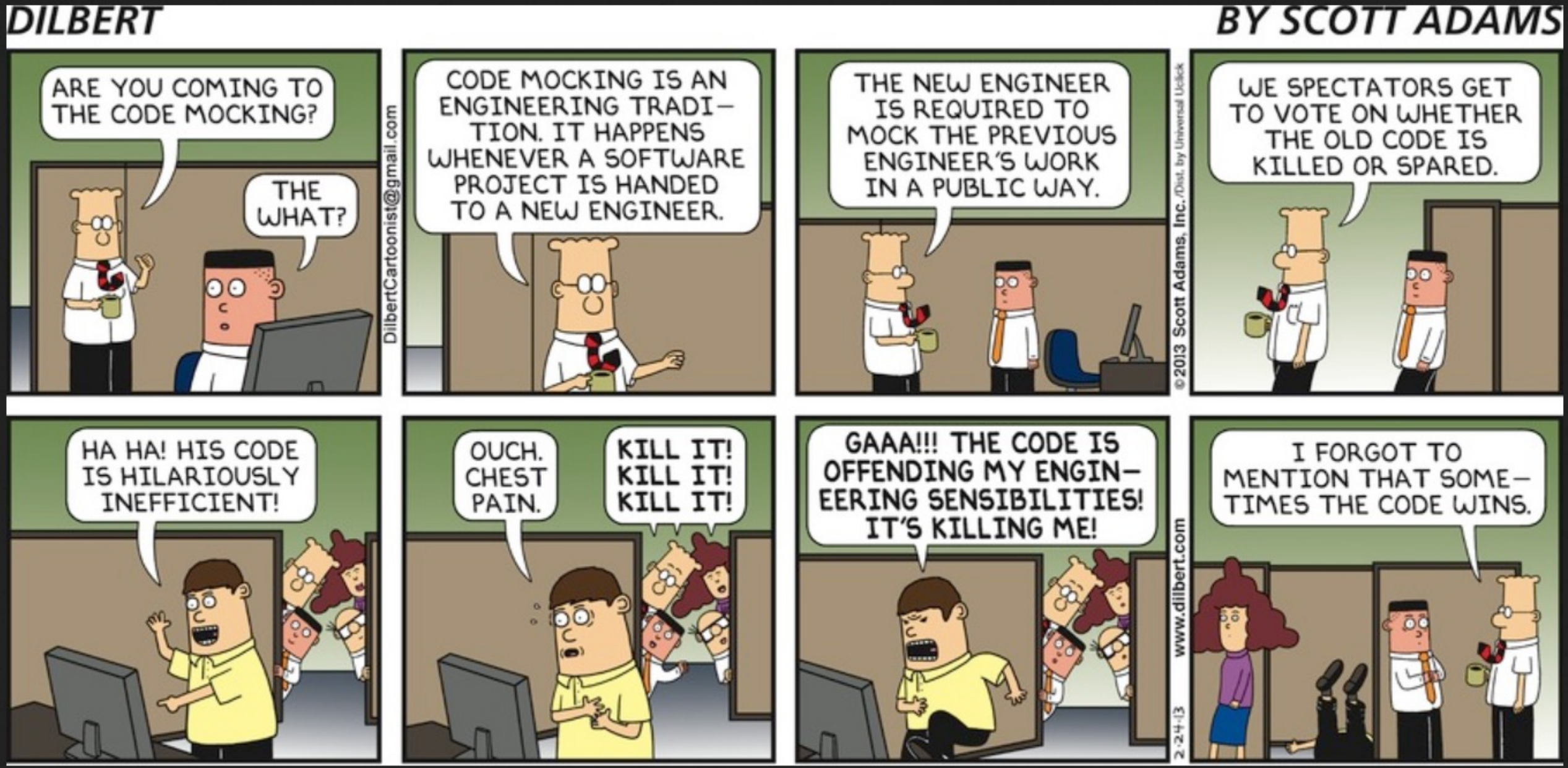
Software architecture

concepts and process

version: 1.0.1



http://dilbert.com/strip/2013-02-24

Software architecture

What

Why

How

When

What is software architecture?

Martin Fowler says...

…the decisions that you **wish** you could get right early

Martin Fowler says...

…the important stuff.
Whatever that is.

Simon Brown says...

…it's anything and everything related to the significant elements of a software system; from the structure and foundations of the code through to the successful deployment of that code into a live environment.

Simon Brown says...

The architectural decisions are those that you can't reverse without some degree of effort.
Or, put simply, they're the things that you'd find hard to refactor in an afternoon.

Simon Brown says…

…architecture provides
structure, firm foundations, vision
and technical leadership

The Frustrated Architect - https://www.infoq.com/presentations/The-Frustrated-Architect

Anecdotes on Predicting the Future

- TV Show has a website [1]
  - Website is usually used for donations, news on show characters, etc. Number of visitors is stable and low.
  - TV Host casually mentions "visit our website for a chance to win a prize"
- Pinterest's growing pains [2]:
  - Initial architecture based on replication of DB was inadequate and difficult to scale.

Politiker og minister Karl Kristian Steincke. [3]
Foto: Folketingets Bibliotek og Arkiv

[1] - https://ilegra.com/en/two-examples-of-bad-software-architecture-that-had-a-negative-impact-on-businesses/
[2] - https://medium.com/pinterest-engineering/sharding-pinterest-how-we-scaled-our-mysql-fleet-3f341e96ca6f
[3] - https://quoteinvestigator.com/2013/10/20/no-predict/

How to Predict the Future

An Introduction

Input for the architecture

- Functional Requirements
  - Provides a lot of what – very little of how
  - Many of the functional req's. probably have little impact on the overall architecture if they are changing
  - Often defined as User Stories or Use Cases

- Non-Functional Requirements
  - Closer to the how than the previous
  - Many of these are very sensitive to the architecture
    - and in reverse can have a big impact on the architecture, if they change
  - Often defined as (quantified) Quality Attributes

Examples

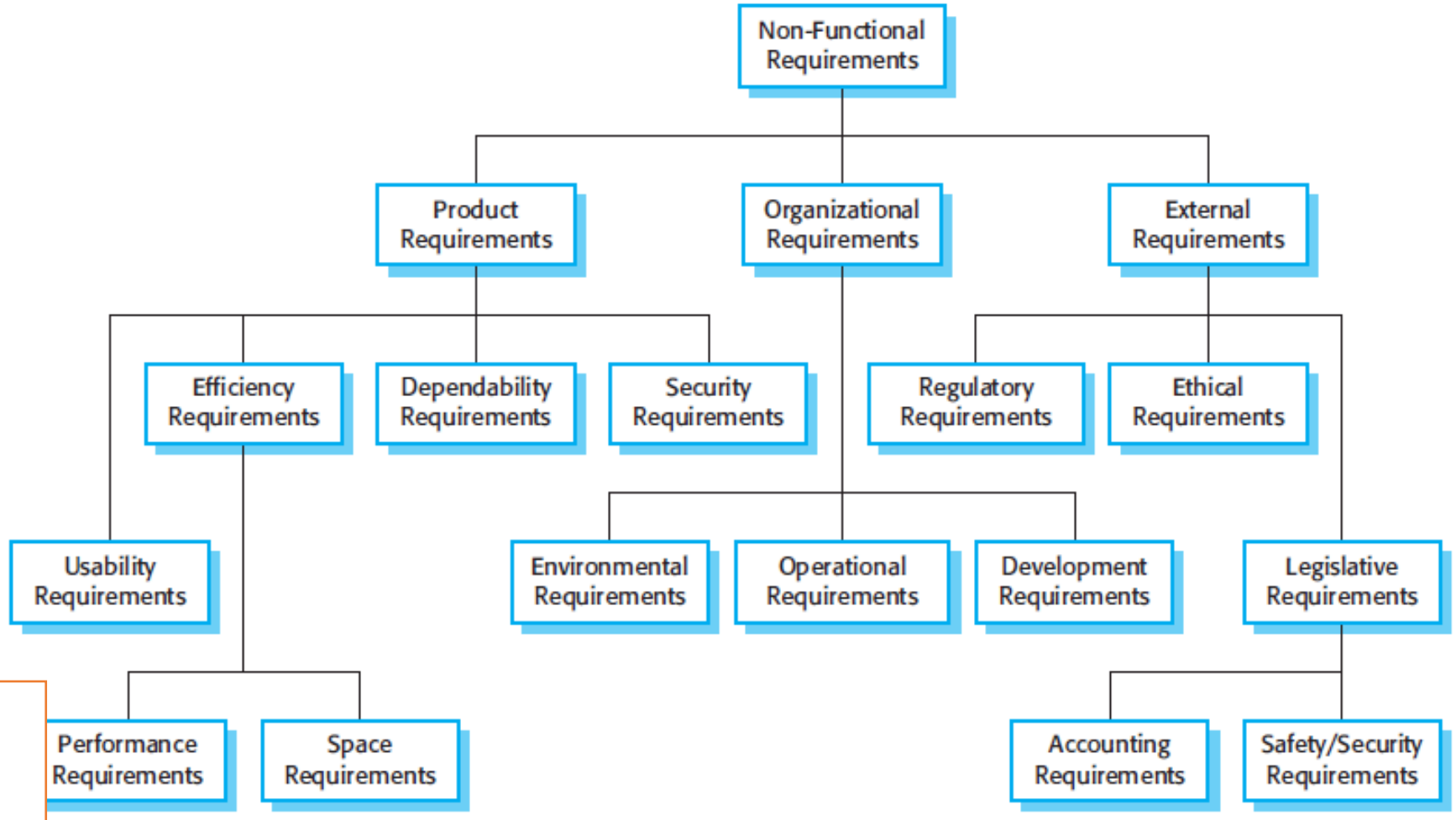
Little impact: User shall be able to search for a video.

