Launching and Sustaining Agile Architecture

Ipek Ozkaya and Robert Nord

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0568

Agile Architecture?

The phrase *“agile architecture”* evokes two concepts:

1. An architecture that is versatile, easy to evolve, and easy to modify, while resilient enough not to degrade after a few changes.
2. An agile way to define an architecture, using an iterative lifecycle, allowing the architectural design to tactically evolve over time, as the problem is better understood.

In the best of worlds, we’d like an agile process that leads to a flexible architecture. This tutorial enables attendees to understand basic architecture concepts that developers use to develop large-scale systems in an agile lifecycle.

Attendees who complete the tutorial should be able to understand the business case for architecture, architecture essentials, and architecting with just enough anticipation as an enabler for agile at scale.

Topics

Motivation

Why? The roles of architecture

What? Defining architecture

How? Essential activities

- An Experience: Smart Decisions Game

When? Release planning

- Light-weight architecture analysis

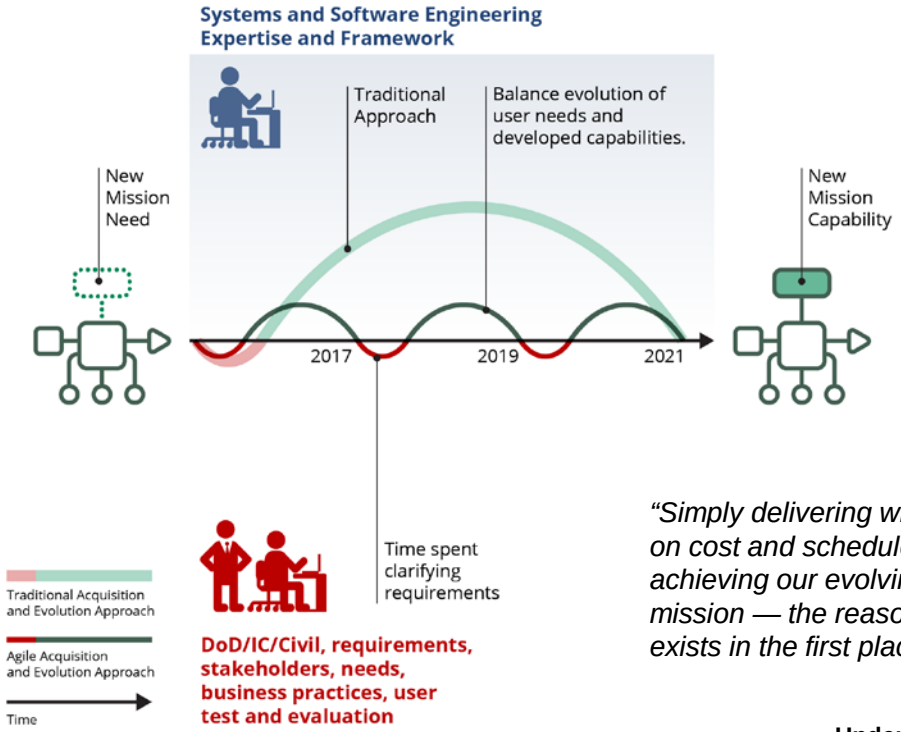
Who? Necessary organic capabilities

Take away

Motivation for Agile and Architecture: Software Engineering and Acquisition

Many regulated environments, like the DoD, NEED innovation and NEED incremental improvements to their systems.

Many of them are now willing to consider changing their approach ***if they can do it without getting in trouble with their governing statutes and regulations.***



"Simply delivering what was initially required on cost and schedule can lead to failure in achieving our evolving national security mission — the reason defense acquisition exists in the first place."

Honorable Frank Kendall
Under Secretary of Defense (AT&L)
2015 Performance of The Defense Acquisition System

Agile Practice

An ***iterative*** and ***incremental*** (evolutionary) approach to software development which is performed in a ***highly collaborative*** manner by ***self-organizing teams*** within an ***effective governance*** framework with ***"just enough"*** ceremony that produces ***high quality*** software in a ***cost effective*** and ***timely*** manner which meets the ***changing needs*** of its stakeholders.

Scott Ambler 2013

Organizational Agility

Agility is the ability to

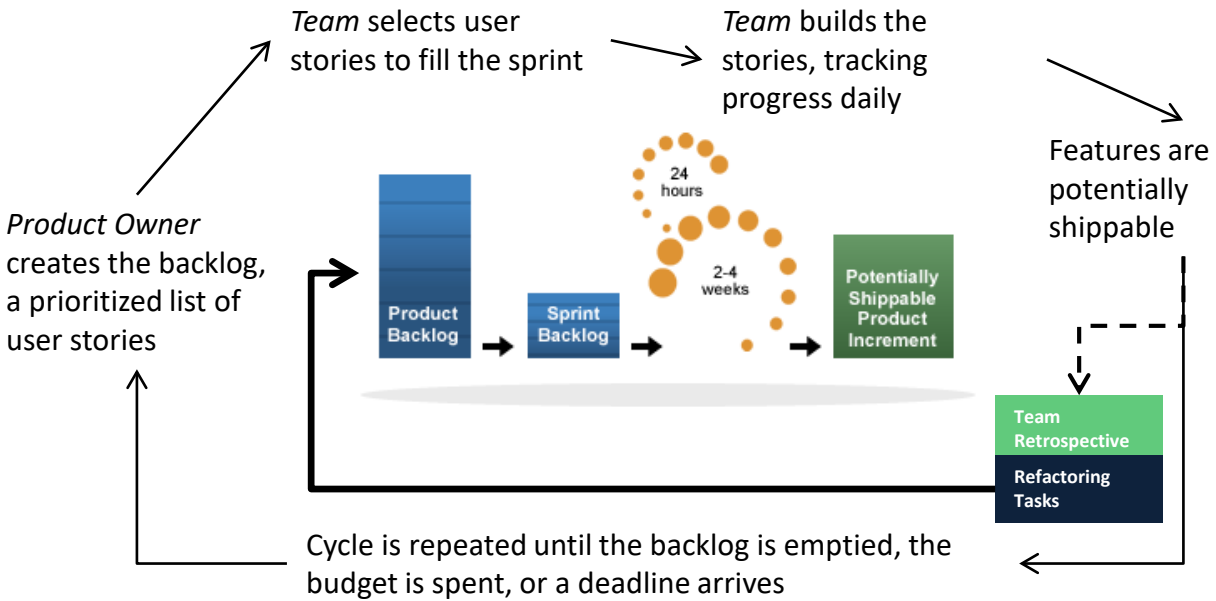
- create and respond to change
- balance flexibility and structure

Core agile cycles

- Envision: product vision, project scope, release plan
- Explore: iteration plan, develop, review and adapt

Jim Highsmith 2010

How Do You Adapt Scrum?



Today's Challenge Dealing with Organizational Change

Yesterday's Agile

Teams got better at building software

- Code quality
- Cohesion
- Velocity
- Improvement

Today's Agile

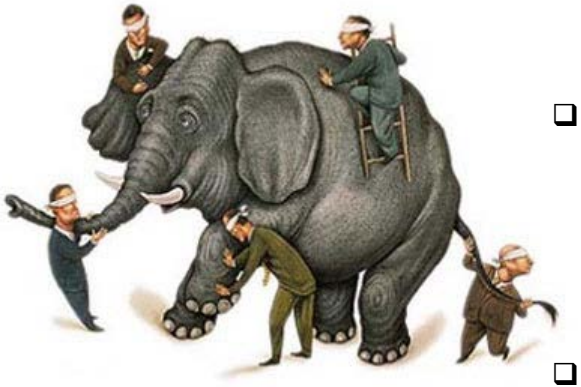
Moving the rest of the business

- Priorities are larger than the development team
- Collaboration is critical
- Timelines have changed

Architecture has a role to play in supporting three aspects of agile at scale: scope, team, and time.

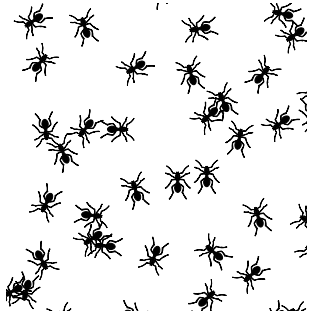
Grant, T. "Navigate the Future of Agile and Lean." *Forrester*, January 10, 2012.

A Closer Look at Scale: Scope



- Is the project in a new domain or technology?
- Are there new requirements such as standards compliance, system testing, and integration lab environments?
- Is there a need to align systems engineering and software development activities?

A Closer Look at Scale: Team



- Are there multiple teams that need to interact, both internal and external to the organization?
- What are the dependencies between the work products of system and software engineers?
- Does the end-to-end delivery of features require resources from multiple teams?

A Closer Look at Scale: Time

- Does the work require different schedule constraints for releases?
- How long is the work product expected to be in service?
- How important are sustainability and evolution?

Introductions

Introduce yourself and share something you know about agile or architecture.
Which of these challenges are you dealing with?

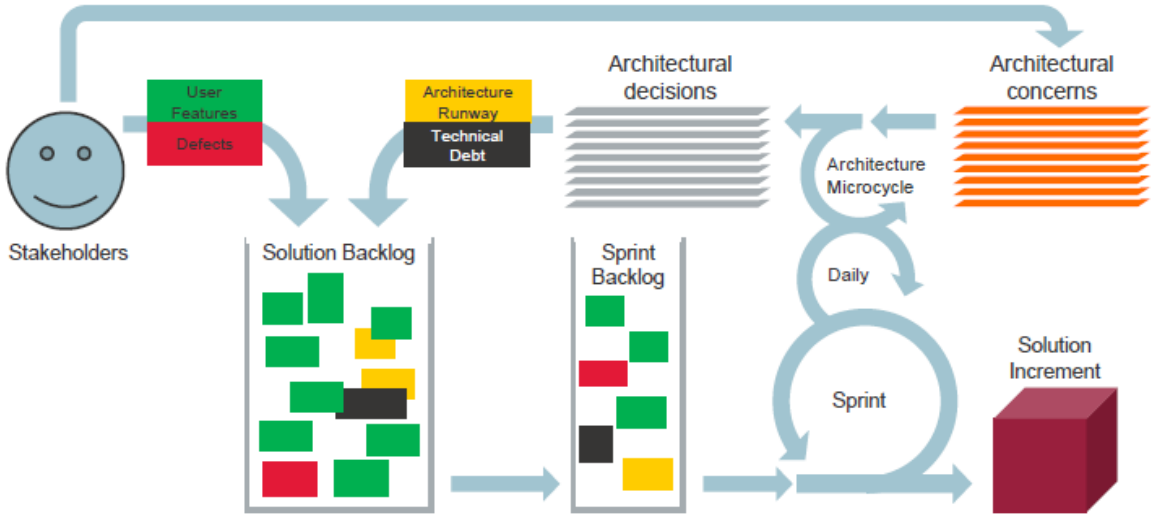
Enhanced Agile Development

As software system size and complexity increase, development practices must adapt to accommodate

- increased technical complexity
- interactions among software sub-systems and components
- larger teams and multiple teams
- coordinated development across multiple, competing objectives

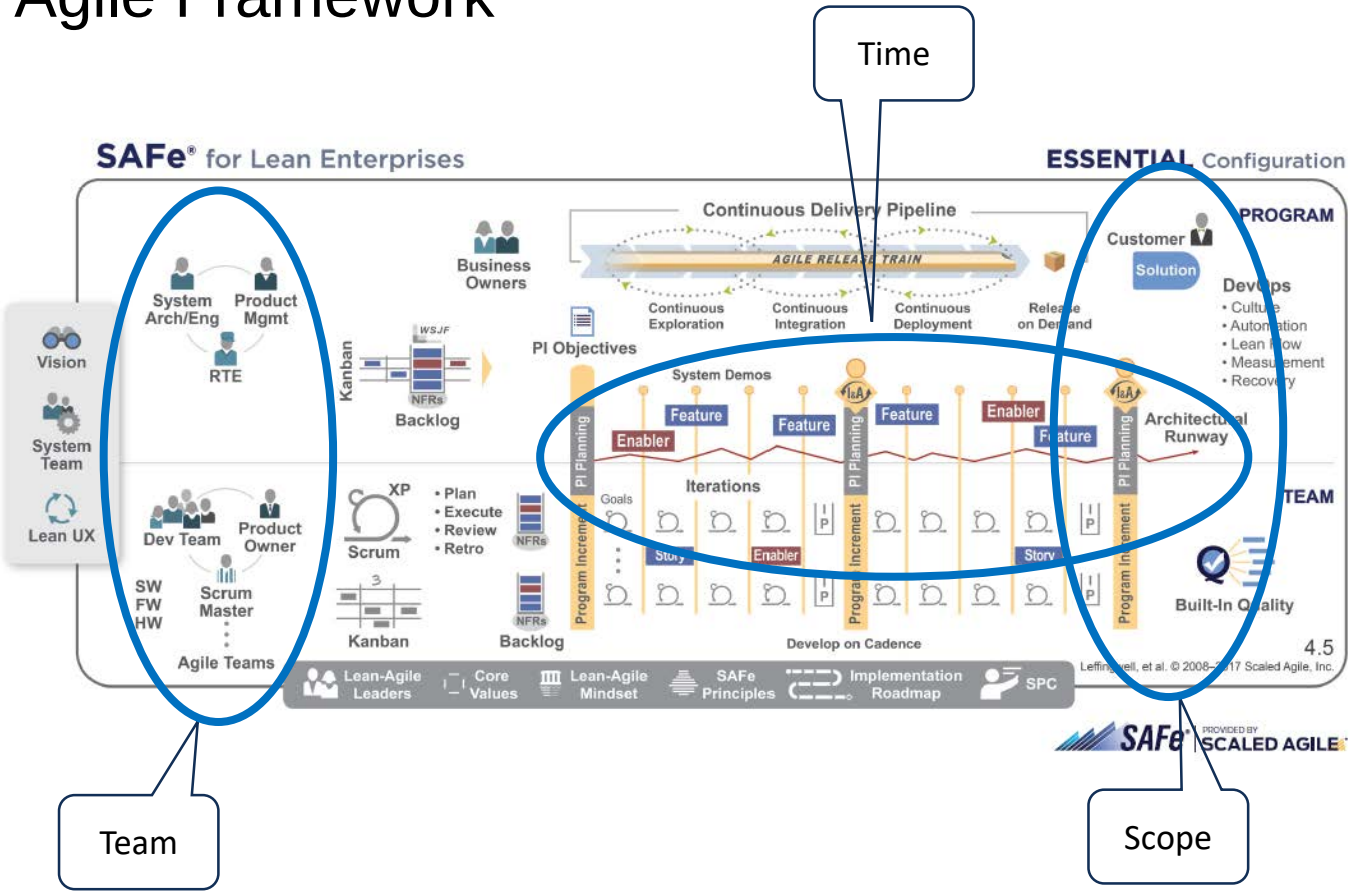
Enhanced agile development adds practices that address these concerns.

SCRUM and the Architecture Microcycle



Poort, E. Selling the Business Case for Architectural Debt Reduction, *Ninth International Workshop on Managing Technical Debt* – XP 2017

Scaled Agile Framework



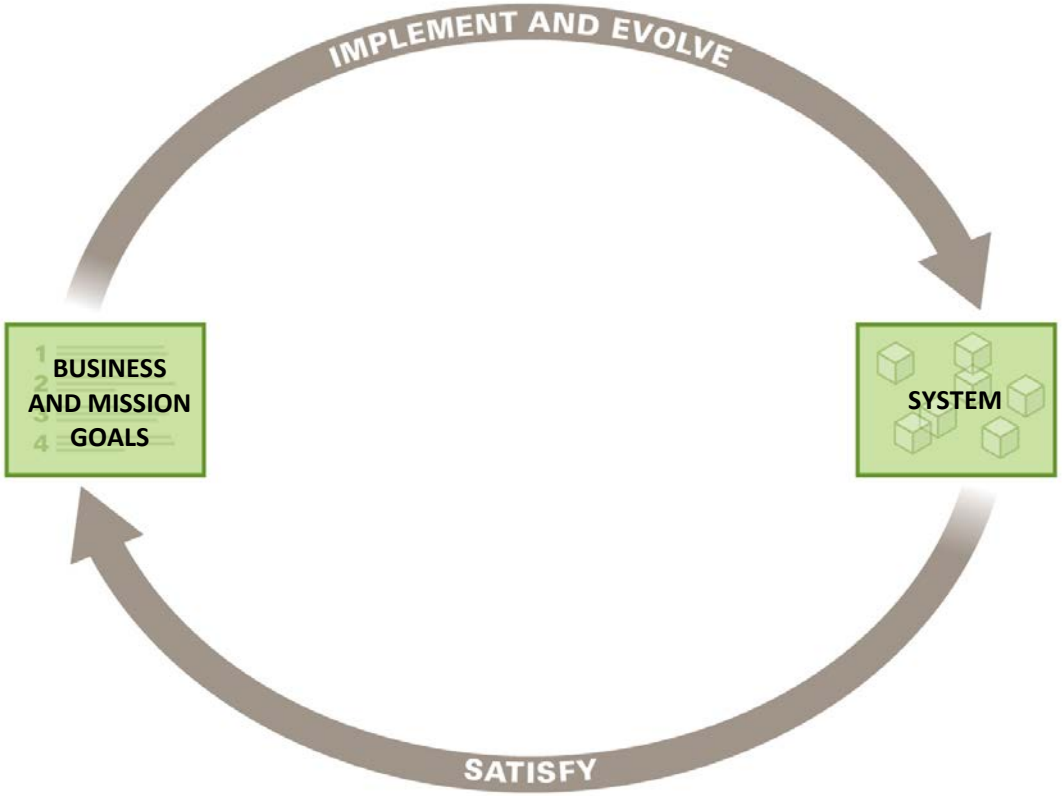
Sounds Expensive!

