Architecting Software the SEI Way

Essential Steps Toward Mastery

February 28, 2012 • 1:00pm–4:30pm EST

Architecture Evaluation: A Tool for Designing Systems That Meet Users' Needs

Felix H. Bachman
Senior Technical Staff

Felix H. Bachmann is a Senior Member of the Technical Staff at the Software Engineering Institute (SEI) working in the Product Line Systems Program on both the Architecture Tradeoff Analysis and Product Line Practice Initiatives. There he is the team lead for architecture-centric product line practices, a co-author of the Attribute-Driven Design Method, a contributor to and instructor for the ATAMSM Evaluator Training, a co-author of Documenting Software Architectures: Views and Beyond.

See his full bio at:
www.sei.cmu.edu/go/architecting-software-the-sei-way

Do You Have the Right Architecture?

Polling Question #1

What is your opinion about architecture evaluation?

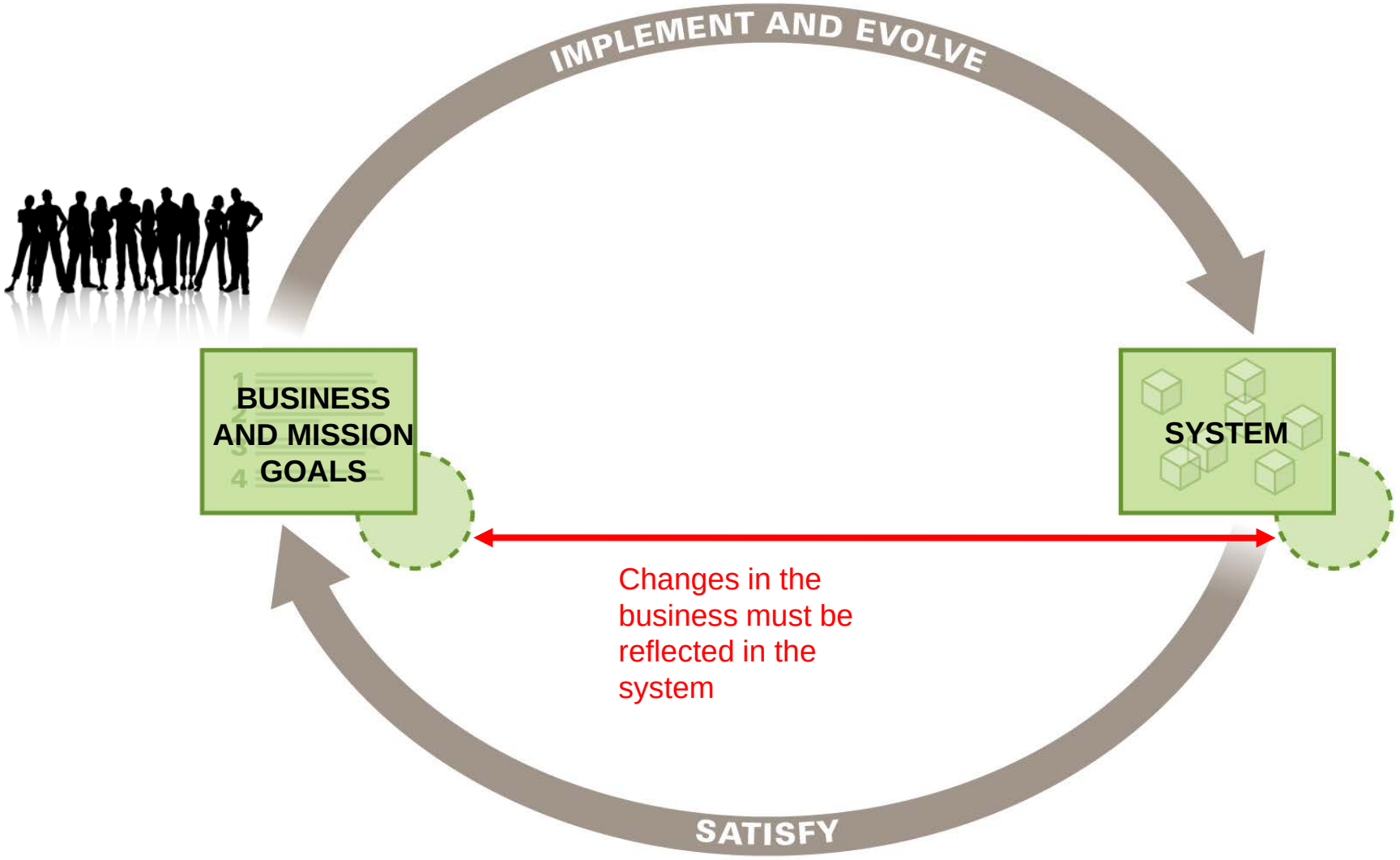
