HSLU Hochschule Luzern

Software Architecture and Techniques

Agile Approaches
Scrum, *eXtreme Programming*, *LeSS*

Marcel Baumann, tangly llc

Architect Hats

- Architect is a **role** in the agile world – *and not a position or a title*
- Domain expert
- Technology expert
- Stakeholder facilitator
- Coach, mentor, teacher

Agile Architecture Principles

- Simple design
- **Emergent architecture**
- Runaway architecture work
- Hexagon approach
- **Relentless focused refactoring**

The quality of the architecture is proportional to the surface of the whiteboard

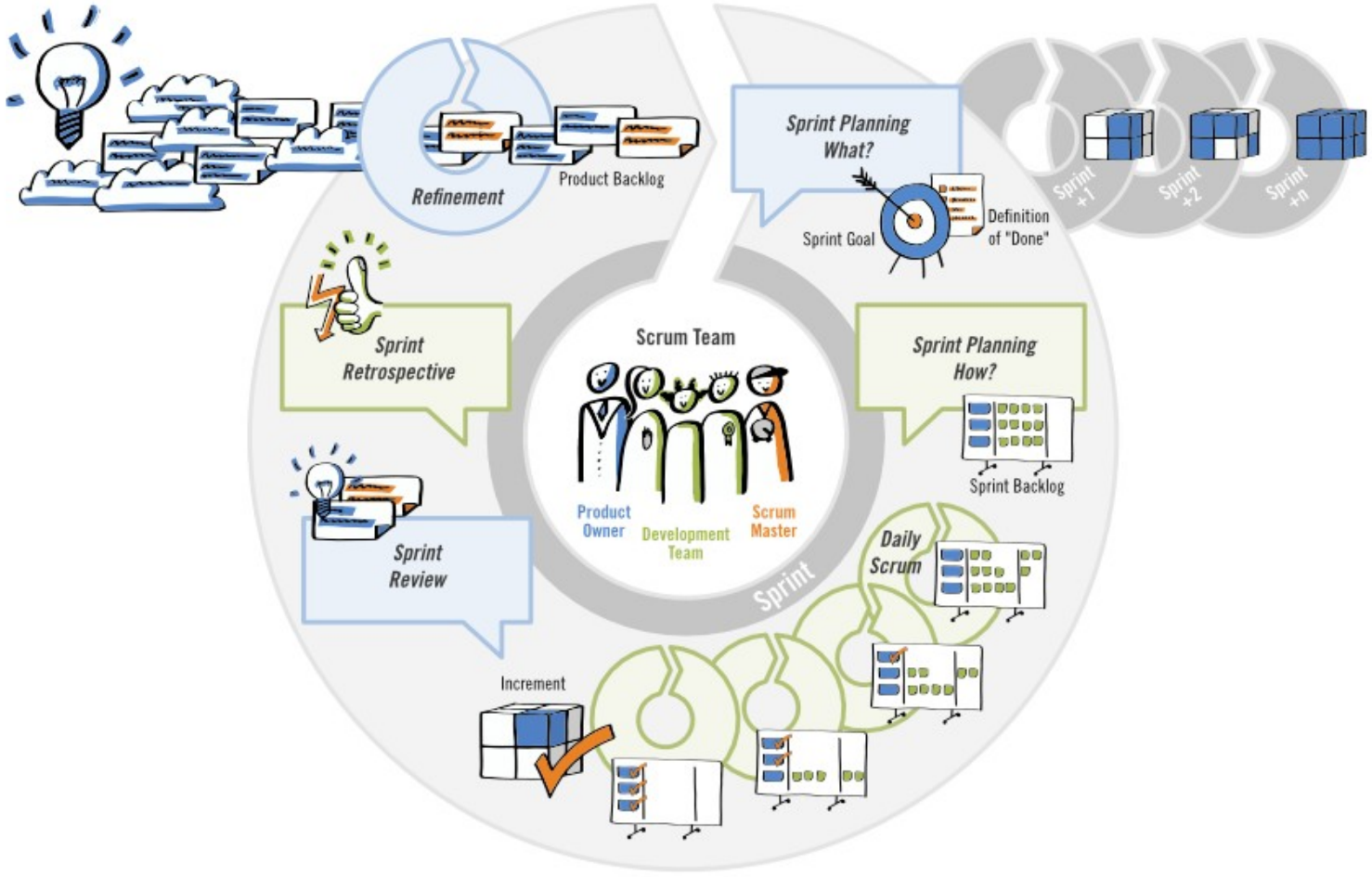
Agile Architecture Techniques

- Continuous integration, delivery and deployment
- Agile implies automation
- Git impact → golden trunk, distributed repository, GitOps
- Potentially shippable product → feature toggle, no *Undone* work
- Clean architecture → clean code → clean coder

