HSLU Hochschule Luzern

Software Architecture and Techniques

What is Agile Architecture?

Marcel Baumann, tangly llc

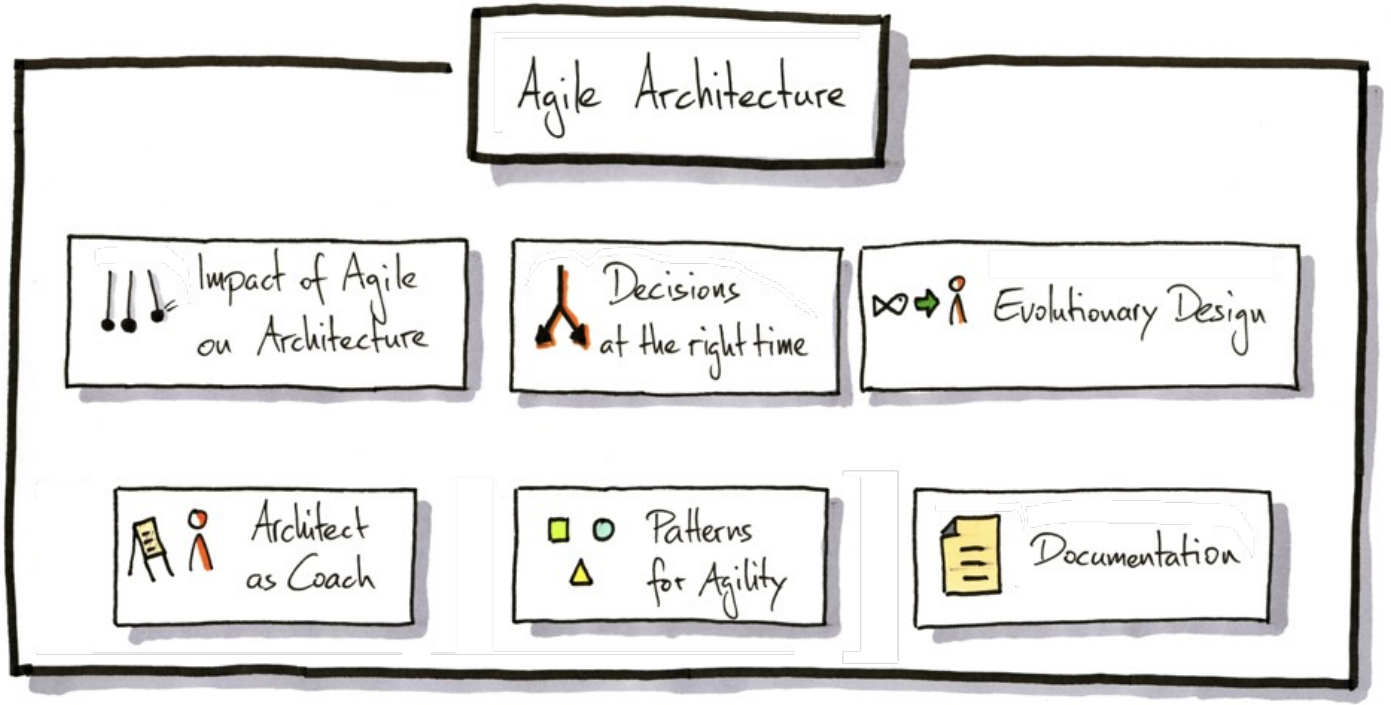
Observations

Every software developer is also a designer, every software developer is also an architect.

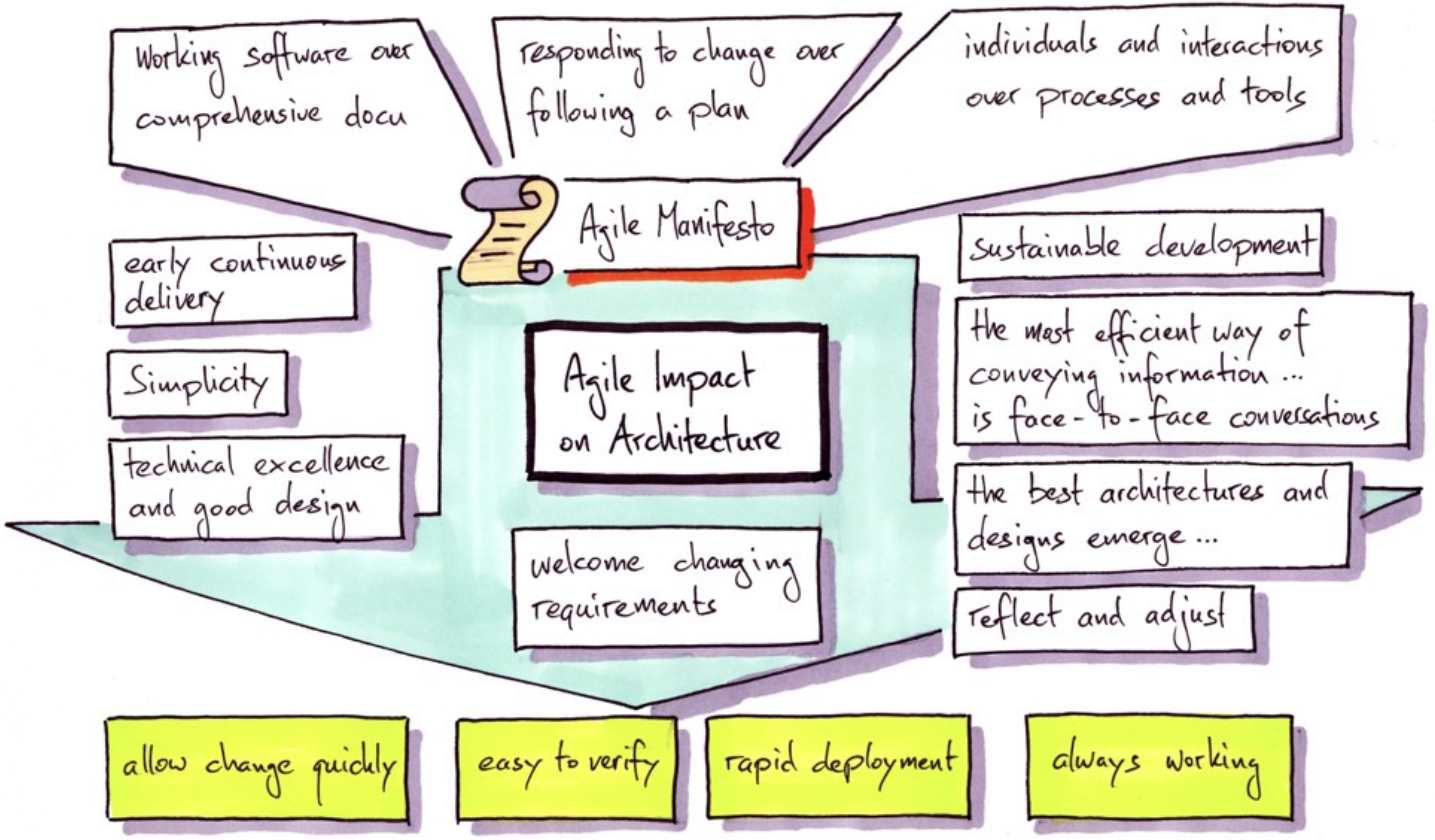
1) The sum of all the source code is the **true** design blueprint or software architecture.
2) The real software architecture **evolves** (better or worse) **every day** of the product, as people do programming.
3) The real living architecture needs to be grown every day through acts of programming by **master programmers**.
4) A software architect who is not in touch with the evolving source code of the product is **out of touch with reality**.
5) **Every programmer is some kind of architect** –whether wanted or not. Every act of programming is some kind of architectural act – good or bad, small or large, intended or not –.