a) It is very helpful

b) It is very helpful, but too much effort

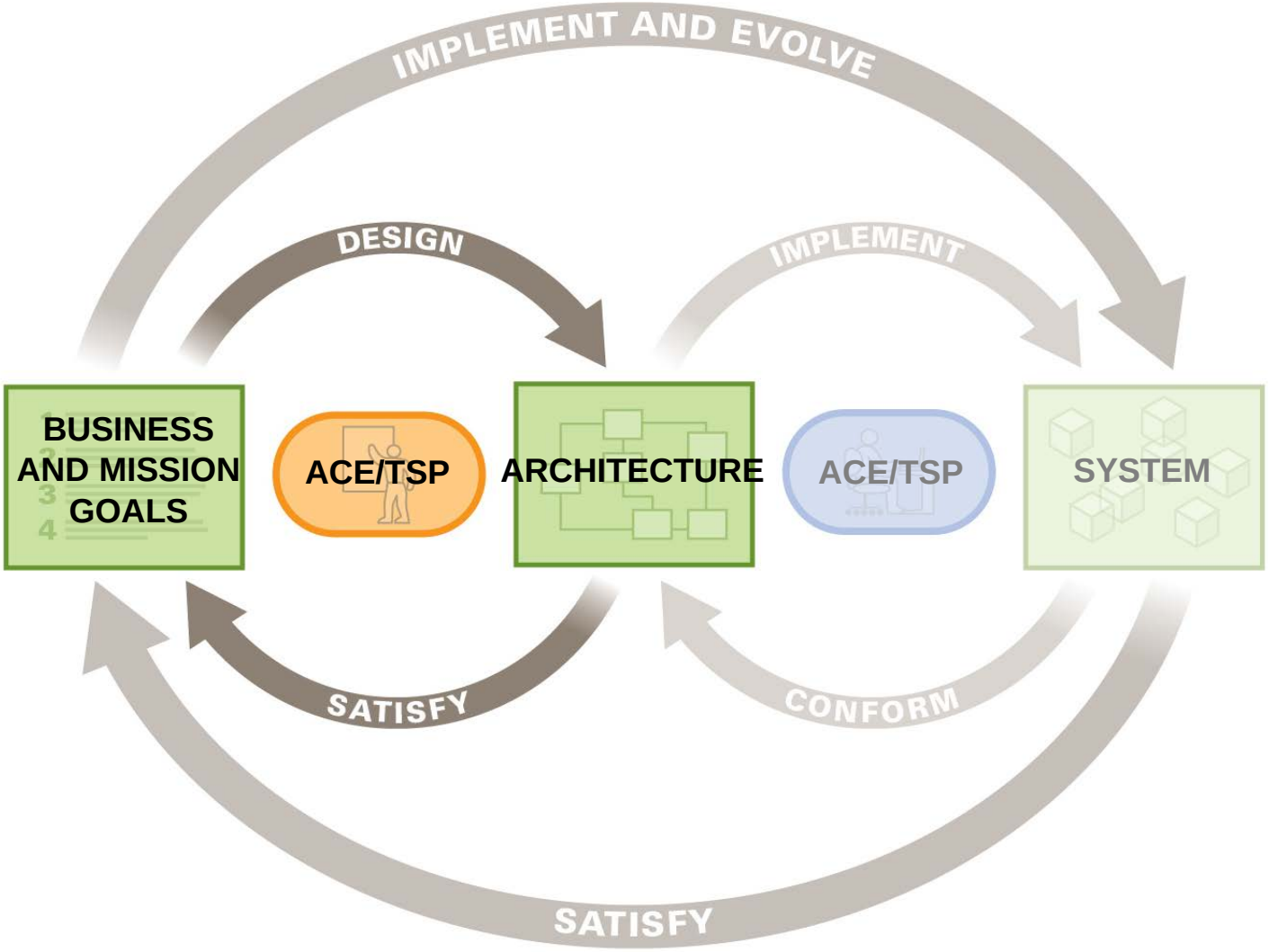
c) It does not have a lot of benefits

d) Never participated in an evaluation

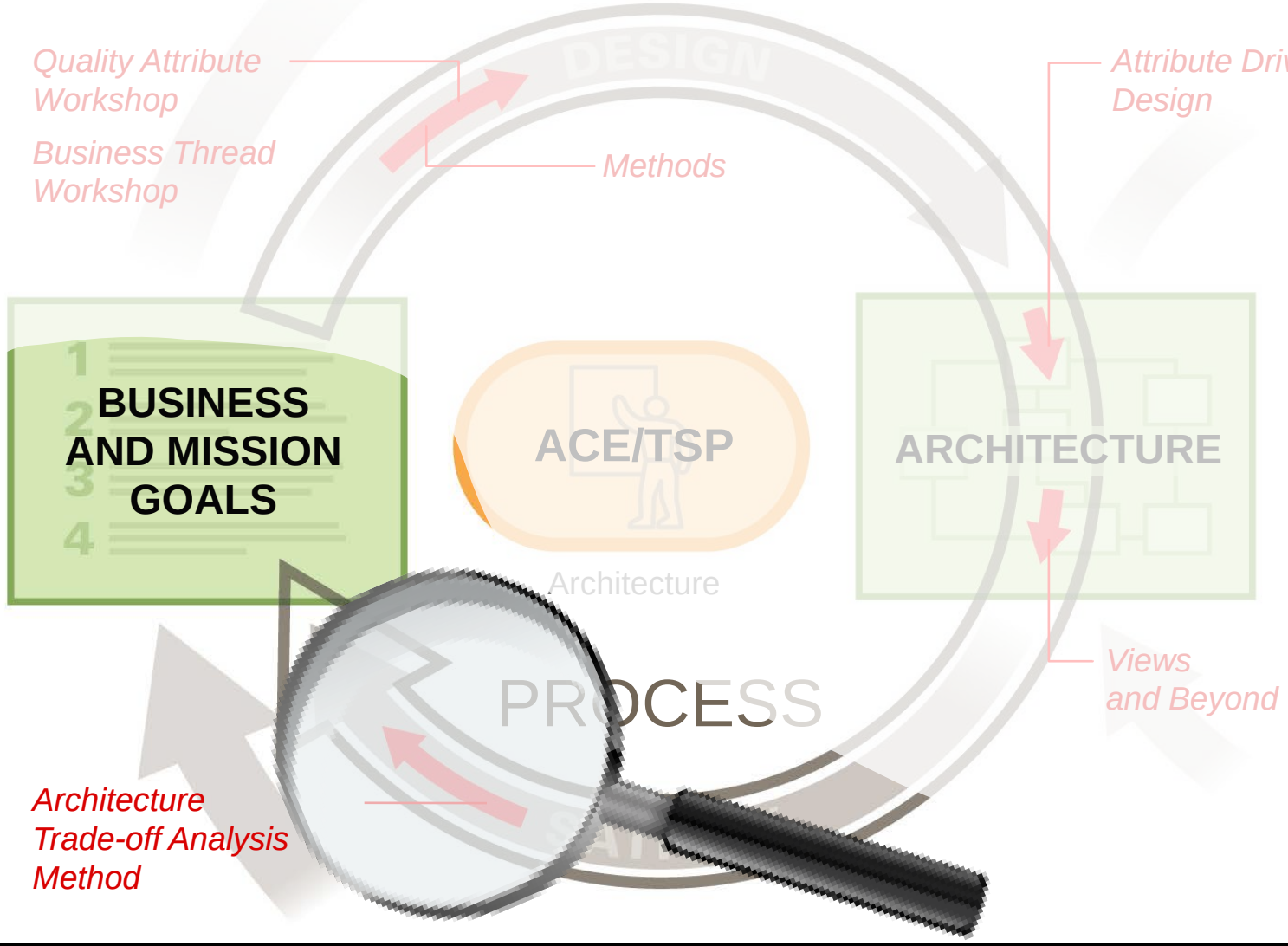
System Purpose



Architecture Centric Engineering



Architecture Centric Engineering



Purpose of Architecture Evaluation

The purpose of an architecture evaluation is to answer the following question:

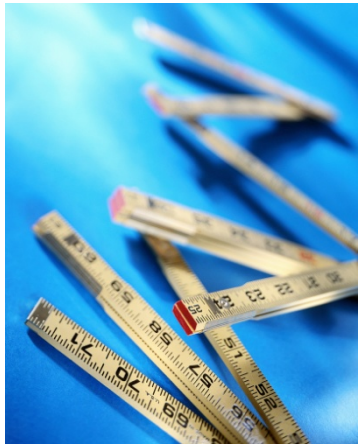
Will the designed system solution have the properties to make the organization successful?

1) Requires a definition of what it means for an organization to be successful
2) Requires an understanding of the designed system properties

System properties include the functions the system must perform, but more importantly, how well those functions work.

Every architecture of the system, whose properties achieve the definition of success, is a good architecture.