Scrum Practices

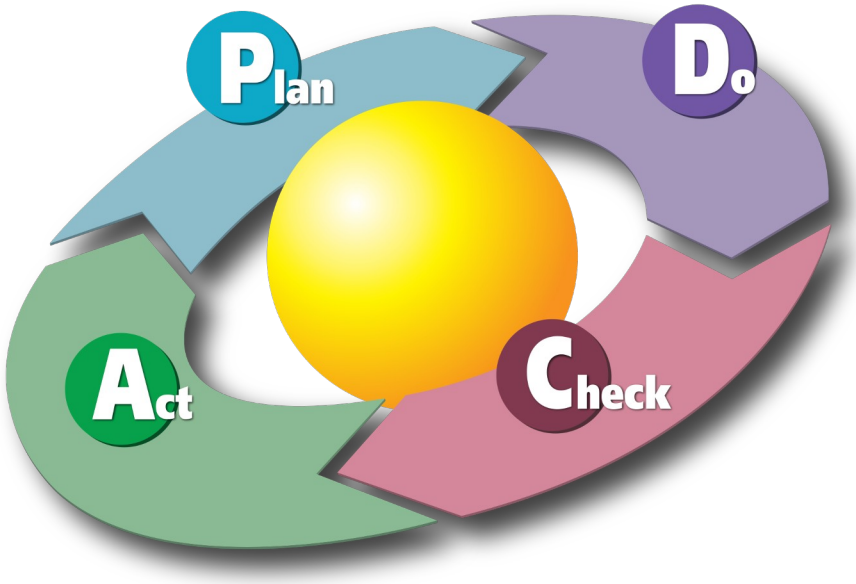
- Scrum does not prescribe any technical practices (see flaccid Scrum)
- Scrum emphasizes vision, context, roadmap
- It should be all about value → *outcome over output*
- Scrum encourages applying eXtreme Programming techniques
- Scrum alliance is working together with LeSS

Scrum


Refinement
Product Backlog
Sprint Planning
What?
Sprint Goal
Definition
of "Done"
Sprint
+1
Sprint
+2
Sprint
+n
Sprint
Retrospective
Scrum Team
Sprint Planning
How?
Sprint Backlog
Product
Owner
Development
Team
Scrum
Master
Sprint
Review
Daily
Scrum
Increment
Sprint

Scrum Approaches

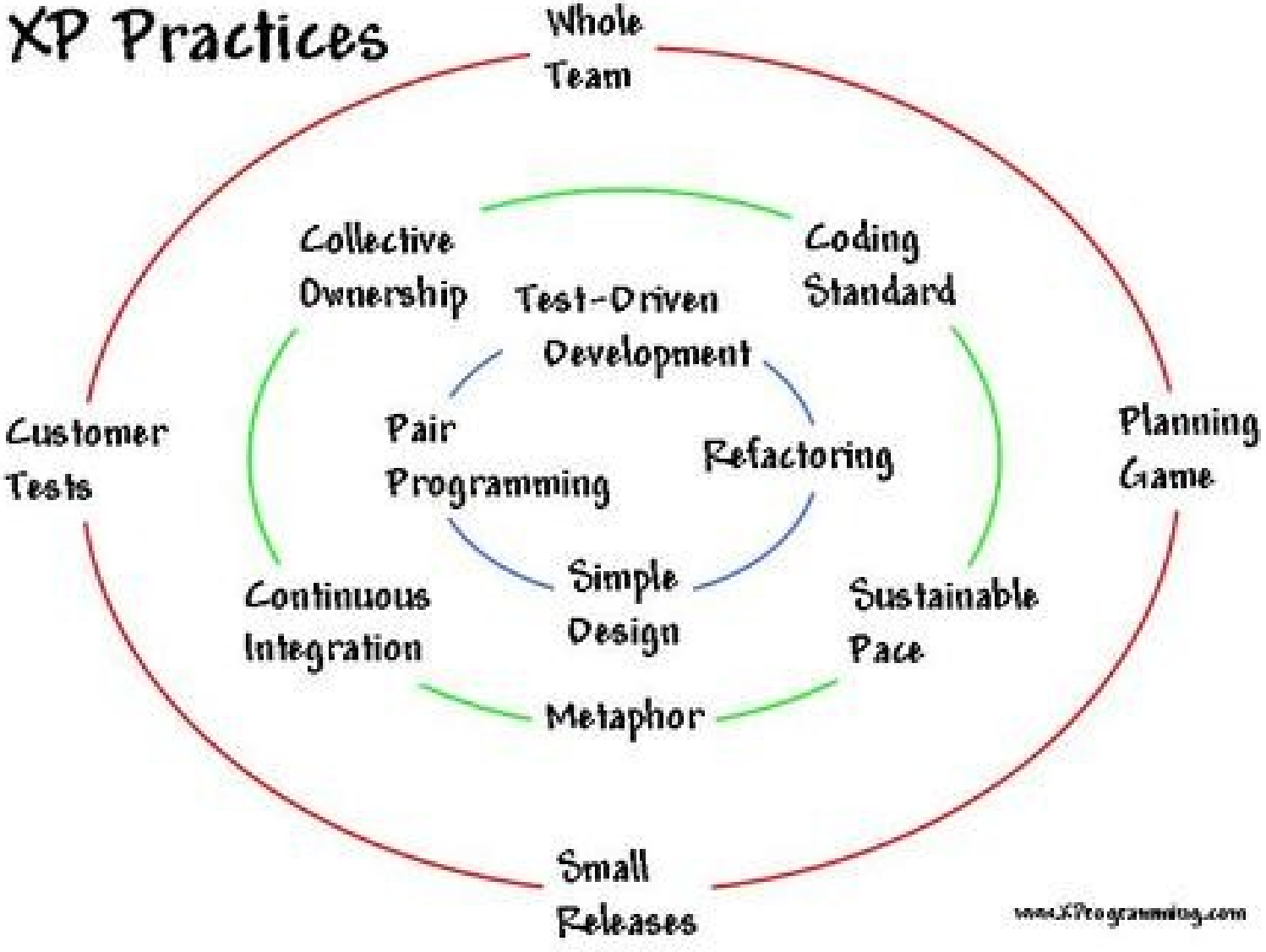
- Scrum emphasizes **learning in the team**
- Scrum builds on continuous improvement
  - Retrospective
  - Review
  - Daily meeting
  - *Always*



