priorities) for the system and its architecture



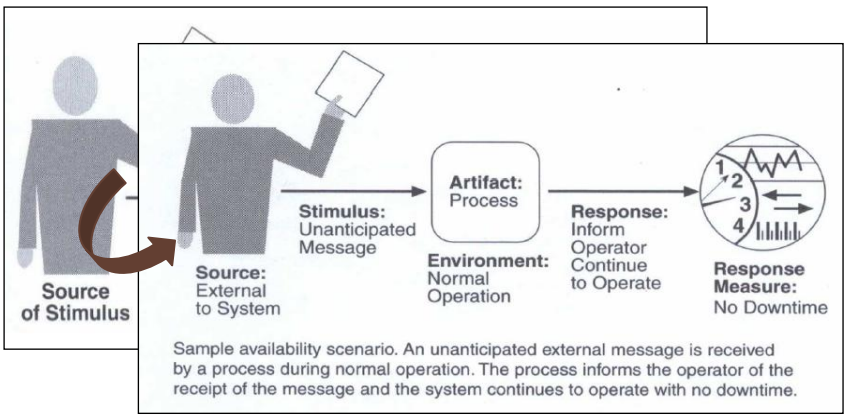
Main "architectural" building blocks

1. Quality attributes & **scenarios**
2. Architectural patterns & **tactics**
3. Architectural structures & **views**



Quality-attribute scenarios

- Textual descriptions that enable us to characterize aspects of quality attributes (in concrete terms), in such a way they can be evaluated and used in design



Modifiability template

Scenario Portion	Possible Values
Source	End-user, developer, system-administrator
Stimulus	Add/delete/modify functionality or quality attribute
Artifact	System user interface, platform, environment
Environment	At runtime, compile time, build time, design-time
Response	Locate places in architecture for modifying, modify, test modification, deploys modification
Response Measure	Cost in effort, money, time, extent affects other system functions or qualities

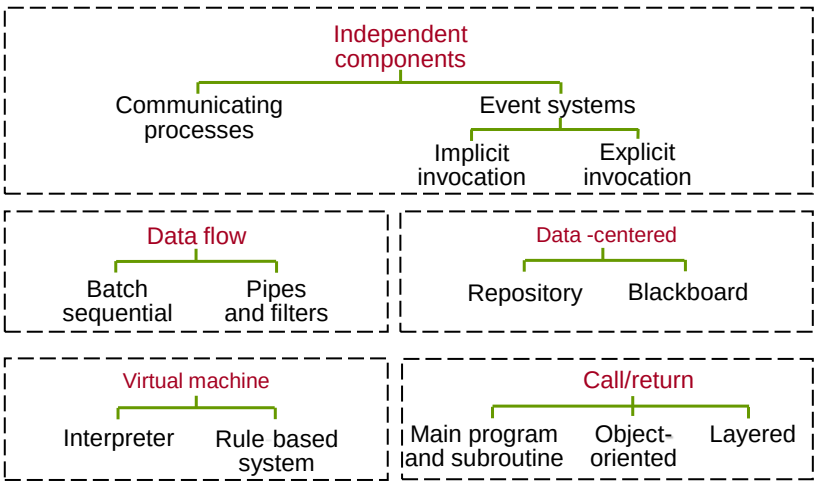
Other scenario templates (SEI)

- ~~Modifiability~~
- Performance
- Availability
- Security
- Usability
- Testability
- …
- *what about "energy consumption"?*

Architectural patterns

- Pioneer work by D. Garlan & M. Shaw in the '90s
  - Identification and cataloguing of high-level patterns (or styles) observed repeatedly in systems
  - The first styles were mostly in the category of components-and-connectors (runtime)
  - Emphasis on constraints/prescriptions, rather than on quality attributes
  - Provided an initial catalog, which has been refined/extended since then

Taxonomy of architectural styles

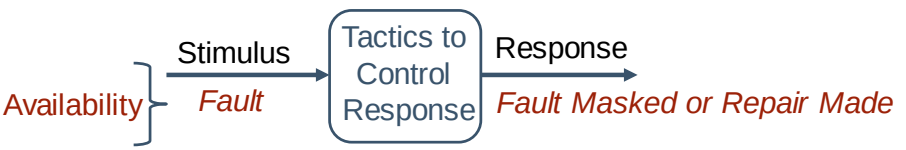


More architectural styles

- Model-View-Controller
- Broker
- Client-server (n-tier)
- …
- Service-oriented architectures?
- Cloud computing?
- MapReduce?
- …