Compared to what?

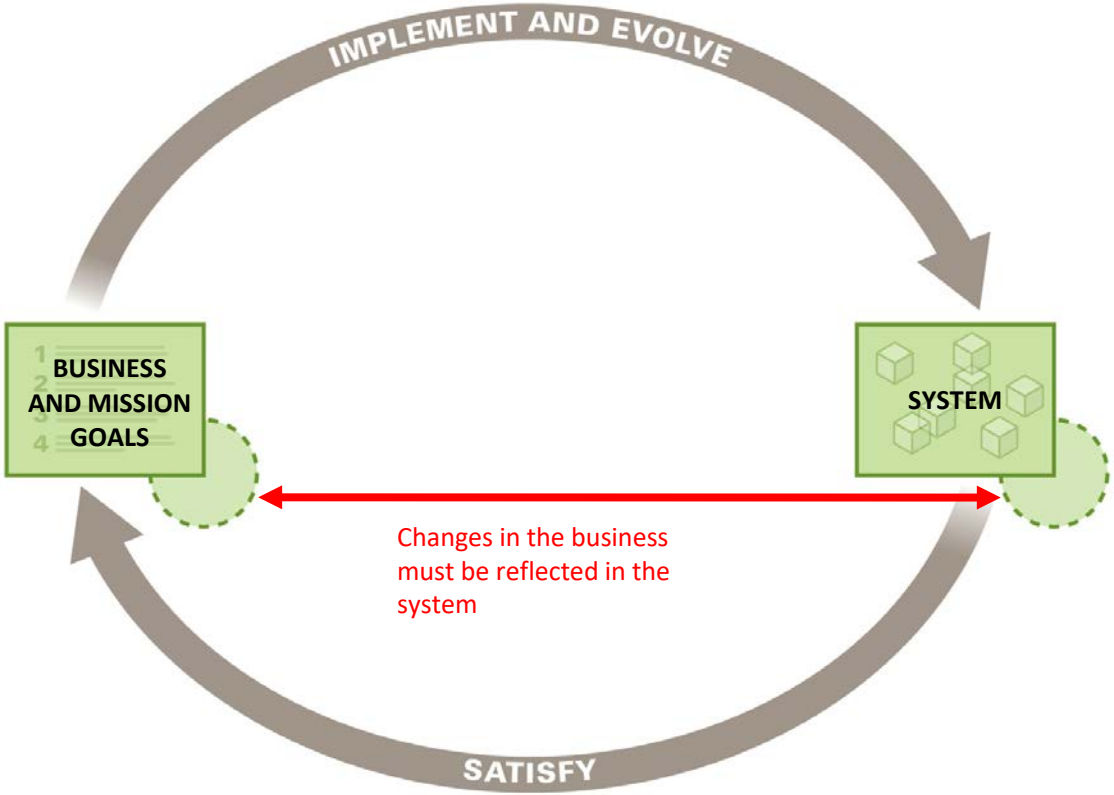
- Over-committing because there is no blueprint for the system?
- Inefficiency from inability to coordinate work?
- Late rework when defects found in test and integration?
- Delivering late and over budget?
- Developing a product that fails to meet stakeholder's needs?

Why? The roles of architecture

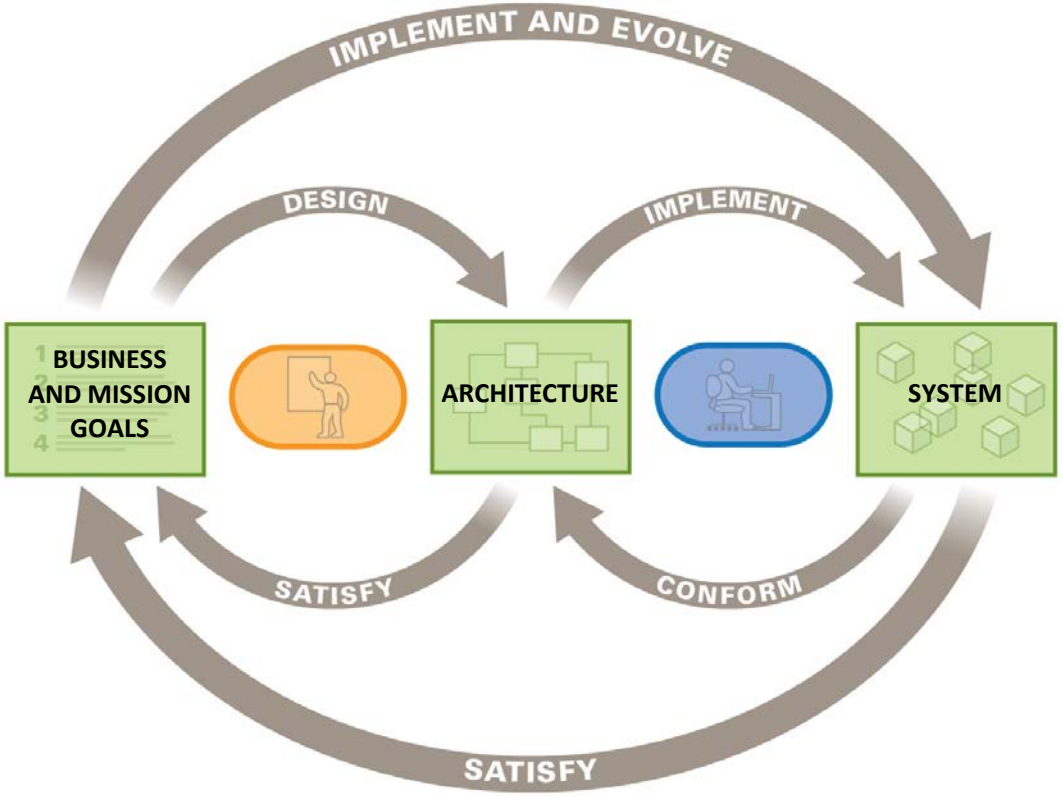
Architecture Practices



Architecture Practices



Architecture Practices



Value Proposition for Architecture

Architecture practice enables the ongoing cost-effective achievement of system-related business goals.

- Sound structure analyses provide objective confidence for achieving system quality.
- Appropriate flexibility enables cost-effective system maintenance and evolution.
- Early identification and mitigation of design risks result in fewer downstream problems and cost savings in integration, test and deployment.

What? Defining architecture

SEI Software Architecture Axioms

SEI's work in software architecture is guided by three foundational principles that highlight architecture's key role in system development and evolution.

1. Software architecture is the bridge between business and mission goals and a software-intensive system.
2. Quality attribute requirements drive software architecture design.
3. Software architecture drives software development through the life cycle.

Architecture – The Bridge

A good architectural representation has

- sufficient detail to reason about mission and business goal satisfaction
- sufficient abstraction to conceptually understand the system
- sufficient detail to appropriately constrain implementation.

All design involves tradeoffs.

Lacking mission and business drivers, the architect has to make assumptions about priorities.

Given well-stated mission and business drivers, the architect has a basis for knowing the priorities among tradeoffs.

“Every system has an architecture…

…encompassing the key abstractions and mechanisms that define that system's structure and behavior… In every case - from idioms to mechanisms to architectures - these patterns are either

intentional

or

accidental”

- Grady Booch in the Preface to *Handbook of Software Architecture*

Architecture and Strategy

An intentional architecture is the embodiment of your business strategy

- Intentional architecture links technology decisions to business goals

An accidental architecture limits strategy options

- Accidental architecture becomes your *de facto* strategy

SEI Software Architecture Axioms

1. Software architecture is the bridge between business and mission goals and a software-intensive system.
2. Quality attribute requirements drive software architecture design.
3. Software architecture drives software development through the life cycle.

Software System Development

Functional Software Requirements

If function were all that mattered, any monolithic software would do, ..***but other things matter***…

The important quality attributes and their characterizations are key.

- Modifiability
- Interoperability
- Availability
- Security
- Predictability
- Portability

analysis, design, development, maintenance and evolution

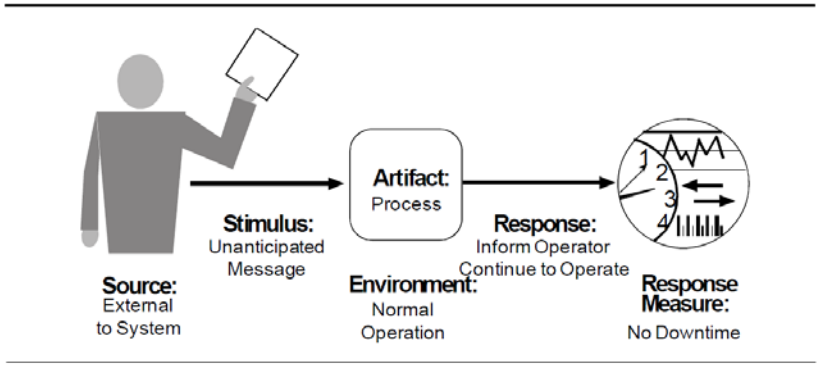
Quality Attribute Drivers → **Software Architecture** → **Software**

has these qualities

Maintainability represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements (ISO 25010).

Users Need Both Functions and Qualities

Required capability
Ease of use
Predictable behavior
Dependable service
Timely response
Protection from intruders
……



Software system/mission goals should address stakeholder needs.

Stakeholder needs often translate to quality attribute requirements.

Scenarios are a powerful way to characterize quality attributes.

Quality Attribute Data from SEI ATAMs

Rank	Quality Attribute Concern	Quality Attribute
1	Reduce coupling	Modifiability
2	Latency	Performance
3	Upgrade and integrate with other system components	Interoperability
4	Designing for portability	Modifiability
5	Ease of operation	Usability
6	Detect faults	Availability
7	Ease of interfacing with other systems or components	Interoperability
8	Designing for extensibility	Modifiability
9	Recover from faults	Availability
10	Resource management	Performance
11	Minimize build, test and/or release duration	Deployability
12	Reusability	Modifiability
13	Prevent faults	Availability
14	Increased processing demands (e.g. add nodes)	Scalability
15	Authorization	Security
16	Resource and data sharing	Interoperability
17	Resist attack	Security
18	Configuration and/or dependency management	Deployability
19	Configurability/compose-ability	Modifiability
20	Backward compatibility and/or rollback strategy	Deployability

Bellomo, S.; Kazman, R.; & Gorton, I. “Insights from 15 Years of ATAM Data: Towards Agile Architecture” *IEEE Software*, 2015.

Discussion: Default Quality Attributes

In the absence of (and often even with) explicit quality attributes, most developers have a (sometimes unconscious) default set of attributes that they value.

- Modularity
- Reusability
- Analyzability
- Modifiability
- Testability
- Readability/understandability
- Efficiency
- Elegance (cleverness?)
- …many possibilities

What are YOUR default quality attributes?

SEI Software Architecture Axioms

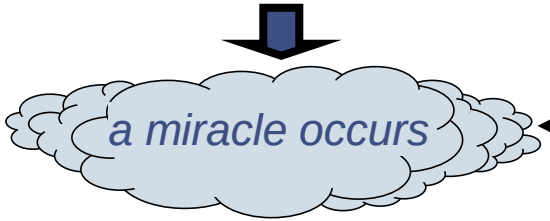
1. Software architecture is the bridge between business and mission goals and a software-intensive system.

2. Quality attribute requirements drive software architecture design.

3. Software architecture drives software development through the life cycle.

Typical Software Development Paradigm

Operational descriptions
High level functional requirements
Systems specifications



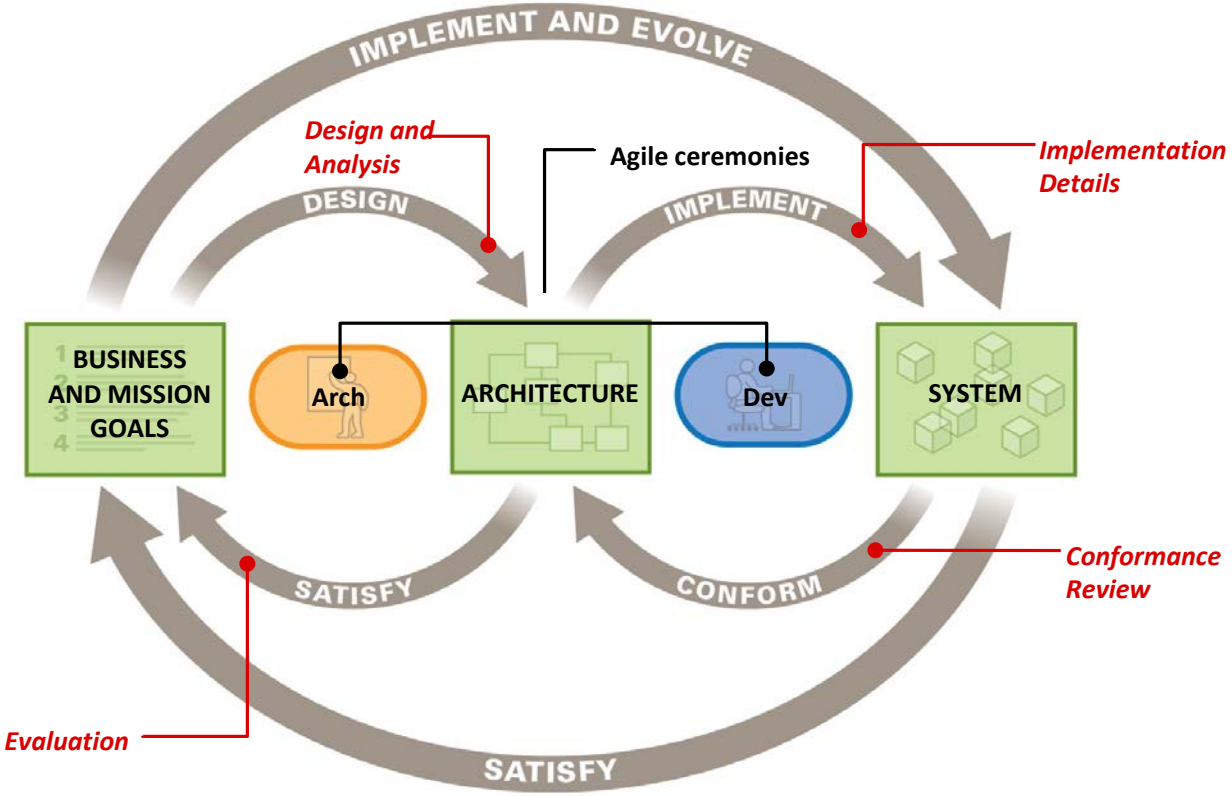
Quality attributes are often weakly articulated and vaguely understood

A specific system architecture
Software architecture emerges

How do you know if the architecture is fit for purpose?

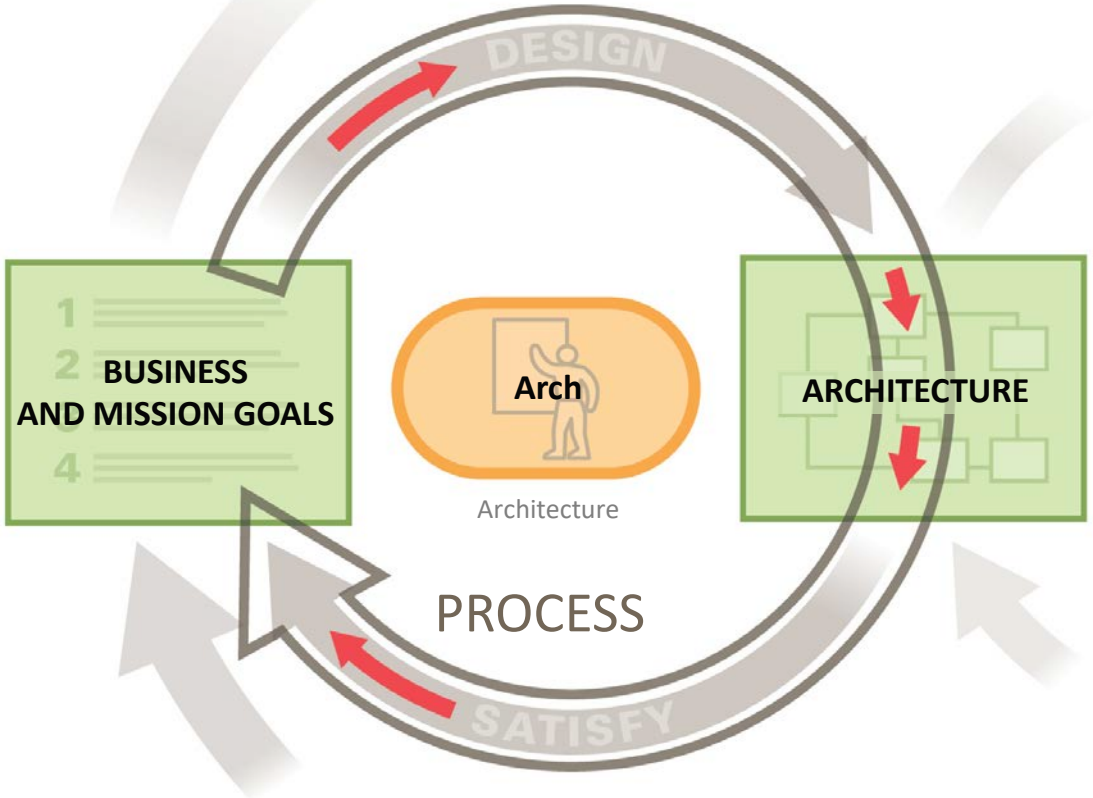
Detailed software design
and implementation