Things to Establish for Architecture Evaluation



An objective measure

An understanding of the architecture

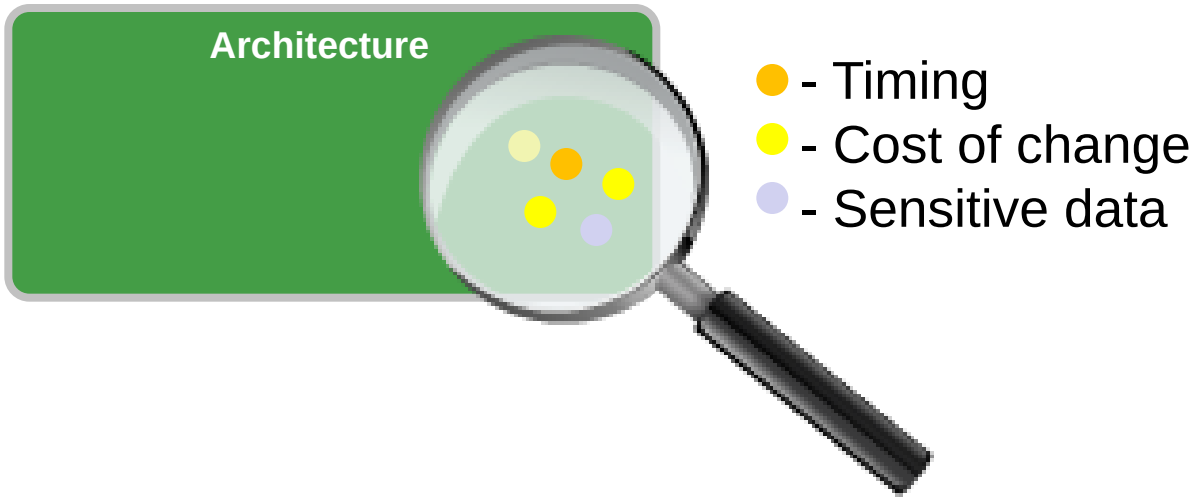
The analysis

An Objective Measure

Building the Yardstick – 1

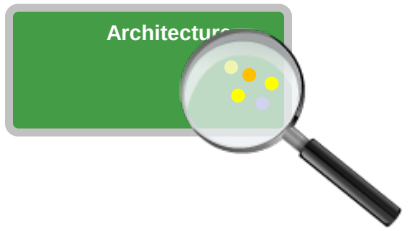
Without a yardstick to measure, every architecture is good or bad.

Will the designed system solution have the properties to make the organization successful?



System properties of interest here are the quality attribute properties of the functions the system has.

Building the Yardstick – 2



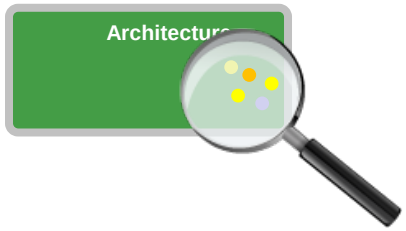
Every function of every system has quality attribute properties.

- Some are known
- Most are unknown

Even if the quality attribute properties are known, how do you know that those are “good” properties?

- If a function provides a result in 0.5 sec. *Is that good?*
- If a function crashes only once per week. *Is that good?*
- If only one credit card transaction per month is compromised. *Is that good?*

Building the Yardstick – 3



Fortunately, most of the system properties do not need to be known as long as they are within a certain range.

- As long as every request is served with an average time of 2 seconds it is a good system.
- It is acceptable that something goes wrong as long as the user does not lose any data

There are only a few functions that must have some important quality attribute properties.

A good yardstick will specify those important quality attributes in a measurable way.

Stakeholders

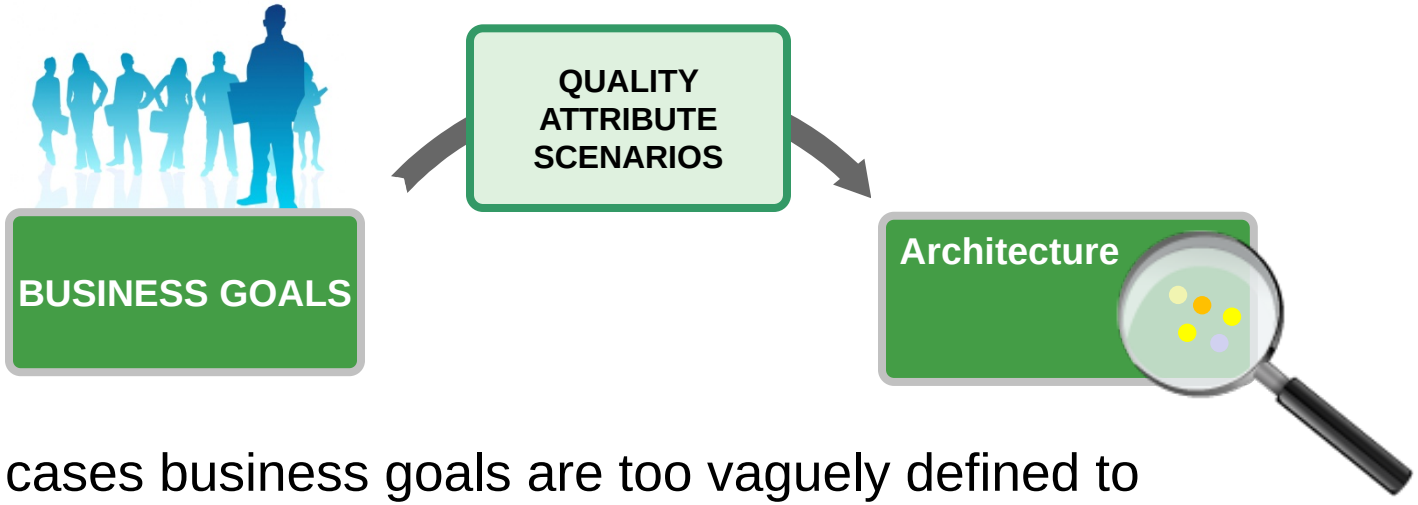
BUSINESS GOALS

Only the stakeholders of an organization can specify what the expected important quality attributes are.

- We need to have a system that can easily be adapted to new market conditions.
- Our systems cannot have any defects in the field
- We guarantee 24/7 availability

These statements (business goals) describe how the organization plans to be successful with their business.

Quality Attribute Scenarios

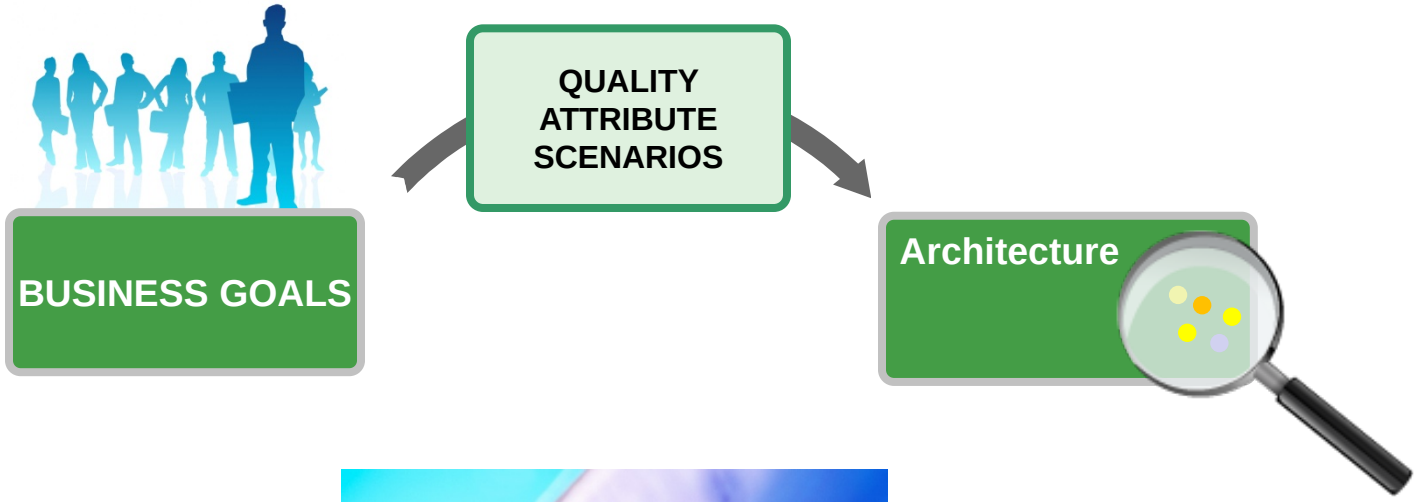


In most cases business goals are too vaguely defined to actually be able to measure their achievement.