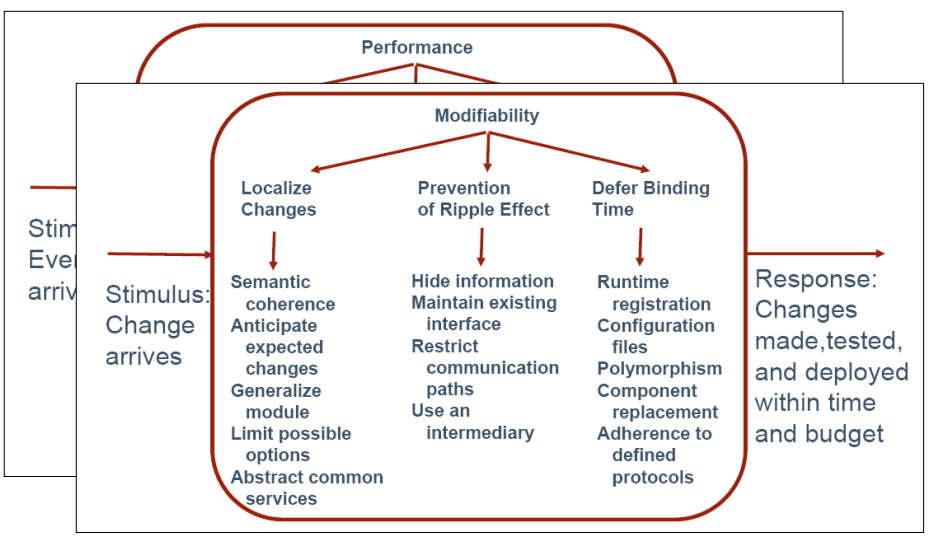
Architectural tactics

- A tactic is a **design decision** that influences the **control** over the response of a **single quality attribute**
- A collection of tactics leads to a "design strategy"
  - An architectural style is often a combination of several tactics
  - Styles target multiple quality attributes at once
  - Tactics can be used to "fine-tune" styles



21 J. Andres Diaz Pace – PUC-Rio, May 2014

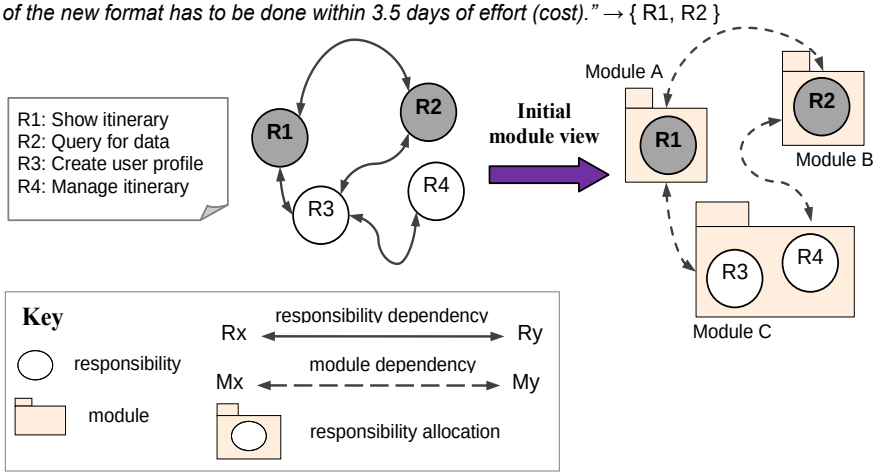
Catalog of tactics



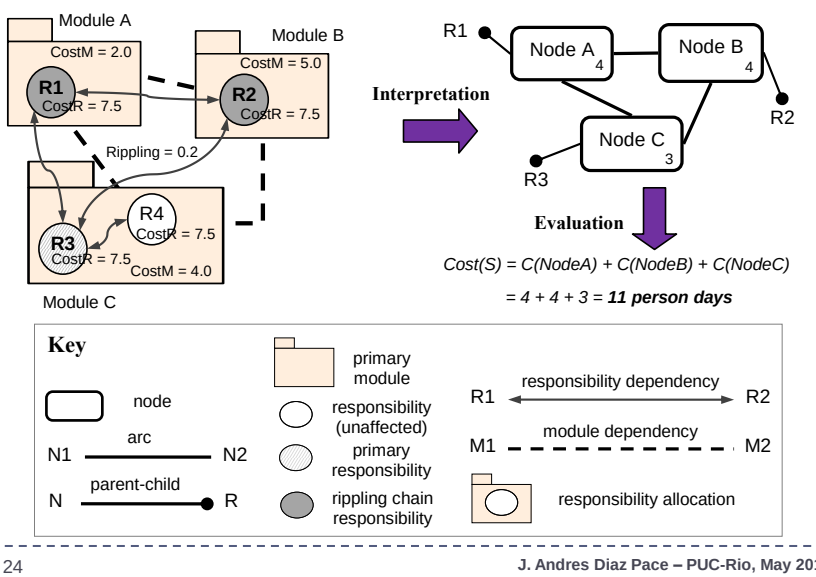
22 J. Andres Diaz Pace – PUC-Rio, May 2014