Medium impact: User shall receive video recommendations.

Little impact: User shall be able to login with Google, Facebook, or Apple, account.

Big impact: System shall play movies to 1000 users simultaneously.

Non-functional requirements



Ex: System shall play movies to 1000 users simultaneously.

Source: Software Engineering, 9th edition – Ian Sommerville

Quality attributes

http://iso25000.com/index.php/en/iso-25000-standards/iso-25010

*-llities

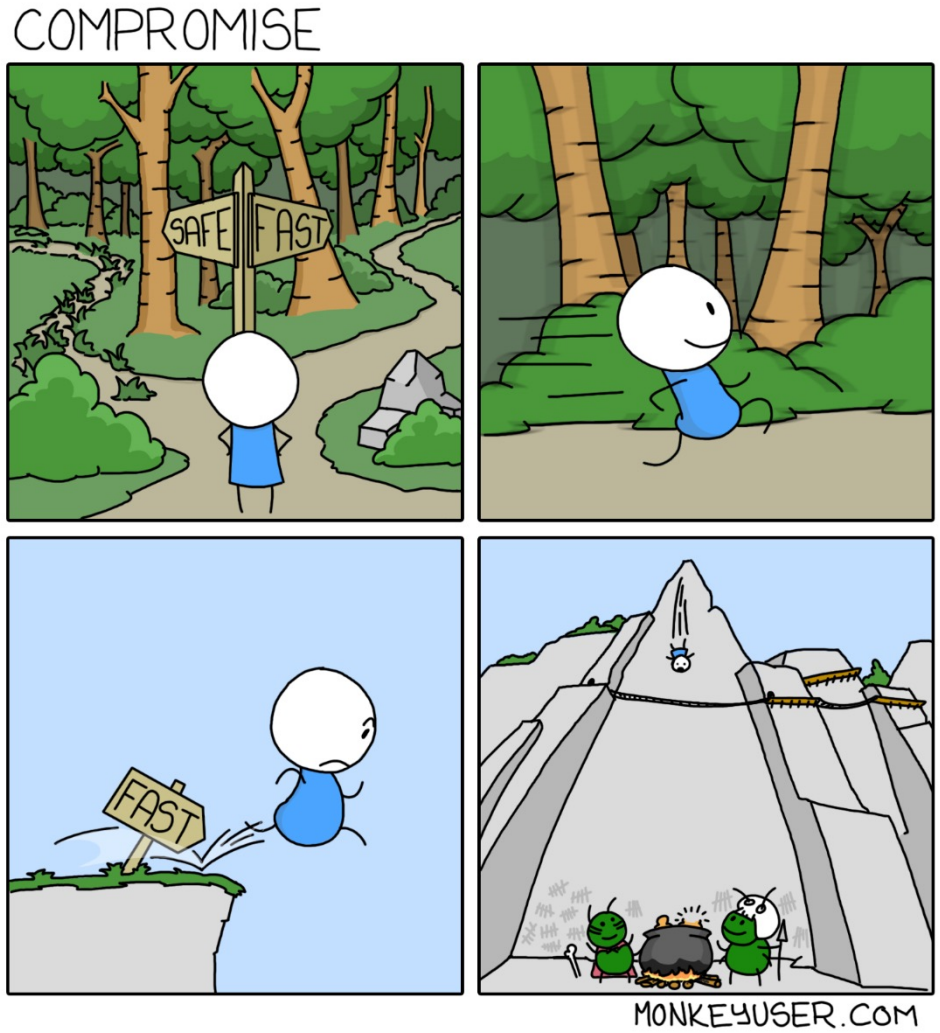
Quality attributes [edit]

Notable quality attributes include:

- accessibility
- accountability
- accuracy
- adaptability
- administrability
- affordability
- agility (see Common subsets below)
- auditability
- autonomy [Erl]
- availability
- compatibility
- composability [Erl]
- configurability
- correctness
- credibility
- customizability
- debuggability
- degradability
- determinability
- demonstrability
- dependability (see Common subsets below)
- deployability
- discoverability [Erl]
- distributability
- durability
- effectiveness
- efficiency
- evolvability
- extensibility
- failure transparency
- fault-tolerance
- fidelity
- flexibility
- inspectability
- installability
- integrity
- interchangeability
- interoperability [Erl]
- learnability
- localizability
- maintainability
- manageability
- mobility
- modifiability
- modularity
- observability
- operability
- orthogonality
- portability
- precision
- predictability
- process capabilities
- producibility
- provability
- recoverability
- relevance
- reliability
- repeatability
- reproducibility
- resilience
- responsiveness
- reusability [Erl]
- robustness
- safety
- scalability
- seamlessness
- self-sustainability
- serviceability (a.k.a. supportability)
- securability (see Common subsets below)
- simplicity
- stability
- standards compliance
- survivability
- sustainability
- tailorability
- testability
- timeliness
- traceability
- transparency
- ubiquity
- understandability
- upgradability
- usability
- vulnerability

https://en.wikipedia.org/wiki/List_of_system_quality_attributes

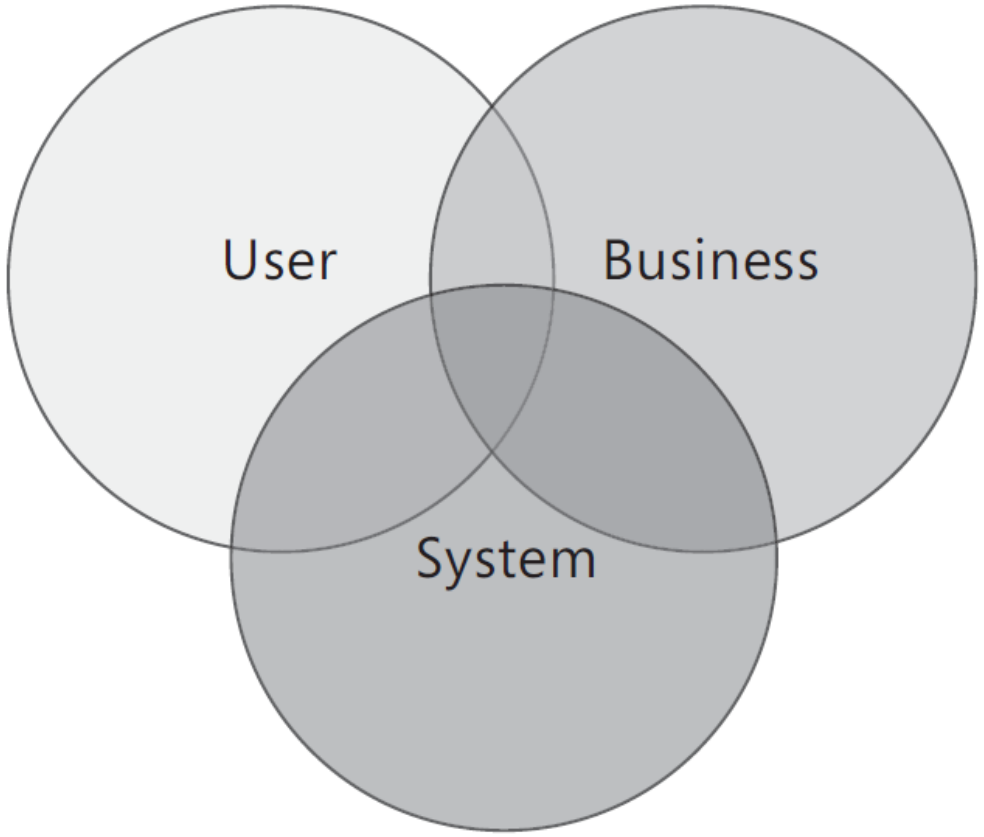
The architecture is a compromise



Quiz (mentimeter)

https://www.monkeyuser.com/2018/compromise/

The architecture is a compromise



- A well specified system contains the requirements and expectations of these stake holders described as:

- User Stories
- Quality Attributes

- The Architecture must be a compromise across these

A noun and a verb

An architecture

To architecture (to architect?)