Quality attribute scenarios bridge business goals and architecture quality attribute properties.

They describe what quality attribute properties the system is expected to possess to run a successful business.

The Yardstick



A customer requires a new auction algorithm. A developer implements and integrates that function within one day of effort.

An error occurs in a fielded system. The system recovers from the error without manual intervention within 5 seconds.

The Yardstick – Summary

Principle 1: Important quality attribute properties of the architecture need to be evaluated.

Principle 2: What is important is derived from the business goals.

Principle 3: Quality attribute scenarios translate business goals into required quality attribute properties.

Understanding the Architecture

Understanding the Architecture – 1

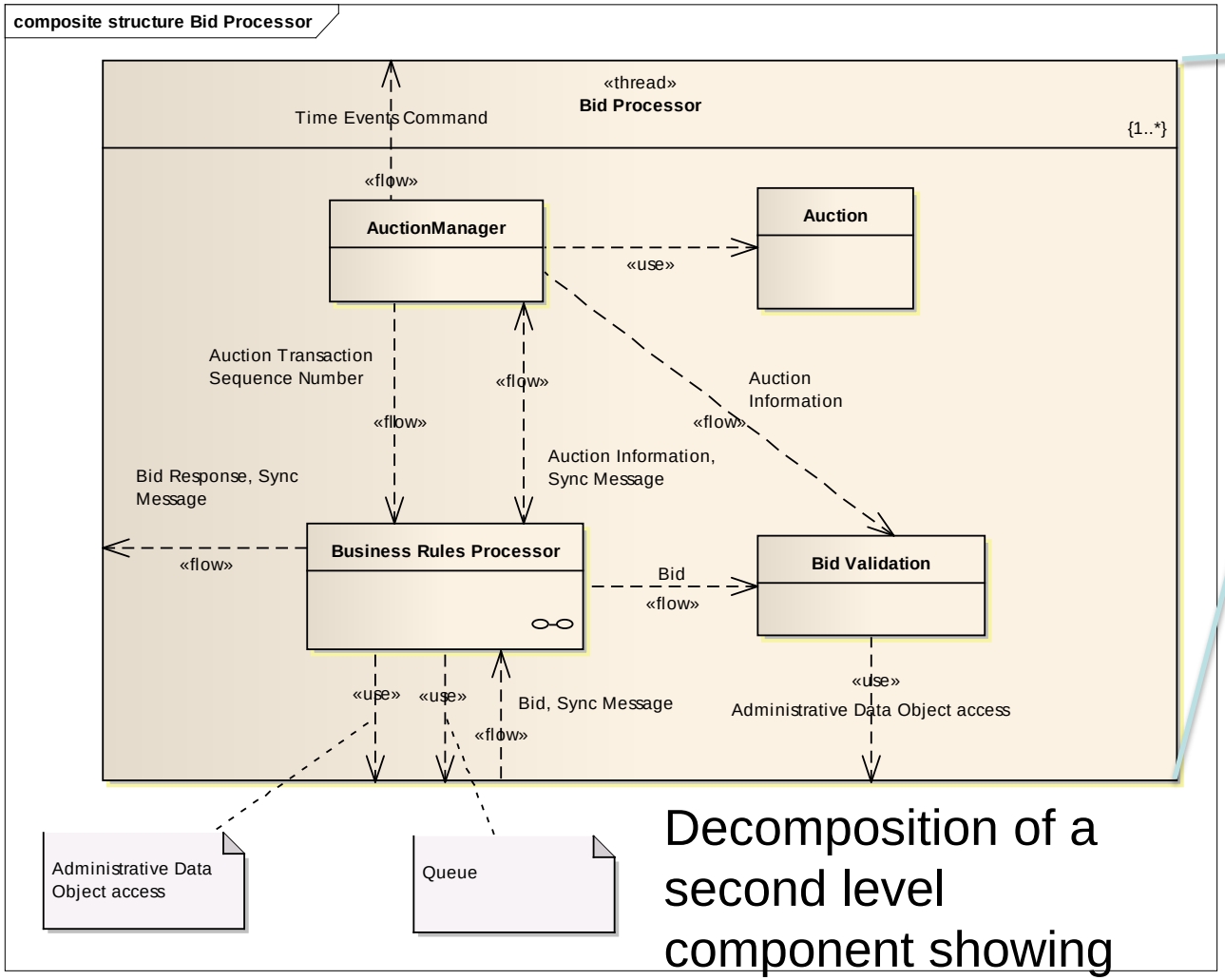
Recall:

To evaluate a system's architecture, an understanding of the system quality attribute properties is required.

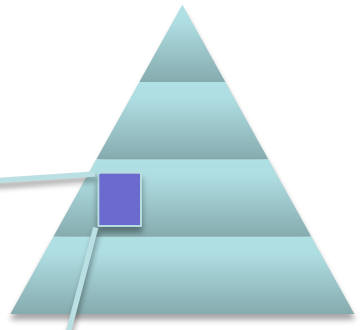
- Very seldom are those properties known and understood
- Almost never are they documented

Architectures of non-trivial systems are complex and it is difficult to understand everything in a short amount of time

Example Component Diagram



Decomposition of a second level component showing the third level



A typical piece of architecture documentation

What are the properties here?