Architecture Practices



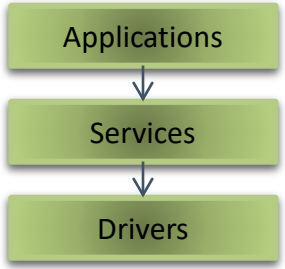
How? Essential activities

Design Cycle



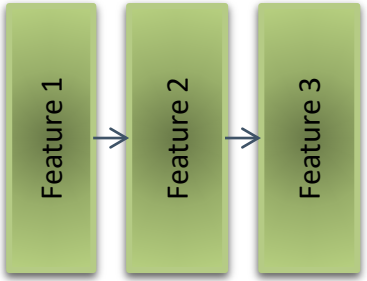
Align Feature and System Decomposition

Infrastructure-driven approach



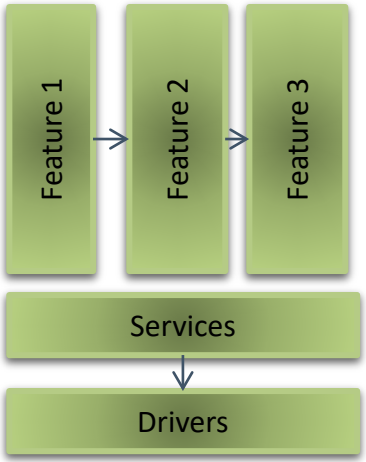
Horizontal decomposition
(e.g., layers)

Feature-driven approach



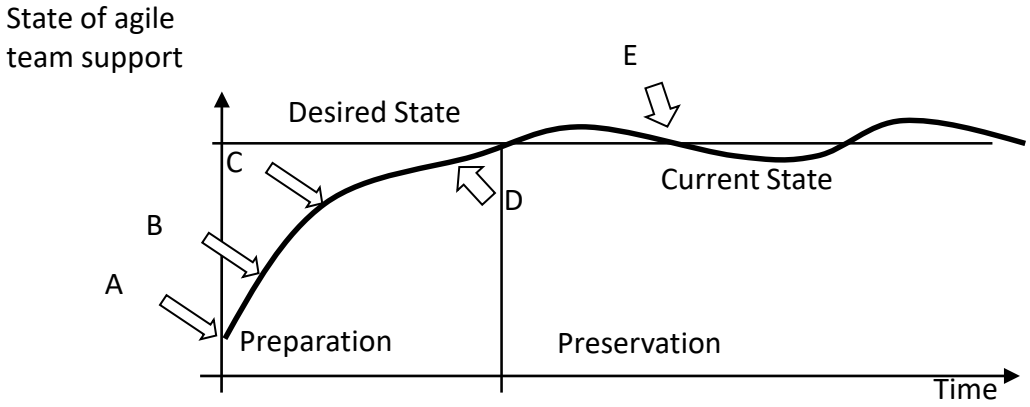
Vertical decomposition
(e.g., subsystems, features)

Hybrid approach



Tension between high-priority features (vertical decomposition) versus common reusable services (horizontal decomposition)

How Much Support for Agile Development?

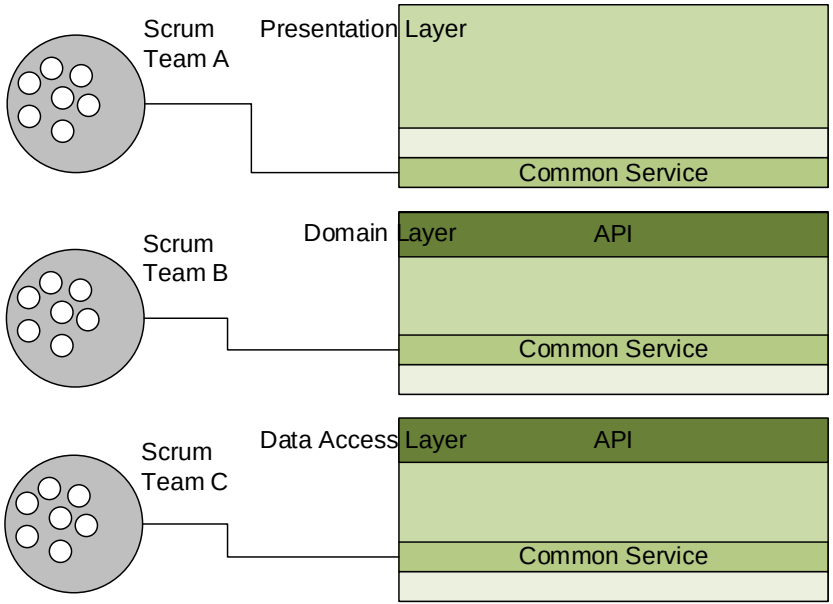


A – No support
B – Most important parts
Ready for the first feature
C – Almost ready with the support
Feature development starts

D – Desired state reached
E – Sustain the state

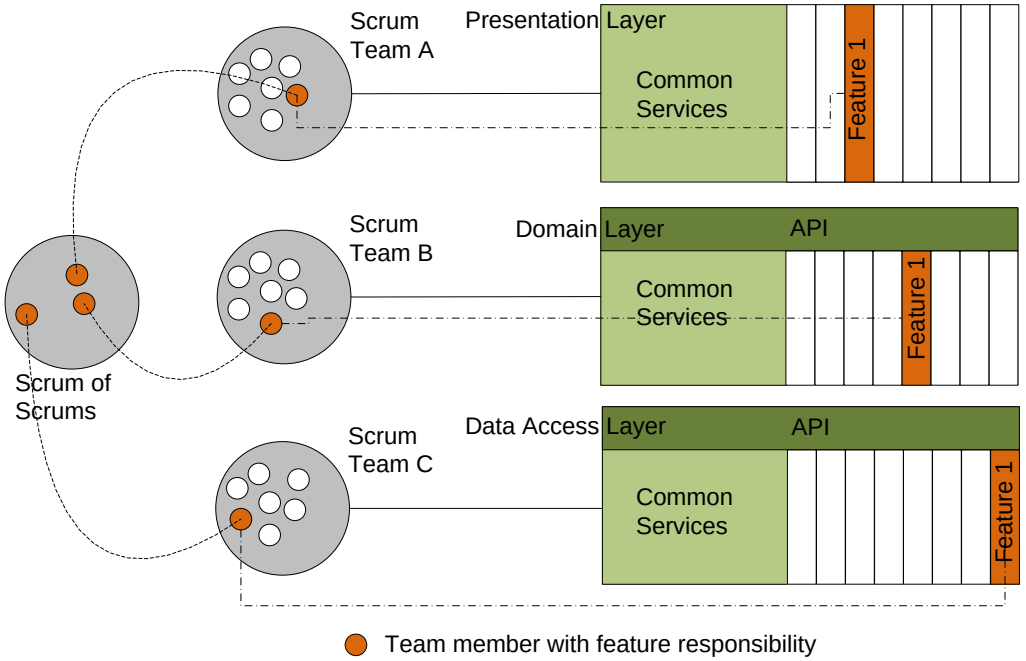
Bachmann, B., Nord, R.L., and Ozkaya, I. "Architectural Tactics to Support Rapid and Agile Stability," *Crosstalk*, May/June, 2012.

Applying the Practices in Concert -1



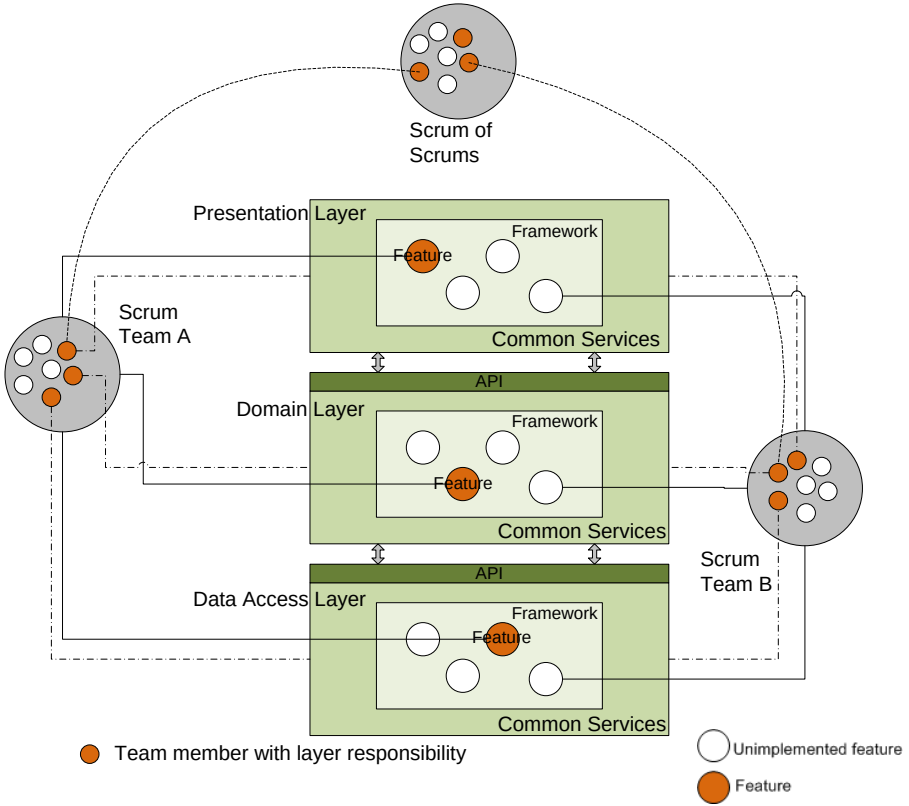
State A – Establishing the infrastructure

Applying the Practices in Concert -2



State B – Progressing architecture and feature development in parallel

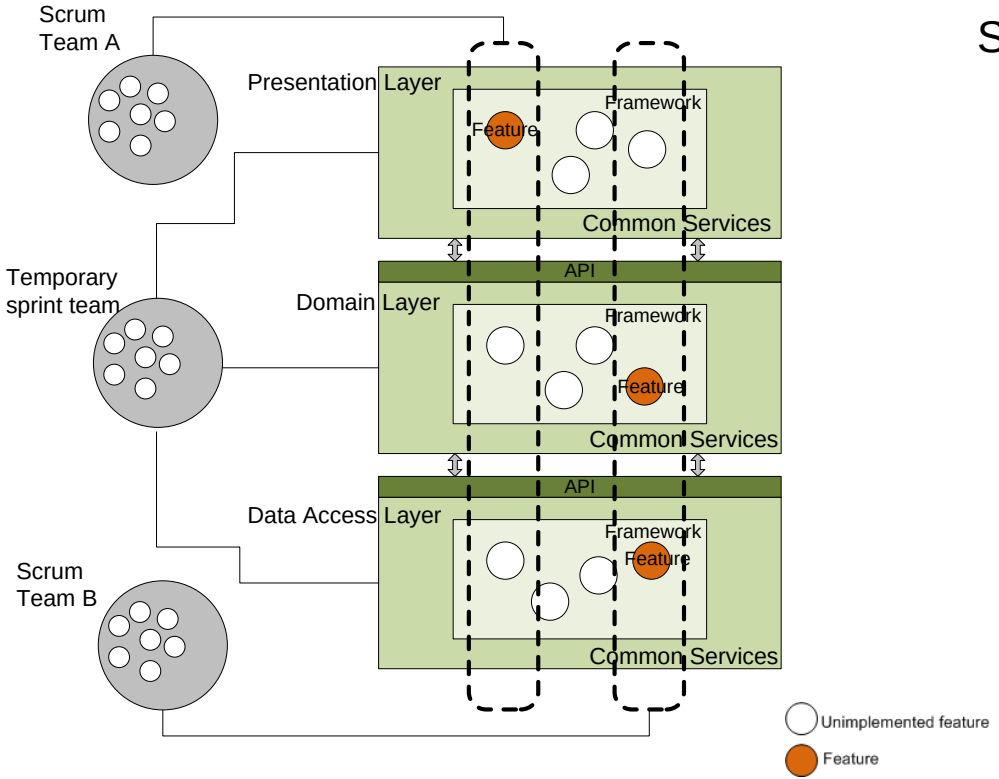
Applying the Practices in Concert -3



State C – Features

- different teams are assigned to different features,
- some team members keep layers and framework consistent

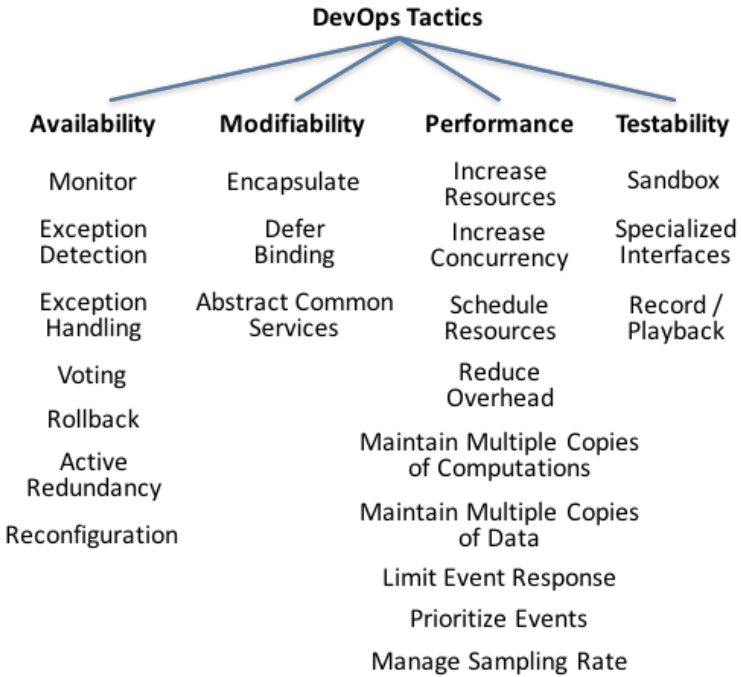
Applying the Practices in Concert -4



State D – Preservation

- different teams are assigned to different features,
- a temporary team prepares layers and frameworks for future feature teams.

Deployability Tactics



Tactics are design decisions that enable quality attributes.

There are tactics for

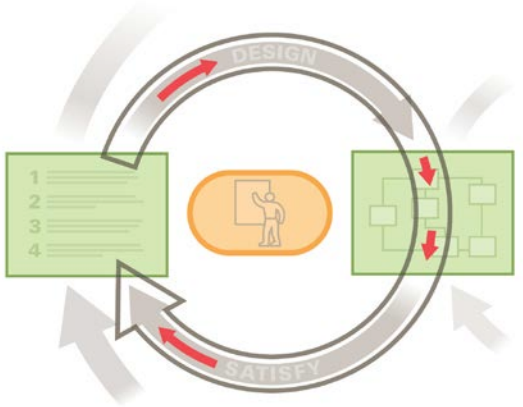
- Availability
- Interoperability
- Modifiability
- Performance
- Security
- Testability
- Usability

Bellomo, S., Kazman, R., Ernst, N., Nord, R.: Toward Design Decisions to Enable Deployability: Empirical Study of Three Projects Reaching for the Continuous-Delivery Holy Grail. In: *First International Workshop on Dependability and Security of System Operation*, pp. 32–37. IEEE Press, New York (2014)

Design and Analysis

The architects ensure that the design is checked in a periodic fashion to see if the quality attribute scenarios are continuing to be fulfilled.

- Step 1: Select scenario to analyze
- Step 2: Elicit architecture approaches
- Step 3: Analyze architecture approaches
- Step 4: Review results



Performing scenario-based peer reviews every second week was never seen as a burden by the architects. They were actually looking forward to the next review because the reviews provided them with valuable input and they could see progress when the list of risks and the to-do list became smaller and smaller over time.

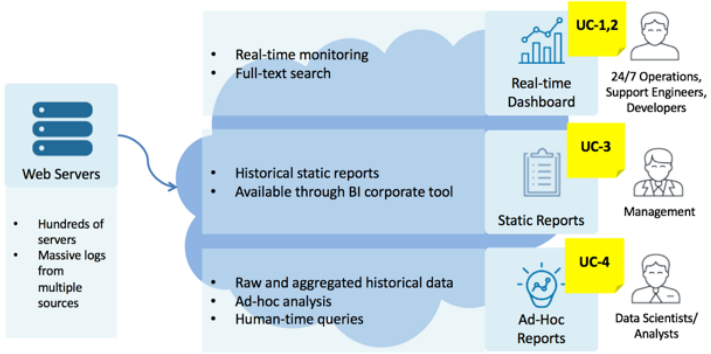
Bachmann, F. Give Stakeholders What They Want: Design Peer Reviews the ATAM Style, *CrossTalk*, Nov/Dec 2011.