eXtreme Programming Practices

- Pair Programming
- Test Driven Development
- CI
- Refactoring
- Coding Standards
- Collective Code Ownership
- Simple Design
- System Metaphor

XP requires Software Craftsmanship


XP Practices
Whole Team
Collective Ownership
Coding Standard
Test-Driven Development
Customer Tests
Pair Programming
Refactoring
Planning Game
Continuous Integration
Simple Design
Sustainable Pace
Metaphor
Small Releases
www.XProgramming.com

XP Approaches

- XP emphasizes **individual** learning
  - Pair programming
  - Coding guidelines
  - Collective ownership of source code and artifacts

Build projects around **motivated** individuals.
Give them the **environment** and **support** they need, and **trust** them to get the job done.

Craftsmanship Approach

- Architect is a **domain expert**
- Architect is a **software craftsman**
- Architect is a **lean leader** – teacher, coach, mentor
- Architect talks with stakeholders and C-level representatives

QUALITY CODING

Manifesto for Software Craftsmanship

raising the bar

As aspiring Software Craftsmen we are raising the bar of professional software development by practising it and helping others learn the craft. Through this work we have come to value:

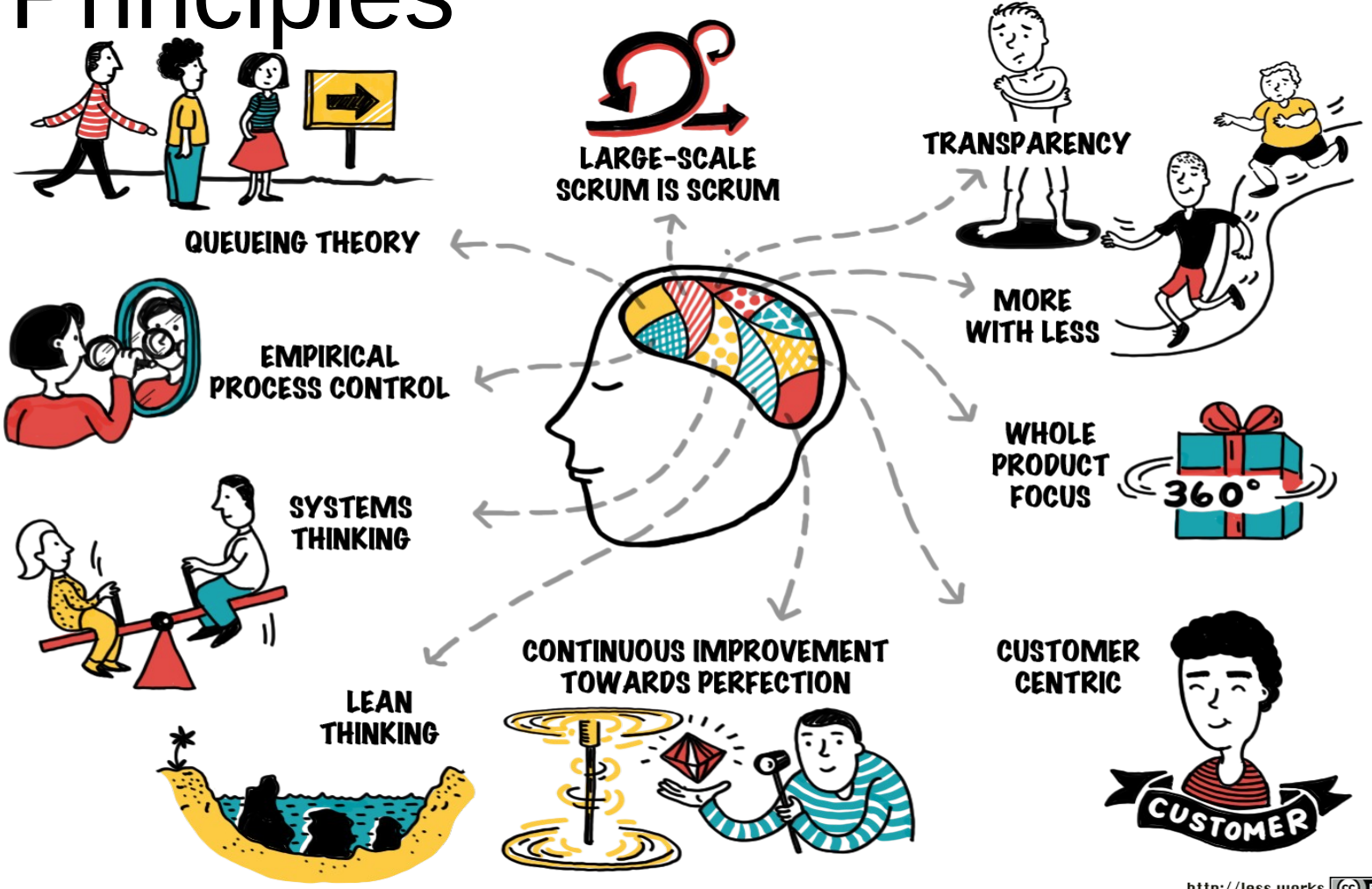
Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