Modifiability: Initial situation

Scenario M1: *"Adding new features requires changes in the data format. The implementation of the new format has to be done within 3.5 days of effort (cost)."* → { R1, R2 }

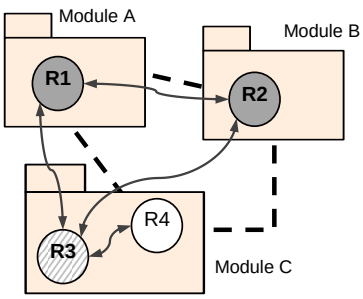


Modifiability: Change impact analysis



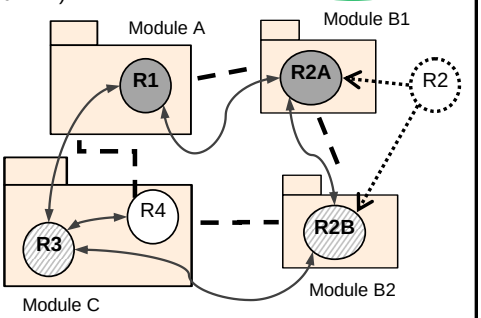
Modifiability: Split responsibility

S → **{ R1, R2 }** cost = 11.12



Split Responsibility R2
(R2 gets refined by responsibilities R2A and R2B)

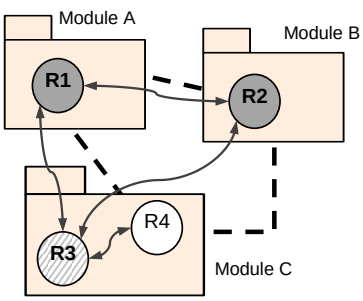
S' → **{ R1, R2A }**
cost = 9.95

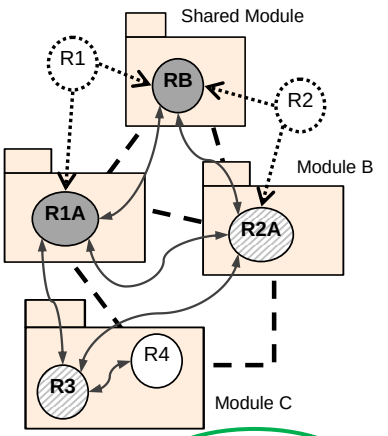


25 J. Andres Diaz Pace – PUC-Rio, May 2014

Modifiability: Abstract common service

S → **{ R1, R2 }** cost = 11.12





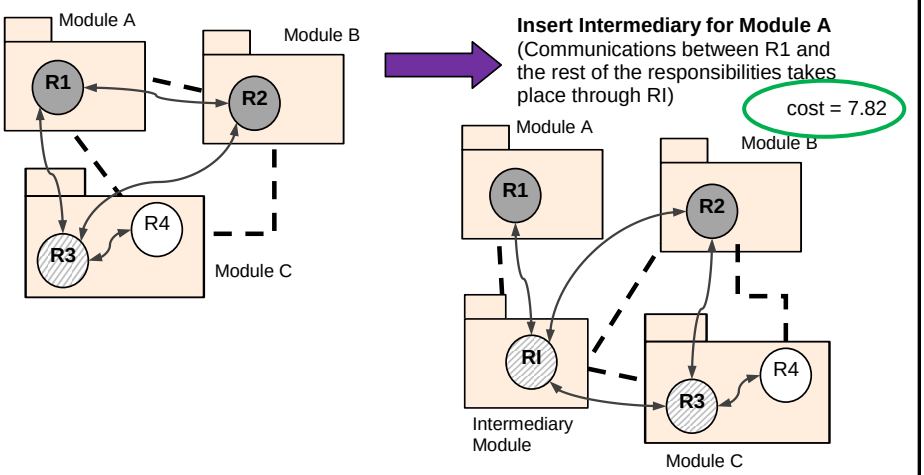
Abstract Common Responsibility RB out of R1 and R2
(R1 gets refined by responsibilities R1A and RB, while R2 gets refined by responsibilities R2A and RB)