Agile Architecture Principles (SAFe)

1) Design **emerges**. Architecture is a **collaboration**.
2) The bigger the system, the longer the runway.
3) Build the **simplest** architecture that can possibly work.
4) When in doubt, **code** or model it out.
5) They build it. They test it, They run it.
6) There is no monopoly on **innovation**.
7) Implement architectural flow.

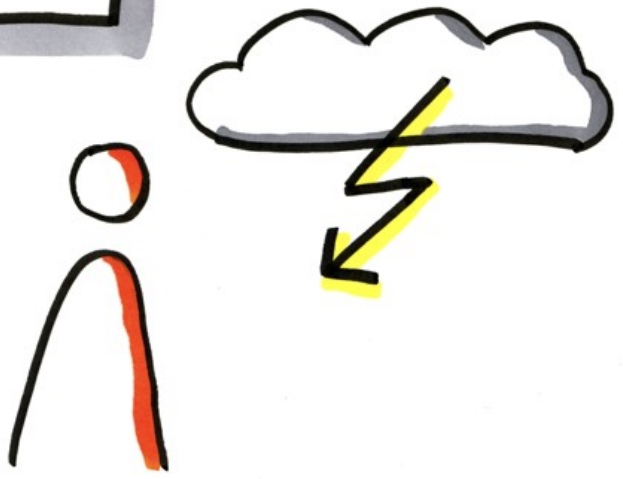


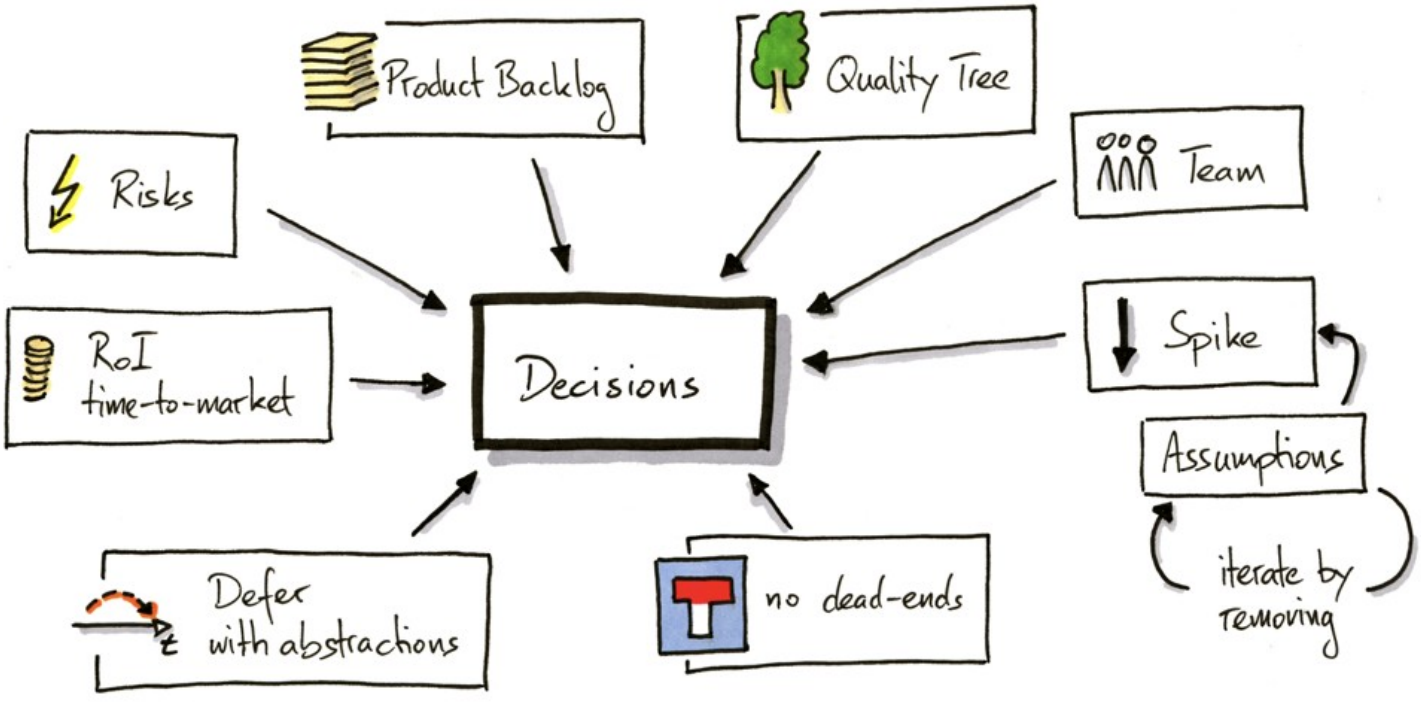
All drawings made by Urs Enzler

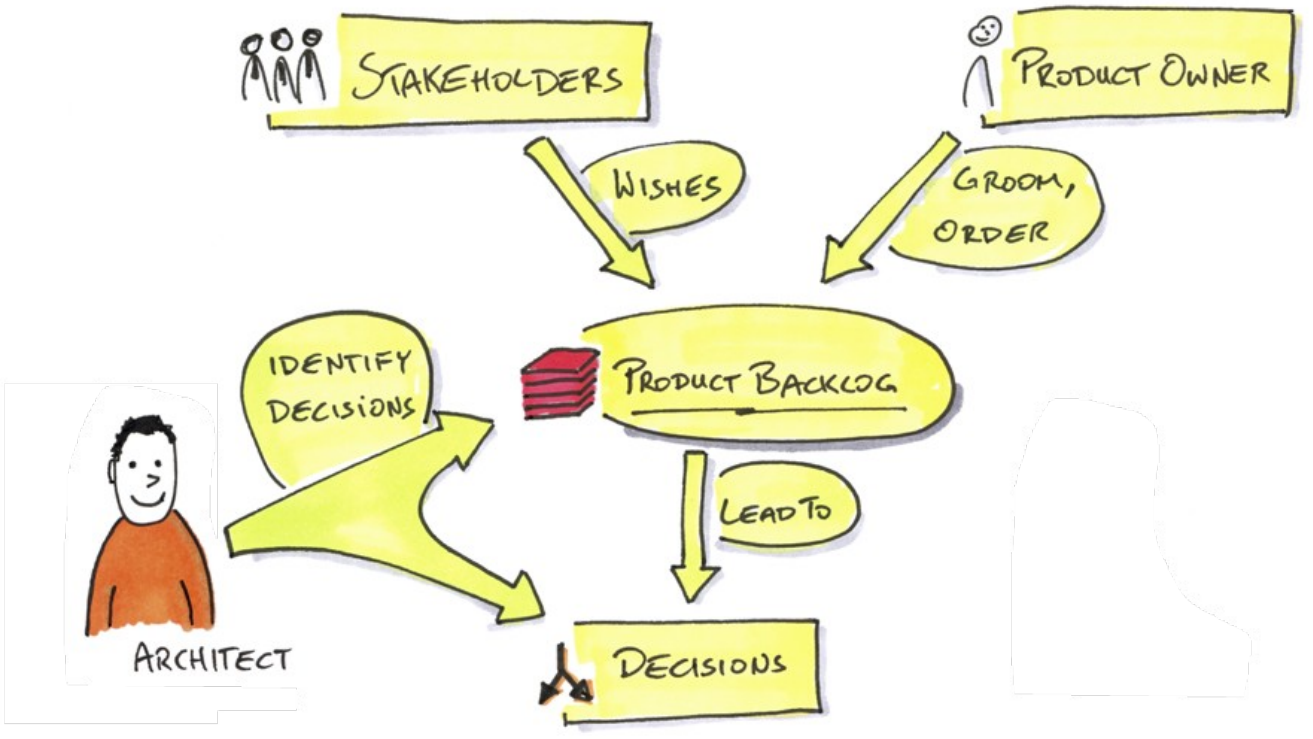

Working software over comprehensive docu
responding to change over following a plan
individuals and interactions over processes and tools
Agile Manifesto
early continuous delivery
Simplicity
technical excellence and good design
Agile Impact on Architecture
welcome changing requirements
Sustainable development
the most efficient way of conveying information ... is face-to-face conversations
the best architectures and designs emerge ...
reflect and adjust
allow change quickly
easy to verify
rapid deployment
always working

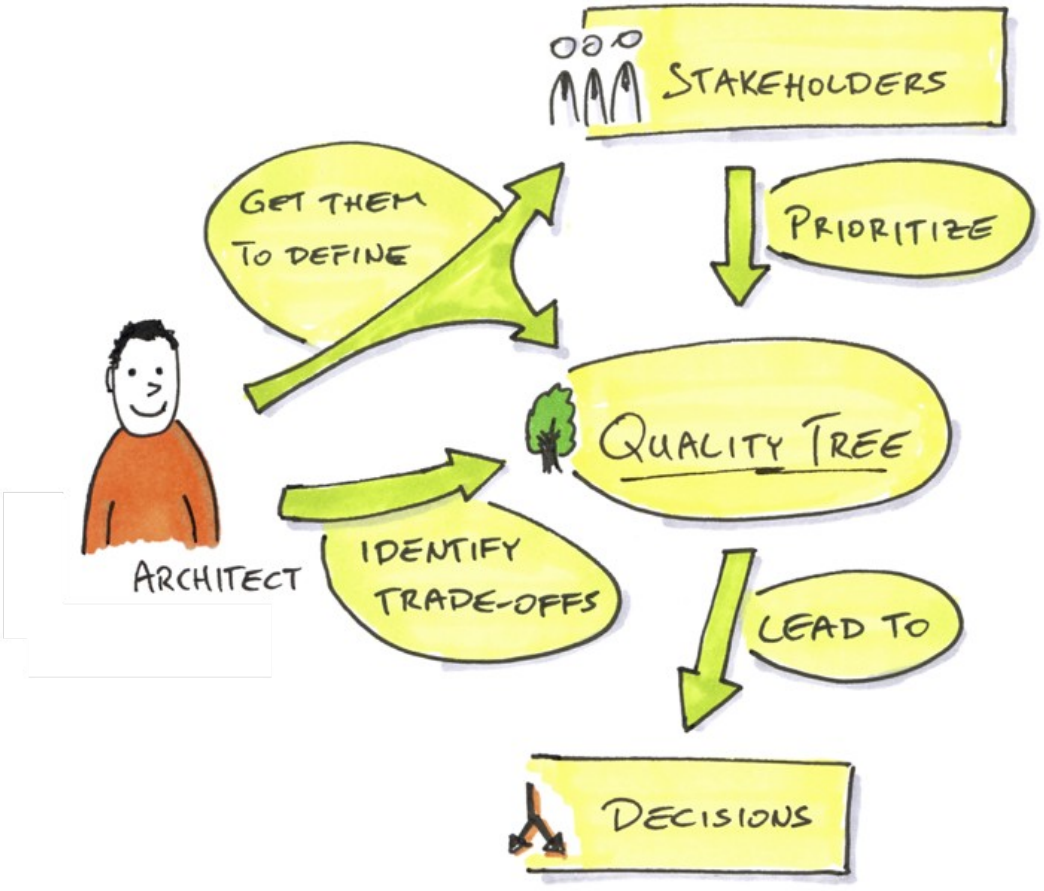
What are your pain points?