Not only customer collaboration,
but also **productive partnerships**

LeSS Principles



LeSS Practices

Architecture and Design

Technical Excellence

Continuous Integration

SPECIFICATION BY EXAMPLE

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

TEST AUTOMATION

TECHNICAL EXCELLENCE

ARCHITECTURE & DESIGN

ACCEPTANCE TESTING

CLEAN CODE

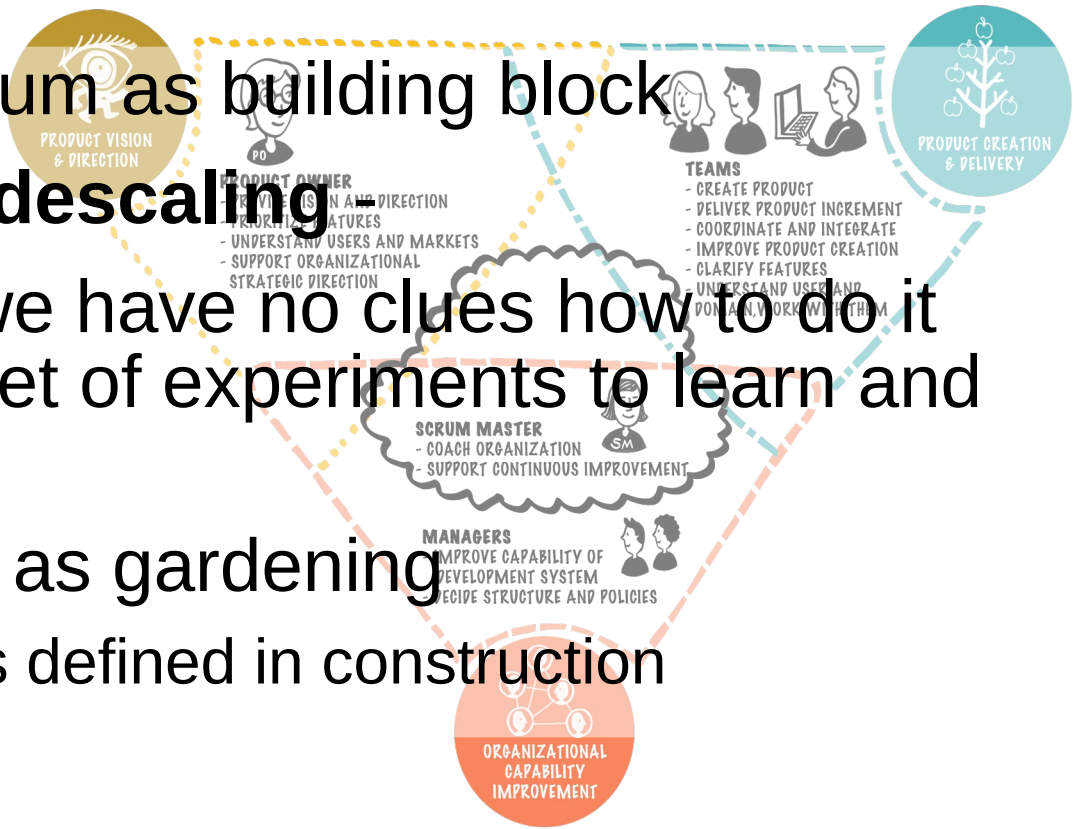
THINKING ABOUT TESTING

TEST-DRIVEN DEVELOPMENT

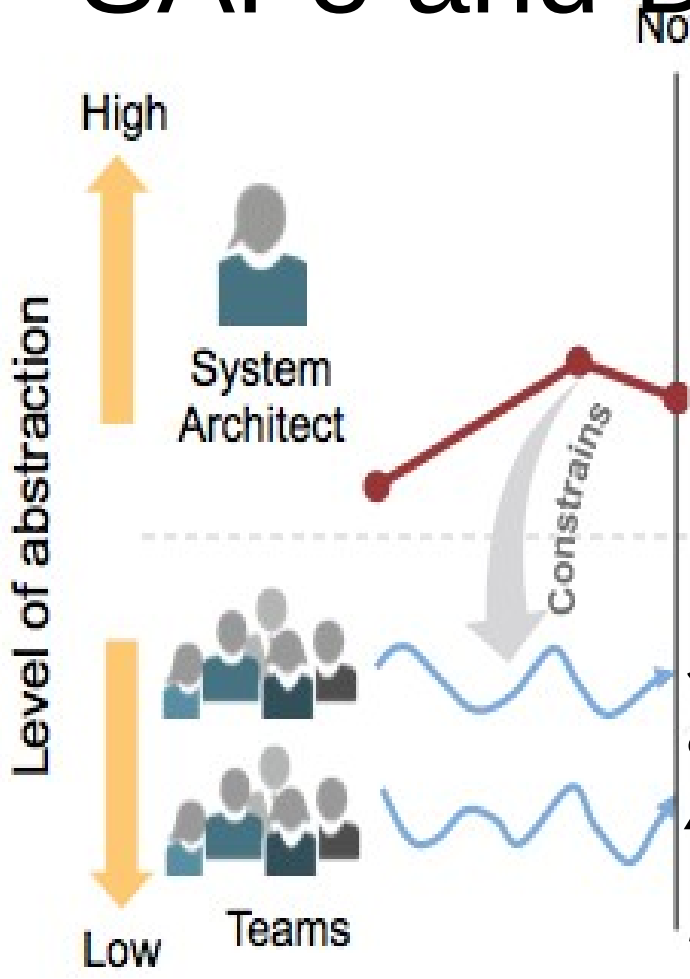
UNIT TESTING

LeSS Ideas

- LeSS emphasizes Scrum as building block
- Do more with LeSS – **descaling**
- LeSS acknowledges we have no clues how to do it and provides a huge set of experiments to learn and tailor.
- Promotes architecture as gardening
  - Discard architecture as defined in construction