S'' → **{ R1A, RB }**
cost = 9.13

26 J. Andres Diaz Pace – PUC-Rio, May 2014

Modifiability: Insert intermediary

$S \rightarrow \{ R1, R2 \}$ cost = 11.12



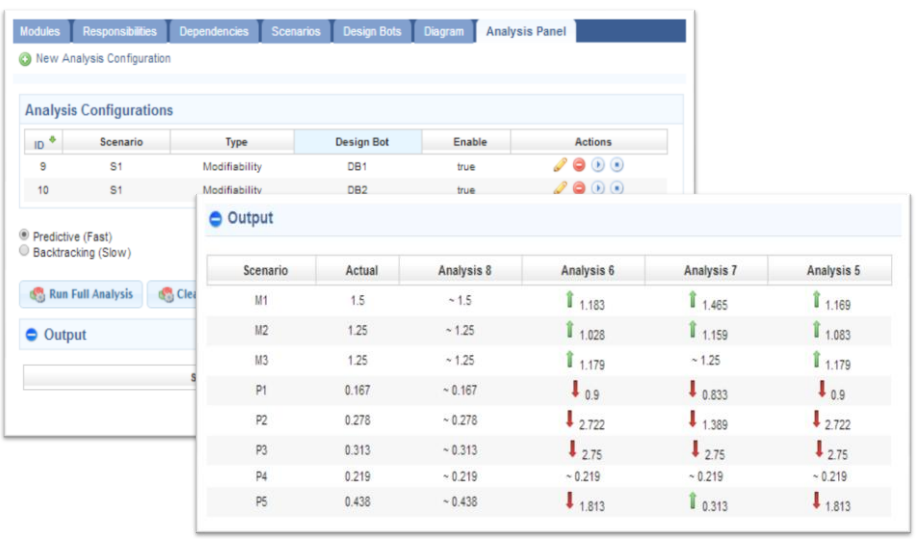
Other groups of tactics (SEI)

- ~~Modifiability~~
- ~~Performance~~
- Availability
- Security
- Usability
- Testability
- …
- *what about "big data"?*

Modifiability: Patterns & tactics

Pattern	Modifiability									
	Increase Cohesion		Reduce coupling					Defer Binding Time		
	Maintain Semantic Coherence	Abstract Common Services	Use Encapsulation	Use a Wrapper	Restrict Comm. Paths	Use an Intermediary	Raise the Abstraction Level	Use Runtime Registration	Use Start-Up Time Binding	Use Runtime Binding
Layers	x	x	x		x	x	x			
Pipe-and-Filter	x		x		x	x			x	
Blackboard	x	x			x	x	x	x		x
Broker	x	x	x		x	x	x	x		
Model-View-Controller	x		x			x				x
Presentation-Abstraction-Control	x		x			x	x			
Microkernel	x	x	x		x	x				
Reflection	x		x							

Research: ArchE + DesignBots

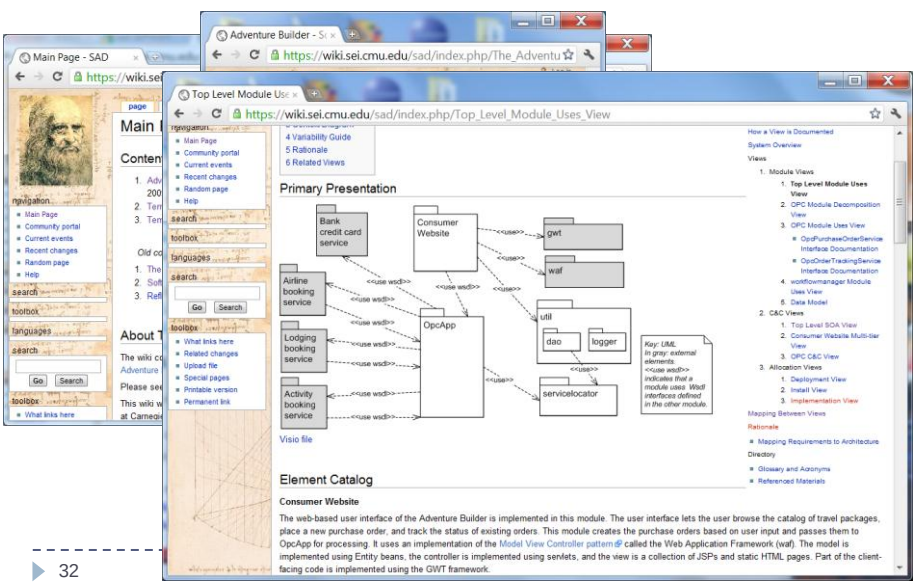


Documenting software architectures

- It produces a **Software Architecture Document** (SAD)
- Why should I care?
  - It is the "container" of the quality-attribute decisions of the system → **knowledge sharing and communication**
  - It supports early analysis and risk identification
  - It defines the work assignments of the project
  - It helps with post-deployment maintenance