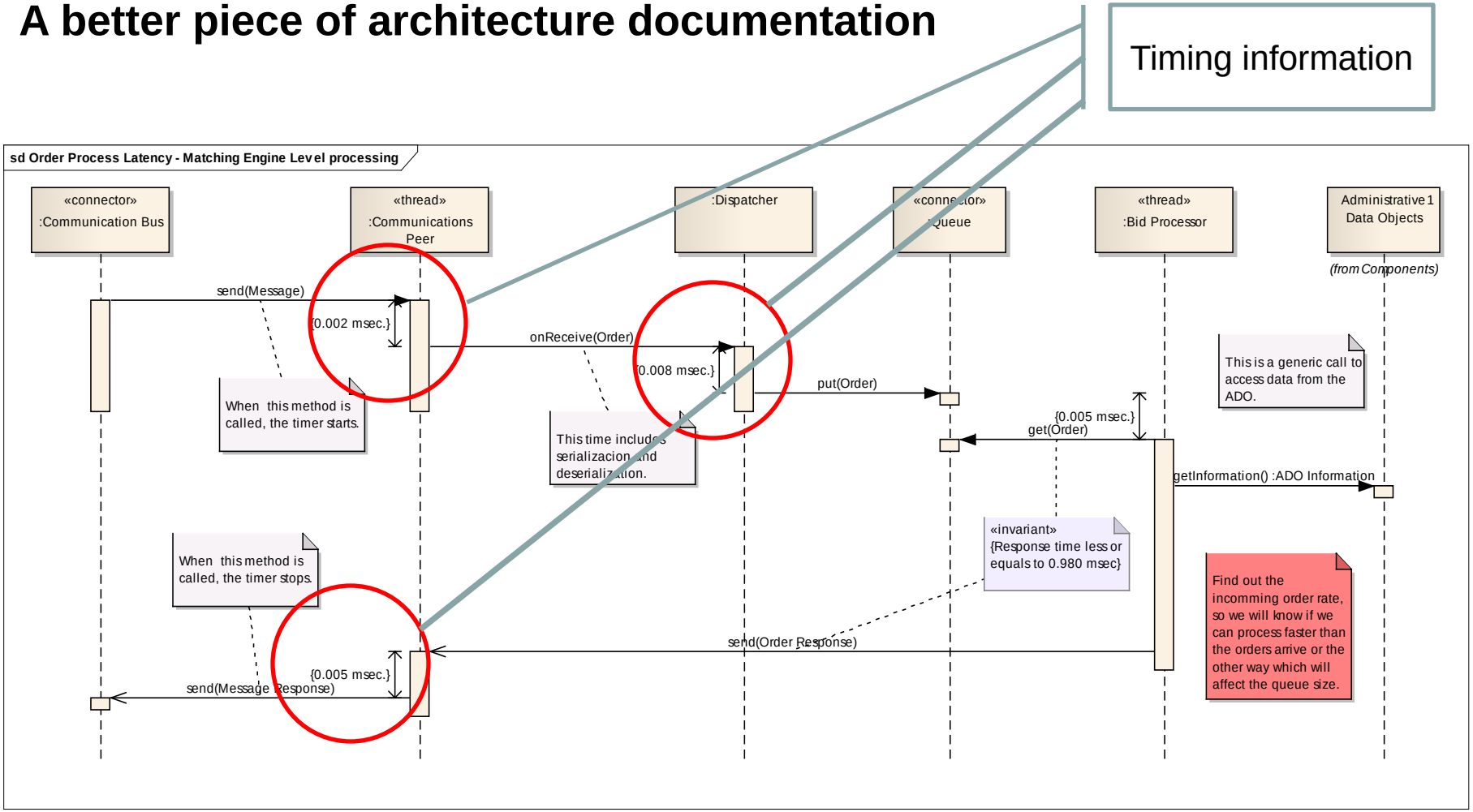
Where are they?

What can we say about:

- Performance?
- Modifiability?
- Security?

Sequence Diagram with Timing

A better piece of architecture documentation



Architecture Approaches

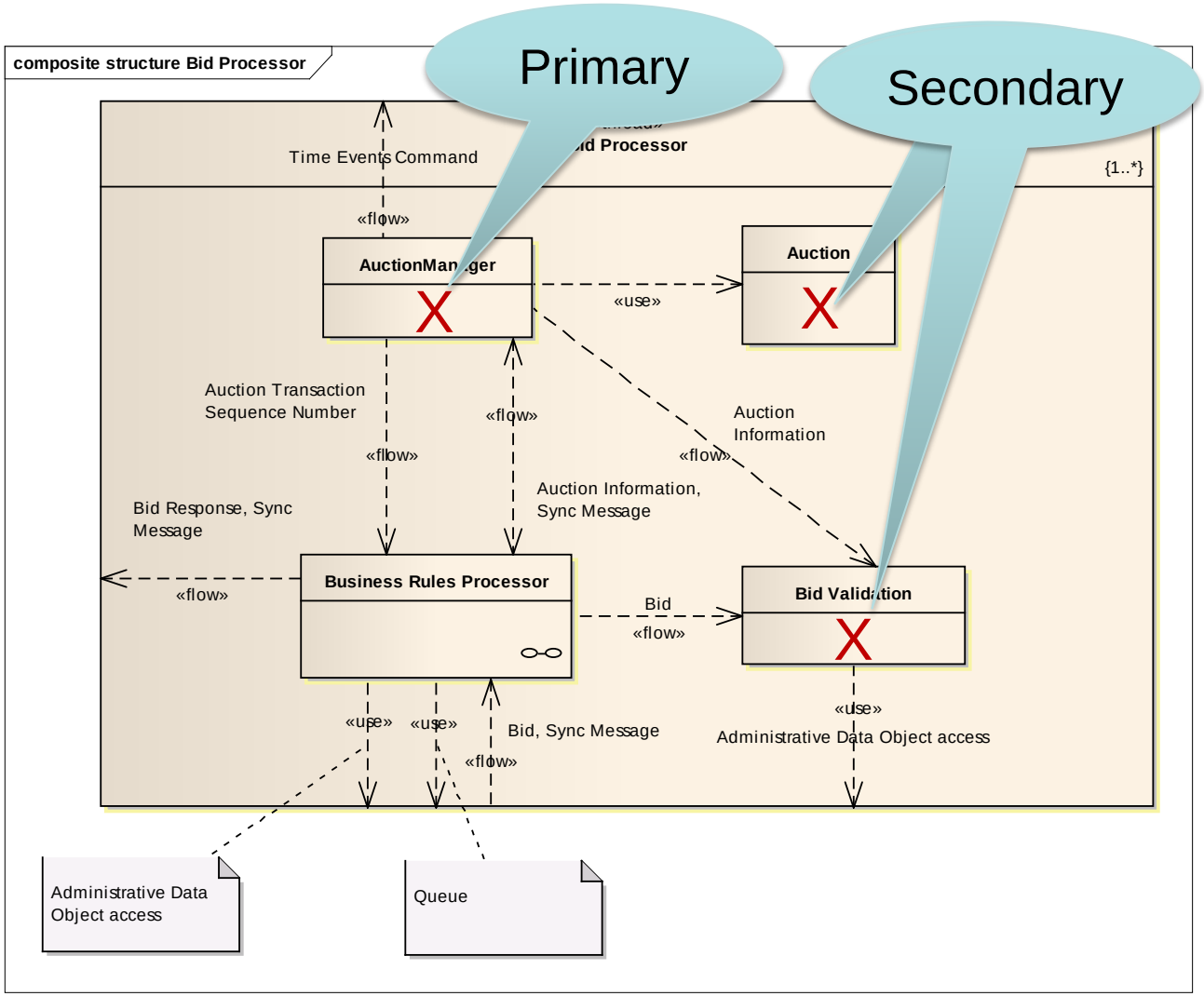
Narrow down the problem by focusing on a quality attribute scenario:

> *A customer requires a new auction algorithm. A developer implements and integrates that function within one day of effort.*

First question to ask:

> What are the components involved in this scenario

Example Component Diagram



Architecture Approaches

Second question to ask:

What are concepts put in place to make this change easy?

- Localized everything that needs to be changed in one component
- Data structures have a generic interface that allows to hide internal changes
- Component also has a versioned interface to allow backward compatibility

These are architecture approaches to support this extensibility scenario.

Architect vs. Documentation

An architecture documentation that can be used for evaluation needs to show where those concepts are and what their properties are:

- Adding a new algorithm means specializing the generic class "AuctionManager". *A typical algorithm can be implemented in four hours.*
- New data structures can be added, but existing data structures cannot be changed without any impact on existing functions. *Very seldom new structures are required. If so, they can be added within one hour.*
- Every component working with the AuctionManager requests an interface of a specific version. *Creating a new version of an interface will take about one day*.