Let’s Have An Experience
Smart Decisions Game

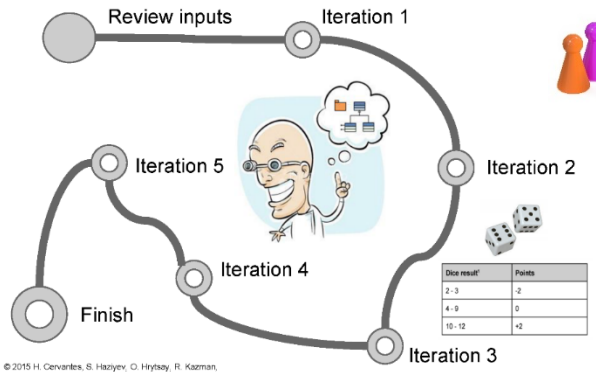
Game Inventory[1]

1. Playing cards

2. Game scenario



3. Game board



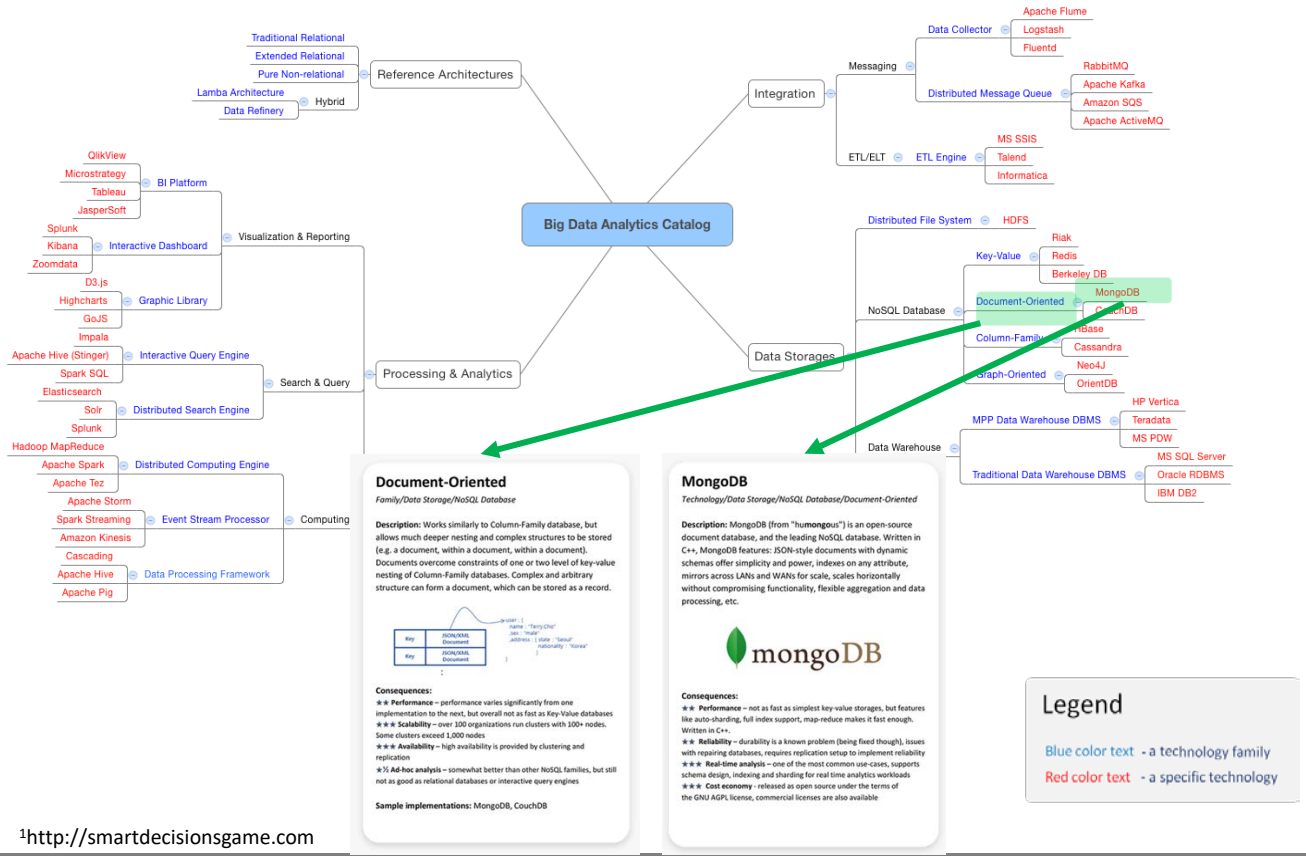
4. Dice and markers

5. Scorecard

	Iteration #1	Iteration #2	Iteration #3	Iteration #4	Iteration #5	
(a) Design Decisions (Names of selected design concept(s))	Lambda Architecture					
(b) Driver selection points (from cards)	2.5+3+3 +3+3=**14.5**					
(c) Instantiation points (from dice)	+2					
(d) Analysis bonus points (from review)	+2					Final score:
(e) Iteration total (b + c + d)	18.5	10	15	11	8	62.5

[1]http://smartdecisionsgame.com

Design Concepts Catalog[1]

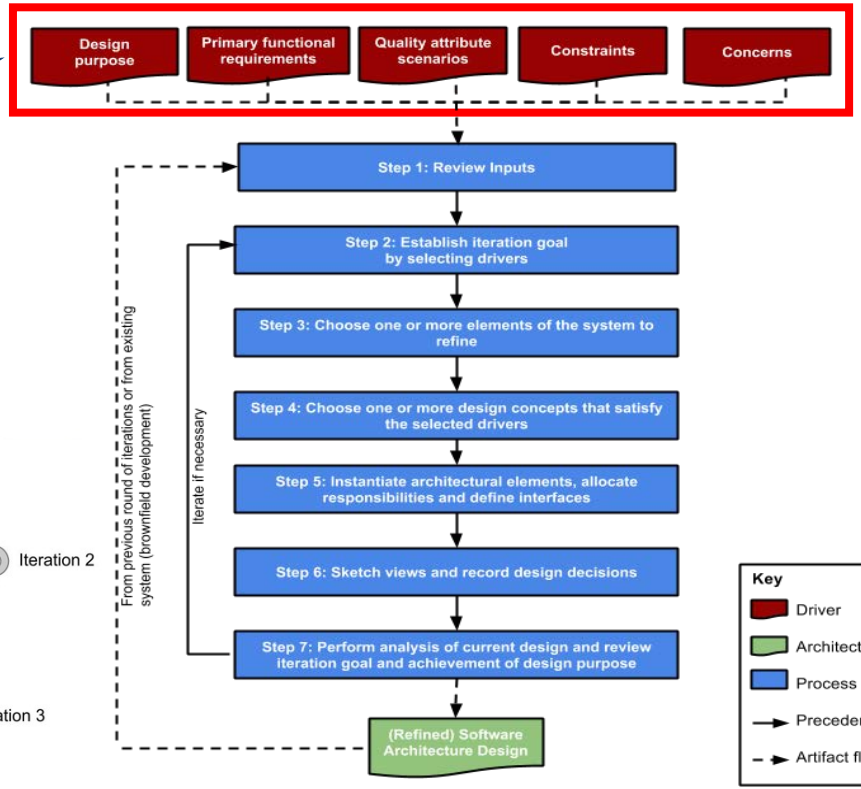


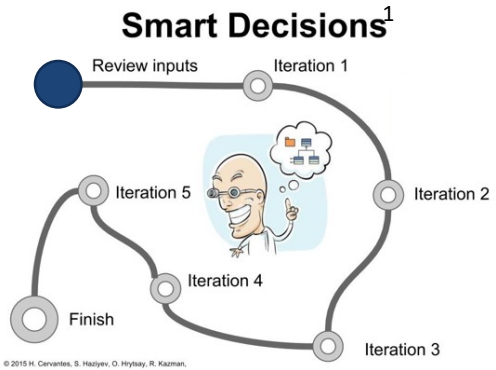
[1]http://smartdecisionsgame.com

Game Rules[1]

ADD Step 1: Review Inputs

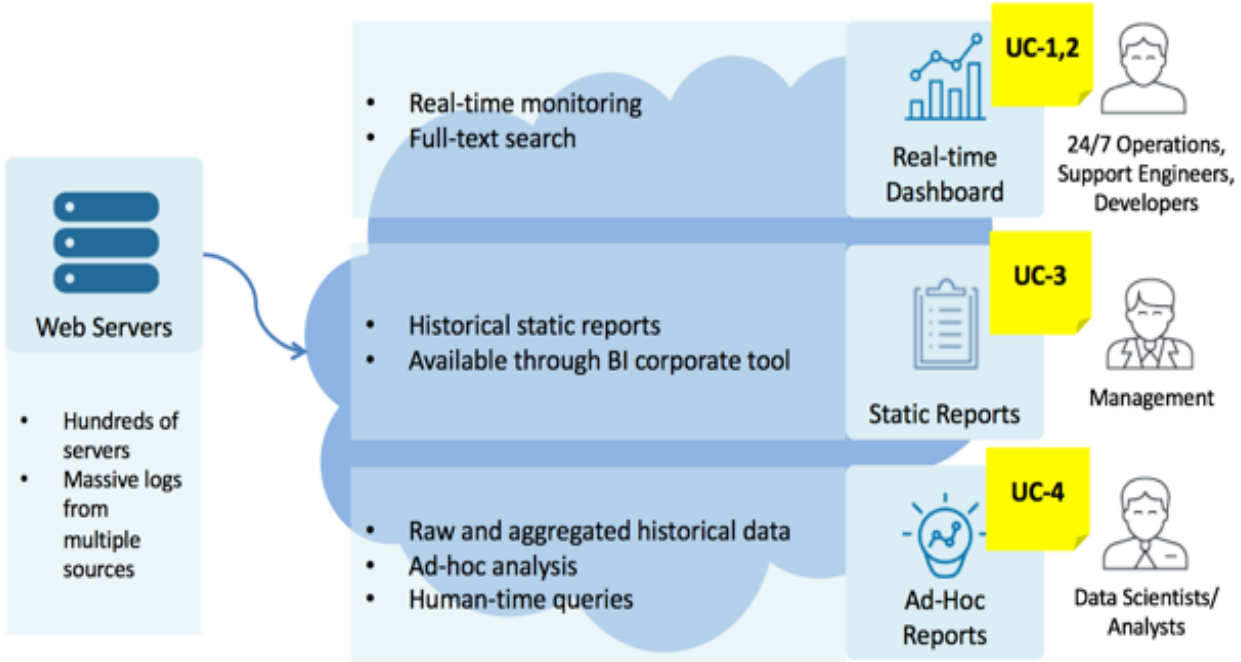
Let's start by reviewing the inputs to the design process…





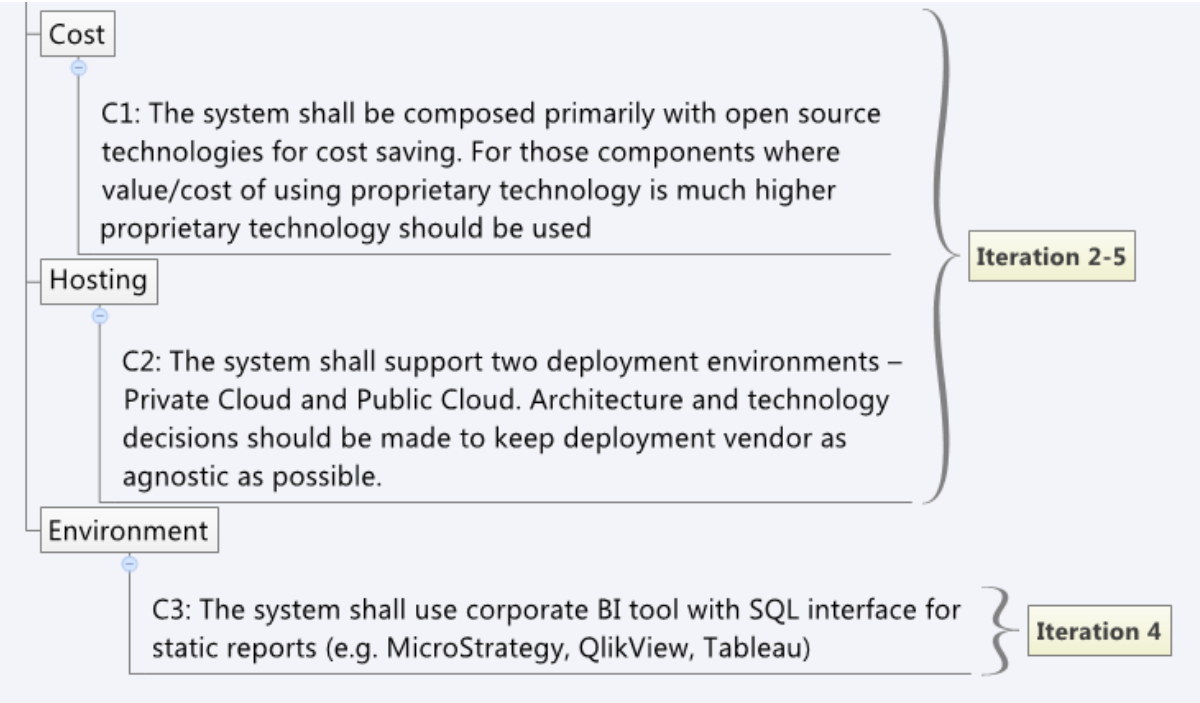
[1]http://smartdecisionsgame.com

Input Requirements: Functional [1]

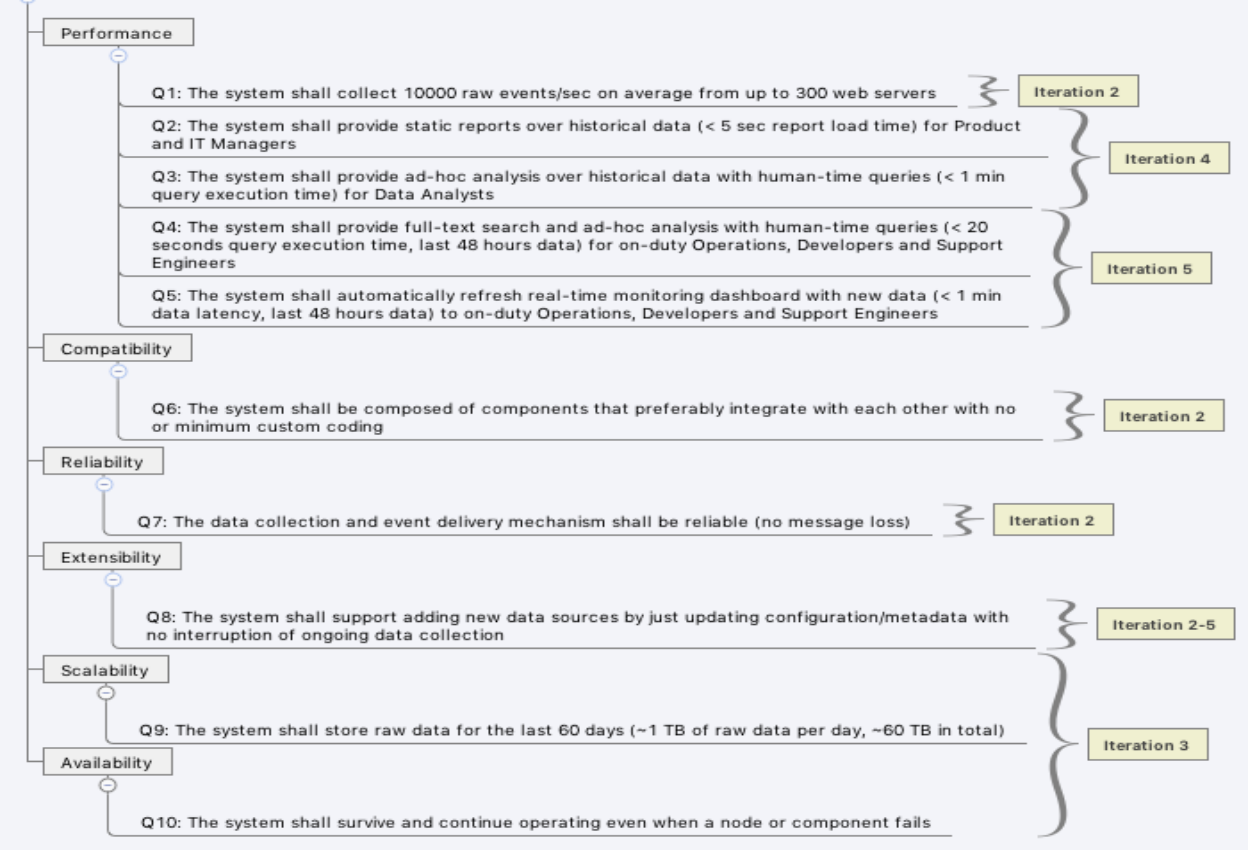


[1] http://smartdecisionsgame.com

Input Requirements: Constraints



Input Requirements: QAs

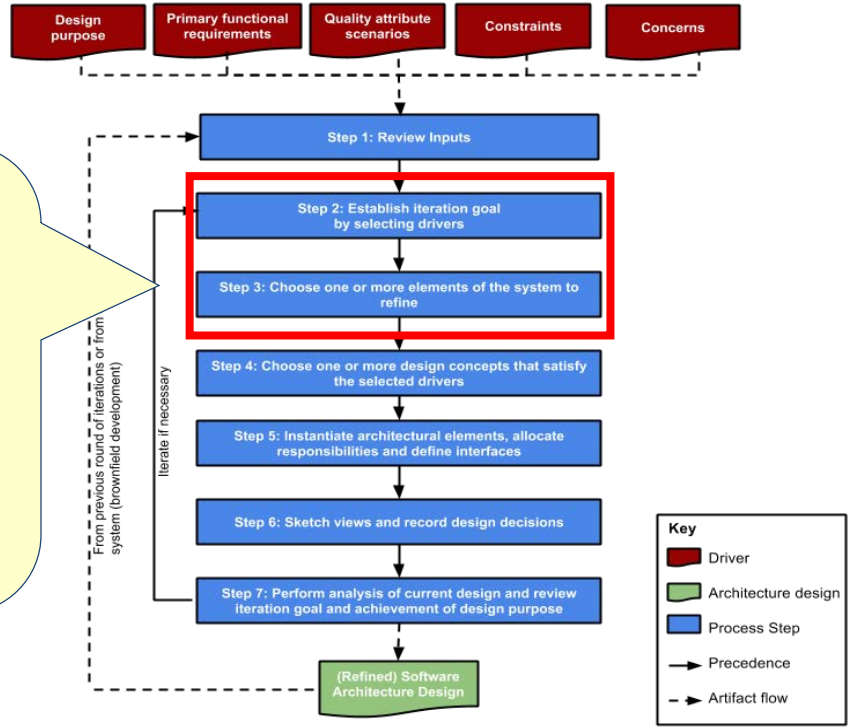


Game Rules[1]