- The SAD has 2 roles:
  - Description
  - Prescription

SAD (SEI) https://wiki.sei.cmu.edu/sad/



Which views are relevant?

- The answer depends on:
  - The specific stakeholders of the system
  - The way in which the documentation will be used
  - The quality attributes that are at work
- Common usages of documentation include:
  - **Education**: new people joins the project
  - **Communication**: among architects, with stakeholders
  - **Analysis**: ensure that certain quality-attribute properties are properly engineered

33 J. Andres Diaz Pace – PUC-Rio, May 2014

Main view-types

- In general, an architectural view consists of a set of elements, relations, and properties, which have a well-defined meaning

Modules → *construction, change impact analysis, testing*
  - Decomposition style
  - Uses style

Components-and-connectors → *performance, availability*
  - Client server, peer-to-peer styles
  - Communicating processes style

Allocation→ *availability, security, performance*
  - Deployment style

34 J. Andres Diaz Pace – PUC-Rio, May 2014

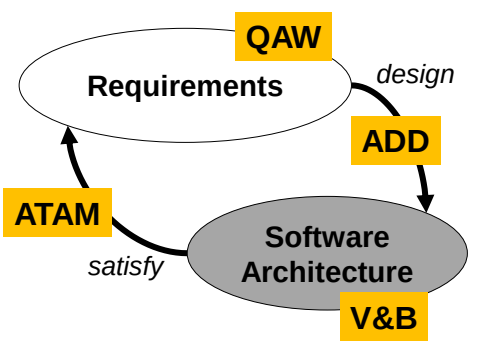
What we have (so far)

- **Quality-attribute scenarios** are drivers for
- **design decisions** (e.g., patterns, tactics)
- which are documented using **architectural views**

- **Methods** can provide "predefined workflows" to speed-up architecting activities
  - Elicitation/prioritization of QA drivers from stakeholders
  - Iterative design decomposition, guided by QAs
    - Not a detailed design, just provide evidence that QAs are met
  - Analysis of a given "design state"
    - Often, the final architecture

35 J. Andres Diaz Pace – PUC-Rio, May 2014

The Design Cycle (SEI)

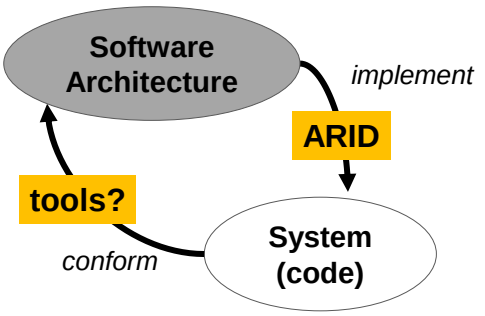


- Definition (and evaluation) of the **design structures** that support the key quality attributes
  - QA elicitation
  - Design
  - Documentation
  - Evaluation

- **The goal is to design a system that is well-aligned with its business goals**

36 J. Andres Diaz Pace – PUC-Rio, May 2014

The Implementation Cycle (SEI)

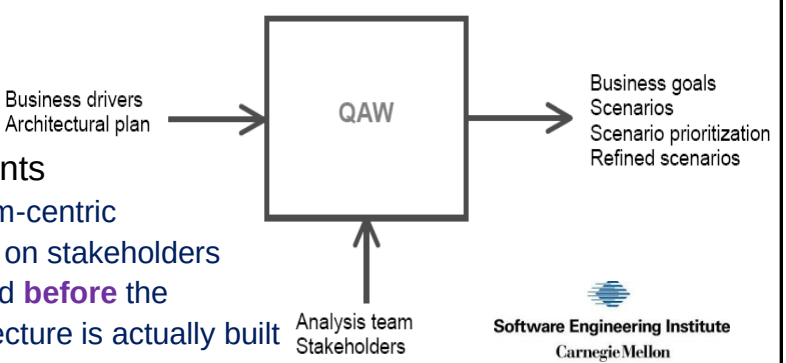


- Mapping between architectural information and code
  - Usually, based on the module view
  - Communication between the architects and developers
  - Conformance checks