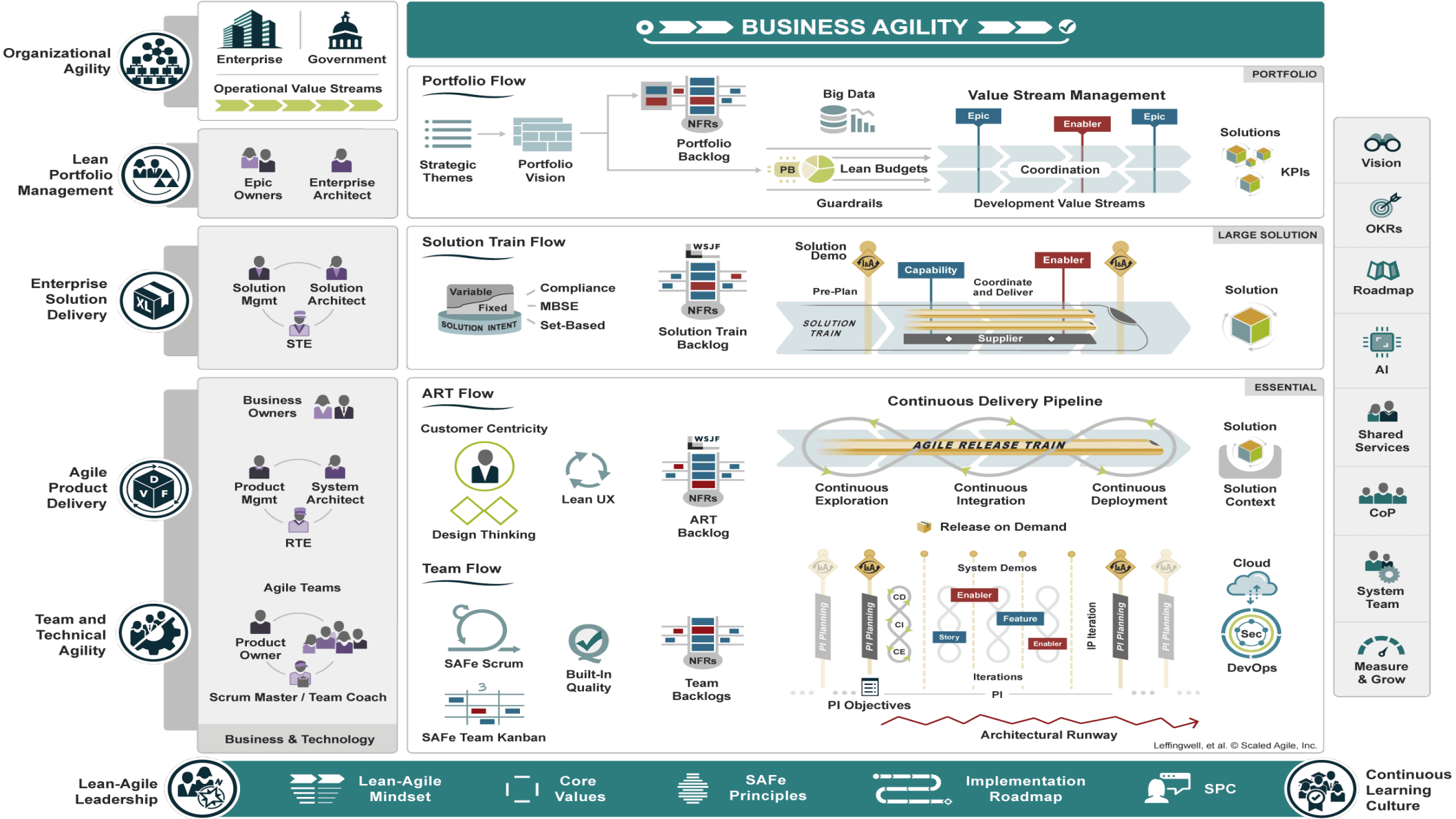
SAFe and DAD



- Their architect role is still classical
  - Runaway and BDUF
  - Architect is a position and a title and is not member of the team → violates Scrum and Manifesto values

On the bright side, SAFe is slowly improving – see the changes in SAFe 5 and 6 -.But still has an Enterprise, a System and a Solution Architect

DAD is now the official agile approach of PMI


BUSINESS AGILITY
Organizational Agility
Enterprise
Government
Operational Value Streams
Lean Portfolio Management
Epic Owners
Enterprise Architect
Enterprise Solution Delivery
Solution Mgmt
Solution Architect
STE
Agile Product Delivery
Business Owners
Product Mgmt
System Architect
RTE
Team and Technical Agility
Agile Teams
Product Owner
Scrum Master / Team Coach
Business & Technology
PORTFOLIO
Portfolio Flow
Strategic Themes
Portfolio Vision
NFRs
Portfolio Backlog
Big Data
PB
Lean Budgets
Guardrails
Value Stream Management
Epic
Enabler
Epic
Coordination
Development Value Streams
Solutions
KPIs
LARGE SOLUTION
Solution Train Flow
Variable
Fixed
SOLUTION INTENT
Compliance
MBSE
Set-Based
WSJF
NFRs
Solution Train Backlog
Solution Demo
Pre-Plan
Capability
Enabler
Coordinate and Deliver
SOLUTION TRAIN
Supplier
Solution
ESSENTIAL
ART Flow
Customer Centricity
Design Thinking
Lean UX
WSJF
NFRs
ART Backlog
Continuous Delivery Pipeline
AGILE RELEASE TRAIN
Continuous Exploration
Continuous Integration
Continuous Deployment
Release on Demand
Solution
Solution Context
Team Flow
SAFe Scrum
Built-In Quality
NFRs
Team Backlogs
SAFe Team Kanban
PI Planning
CD
CI
CE
System Demos
Enabler
Feature
Story
Enabler
Iterations
IP Iteration
PI
PI Objectives
Architectural Runway
Cloud
Sec
DevOps
Vision
OKRs
Roadmap
AI
Shared Services
CoP
System Team
Measure & Grow
Lean-Agile Leadership
Lean-Agile Mindset
Core Values
SAFe Principles
Implementation Roadmap
SPC
Continuous Learning Culture
Leffingwell, et al. © Scaled Agile, Inc.