ADD Step 2: Establish iteration goal by selecting drivers
ADD Step 3: Choose one or more elements of the system to refine

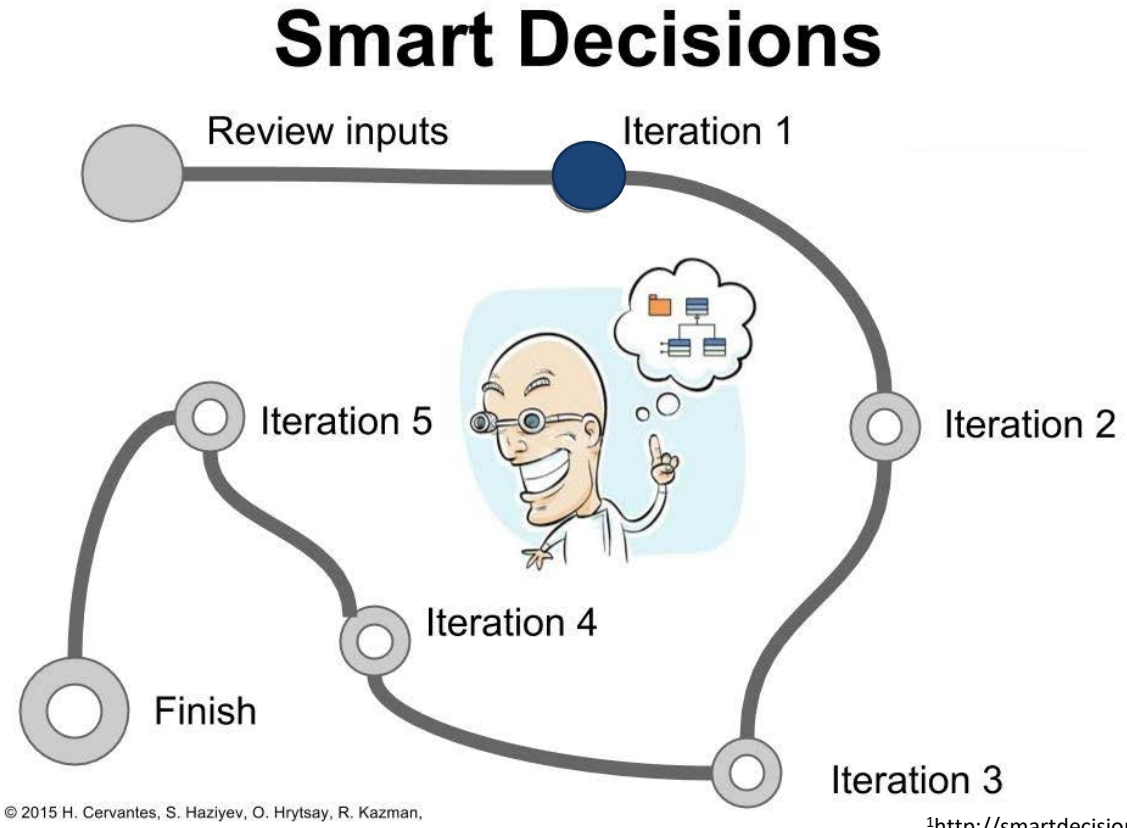
The game is played in rounds that represent the iterations.

For each round the game provides
- Iteration goal (i.e., selected drivers)
- Element to refine



[1]http://smartdecisionsgame.com

Let's Start![1]



[1]http://smartdecisionsgame.com

Iteration 1 Goal: Logically Structure the System

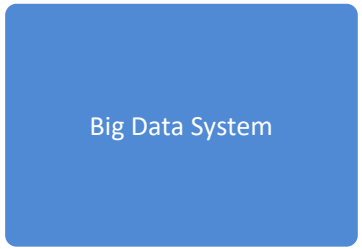
Drivers for the iteration:

- Ad-hoc analysis
- Real-time analysis
- Unstructured data processing

} Primary use cases

- Scalability } QA
- Cost Economy } Constraint

Element to refine:

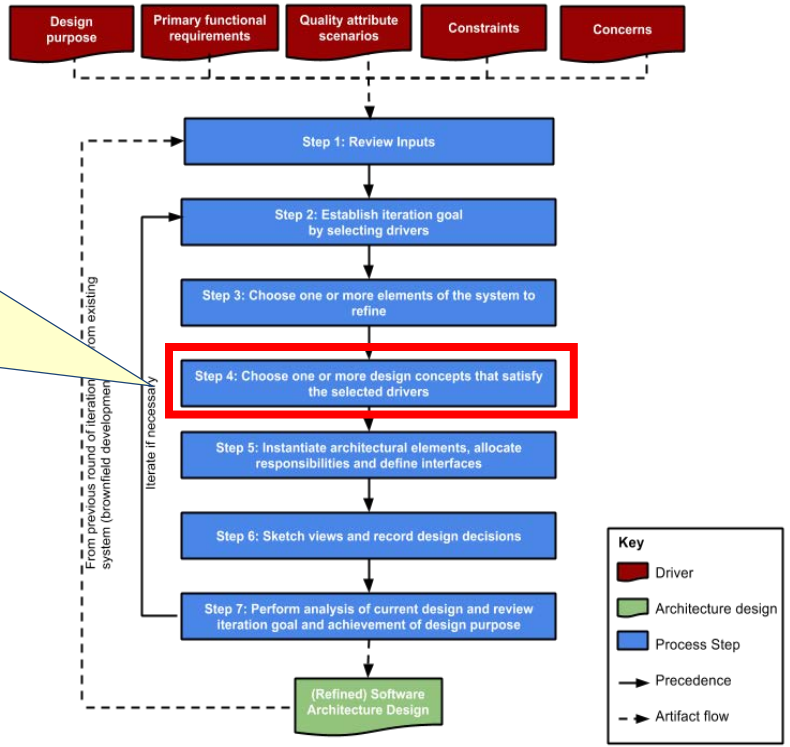


Game Rules[1]

ADD Step 4: Choose one or more design concepts that satisfy the selected drivers

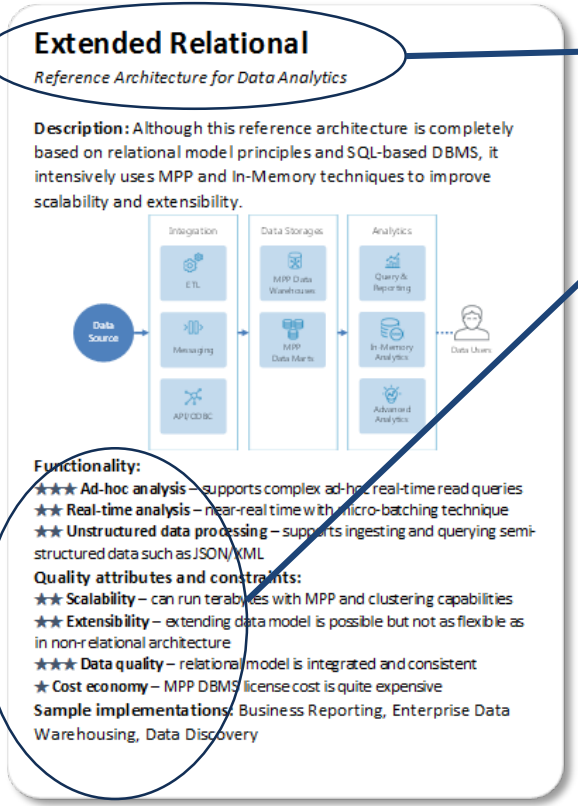
You will make decisions selecting design concepts:

- **Patterns:**
  - **Reference architectures**
  - **Technology families**
- **Technologies**



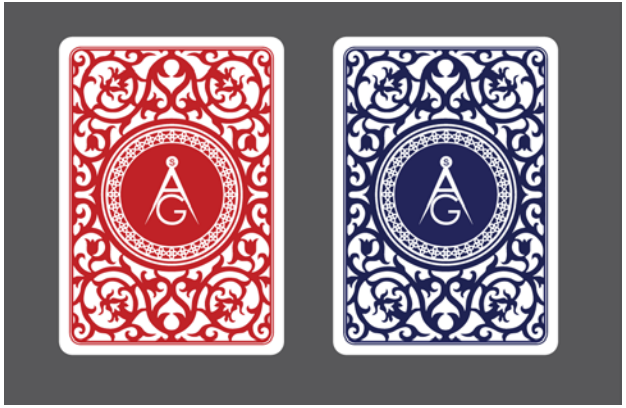
[1]http://smartdecisionsgame.com

Game Rules: Design Concepts Cards[1]



Name and type of design concept

Influence on drivers



Technologies **Patterns**

[1]http://smartdecisionsgame.com

Iteration 1 Goal: Logically Structure the System

Drivers for the iteration:

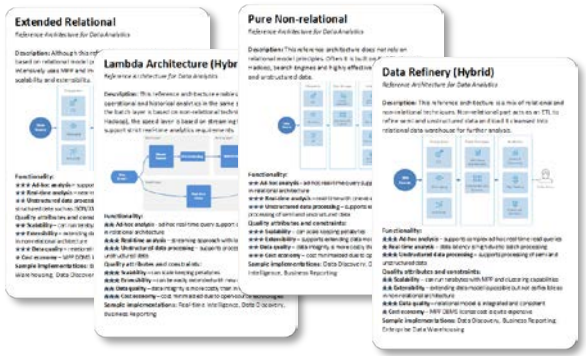
- Ad-hoc analysis
- Real-time analysis
- Unstructured data processing
- Scalability
- Cost economy

Element to refine:

Big Data System

To Do:

Select 1 **Reference Architecture** Card[1]

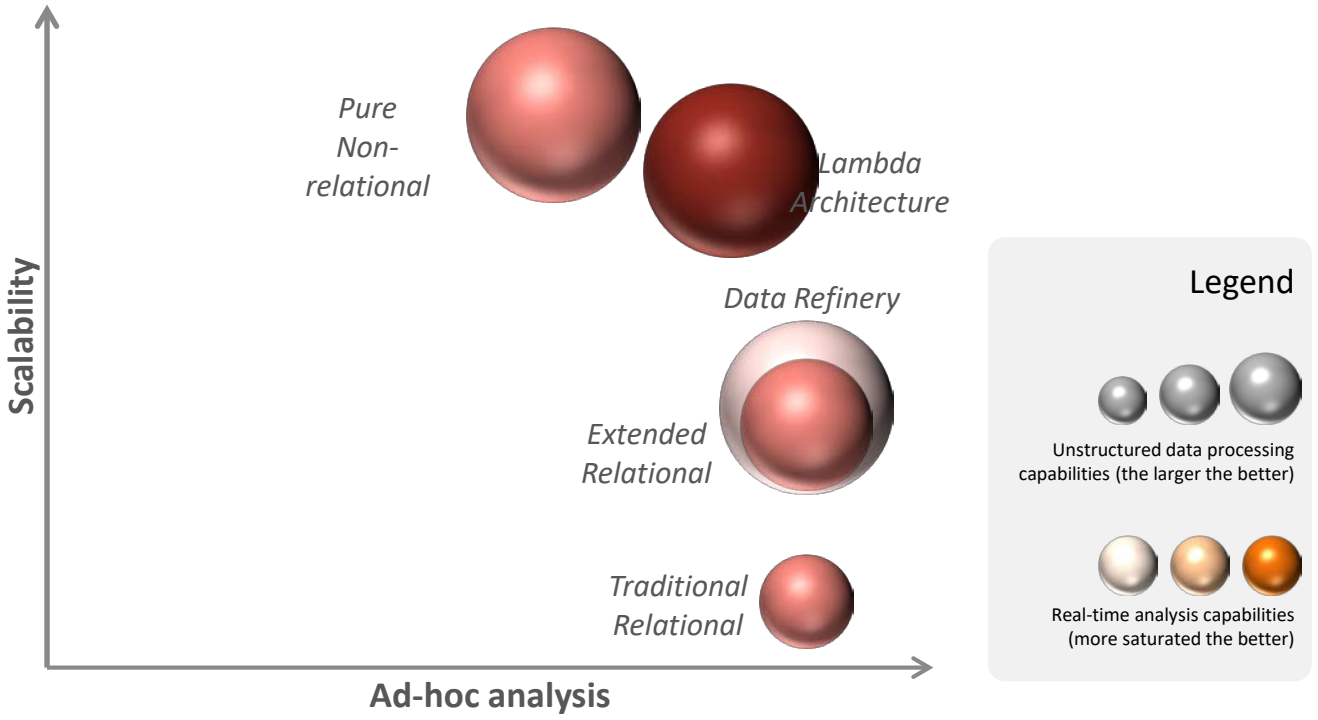


Alternatives:

- Extended Relational
- Pure Non-Relational
- Data Refinery
- Lambda Architecture

[1]http://smartdecisionsgame.com

Big Data Analytics Reference Architectures Trade-Offs



Fill in the Scorecard[1]

	Iteration #1	Iteration #2	Iteration	Iteration	Iteration	
(a) Design Decisions *(Names of selected design concept(s))*	Lambda Architecture					
(b) Driver selection points *(from cards)*	2.5+3+3 +3+3=**14.5**					
(c) Instantiation points *(from dice)*						
(d) Analysis bonus points *(from review)*						
(e) Iteration total *(b + c + d)*						

Some iterations require you to draw two cards. For these iterations you will need to

- Record the name of both design concepts
- Add the points for both of the cards

Please note that some drivers may not be associated with both cards, for example:

- Performance (for Family and Technology)
- Compatibility (for Family)
- Reliability (for Technology)

In these cases, you only count points for the drivers that are associated with the card.

Fill in (b) by adding the points for the drivers considered for the iteration, in this case:

- Ad-hoc analysis (2.5)
- Real-time analysis (3)
- Unstructured data processing (3)
- Scalability (3)
- Cost economy (3)

Record design decisions in (a)

= 1 Point

[1]http://smartdecisionsgame.com

Next Steps[1]

ADD Step 5: Instantiate elements, allocate responsibilities, define interfaces
ADD Step 6: Sketch views and record design decisions

You will:
- Record the design decision
- Throw two dice to simulate both market uncertainty and how well you instantiate your selected design concepts

[1]http://smartdecisionsgame.com

Fill in the Scorecard[1]

	Iteration #1	Iteration #2	Iteration #3	Iteration #4	Iteration #5	
(a) Design Decisions *(Names of selected design concept(s))*	Lambda Architecture					
(b) Driver selection points *(from cards)*	2.5+3+3 +3+3=**14.5**					
(c) Instantiation points *(from dice)*	+2					
(d) Analysis bonus points *(from review)*						**Final score:**
(e) Iteration total *(b + c + d)*						

Roll two dice once and add or subtract points according to the following table, and fill in (c).

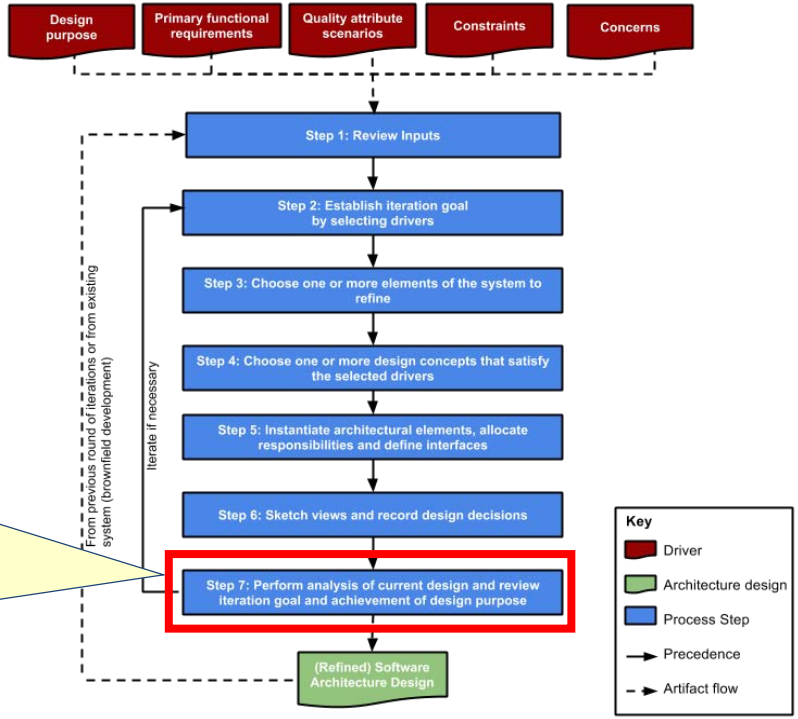
Dice result[1]	Points
2 - 3	-2
4 - 9	0
10 - 12	+2

[1]http://smartdecisionsgame.com

Next Step[1]

ADD Step 7: Perform analysis of current design and review iteration goal and design purpose

We will review the decisions together. The first iteration will be reviewed now, and the rest will be reviewed at the end of the game.



[1]http://smartdecisionsgame.com

Iteration 1 Review[1]

Score ad-hoc analysis, real-time analysis, unstructured data processing, scalability, and cost economy.