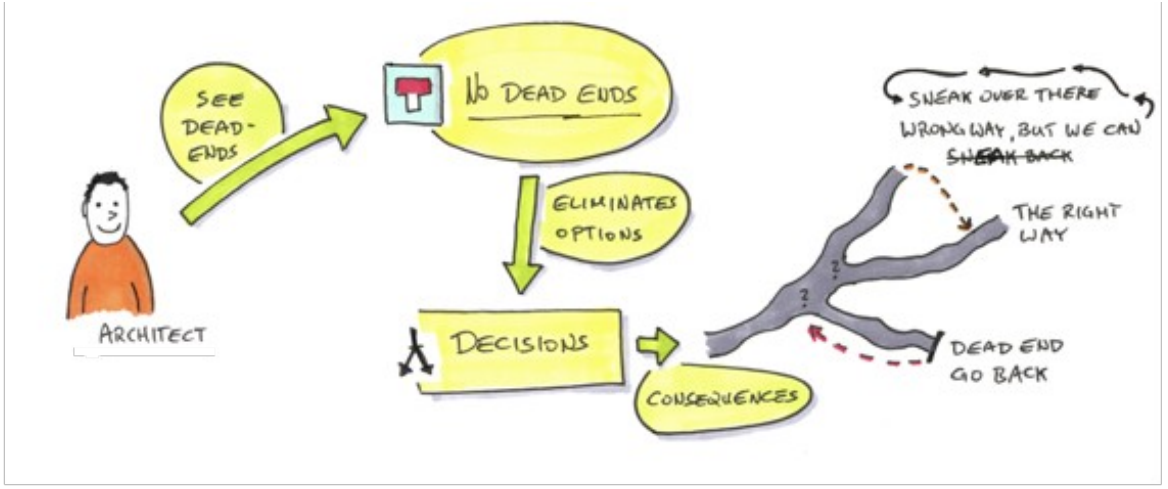
allow change quickly
easy to verify
rapid deployment
always working