Refactoring & Clean Code

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

Refactoring: Improving the Design of Existing Code, 1999

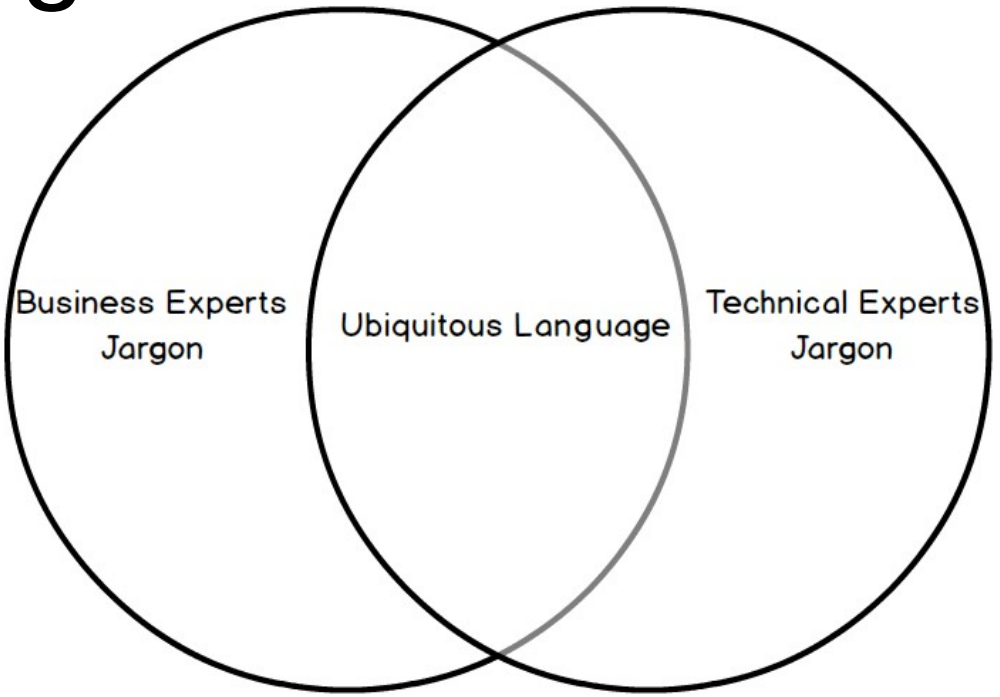
Agile Architecture Approach - DDD

- Domain Driven Design and Architecture
  - Bounded Domains
  - Event Storming
- Software craftsmanship, clean code, clean coder