As a noun:

structure

(and behavior)

As a verb:

process

"I'll go talk to the stakeholders and find out their requirements... in the meantime, you guys start coding."

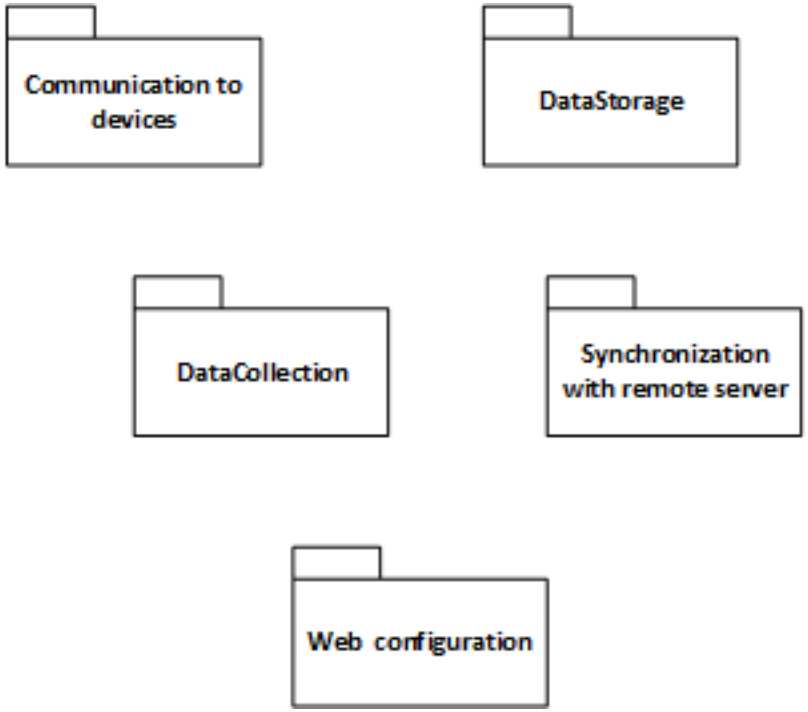
https://www.modernanalyst.com/Resources/BusinessAnalystHumor/tabid/218/ID/2798/While_I_get_the_business_requirements.aspx

Structure

(and behavior)

Structure

initial design for a data collection system



Examine the functional requirements and put functionality into 'boxes' or modules.

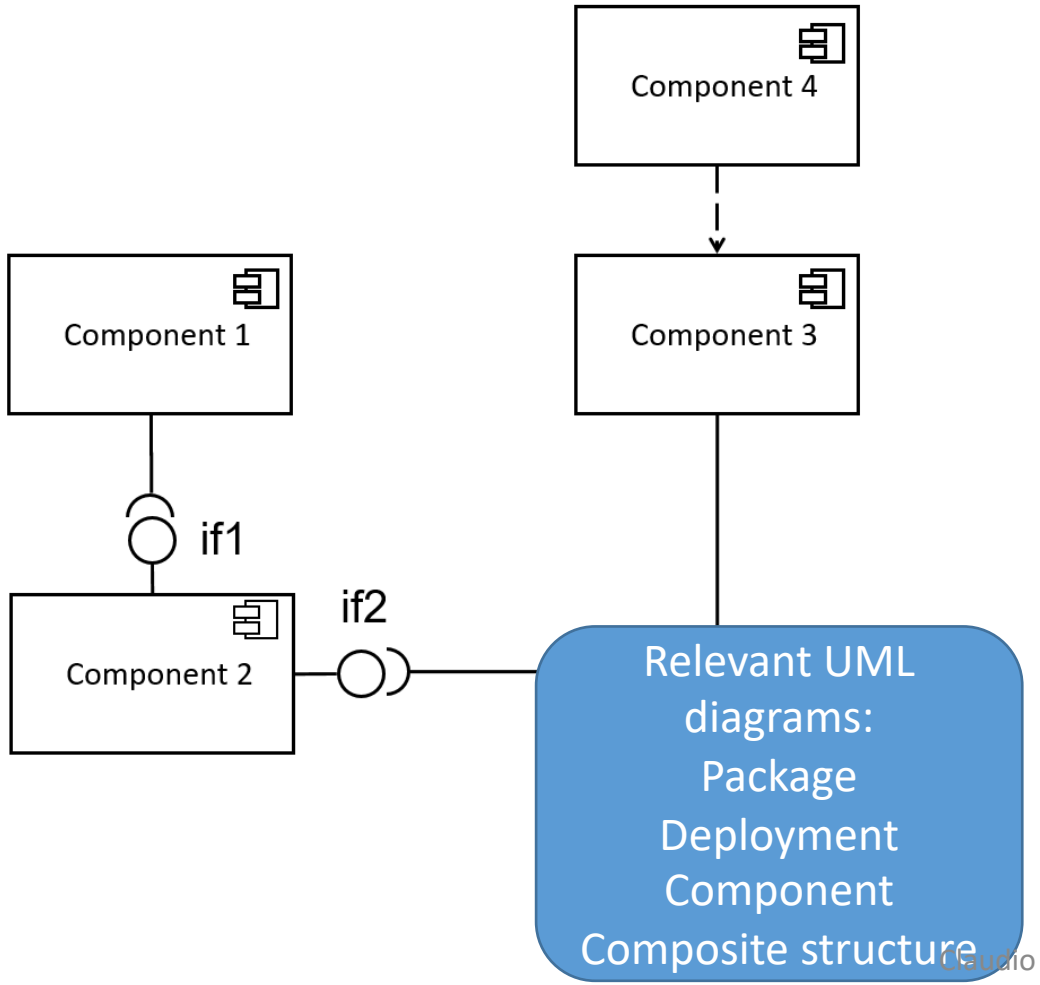
What belongs together?

What functionality depends on other functionality?

What should be exposed at the boundaries of the boxes?

The boxes could be UML packages.

Structure



Examine the functional requirements and put functionality into 'boxes' or modules.

What belongs together?

What functionality depends on other functionality?

What should be exposed at the boundaries of the boxes?

Or UML Components.

SOLID principles and architecture

The SOLID principles also apply at the architecture level.