Design decision	Driver points	Bonus points	Comments
Extended Relational	3+2+2+2+1=**10**	**-4**	This reference architecture is less appropriate for this solution mostly because of cost and real-time analysis limitation.
Pure Non-Relational	2+2.5+3+3+3=**13.5**		This reference architecture is closer to the goal than the others except Lambda Architecture.
Lambda Architecture (Hybrid)	2.5+3+3+3+3=**14.5**	**+2**	This is the most appropriate reference architecture for this solution! From the provided reference architectures Lambda Architecture promises the largest number of benefits, such as access to real-time and historical data at the same time.
Data Refinery (Hybrid)	3+1+3+2+1=**10**	**-4**	This reference architecture is less appropriate for this solution mostly because of cost and real-time analysis limitation.

[1]http://smartdecisionsgame.com

Fill in the Scorecard[1]

	Iteration #1	Iteration #2	Iteration #3	Iteration #4	Iteration #5	
(a) Design Decisions *(Names of selected design concept(s))*	Lambda Architecture					
(b) Driver selection points *(from cards)*	2.5+3+3 +3+3=**14.5**					
(c) Instantiation points *(from dice)*	+2					
(d) Analysis bonus points *(from review)*	+2					**Final score:**
(e) Iteration total *(b + c + d)*	18.5					

Add bonus points, if any and fill in (d)

Sum the points and calculate the total for the iteration in (e)

[1]http://smartdecisionsgame.com

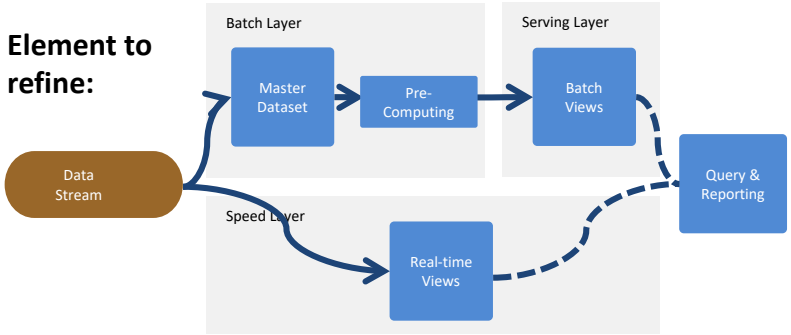
Iteration 2 Goal: Refine the Data Stream Element

EXERCISE

Drivers for the iteration:

- Performance (for Family and Technology)
- Compatibility (for Family)
- Reliability (for Technology)

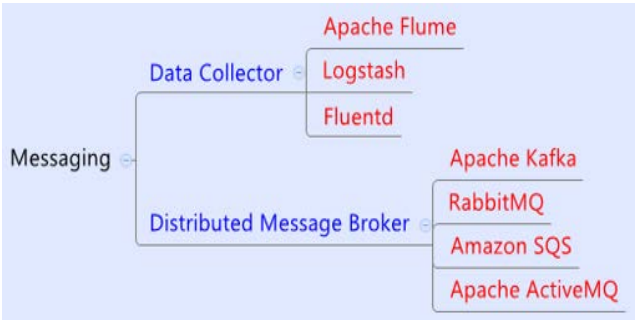
Element to refine:



To Do:

Select 2 cards: 1 Family card and 1 associated Technology card[1]

Alternatives:



Tip:

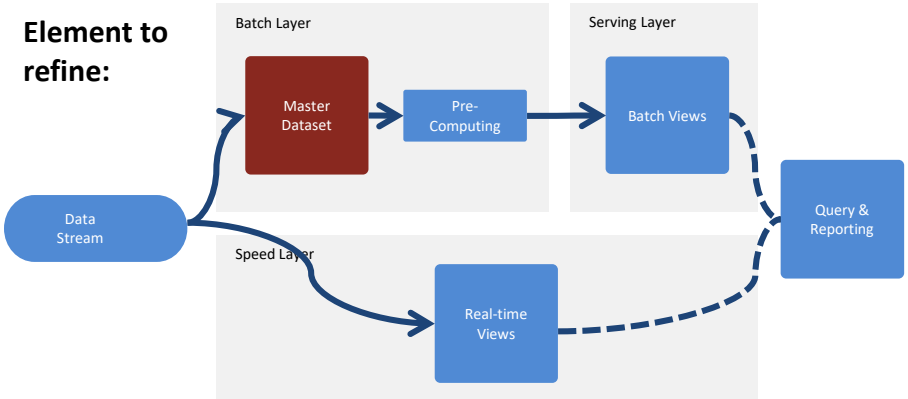
- Look for an option that can be deployed on-premise and on-cloud

[1]http://smartdecisionsgame.com

Iteration 3 Goal: Refine Master Dataset Element

EXERCISE

Element to refine:

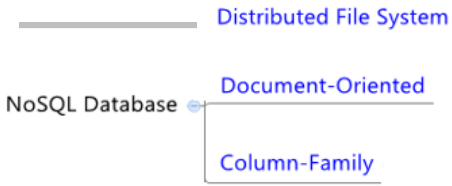


Drivers for the iteration:

- Scalability
- Availability

Select 1 Family card[1]

Possible alternatives:



Tip:

- Look for an option with better extensibility (easy storing of new data formats)

[1]http://smartdecisionsgame.com

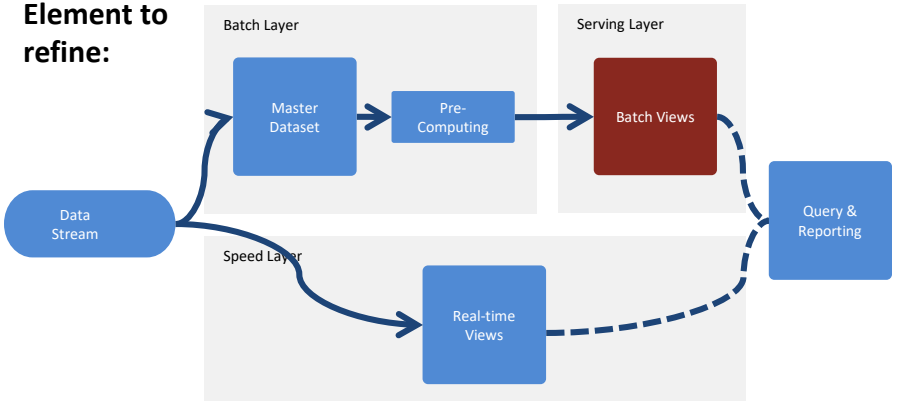
Iteration 4 Goal: Refine Batch Views Element

EXERCISE

Drivers for the iteration:

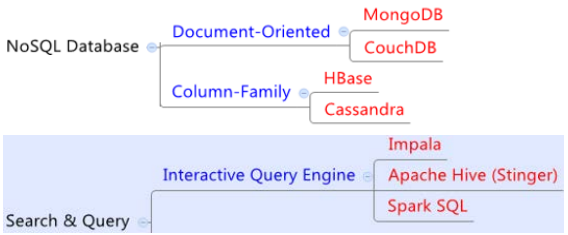
- Ad-hoc analysis (for Family)
- Performance (for Family and Technology)

Element to refine:



To Do: Select 2 cards: 1 Family and 1 associated Technology card[1]

Possible alternatives:



Tip:

- Look for an option that provides ad-hoc analysis and still good performance for static reports
- Impala or Hive are incompatible with Document-Oriented NoSQL databases in the Master Dataset

[1]http://smartdecisionsgame.com

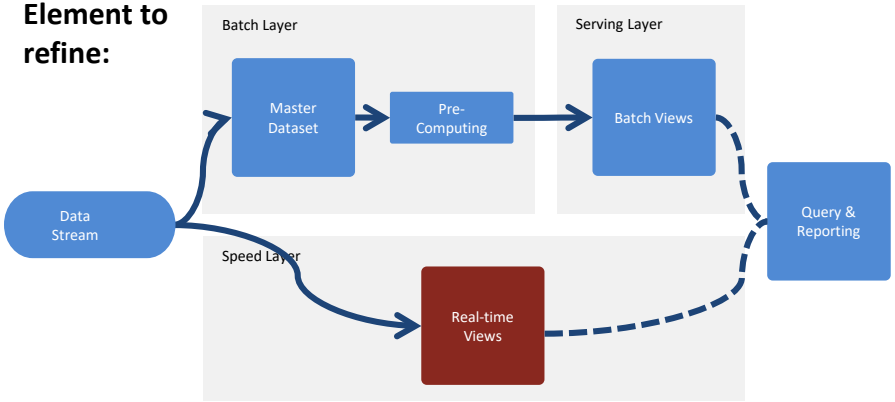
Iteration 5 Goal: Refine Real-Time Views Element

EXERCISE

Drivers for the iteration:

- Ad-hoc analysis (for the family)
- Real-time analysis (for the technology)

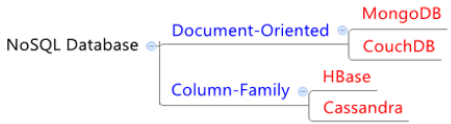
Element to refine:



To Do:

Select 2 cards: 1 Family and 1 associated Technology card[1]

Possible alternatives:



Tip:

- Look for an option that provides full-text search capabilities and extensibility (new data formats and dashboard views)

[1]http://smartdecisionsgame.com

Fill in the Scorecard[1]

EXERCISE

	Iteration #1	Iteration #2	Iteration #3	Iteration #4	Iteration #5	
(a) Design Decisions *(Names of selected design concept(s))*	Lambda Architecture					
(b) Driver selection points *(from cards)*	2+3+3 +3+3=**14**					
(c) Instantiation points *(from dice)*	+2					
(d) Analysis bonus points *(from review)*	+2					**Final score:**
(e) Iteration total *(b + c + d)*	18	10	15	11	8	62.5

Calculate the final score

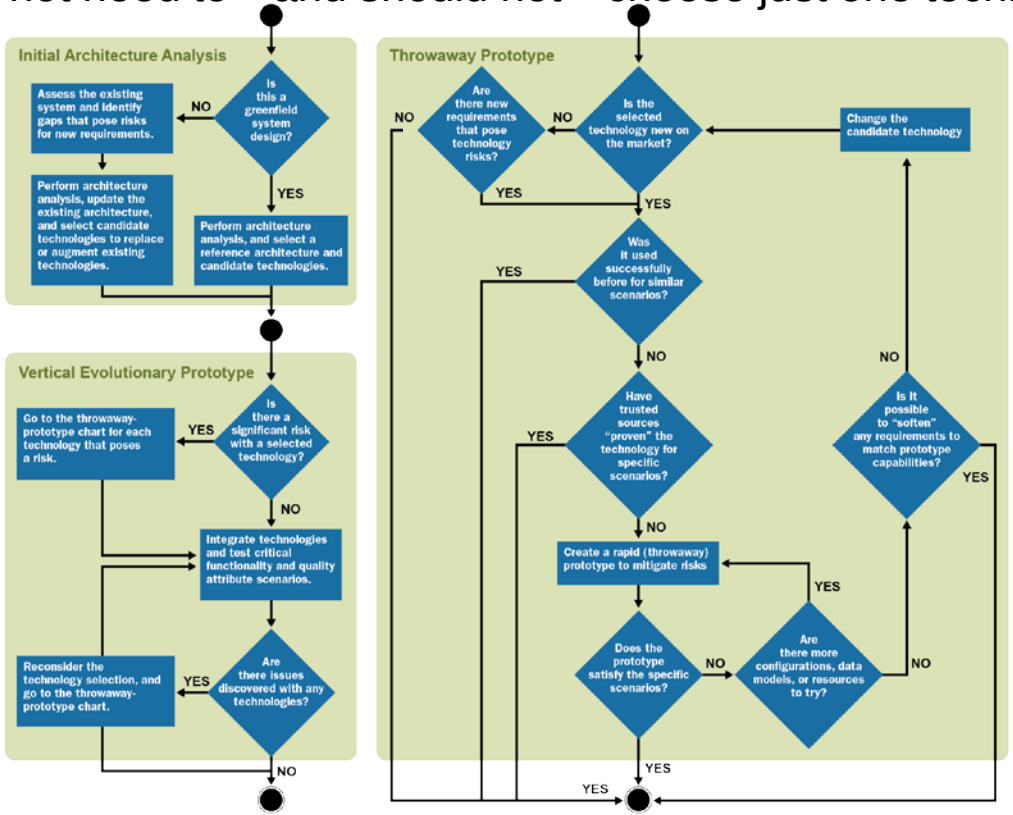
- Add 4 to the team who finished first
- Subtract 4 from the team who finished last

[1]http://smartdecisionsgame.com

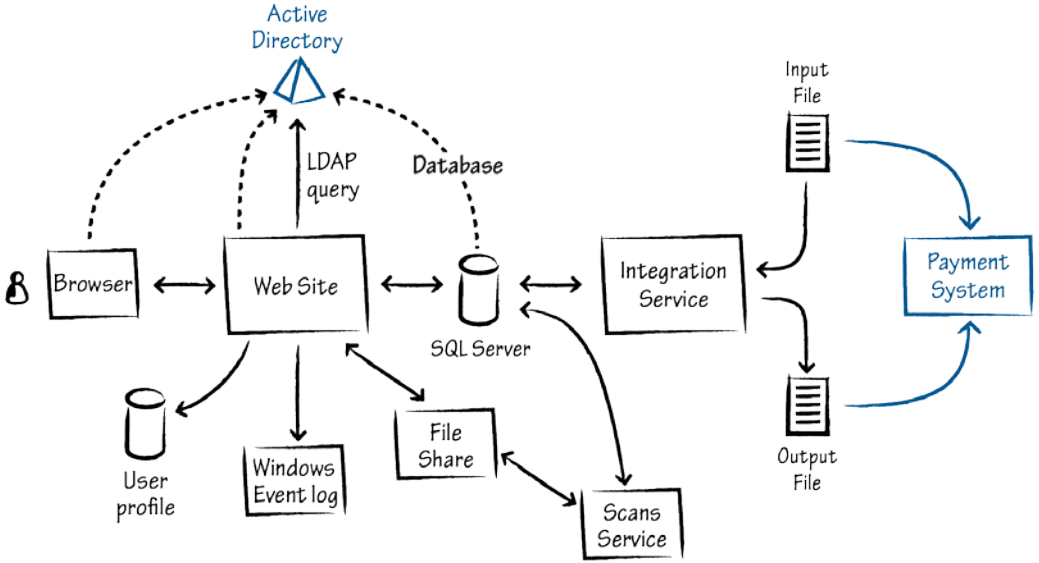
Debrief

Design and Analysis Inform Each Other

You do not need to—and should not—choose just one technique:



Documenting During Design



Element	Responsibility
Browser	Responsible for rendering information and gathering user input
Integration Service	This element modifies and retrieves data from the database in the form of files
…	…

Tracking Progress

When designing there are three key questions to answer:

- How much design do we need to do?
- How much design has been done so far?
- Are we finished?

Agile practices such as the use of backlogs and kanban boards can help you track design progress and answer these questions.

Design Backlog

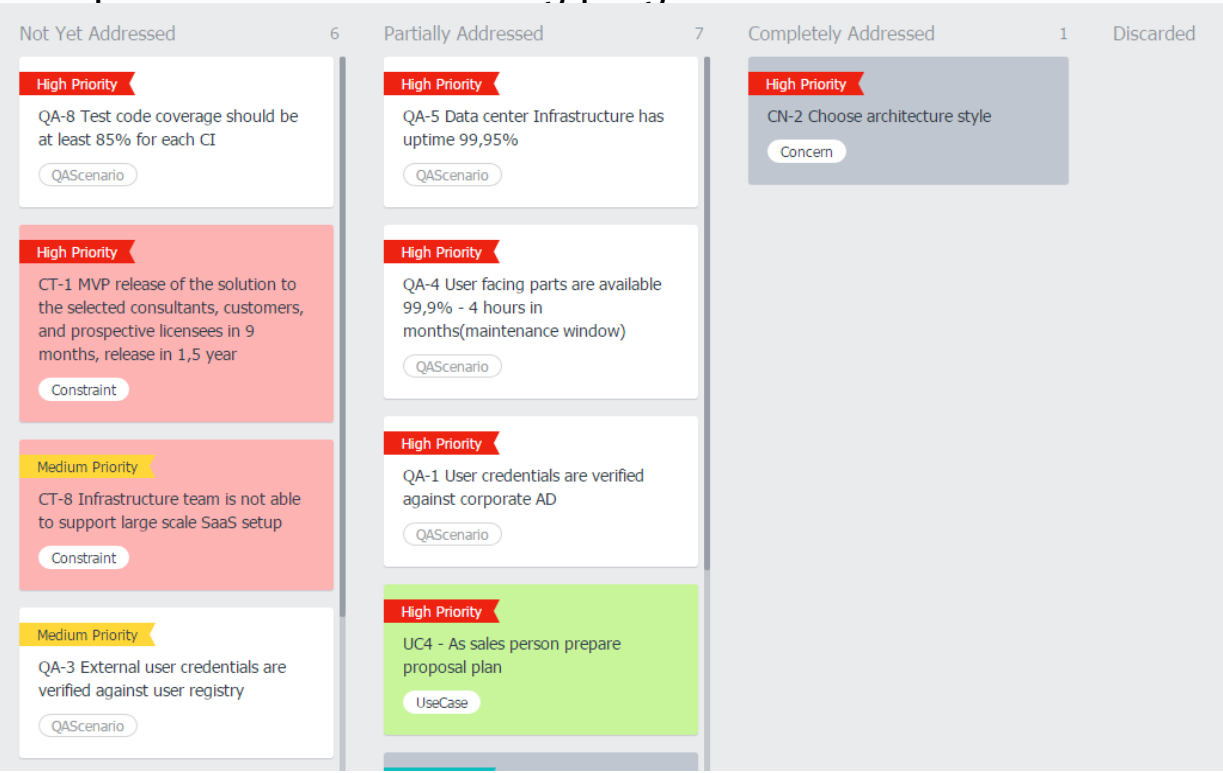
You should create a list of the actions that still need to be performed as part of the architecture design process.