If this information is not in the architecture document then the answers have to come from the architect.

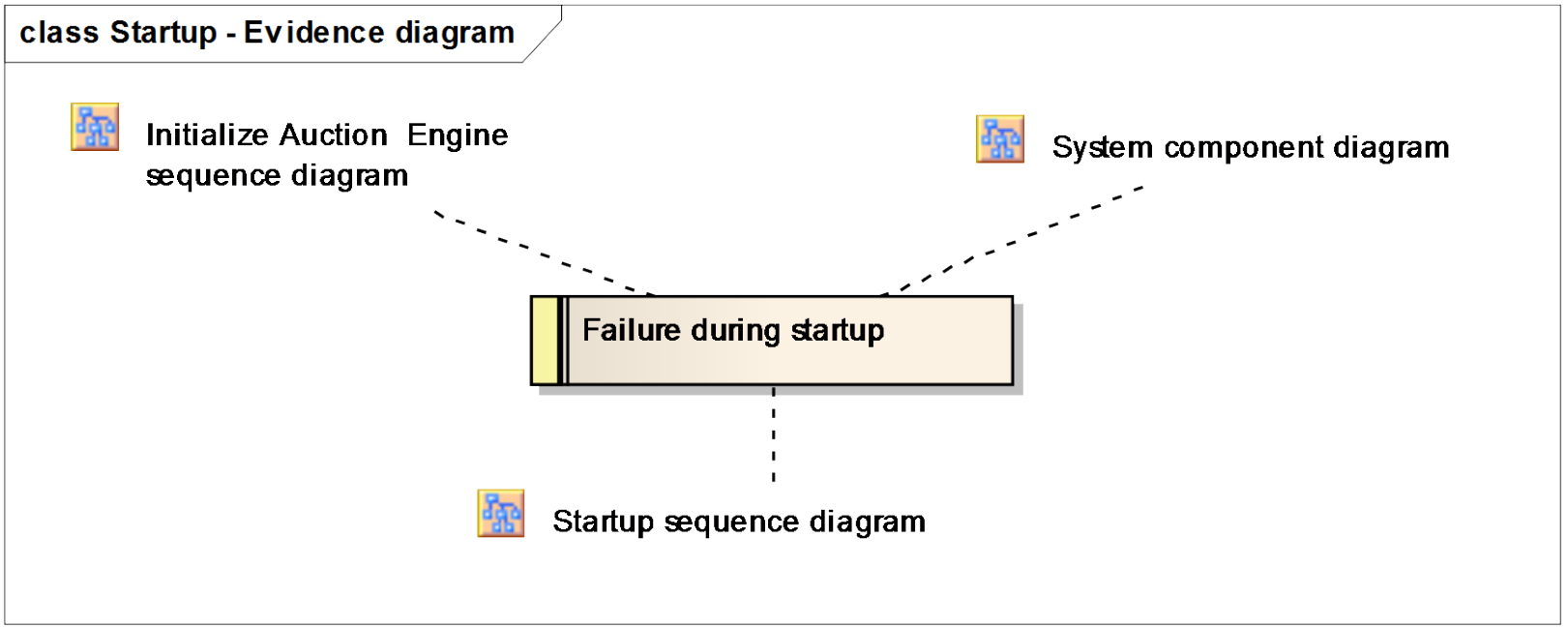
Side Effects

Every architecture approach used also has negative impact on other quality attributes:

- Putting everything that needs to change in one place may introduce unnecessary dependencies to other components. Bad for security and other types of changes
- Data structures with generic interfaces may impose a performance penalty
- Versioned interfaces increase complexity, which is more difficult to test and the change for system crashes increases

The architect needs to be aware of these issues, needs to put mitigations into place, and document them.

Scenario based View Packets



For evaluation purposes, architecture documentation should be organized as view packets around quality attribute scenarios.

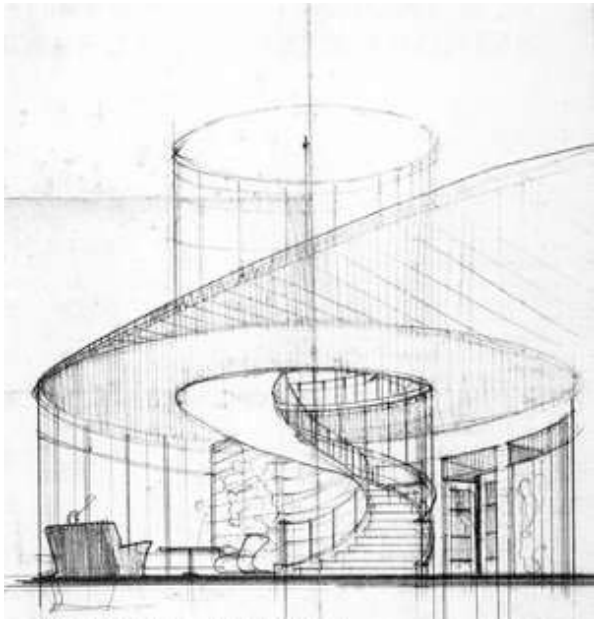
- A view packet contains parts of other views for a specific purpose. (In our case for evaluation purposes)

Understanding the Architecture – Summary

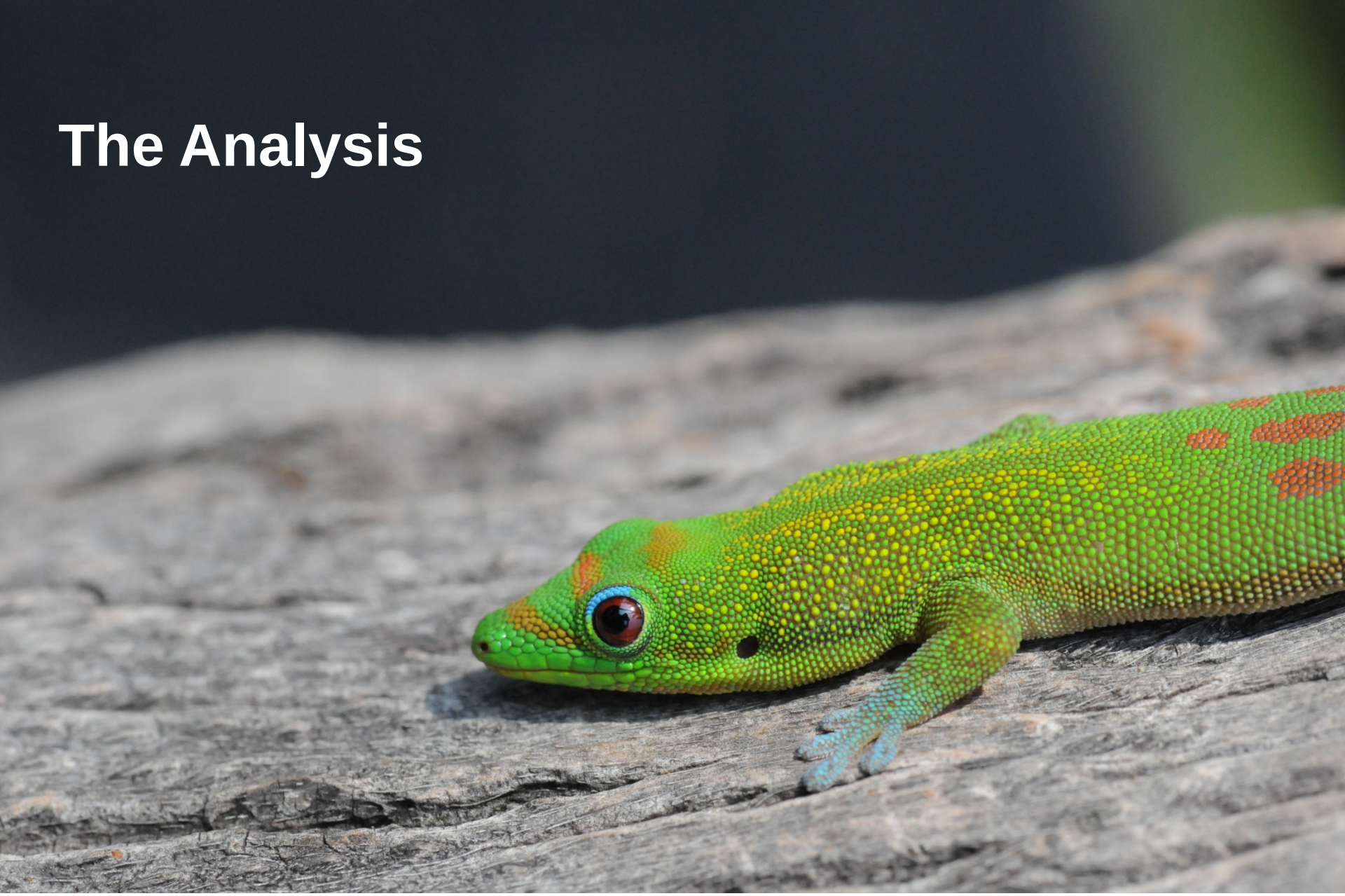
Principle 4: Identify relevant components by using scenarios.