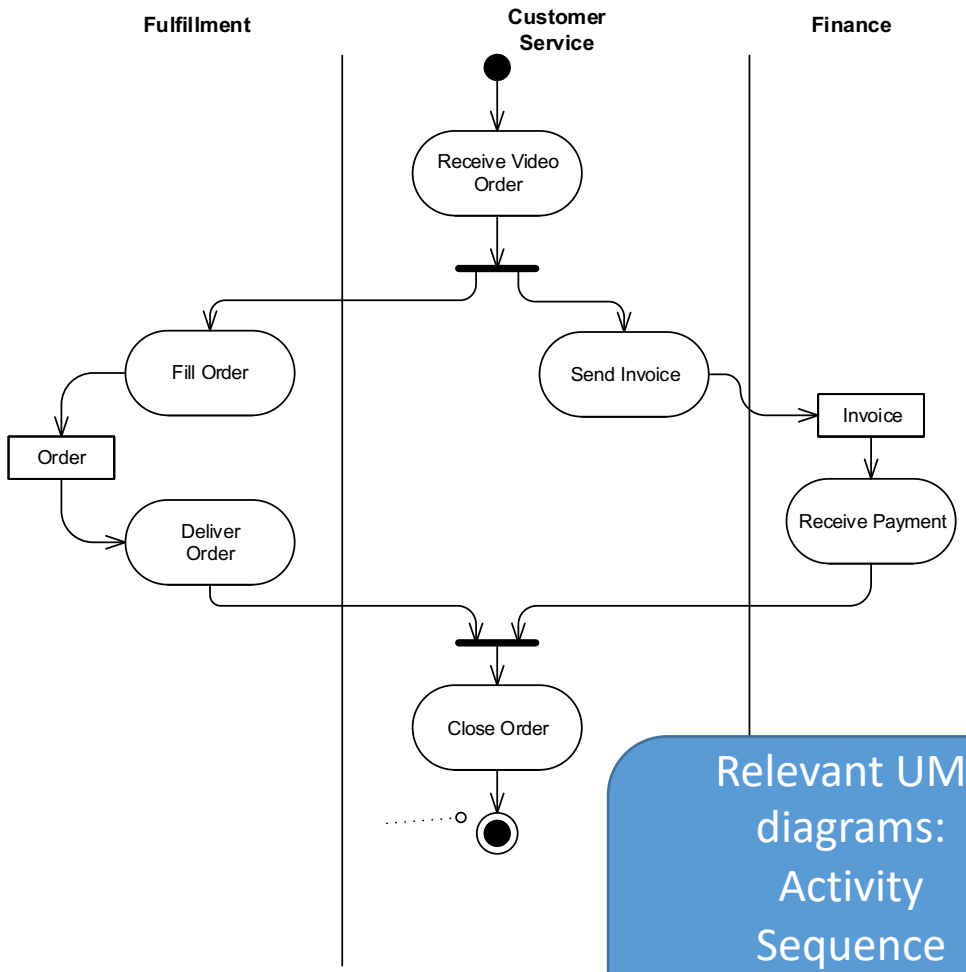
We're just talking **modules**

and not classes.

High cohesion
and
Low coupling

Single Responsibility Principle
Open – Closed Principle
Lisskov's Substitution Principle
Interface Segregation Principle
Dependency Inversion Principle

Behavior



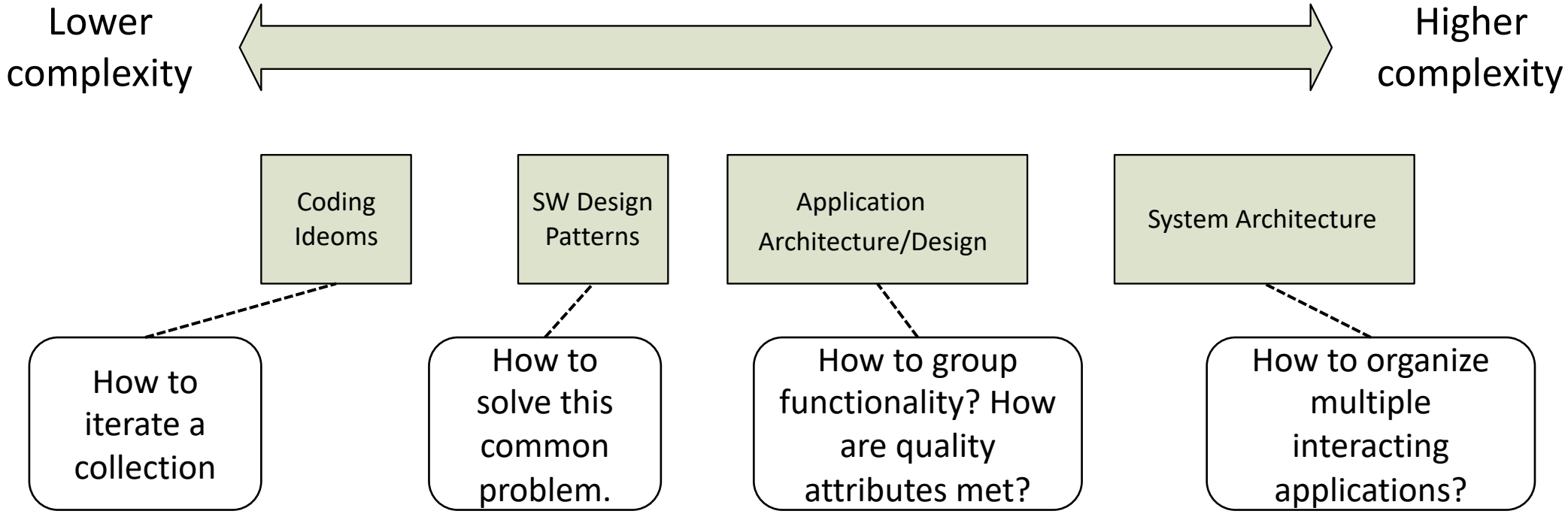
Relevant UML diagrams:
Activity
Sequence
State
Communication

How does data flow between the modules?

What is the flow through the software?

Use UML diagrams for behavior.

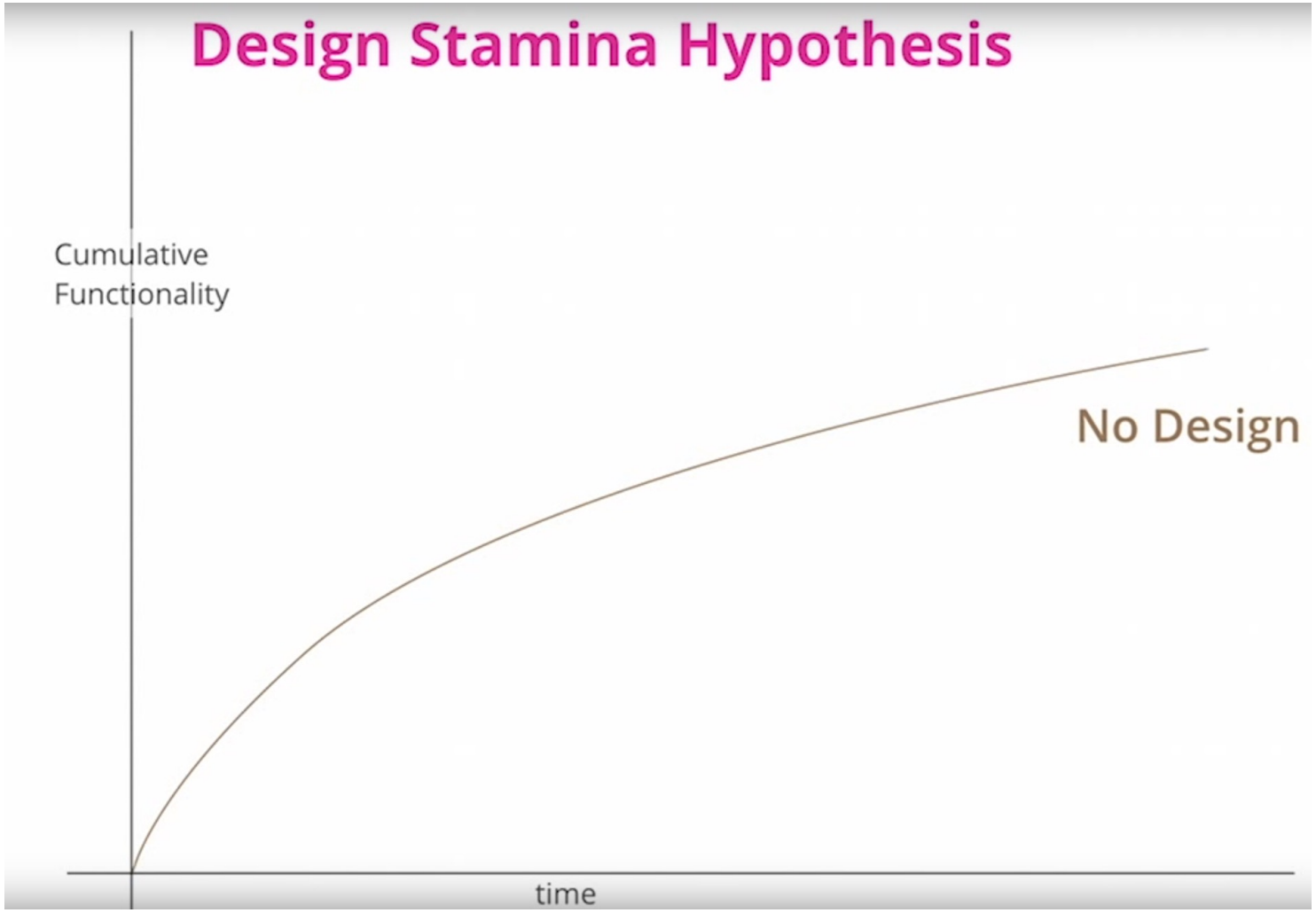
Design or architecture?