- **The goal is to implement the system following the guidelines/rules of the architecture**

Quality Attribute Workshop (QAW)

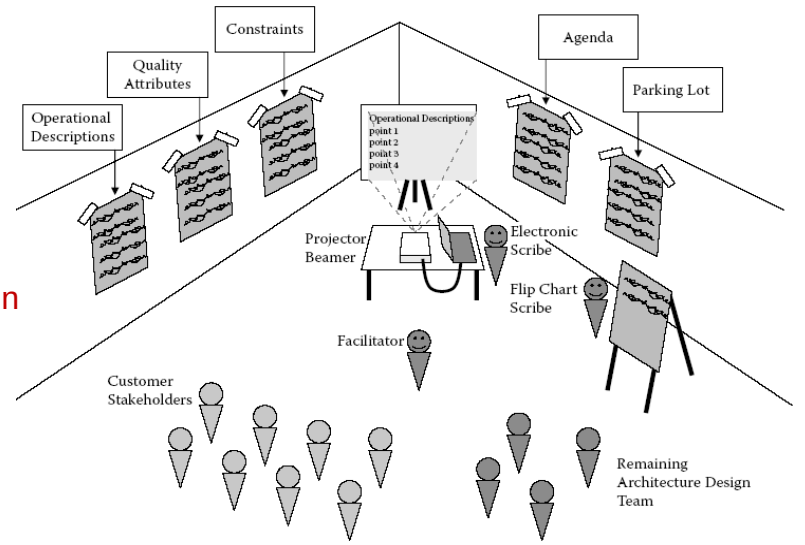
- An SEI facilitated method that involves the **system stakeholders** from early stages, in order to discover relevant quality attributes for the system



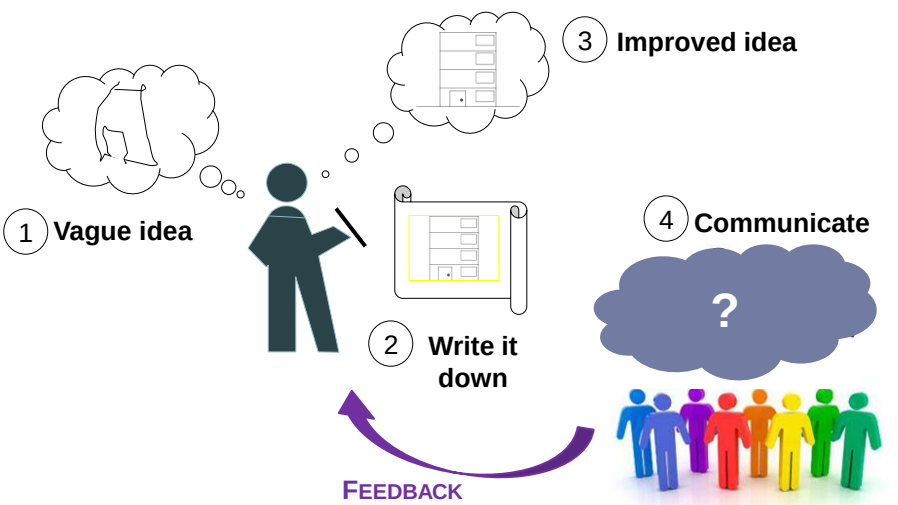
- Key points
  - System-centric
  - Focus on stakeholders
  - Applied **before** the architecture is actually built

Typical workshop setup

Also applied in ATAM

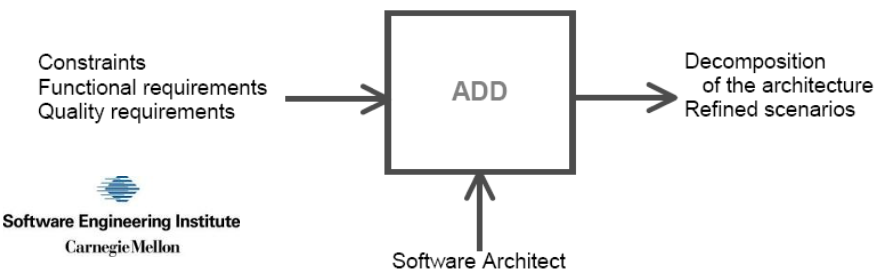


The value of designing (& documenting)