Principle 5: Identify architecture approaches with their quality attribute properties

Principle 6: Identify the side effects of those architecture approaches.



The Analysis



Matchmaking – 1

So far:

- We created the yardstick
- We extracted the important quality attribute properties of interest from the architecture.

The architecture evaluation process has to answer the following questions:

> *Do the quality attribute properties of the identified components support the given quality attribute scenario sufficiently?*
>
> *Is the negative impact of the chosen architecture approaches on other quality attribute scenarios acceptable?*

Matchmaking – 2

It is the architect of the system that has to answer these questions.

- If the answers are documented, then examining the documentation is sufficient for the evaluation

- If not, the architect has to provide the answers in an interview.

The evidence provided must be sufficient to convince quality attribute experts and the stakeholders.

Risks

No system design is without any risks.

Successful organizations understand and handle the risks.

An architectural risk is when

- The system properties do not completely satisfy the given quality attribute scenarios
- The approaches used for one scenario negatively impact other scenarios.

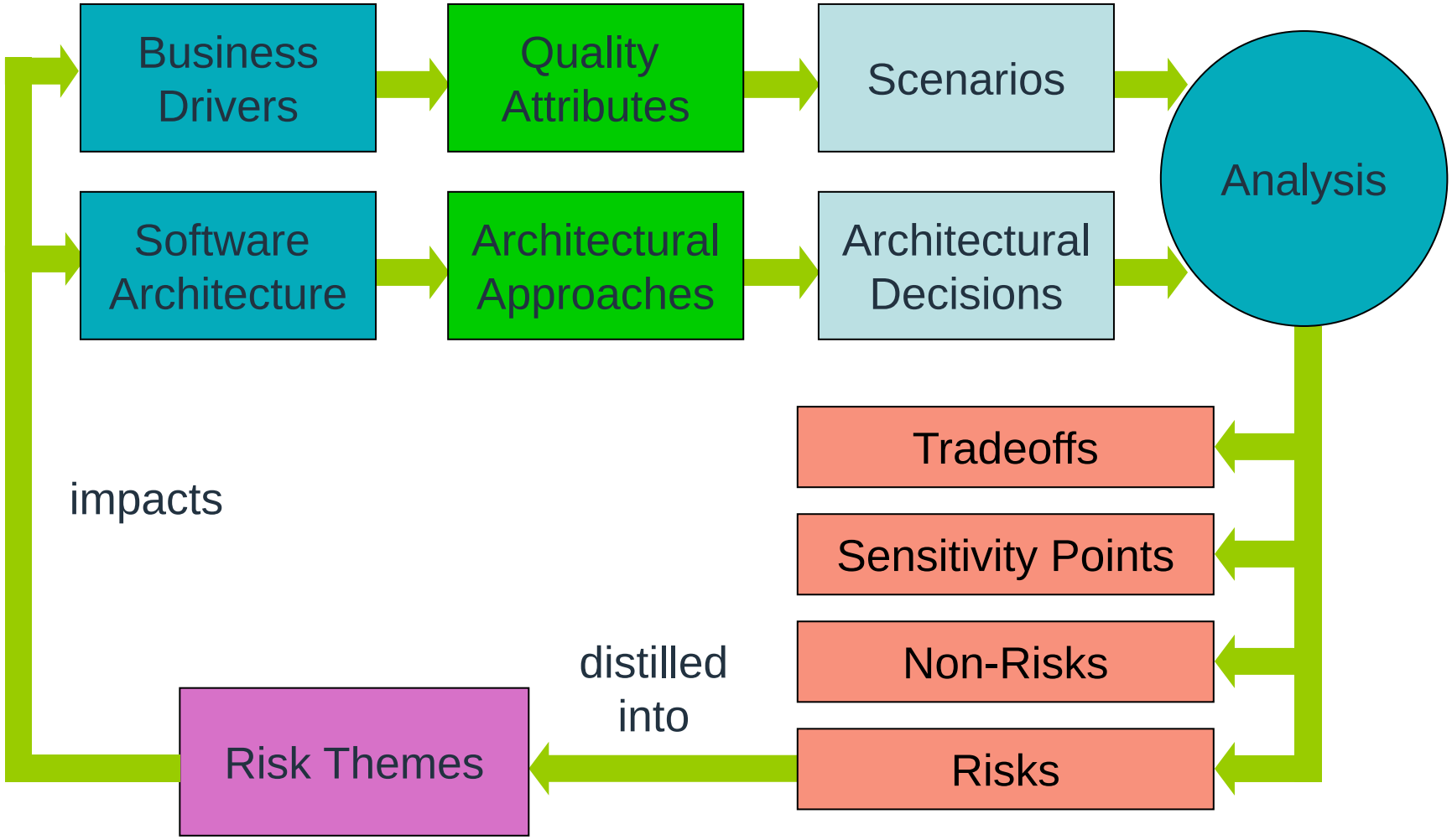
If an important scenario cannot be supported, then one or more business goals are impacted.