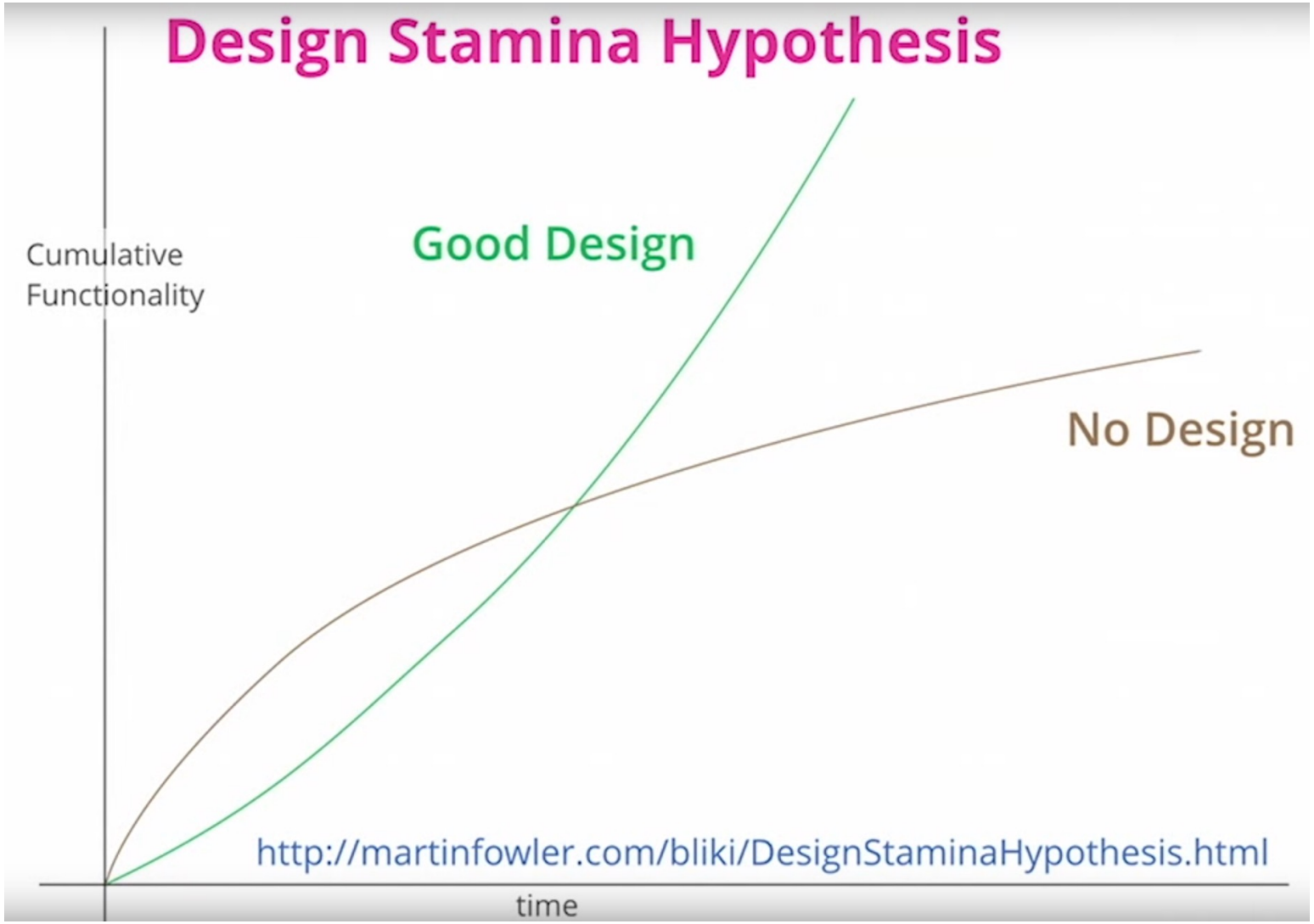
Source: me.me

Why to architect?

Design Stamina Hypothesis



Design Stamina Hypothesis



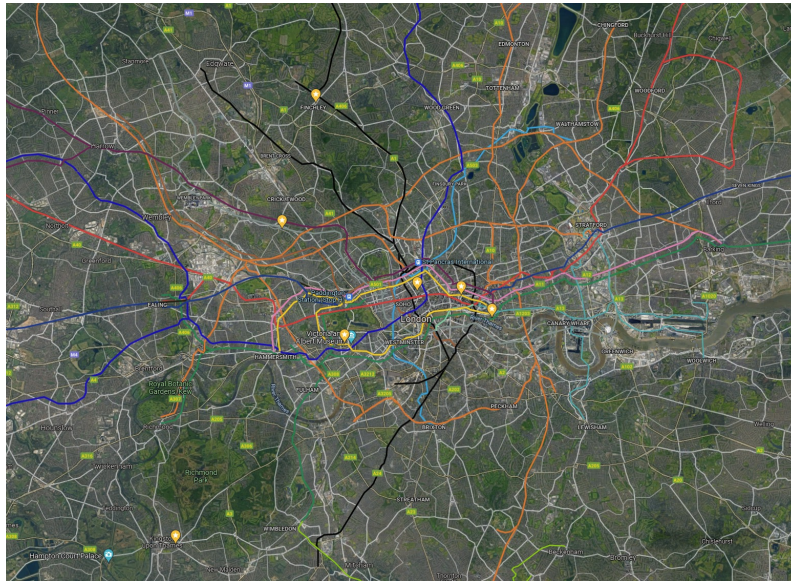
How much to architect?

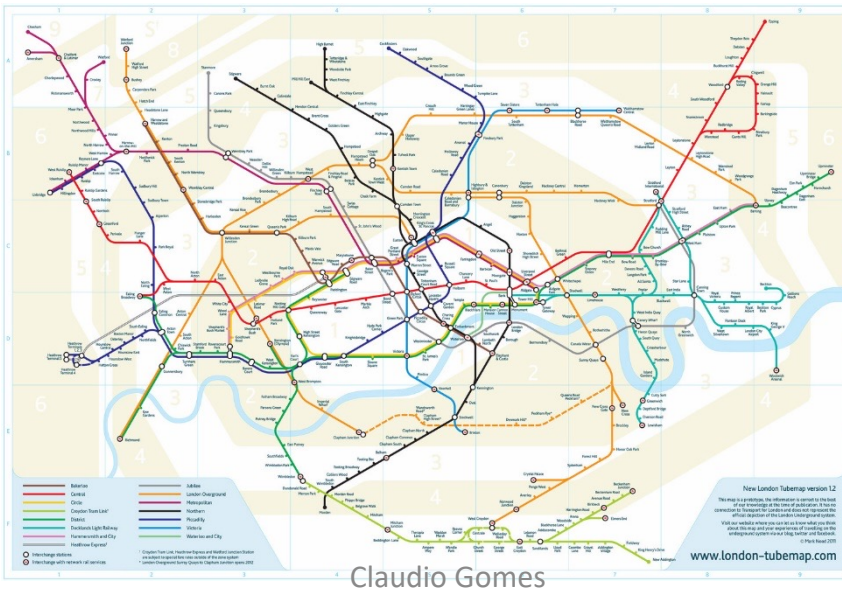
Models

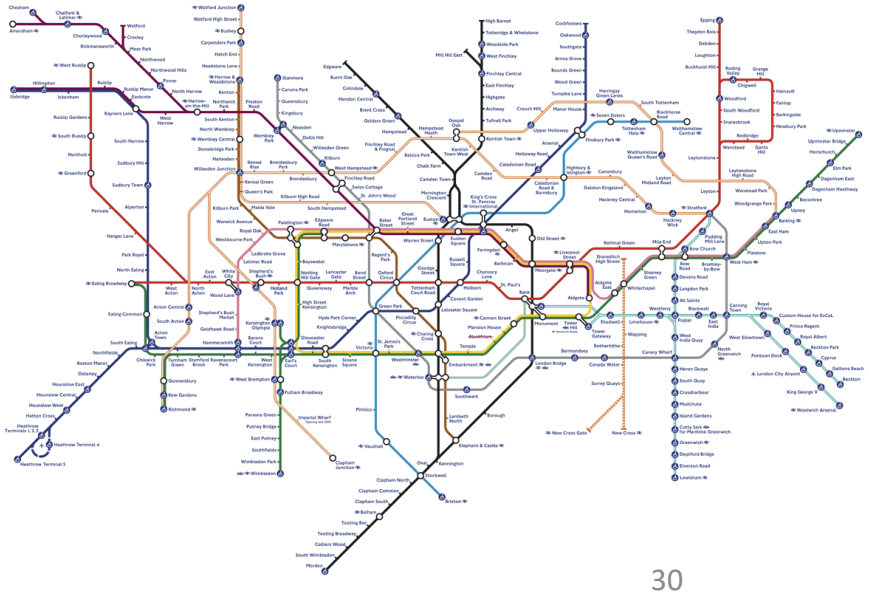
"Everything should be made as simple as possible, but not simpler.",
Albert Einstein, Louis Zukofsky, Roger Sessions, William of Ockham

- An architecture is a set of models, and a model has three characteristics:
  - Must be based on the system
  - Must reflect a relevant subset of the system's properties
  - Can be used to think about the system, in a limited context.