Initially, populate the design backlog with your drivers, but other activities such as the following can be included:

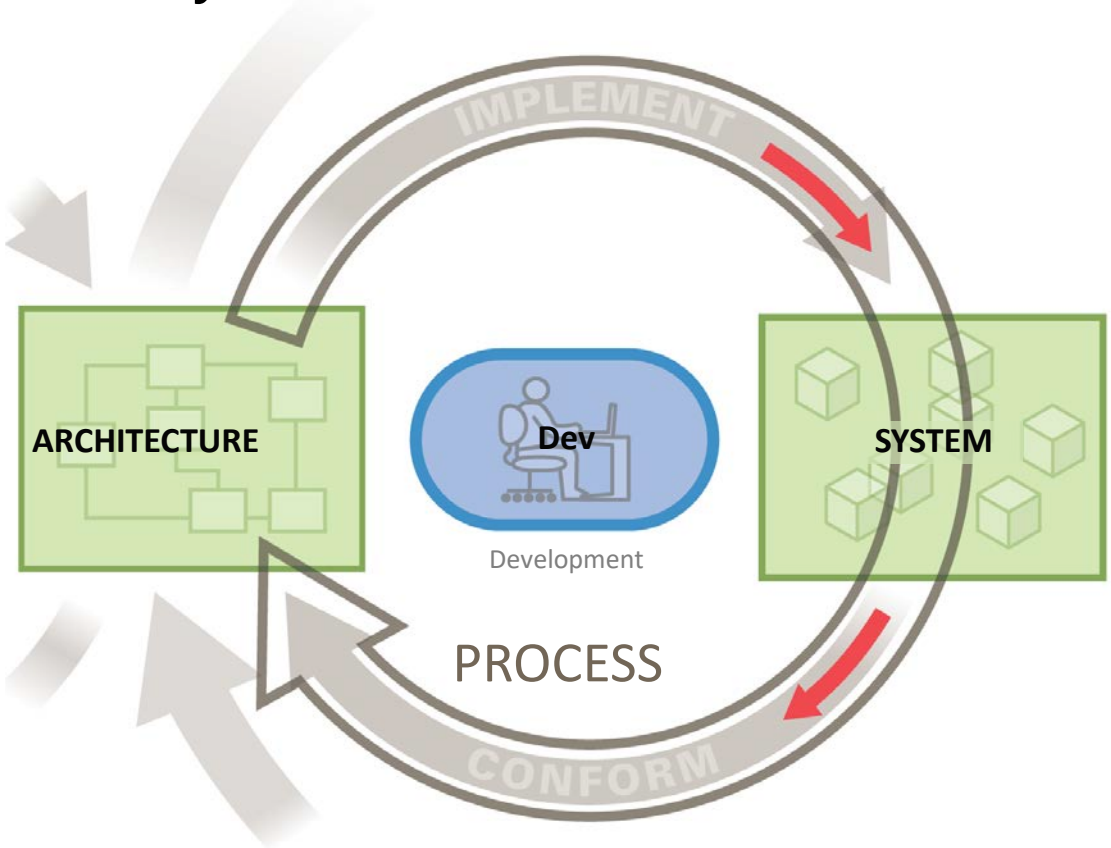
- Creation of a prototype to test a particular technology or to address a specific quality attribute risk
- Exploration and understanding of existing assets (possibly requiring reverse engineering)
- Issues uncovered in a review of the design (recall that we analyze as we are designing)
- Review of a partial design that was performed on a previous iteration

Using a Design Kanban Board

One possible tool for tracking progress is a Kanban board.



Implementation Cycle



Implementation Details

The design cycle of architecture development led to:

- solution organized around quality attribute scenarios
- detail sufficient to provide the required evidence

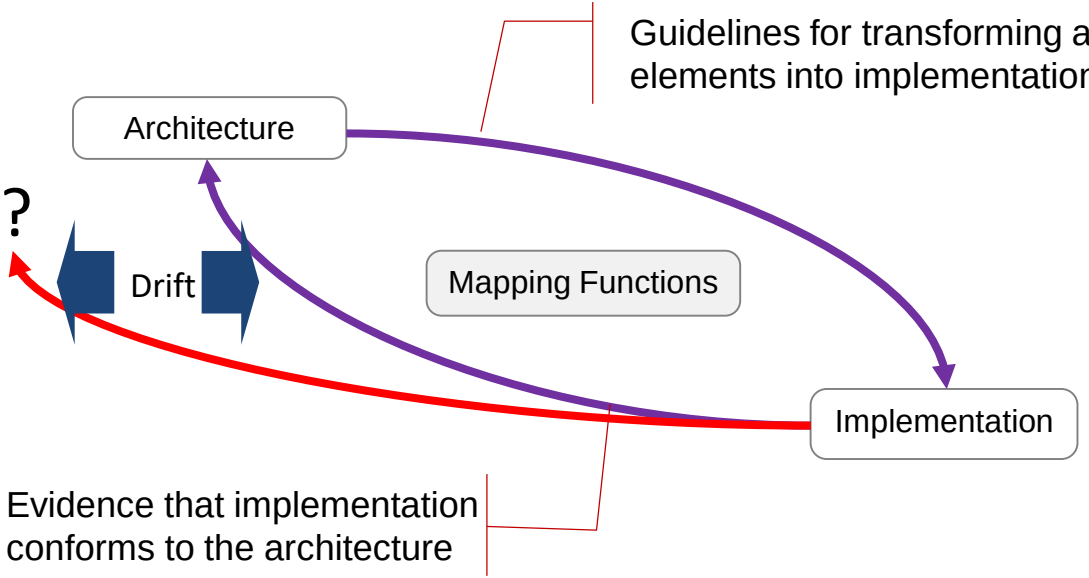
Additional information is needed to begin implementation:

- more details for module interfaces and responsibilities
- documentation organized around work packages
- feedback between developers and architects

An active design review is used to:

- effectively communicate the designed solution to developers
- get feedback about the documentation and areas for improving it

Conformance Review

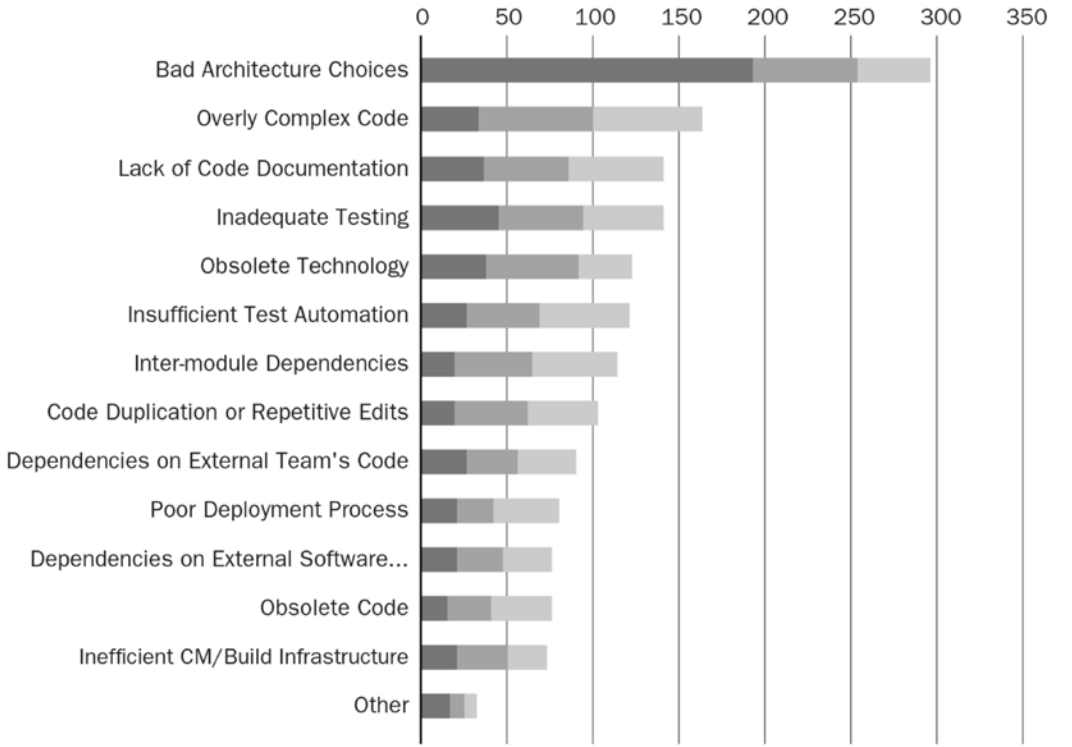


For the implementation to exhibit the quality attributes engineered at the architectural level, it must conform to the architecture.

There will be discrepancies between the architecture and the implementation; also known as “architectural drift.”

When? Release planning

Software Architecture and Design Trade-offs Matter



Results from over 1800 developers from two large industry and one government software development organization.

"Measure it? Manage it? Ignore it? Software Practitioners and Technical Debt" N. Ernst, S. Bellomo, I. Ozkaya, R. Nord, I. Gorton, Int. Symp on Foundations of Software Engineering 2015.

Roadmap and Architecture

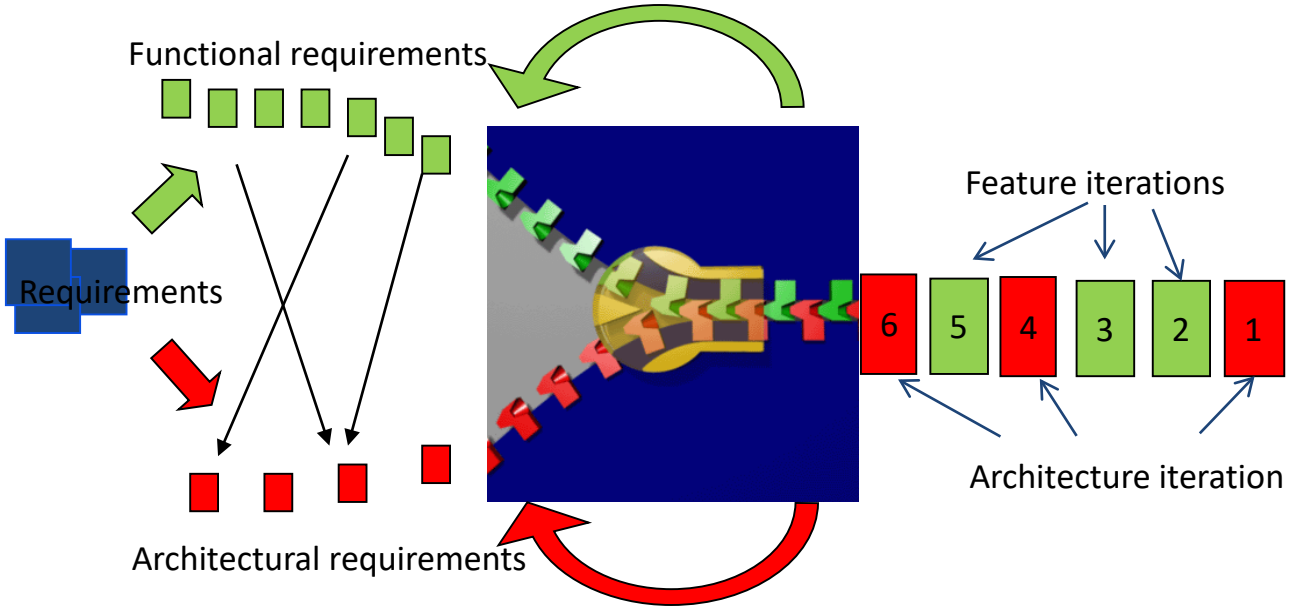
Roadmapping

- Enables making short-term decisions in a long-term context
- Balances global and local optimization

Everything is always changing – why should I plan for the future?

- Make everything equally hard to respond to, or…
- Use architecture to enable anticipated changes to be made more efficiently
- Pay special attention to technical debt that may accumulate due to architecture decisions
- Incorporate light-weight analysis techniques into sprint and release planning

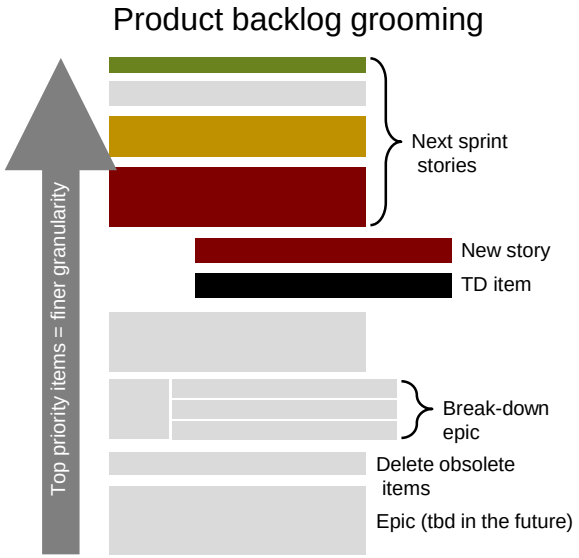
The Zipper Metaphor



Nord, R.L., Ozkaya, I. and Kruchten, P. Agile in Distress: Architecture to the Rescue. T. Dingsøyr et al. (Eds.): *XP 2014 Workshops*, LNBIP 199, pp. 43–57, 2014. Springer International Publishing Switzerland 2014

Essential Software Artifacts

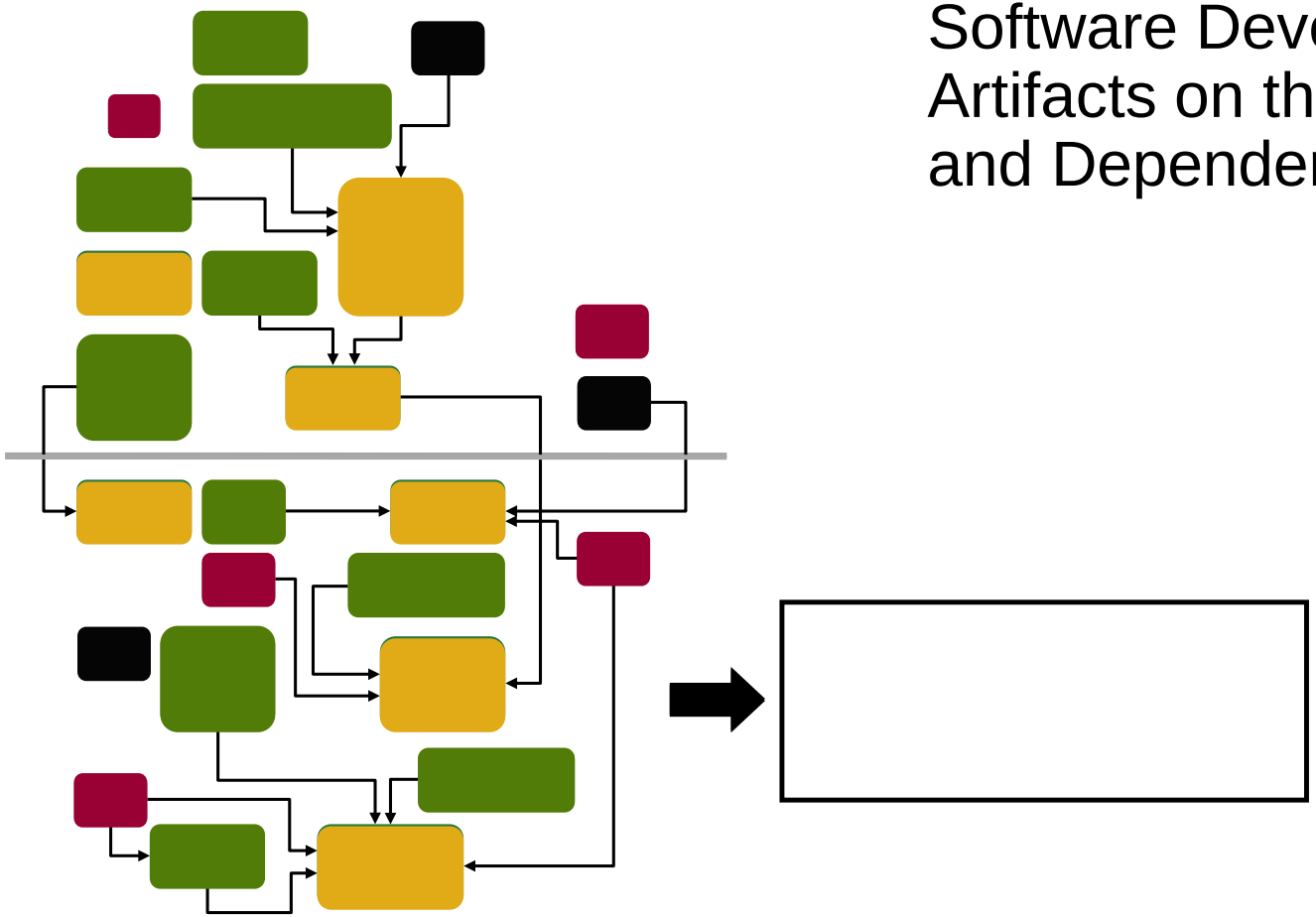
Incorporate Technical Debt Management to Release Planning



Discussion of backlog items should include an explicit focus on architecture and any technical debt items in addition to new features and defects.