Analysis – Summary

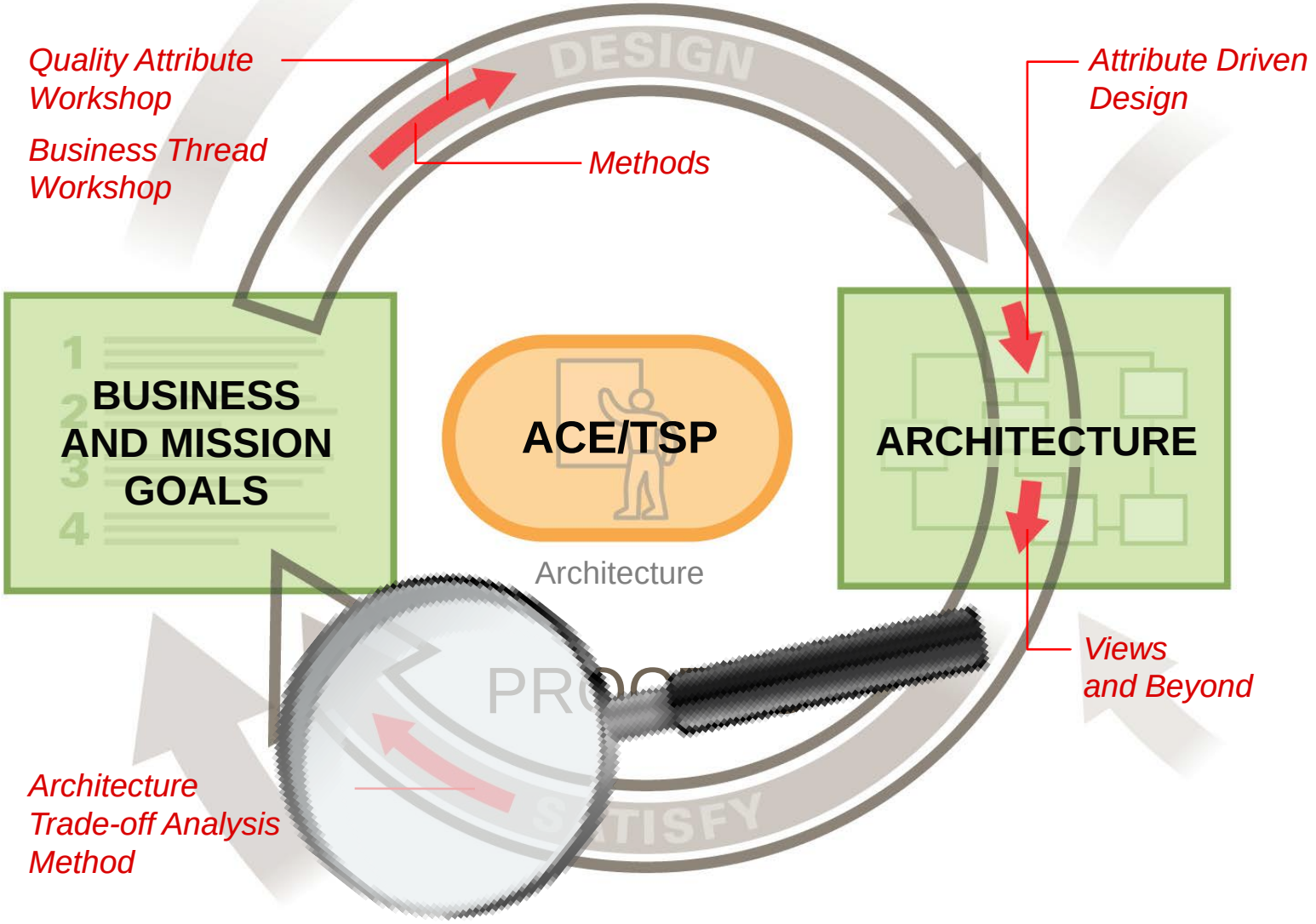
Principle 7: The architect provides evidence that the system's quality attribute properties will indeed fulfill the scenarios.

Principle 8: Mismatches between architecture properties and scenarios become risks to the business goals

Architecture Evaluation using the ATAM®



Architecture Centric Engineering



Architecture Evaluation in the Small

Before designing the architecture of a system, the desired quality attribute properties need to be clear.

- Without them it is not clear which goal to achieve with the design.
- Without a goal, every design is good and bad.

Quality attribute scenarios are a **required** input for the design phase. This is true for both,

- Green field development (create a new architecture).
- Brown field development (adjust an existing architecture).

Architecture Design – Green Field

Stakeholders representing the organization's business goals provide the desired quality attribute requirements.

A Quality Attribute Workshop (QAW) is the tool to elicit those requirements quickly as a prioritized, measurable set of quality attribute scenarios.

These quality attribute scenarios are a starting point and they will be refined during the architecture design.

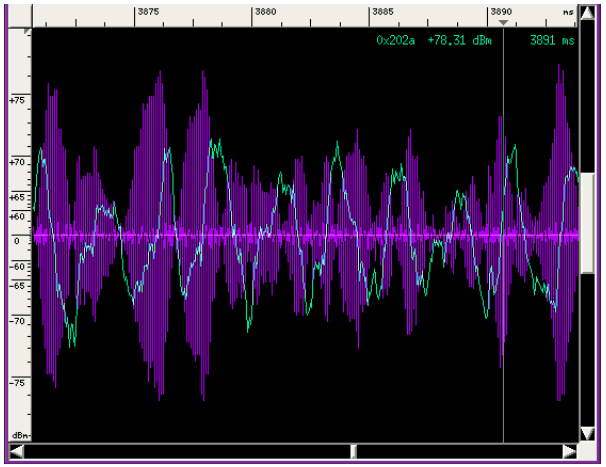
At average, every one of the QAW scenarios will lead to three more refined scenarios.

An architecture team of four people can finish an architecture design for a single refined scenario in about one week average.

Architecture Design – Brown Field

A request to change an existing architecture is usually because of a problem or an enhancement to the system.

Create a problem statement of the issue as it can be observed at the existing system.



Create a set of quality attribute scenarios that state the desired observable behavior after the redesign is done.

No matter if green field or brown field, the starting point for architecture work is a set of quality attribute scenarios.

Scenario Based Peer Review – 1

Waiting until all the architecture design is done before doing an architecture evaluation bears the risk that important aspects have been forgotten.

The quicker the feedback, the less rework

Peer reviews using architecture evaluation techniques provide the necessary feedback.

- During the review the architects need to make the case to an expert why a scenario is fulfilled.
- Written evidence (architecture documentation) is necessary.
- If the case cannot be made within one hour then something is wrong with the solution.

Scenario Based Peer Review – 2

A two weeks cycle for peer reviews seems to be appropriate.

- In two weeks, architects can handle two scenarios.
- Two scenarios can be reviewed within two hours.
- Reviews every second week for two hours insure the design is on track.

Steps of the peer review.

- Select the scenario to review.
- List the architecture approaches used to support this scenario.
- Show and list the documentation that shows how the solution works.
- List the risks that come with the solution.
- List to-do items
- Make a decision to approve or to repeat the review.

Peer Review Results

After a peer review it is clear

- what architecture documentation is required for a specific scenario.
- what the open action items are,
- which risks need to be mitigated,
- if a scenario needs to be re-negotiated with the stakeholders.

Keeps the system's architecture design on track!

SATURN 2012

Software Engineering Institute | Carnegie Mellon

As projects continue to grow in scale and complexity, effective collaboration across geographical, cultural, and technical boundaries is increasingly prevalent and essential to system success. SATURN 2012 will explore the theme of “Architecture: Catalyst for Collaboration.”