Figure out what the goal of the architecture is.



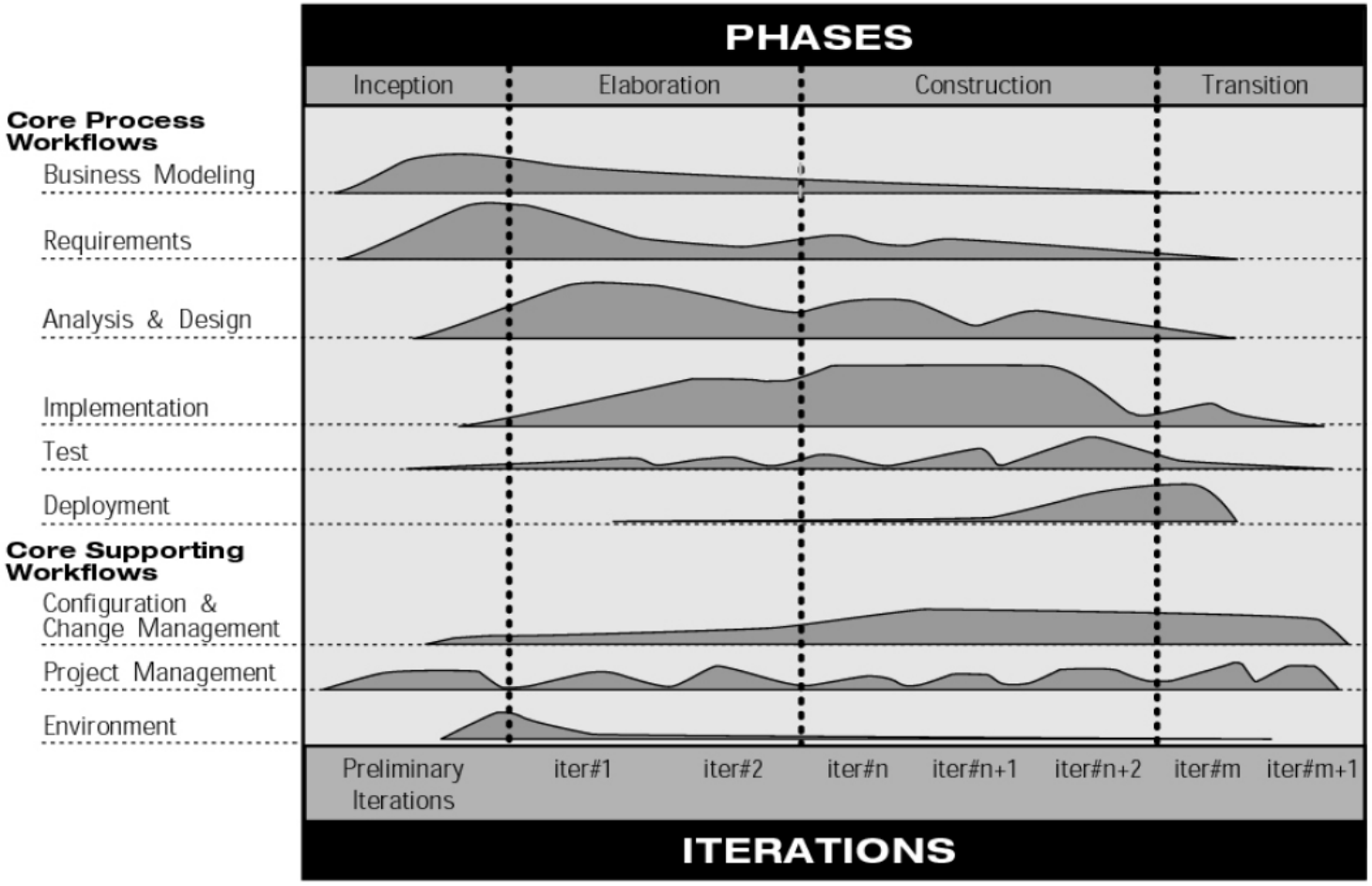




Source: google maps

How to architect?

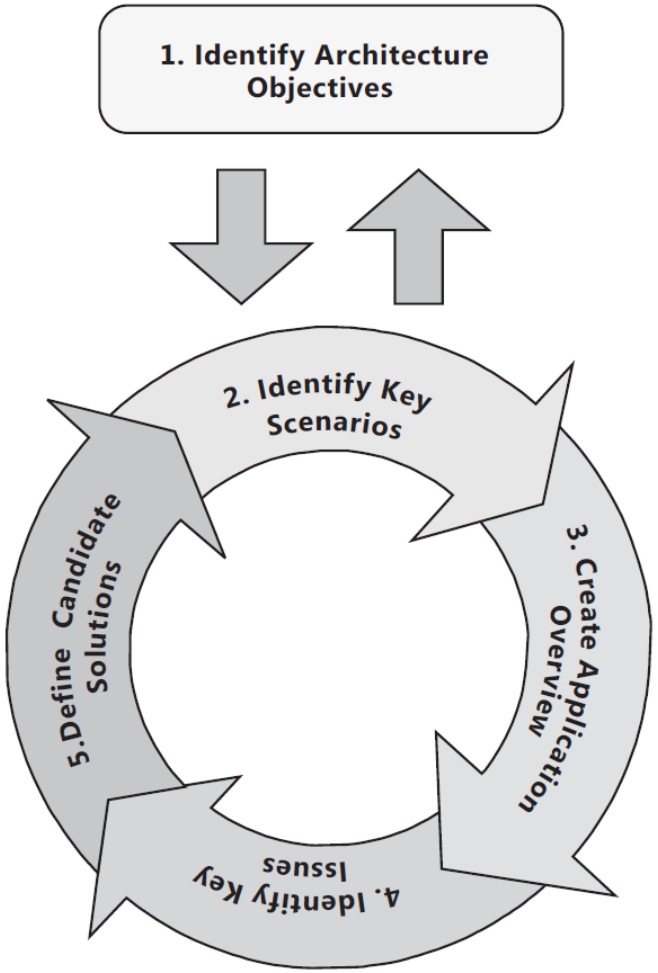
Rational Unified Process (RUP)



https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

The nine core process workflows

An architecture process (from Microsoft)



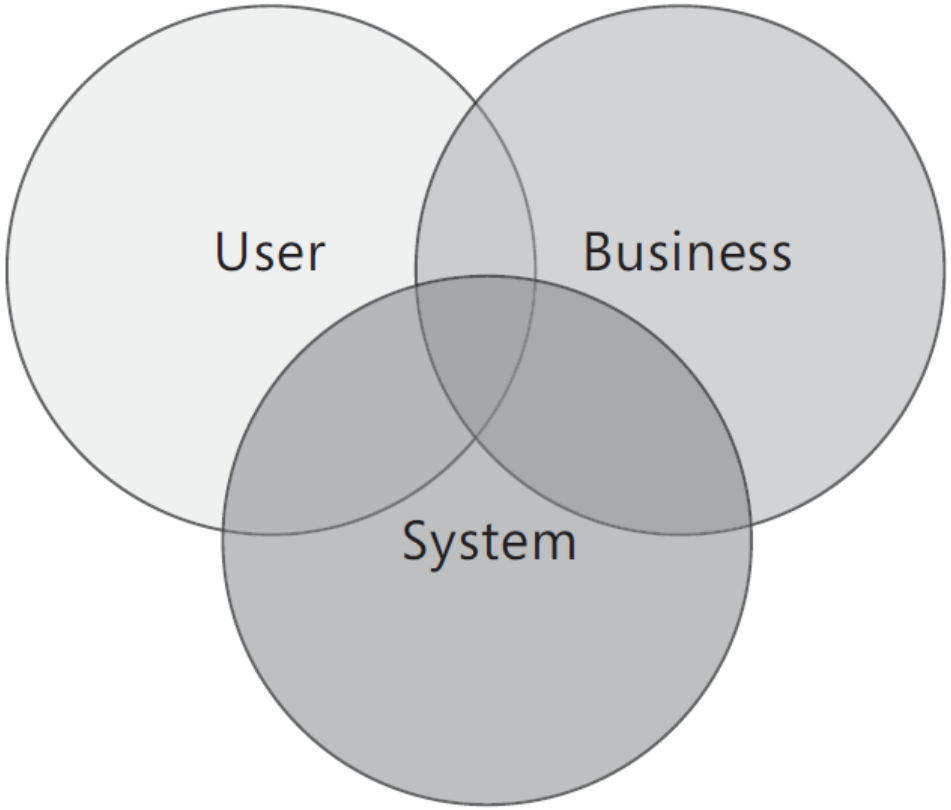
Iterative and incremental.