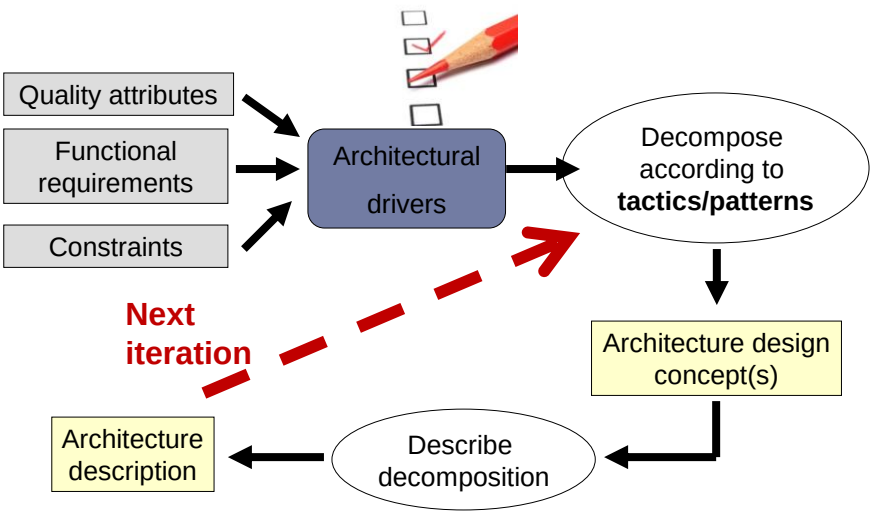
Attribute Driven Design (ADD)

- An SEI method that follows a recursive design process, based in the decomposition of the solution by means of architectural patterns and tactics
  - The quality-attribute scenarios steer the iterations and decomposition
  - The process ends once all quality-attribute drivers are satisfied



41 J. Andres Diaz Pace – PUC-Rio, May 2014

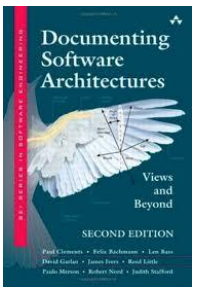
ADD: Flow of iterations



42 J. Andres Diaz Pace – PUC-Rio, May 2014

Views & Beyond (V&B) – The view zoo

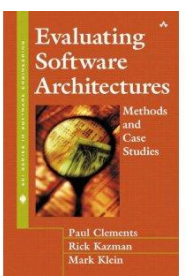
- Module viewtype
  - Decomposition style
  - Uses style
  - Generalization style
  - Layered style
- Component-and-connector viewtype
  - Pipe-and-Filter style
  - Shared-Data style
  - …
- Allocation viewtype
  - Deployment style
  - Implementation style
  - Work assignment style



Software Engineering Institute
Carnegie Mellon

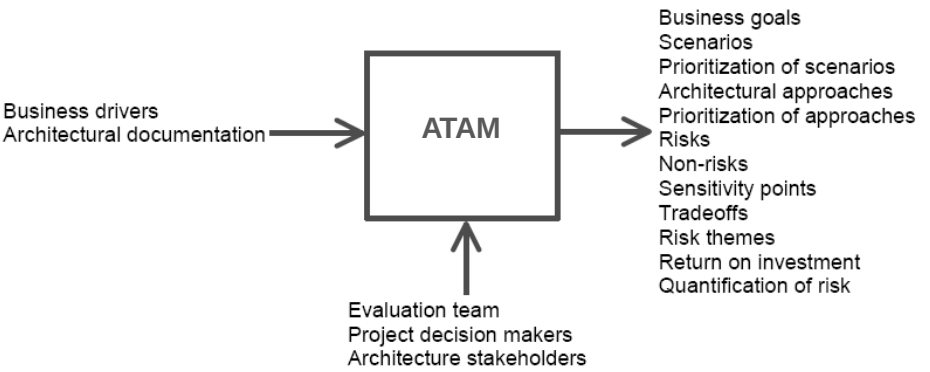
Architecture Tradeoff Analysis Method (ATAM)

- Why evaluating an architecture?
  - All design solutions involve **tradeoffs**!
    - In part, because of quality-attribute considerations
- The architecture is produce in early stages and captures **-significant design decision**
- It is an SEI method for analyzing the **consequences of architectural decisions** with respect to quality attributes
- Early **risk detection**
  - Trends rather than precise results
- Forces the recording of architectural information