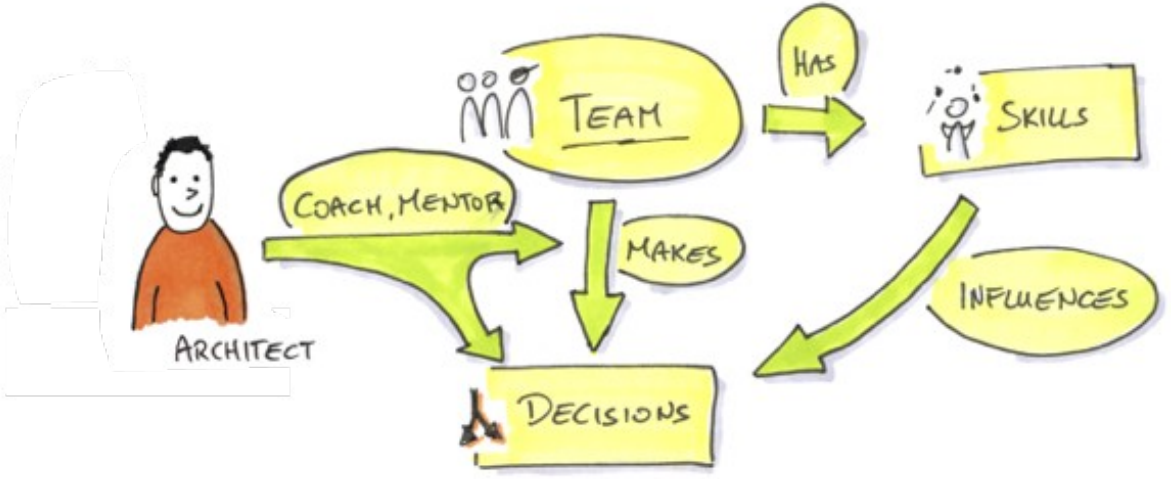


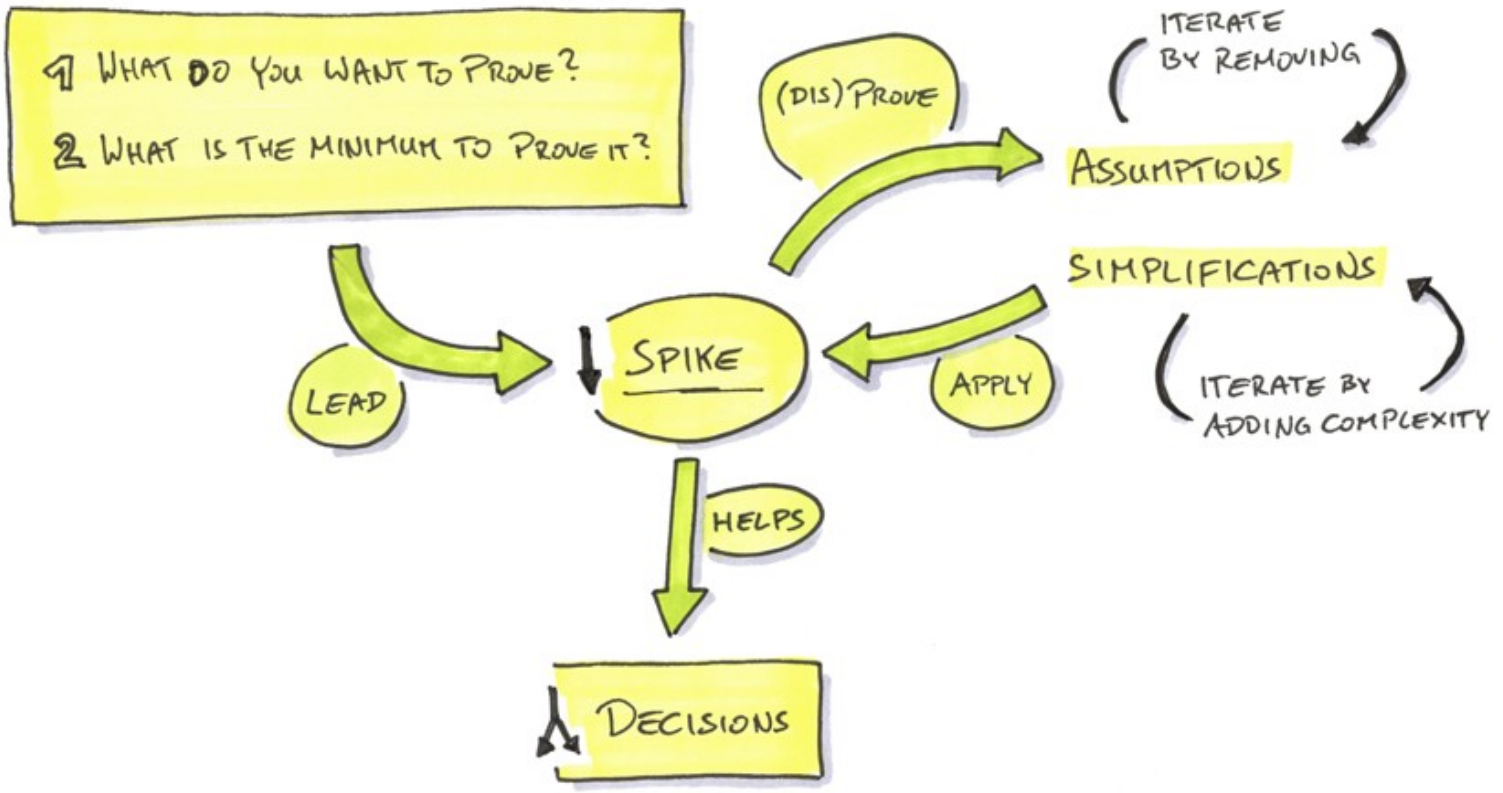

Product Backlog
Quality Tree
Risks
Team
RoI
time-to-market
Decisions
Spike
Assumptions
iterate by
removing
Defer
with abstractions
no dead-ends

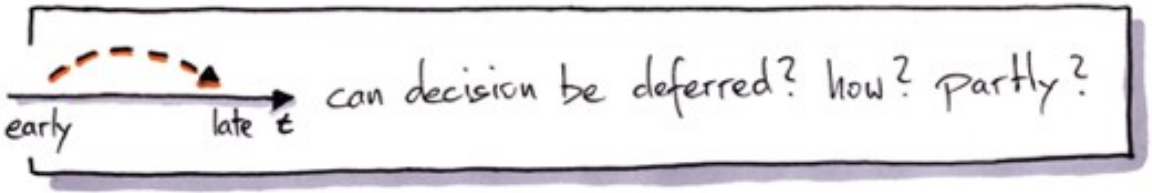

STAKEHOLDERS
PRODUCT OWNER
WISHES
GROOM, ORDER
IDENTIFY DECISIONS
PRODUCT BACKLOG
LEAD TO
ARCHITECT
DECISIONS


STAKEHOLDERS
GET THEM TO DEFINE
PRIORITIZE
QUALITY TREE
ARCHITECT
IDENTIFY TRADE-OFFS
LEAD TO