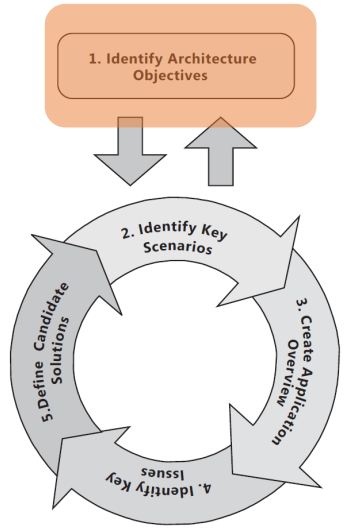
Test against:

- requirements
- known constraints
- quality attributes

Source: Microsoft Application Architecture Guide, 2nd Edition

1. Architecture objectives - input





Goals and constraints shape your architecture and design process.

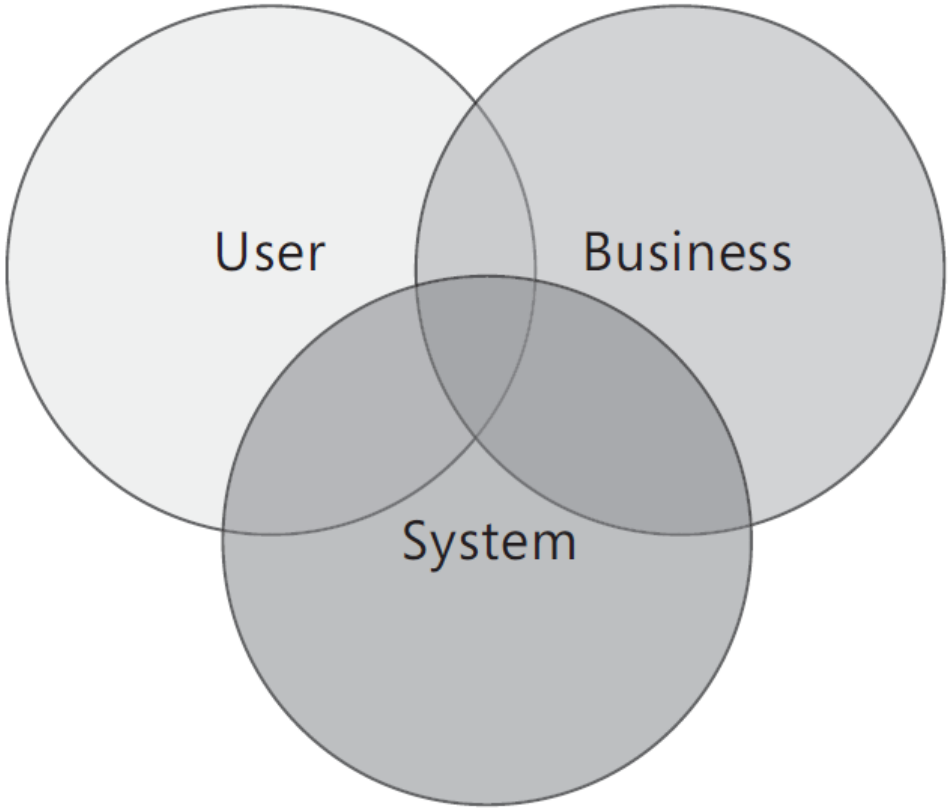
The architecture must satisfy:

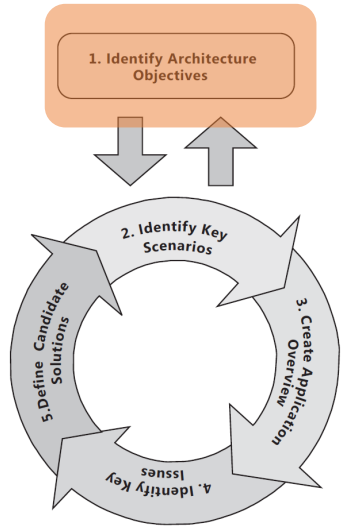
- Functional requirements
- Non-functional requirements
  - External requirements (e.g. standards)
  - Organisational requirements
  - Product requirements (quality attributes)

Cross cutting concerns!

Source: Microsoft Application Architecture Guide, 2nd Edition

1. Architecture objectives – output





Who will use the output of this iteration?

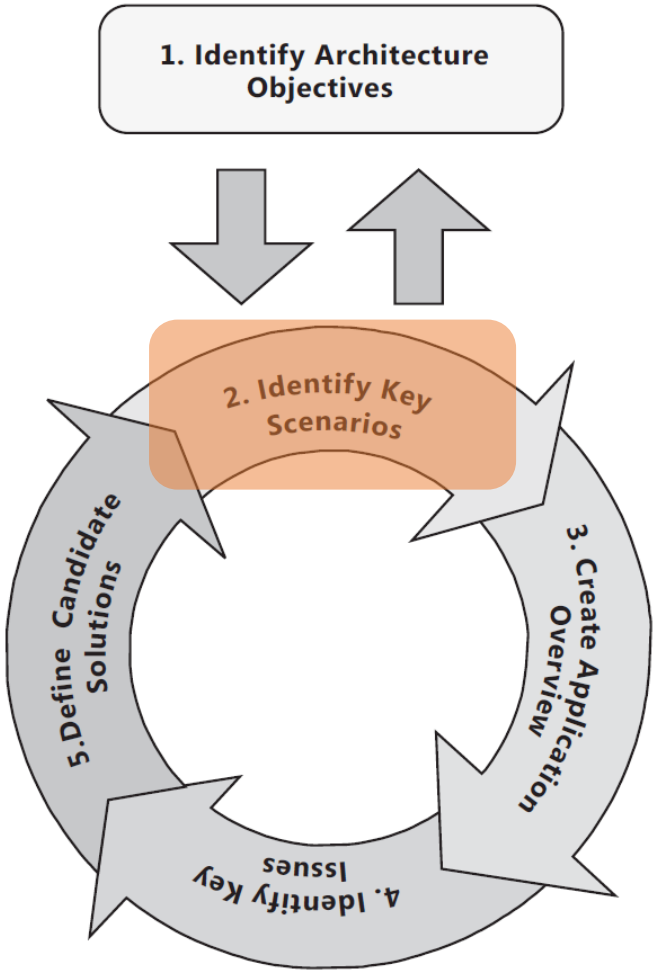
- Management?
- Testers?
- Developers?
- Other architects?

Are you:

- Creating a complete application design?
- Building a prototype?
- Examining technical risks?
- Testing potential options?
- Building shared models to gain an understanding of the system?