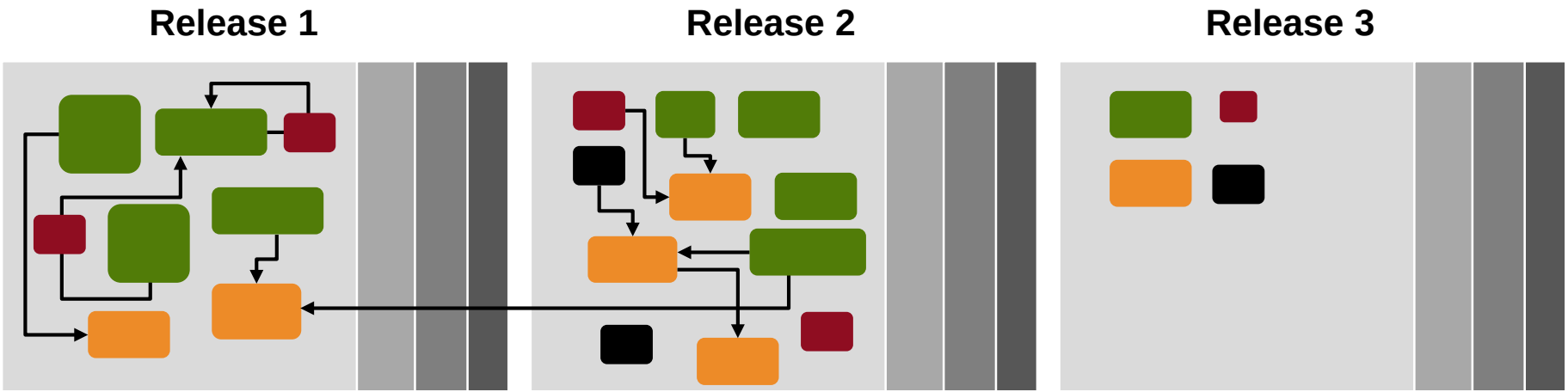
Architecture stories and technical debt should be explicitly recorded, similar to new user stories, defects, and the like.

Software Development Artifacts on the Backlog and Dependencies

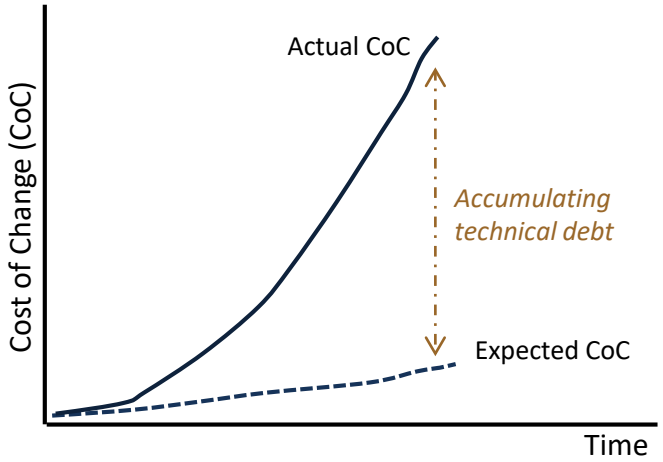


Impact on Product Development

Impact forces choices that bring into focus issues of cost and value, current needs, and future potential.



The Cost of Accepting Technical Debt

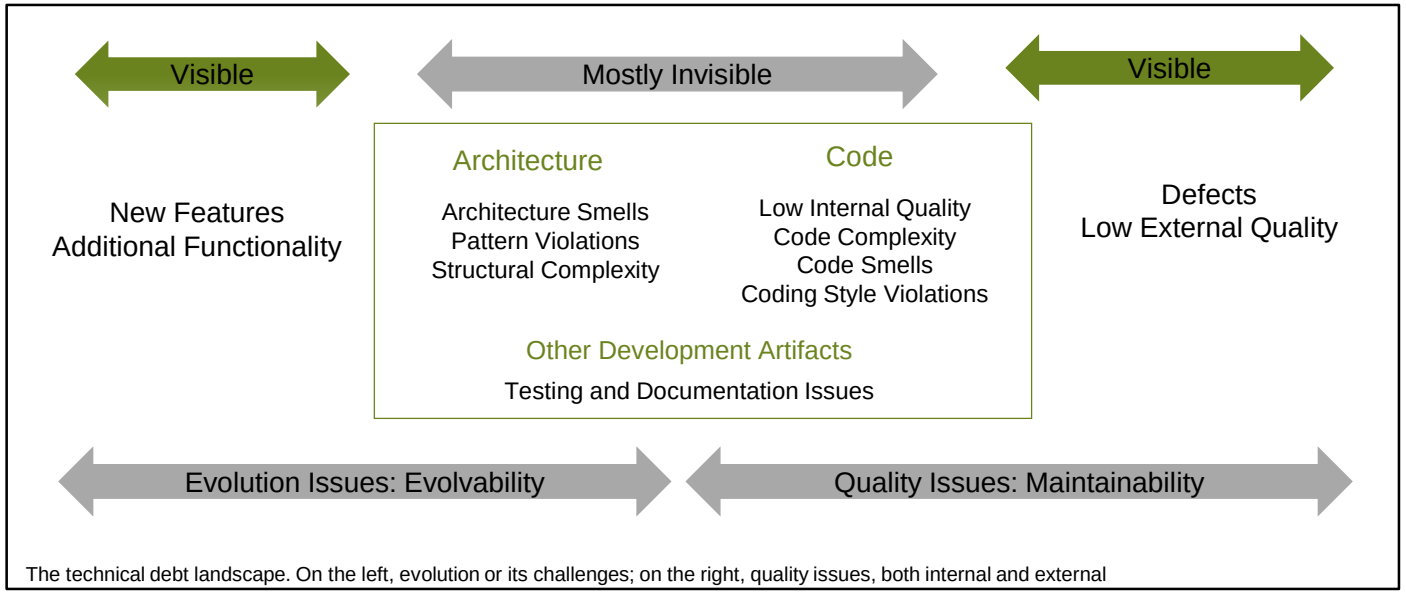


For each instance of technical debt

- Understand range of consequences
- Measure what you can
- Qualitatively assess what you can't
- Reconcile data with assessments

Make informed trade-off decisions about remediation.

From Symptoms to Specifics



The technical debt landscape. On the left, evolution or its challenges; on the right, quality issues, both internal and external

Kruchten, P. Nord, R.L., Ozkaya, I. 2012. Technical Debt: From Metaphor to Theory and Practice, IEEE Software, 29(6), Nov/Dec 2012.

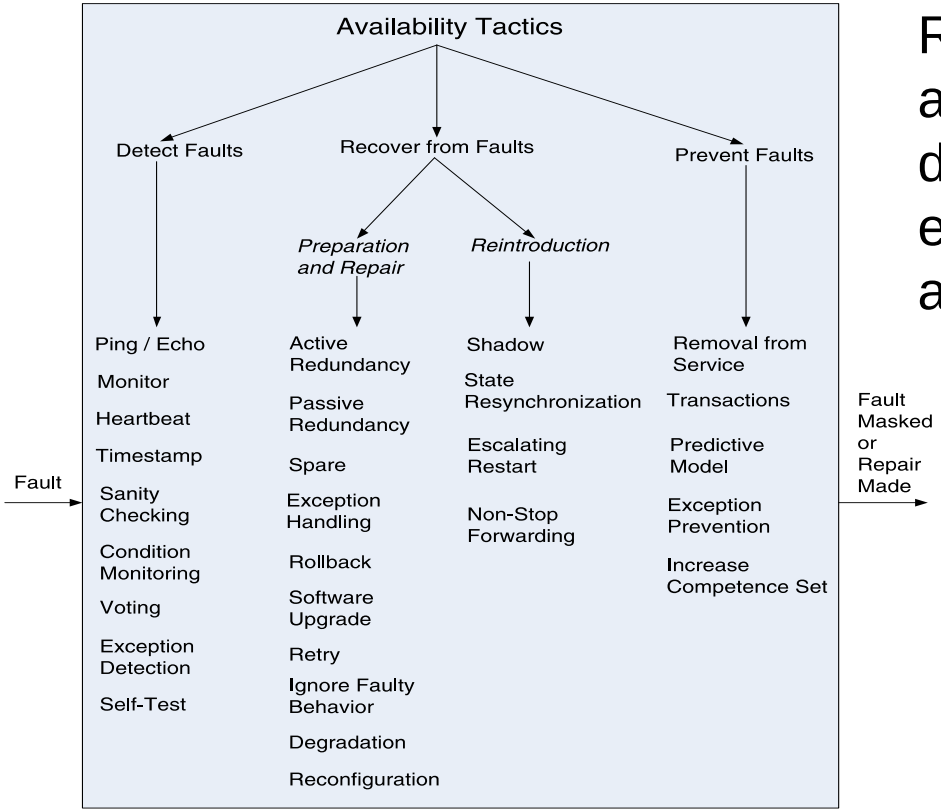
Light-weight Analysis Technique
Tactics-Based Questionnaires

We can employ tactics as an a guide to analysis. By turning every tactic into a question, we create a set of QA-specific questionnaires.

These are employed as follows:

1. *The reviewers determine a number of quality attributes to drive the review.* These quality attributes will determine the selection of tactics-based questionnaires to use.
2. *The architect presents the portion of the architecture to be evaluated.* The reviewers individually ensure that they understand the architecture. Questions at this point are just for understanding.
3. *For each question from the questionnaire, the designer walks through the architecture and explains whether and how the tactic is addressed.* The reviewers ask questions to determine how the tactic is employed, the extent to which it is employed, and how it is realized.
4. *Potential problems are captured.* Real problems must be fixed or a decision must be explicitly made to accept the risks.

Example – Availability Tactics



Recall that *tactics* are design decisions that enable quality attributes.

Availability Tactics-Based Questions

•Does the system use **ping/echo** to detect a failure of a component or connection, or network congestion?

•Does the system use a component to **monitor** the state of health of other parts of the system? A system monitor can detect failure or congestion in the network or other shared resources, such as from a denial-of-service attack.

•Does the system use a **heartbeat**—a periodic message exchange between a system monitor and a process—to detect a failure of a component or connection, or network congestion?

•Does the system use a **timestamp** to detect incorrect sequences of events in distributed systems?

•Does the system employ **rollback**, so that it can revert to a previously saved good state (the “rollback line”) in the event of a fault?

…

Lightweight Analysis Exercise

Let’s try a lightweight analysis.

1. Form groups of 2-5 people.

   One person is the architect.
   One person is the scribe (and questioner).
   Other group members are questioners.

2. Choose a system that the architect is intimately familiar with.

Lightweight Analysis Exercise

3. Choose one of the following quality attributes:
   - Availability
   - Interoperability
   - Modifiability
   - Performance
   - Security
   - Testability
   - Usability

4. Now turn each tactic into an interview question.

5. Record the responses to each question using the worksheet in the LAA section of the Supplementary materials.

Who? Necessary organic capabilities

Architects: Anchors or Accelerators to Organizational Agility?

You can’t outsource architecture oversight

- You need the organic capabilities to own the architecture, though you can get help and contract out tasks
- Architecture is strategy
- System outlives contracts

How can architects accelerate agility in organizations?

- Be agile
- Be architects of structure, time, and transition
- Create agile design guidelines

Jim Highsmith 2010

Architecture Quick Look

Business and development stakeholders

- investigate agile software development and architecture principles and practices
- identify risks and factors worthy of attention
- set priorities in improving software development

Why? The risks uncovered in this analysis will guide the application of the other architecture practices.

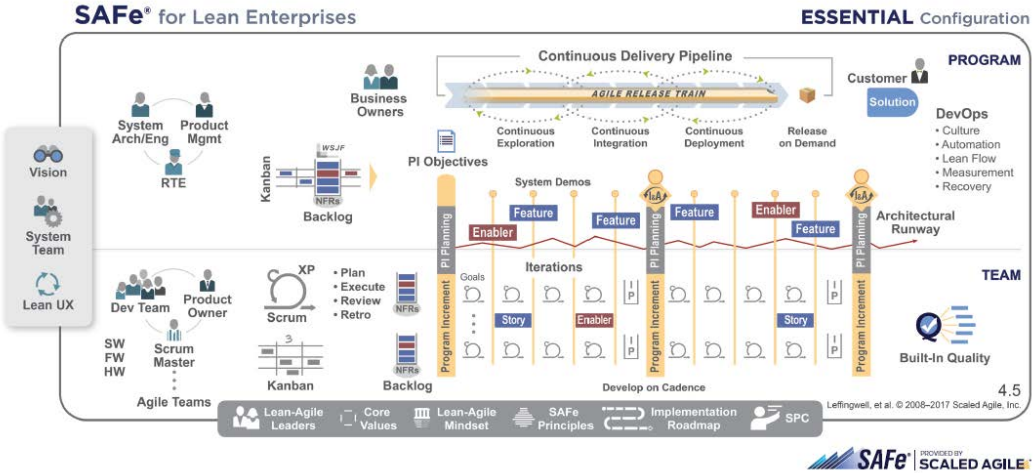
Ozkaya, I., Gagliardi. M., and Nord, R.L. 2013. Architecting for Large Scale Agile Software Development: A Risk-Driven Approach, *Crosstalk* 26, 3 (May/June 2013): 17-22.

Understanding Program Readiness for Agile

SEI Agile in Government. https://sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=21345

Take away

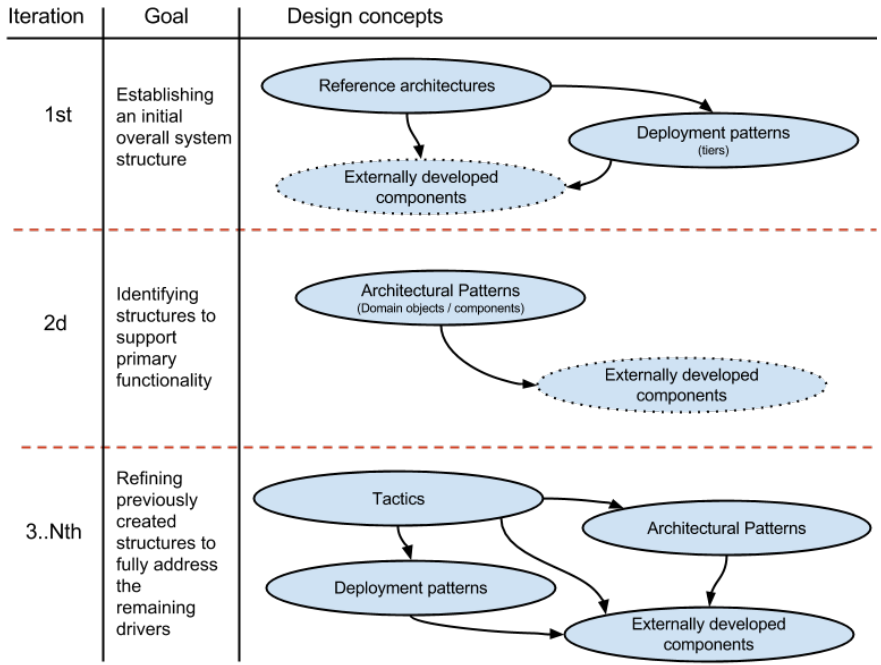
Agilely architecting …



The essence of agile architecting is to conduct these activities concurrently with the right balance:

- requirements originating from the problem space inform architecture and development,
- explorations originating from architecture and implementation investigations assist in eliciting and detailing requirements.

… an agile architecture



Architecture understood as design concepts that influence the time and cost to implement, test, and deploy changes:

- reference architectures
- architectural design patterns
- tactics
- design principles
- externally developed components

Cervantes, H., and Kazman, R. 2016. Designing Software Architectures: A Practical Approach, SEI Series in Software Engineering, Addison-Wesley, 2016.

Agilely architecting an agile architecture

Agilely architecting an agile architecture has four key requirements:

1. Focus on ***key quality attributes*** and incorporate these into technical explorations within prototyping and spikes
2. Understand that a successful product is a combination of customer-visible features and the underlying ***infrastructure that enables*** those
3. Recognize that an architecture that enables ease of maintainability and evolvability is the result of ***ongoing, explicit attention***
4. Continuously manage ***dependencies*** between functional and architectural requirements and ensure that the architectural foundation is put in place in a just-in-time manner

Bellomo, S., Kruchten, P., Nord, R.L., Ozkaya, I.: How to Agilely Architect an Agile Architecture? *Cutter IT J.* 27, 12–17 (2014)

Contact Information

Ipek Ozkaya
Architecture Practices Initiative
Email: ozkaya@sei.cmu.edu

Robert Nord
Architecture Practices Initiative
Email: rn@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/architecture

Customer Relations
Email: info@sei.cmu.edu
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257

Quality Attribute Scenarios (reference)

1. ***Stimulus*** - the condition that affects the system
2. ***Response*** - the activity that results from the stimulus
3. ***Source of Stimulus*** - the entity that generated the stimulus
4. ***Environment*** - the condition under which the stimulus occurred
5. ***Artifact*** - the entity that was stimulated
6. ***Response Measure*** - the measure by which the system's response will be evaluated

Architecture Practices (reference)

Core Practices

Elicit and capture business and mission goals in the form of quality attribute scenarios.

Iteratively and incrementally transform scenarios into architectural structure and content.

Evaluate the architecture for risks to achievement of the scenarios.

Transition the architecture to implementation, build it, and verify compliance.

Supporting Methods

Quality Attribute Workshop (QAW)
Business Thread Workshop (BTW)
Architecture Roadmap

Attribute-Driven Design (ADD)
Views and Beyond (V&B)

Architecture Tradeoff Analysis Method (ATAM)
Scenario-based peer reviews

Active Design Review (ARID)
Design and implementation rules
Conformance reviews

www.sei.cmu.edu/architecture