DECISIONS


SEE DEAD-ENDS
NO DEAD ENDS
ELIMINATES OPTIONS
ARCHITECT
DECISIONS
CONSEQUENCES
SNEAK OVER THERE
WRONG WAY, BUT WE CAN SNEAK BACK
THE RIGHT WAY
DEAD END GO BACK

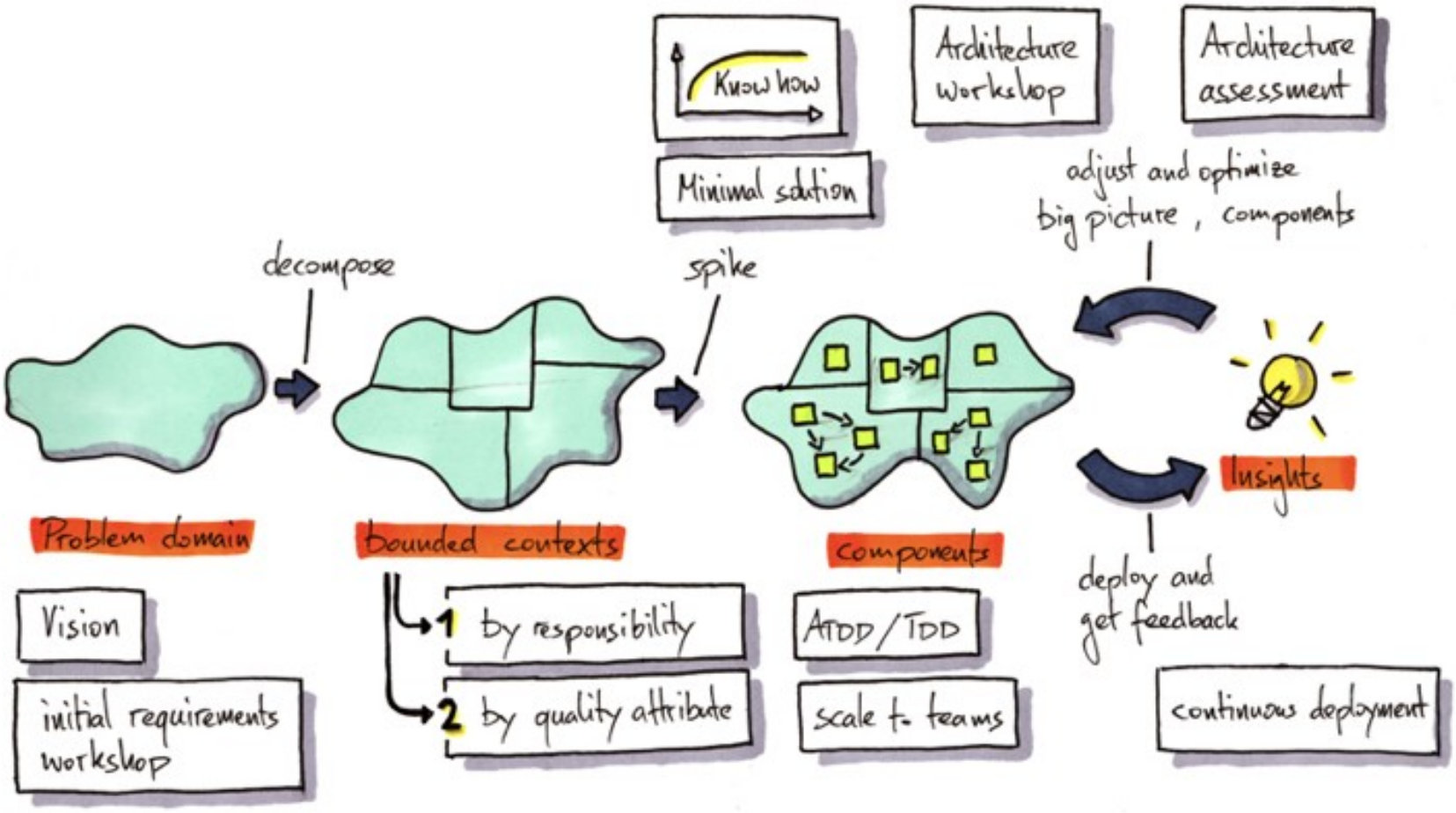

TEAM
HAS
SKILLS
COACH, MENTOR
MAKES
INFLUENCES
ARCHITECT
DECISIONS


1 WHAT DO YOU WANT TO PROVE?
2 WHAT IS THE MINIMUM TO PROVE IT?
(DIS) PROVE
ITERATE BY REMOVING
ASSUMPTIONS
SIMPLIFICATIONS
ITERATE BY ADDING COMPLEXITY
SPIKE
LEAD
APPLY
HELPS
DECISIONS