Source: Microsoft Application Architecture Guide, 2nd Edition

2. Key scenarios



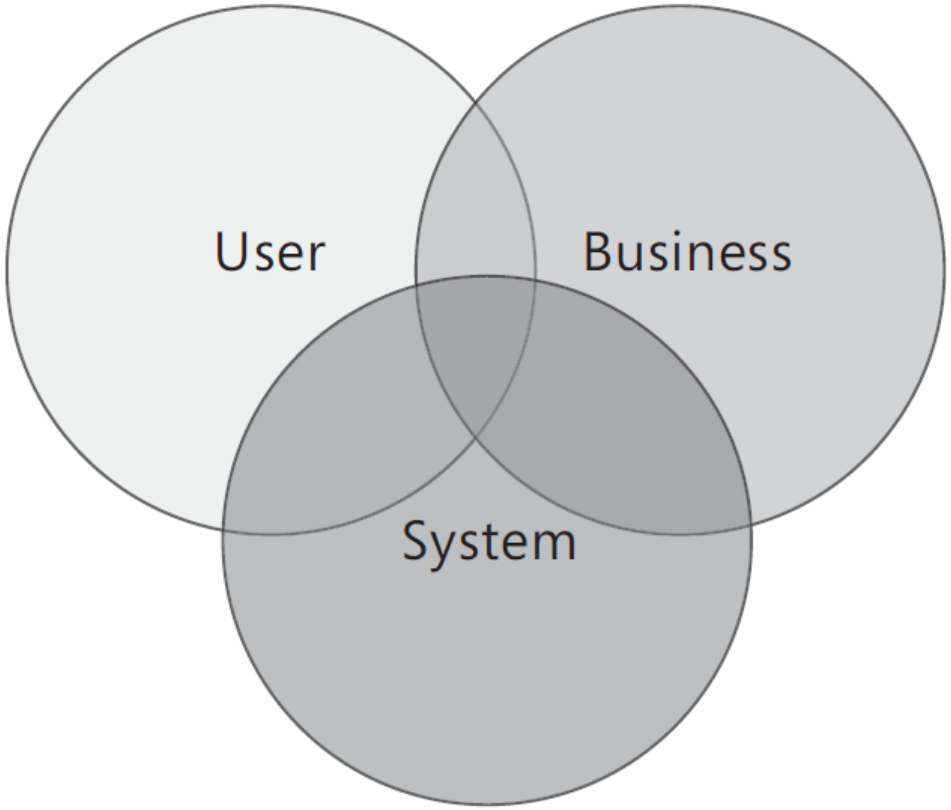
What are Key Scenarios?

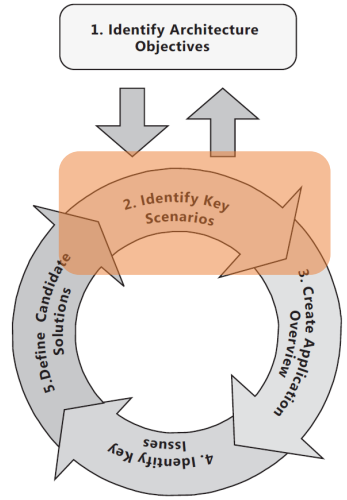
This is where use cases and quality attributes meet!

To identify them takes practice, but here are some pointers:

Source: Microsoft Application Architecture Guide, 2nd Edition

2. Key scenarios





Critical functionality.

Critical non-functional reqs

Exploration (unknown areas)

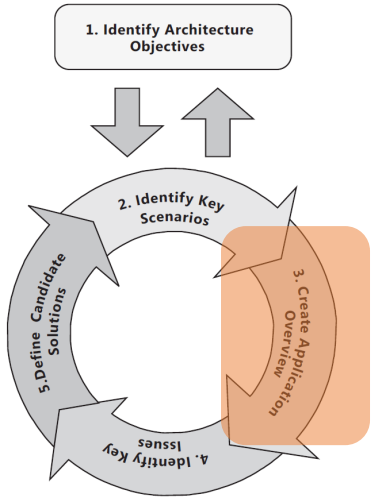
Risk mitigation

Look for intersections between the user, business and system views.

Prefer exercising multiple layers in the architecture when you select scenarios for the current iteration

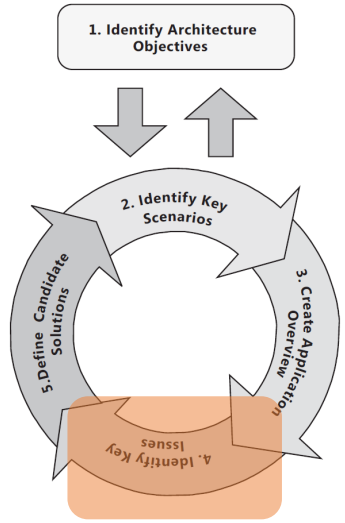
Source: Microsoft Application Architecture Guide, 2nd Edition

3. Create an application overview



- What is an application overview?
- It is one or more proposals for an architecture
- Determine your application type.
- Identify your deployment constraints.
- Determine relevant technologies.
- Identify important architecture design styles.
  - Use your toolbox of Architectural Patterns – more next lecture

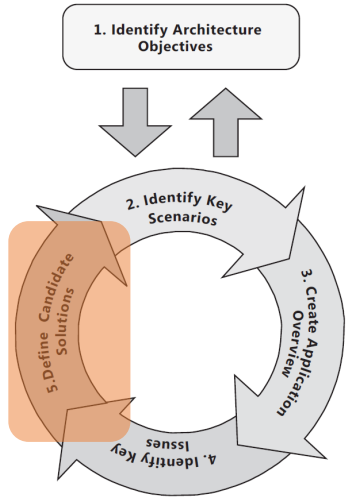
4. Identify Key Issues



- What are Key Issues?

- Problems that must be solved with this (version of the) architecture
- E.g. Key Scenarios, are they solved?
- Pose relevant hypothetical future changes:
  - "Can I swap from one third party service to another?,"
  - "Can I add support for a new client type?,"
  - "Can I quickly change my business rules relating to billing?,"
  - "Can I migrate to a new technology for X?"
- Try it out, analyze and document outcomes

5. Define Candidate Solutions



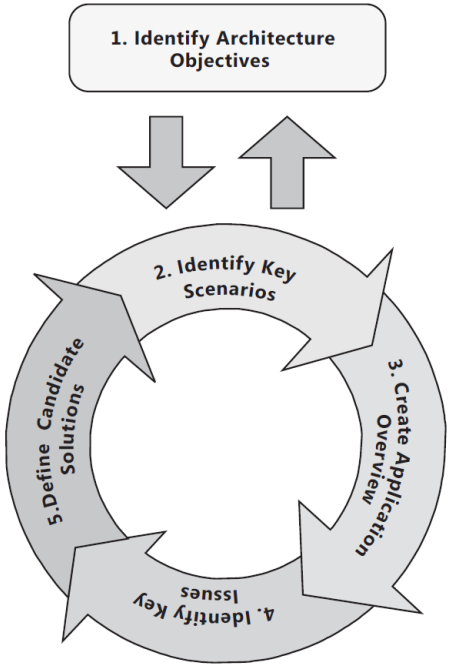
- Propose candidate solutions to key issues.
- Evaluate against "baseline" architecture:
  - Does this architecture succeed without introducing any new risks?
  - Does this architecture mitigate more known risks than the previous iteration?
  - Does this architecture meet additional requirements?
  - Does this architecture enable architecturally significant use cases?
  - Does this architecture address quality attribute concerns?
  - Does this architecture address additional crosscutting concerns?
- May involve coding to validate assumptions (architectural spike)

Communicate architecture

- Next lecture.

Exercise

VideoFlix

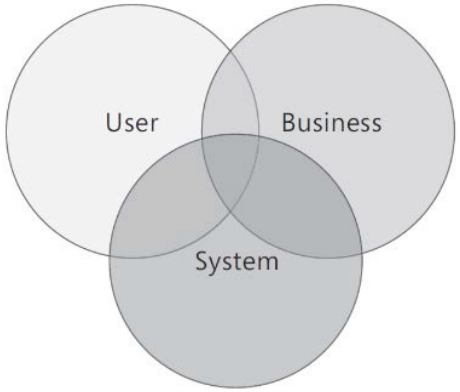


Exercise 1 and 2:

Identify key scenarios for a new video streaming service.

Remember that the objective of this iteration is to build a prototype.

Padlet for brainstorming:
https://aarhusuni.padlet.org/henrikbitschkirk/videoflix-key-scenarios-lwm96w35d84wjmau



Follow up on key scenarios

Homework

Exercise 3 and 4

Identify *and quantify* quality attributes

Homework

Quantifying the quality attributes means to make them measurable.

Take as example scalability:
”VideoFlix has to stream to a lot of viewers at the same time” <- You can’t measure that…
”VideoFlix has to stream the test video clip to 100.000 viewers in 1080p@30fps” <- This is measurable, even though it may be hard to execute the test.

Exercise 3 and 4

Identify *and quantify* quality attributes

AARHUS UNIVERSITY

References and image sources

Video(s):

Martin Fowler - Making architecture matter:
https://www.youtube.com/watch?v=DngAZyWMGR0

Simon Brown - The Frustrated Architect:

https://www.infoq.com/presentations/The-Frustrated-Architect

Images (s):

Kalaha: https://www.lekolar.dk/sortiment/leg/spil-og-puslespil/bord-og-bratspil/kalaha/