- Technology stack in architecture – Forget about the illusion architecture is technology neutral
  - Look at Google 40’000 developers, 8 technology stacks

Ubiquitous Language

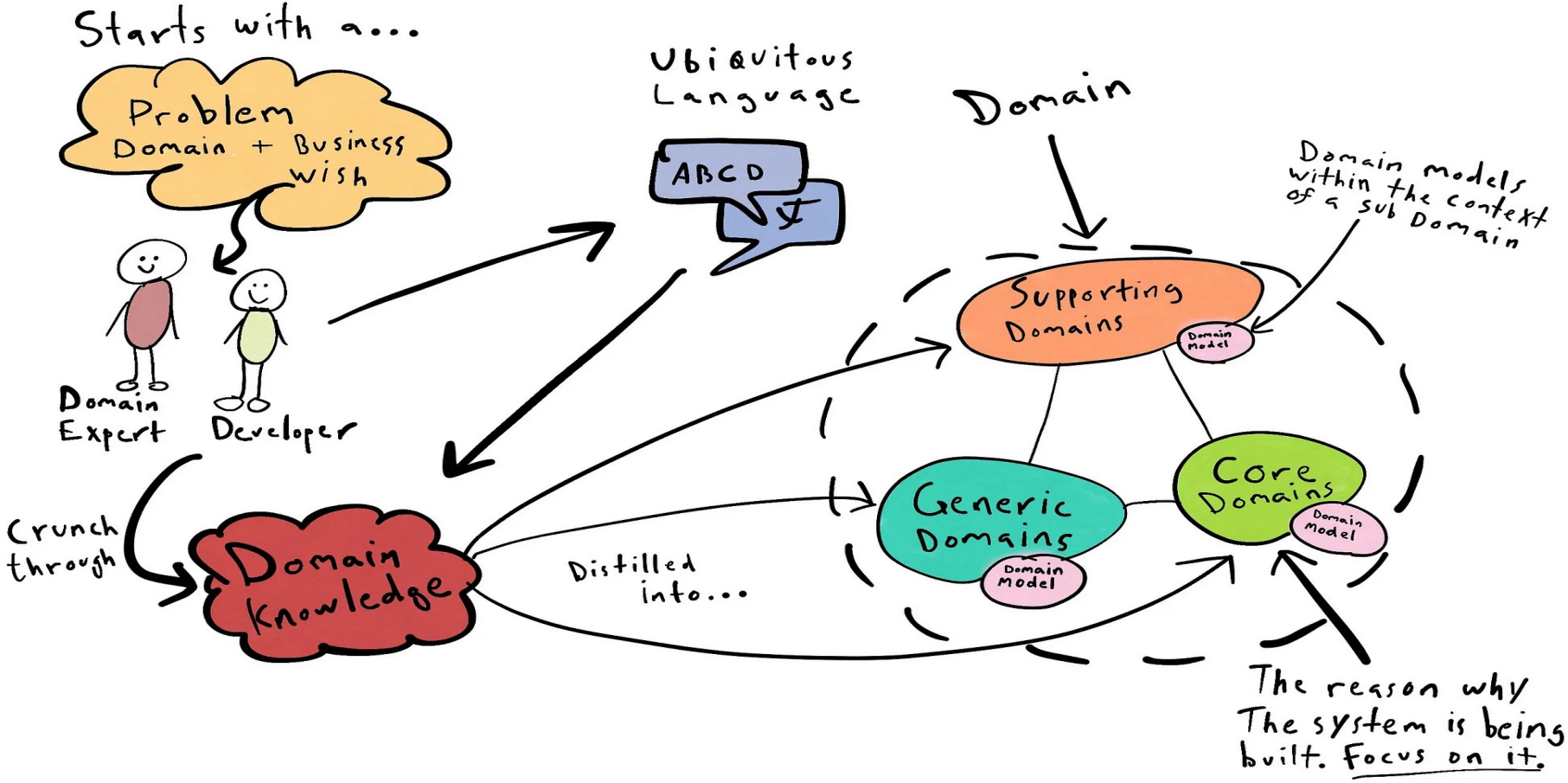
If you can’t **explain** it to a six-year-old, you don’t understand it yourself.
— Albert Einstein