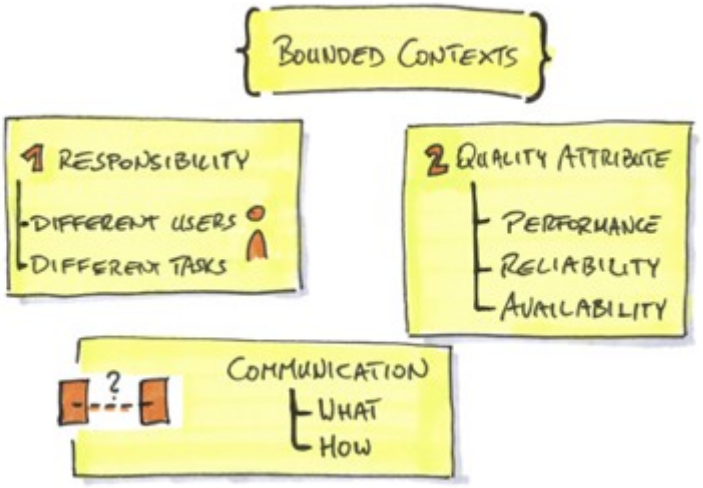


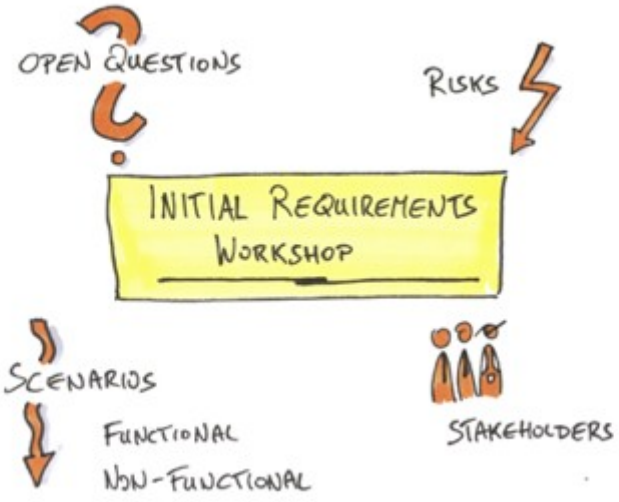
- persist data of your system to survive restart
- how to translate UI and data
- communication between parts of your system
- scaling (run on multiple threads, processes, machines)
- security (how to authenticate, authorize)
- journaling (Activities, data)
- reporting
- data migration / data import
- releasability
- backwards compatibility
- response times
- Archiving data