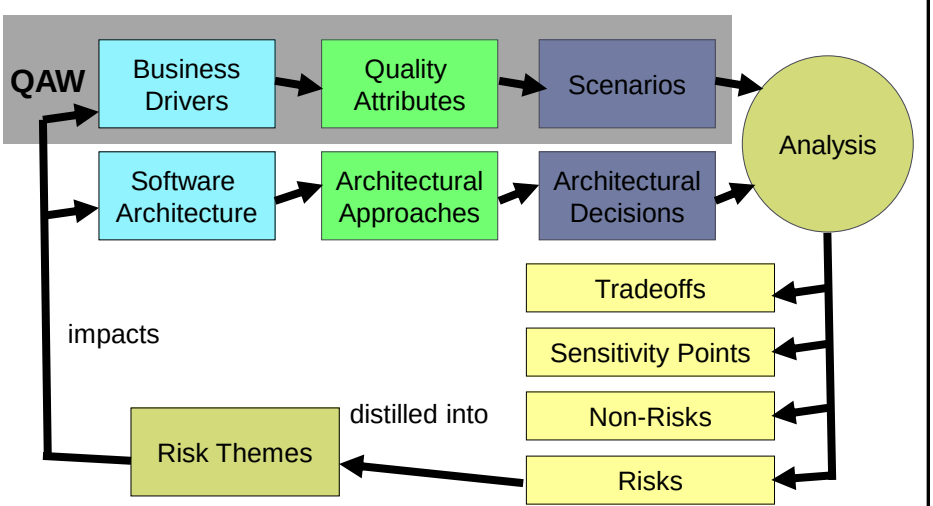


Software Engineering Institute
Carnegie Mellon

ATAM: Inputs/outputs & roles

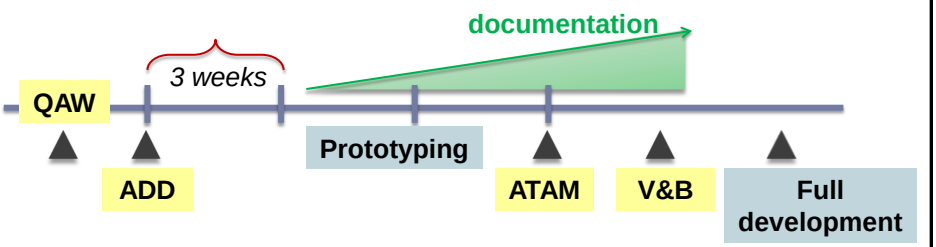


ATAM: Flow of activities



Can these methods be combined?

- The tailoring depends on the characteristics of the project
  - Schedule
  - Criticality of the quality attributes
  - Prior experience of personnel
  - System size



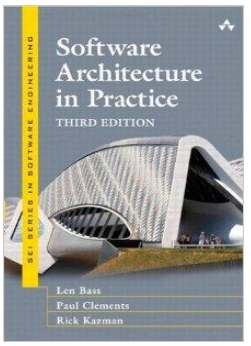
47 J. Andres Diaz Pace – PUC-Rio, May 2014

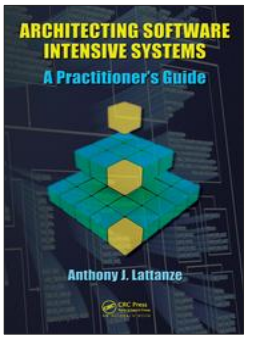
Summary

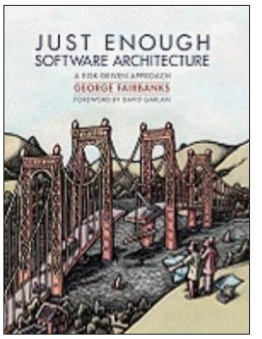
- The quality and longevity of a software system is determined to a great extent by its architecture
  - Early identification of architectural risks saves time and money
- Benefits of architectural analysis and design
  - Forces an articulation between business goals and QAs
  - Results in the prioritization of (conflicting) goals/scenarios
  - Forces a clear explanation of architectural approaches used, which then provides a plan/guidelines to development teams
  - Improves the quality of the product
- Different artifacts, techniques, and methods
  - Either proposed by SEI or by others

48 J. Andres Diaz Pace – PUC-Rio, May 2014

For more information







49 J. Andres Diaz Pace – PUC-Rio, May 2014

Final comments

"If you think **good** architecture is expensive, try bad architecture"

-- Brian Foote and Joseph Yoder

49 J. Andres Diaz Pace – PUC-Rio, May 2014

Thank you!

adiaz@exa.unicen.edu.ar
or
adiazpace@gmail.com