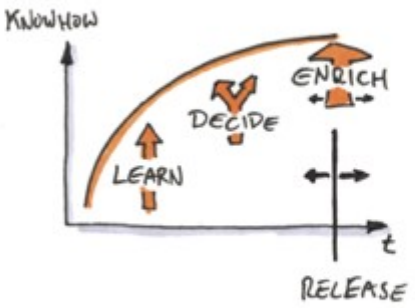
design to be independent on decision

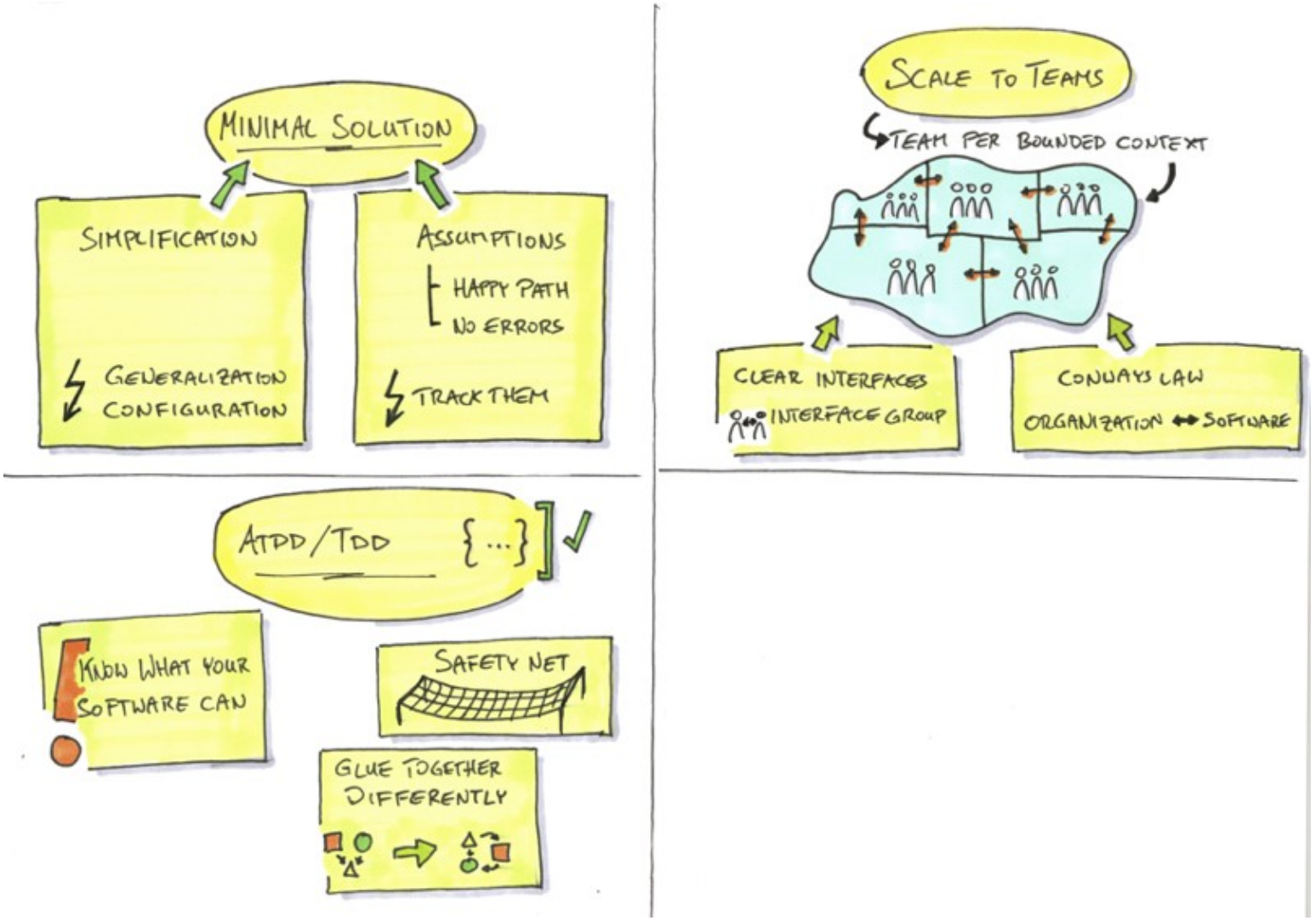

Know how
Minimal solution
Architecture workshop
Architecture assessment
adjust and optimize
big picture , components
decompose
spike
Insights
Problem domain
bounded contexts
components
deploy and
get feedback
Vision
initial requirements workshop
1 by responsibility
2 by quality attribute
ATDD / TDD
scale to teams
continuous deployment

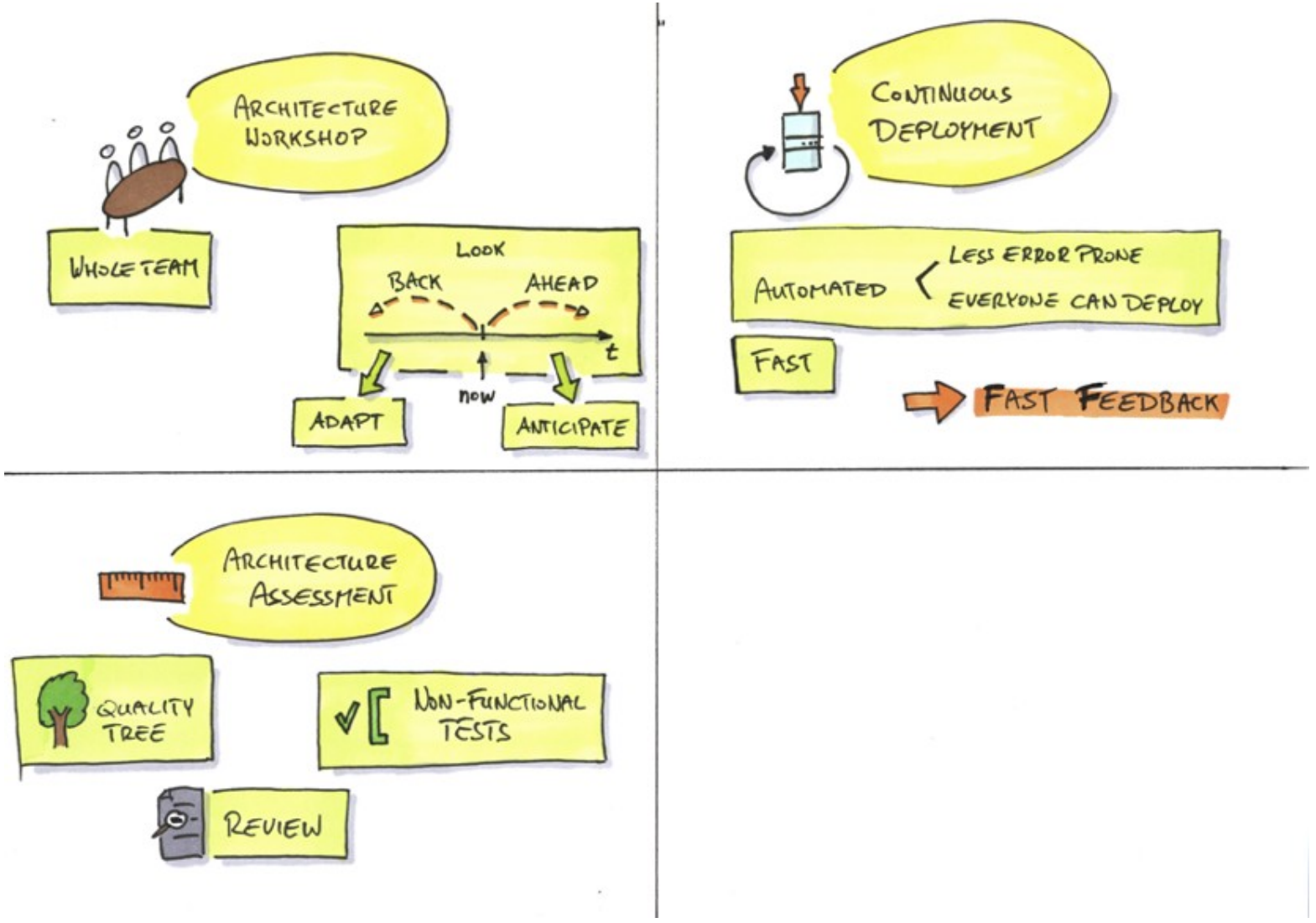

VISION
DEAD-ENDS
DECISIONS


BOUNDED CONTEXTS
1 RESPONSIBILITY
DIFFERENT USERS
DIFFERENT TASKS
2 QUALITY ATTRIBUTE
PERFORMANCE
RELIABILITY
AVAILABILITY
?
COMMUNICATION
WHAT
HOW

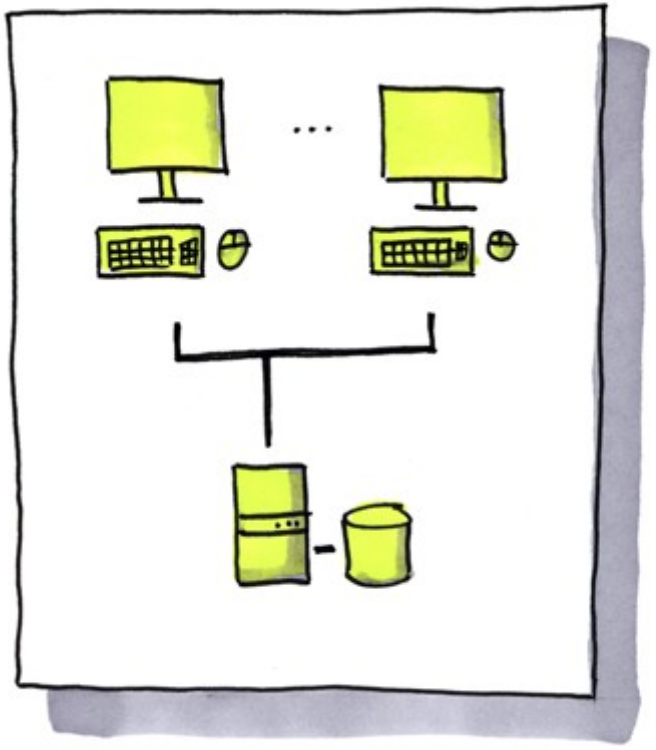

OPEN QUESTIONS
RISKS
INITIAL REQUIREMENTS WORKSHOP
SCENARIOS
FUNCTIONAL
NON-FUNCTIONAL
STAKEHOLDERS


KNOWHOW
ENRICH
DECIDE
LEARN
t
RELEASE


MINIMAL SOLUTION
SIMPLIFICATION
GENERALIZATION
CONFIGURATION
ASSUMPTIONS
HAPPY PATH
NO ERRORS
TRACK THEM
SCALE TO TEAMS
TEAM PER BOUNDED CONTEXT
CLEAR INTERFACES
INTERFACE GROUP
CONWAYS LAW
ORGANIZATION ↔ SOFTWARE
ATDD/TDD
{...}
KNOW WHAT YOUR SOFTWARE CAN
SAFETY NET
GLUE TOGETHER DIFFERENTLY


ARCHITECTURE WORKSHOP
WHOLE TEAM
LOOK
BACK
AHEAD
t
now
ADAPT
ANTICIPATE
CONTINUOUS DEPLOYMENT
AUTOMATED
LESS ERROR PRONE
EVERYONE CAN DEPLOY
FAST
FAST FEEDBACK
ARCHITECTURE ASSESSMENT
QUALITY TREE
NON-FUNCTIONAL TESTS
REVIEW

define evolution steps



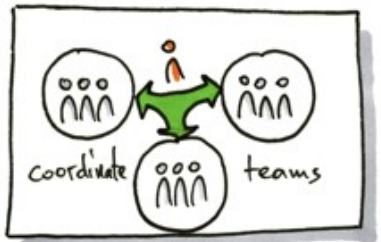
Sirius Cybernetics

Druid Resources Department (DRD)

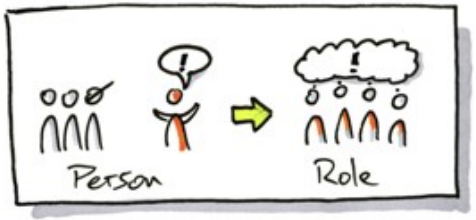
Step I manage data of all druids

- assembly date
- retirement date
- serial number
- jobs
  - work place
  - from / to date
  - customer feedbacks

Step II journaling



Architect



lead technically

technology evangelist
engineering practises
technical spikes
non-functional specs
write code

understand stakeholders

big picture
talk to all stakeholders
learn about all view points
understand the user
help the PO

coach the team

architecture is team work
coordinate
pair program
educate
support SM

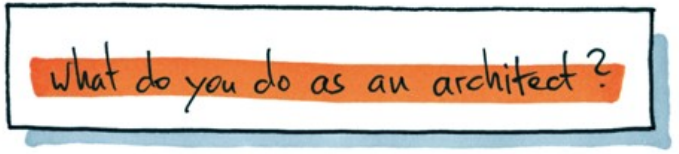
communicate

use effective tools

be part of the team

learn to be a good designer

make sure decisions are made



compare with Agile Architect checklist

Agile Architecture Patterns

Modularity

- exchangeability
  - clear interfaces
  - clear responsibilities
- fits in my head
- risk and cost of local change is low
- duplication ⟷ modularity vs. integration
- coupling < space / time
- split brain

Clean Architecture

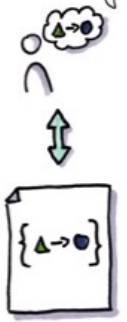
- direction of dependencies
  - no tangles
  - concrete → abstract
- decouple from environment
- testable
- scenario driven
- inversion of control
- passing data over boundaries

Deployability

- fast
- frequent
- versioning
- agnostic of target platform
- one branch for all customers

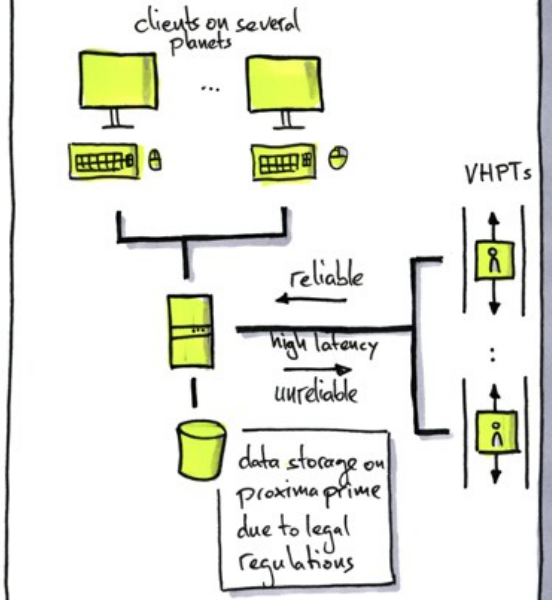
Software reflects user's mental model

- ubiquitous language



1) identify bounded contexts

2) define evolutionary architecture steps



Maintenance
- see activities of VHPTs
  - filters:
    - VHPT name
    - period
    - kind of event
- run diagnostics program on VHPT
- receive warning when VHPT out-of-order

VHPT
- sends data in "real" time
  - door opened / closed
  - movement
  - out-of-order notification
  - mood of person

Sales
- report about mood of passengers

Accounting
- see out-of-order data
- set a random VHPT to out-of-order

manual
specifications
auto-documentation
Documentation
Questions: (by Michael Nygard)
• who is the consumer?
• what do they need?
• how do you deliver it to them?
• how do you know when they are ready for it?
• how do you produce it?
• what input do you need to produce it?
t
only document what you did, not what you want to do
shared
Whole team participates

Links

- Blog *Agile Architecture with Scrum*

Exercises

- Read cheat sheet “Agile Architecture”
- Apply the learnt principles to your actual product
  - Improve one Java class following clean code
  - Build it through your CI/CD pipeline