“You should **name** a variable using the same care with which you name a first-born child.”
— Robert C. Martin

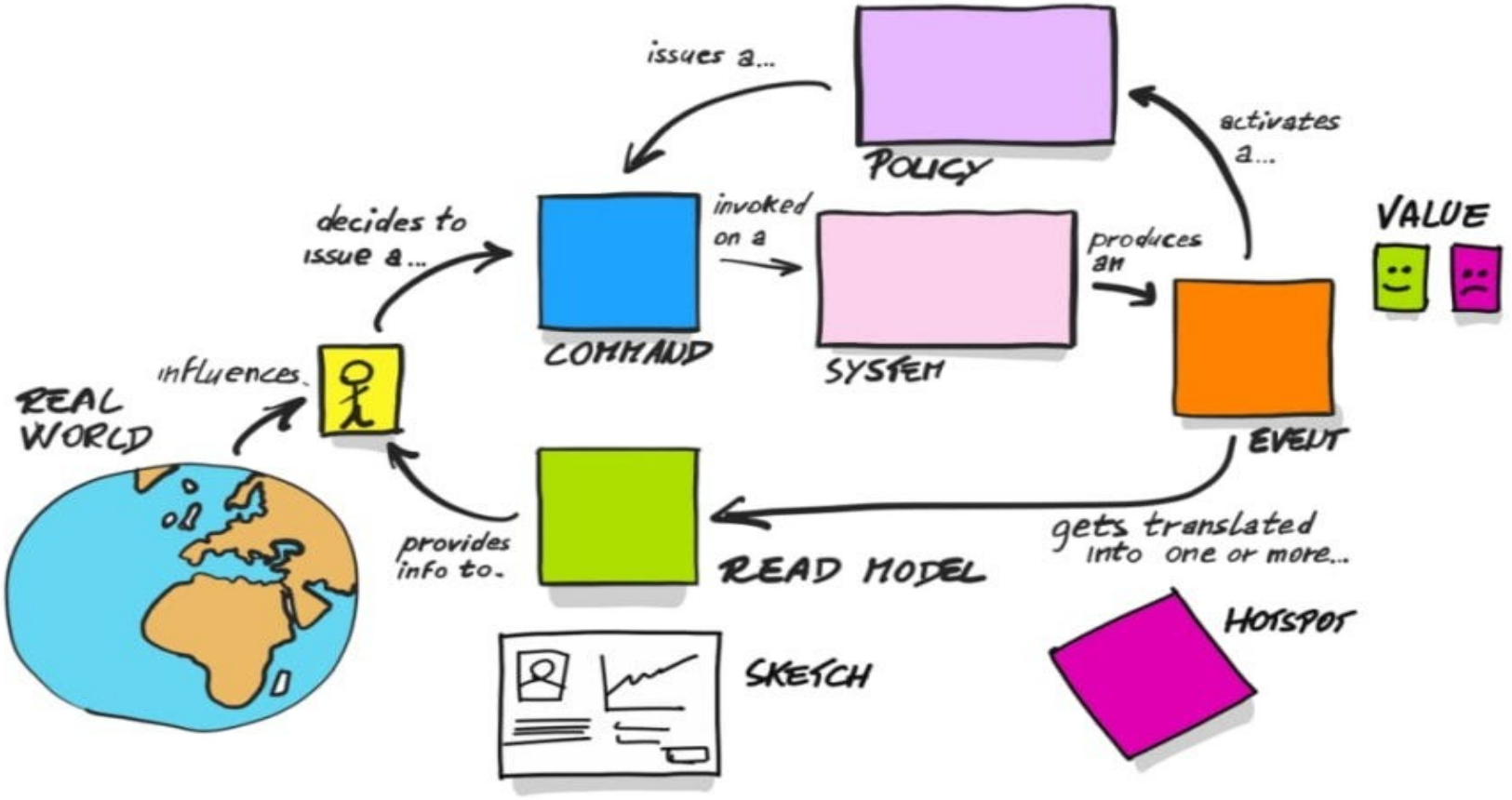


The fundamental horror of anemic model is that it is so contrary to the basic idea of object-oriented designing. The anemic domain model is just a procedural style design […]. What is worse, many people think that anemic objects are real objects, and thus **completely miss the point of what object-oriented design is all about**. — Martin Fowler

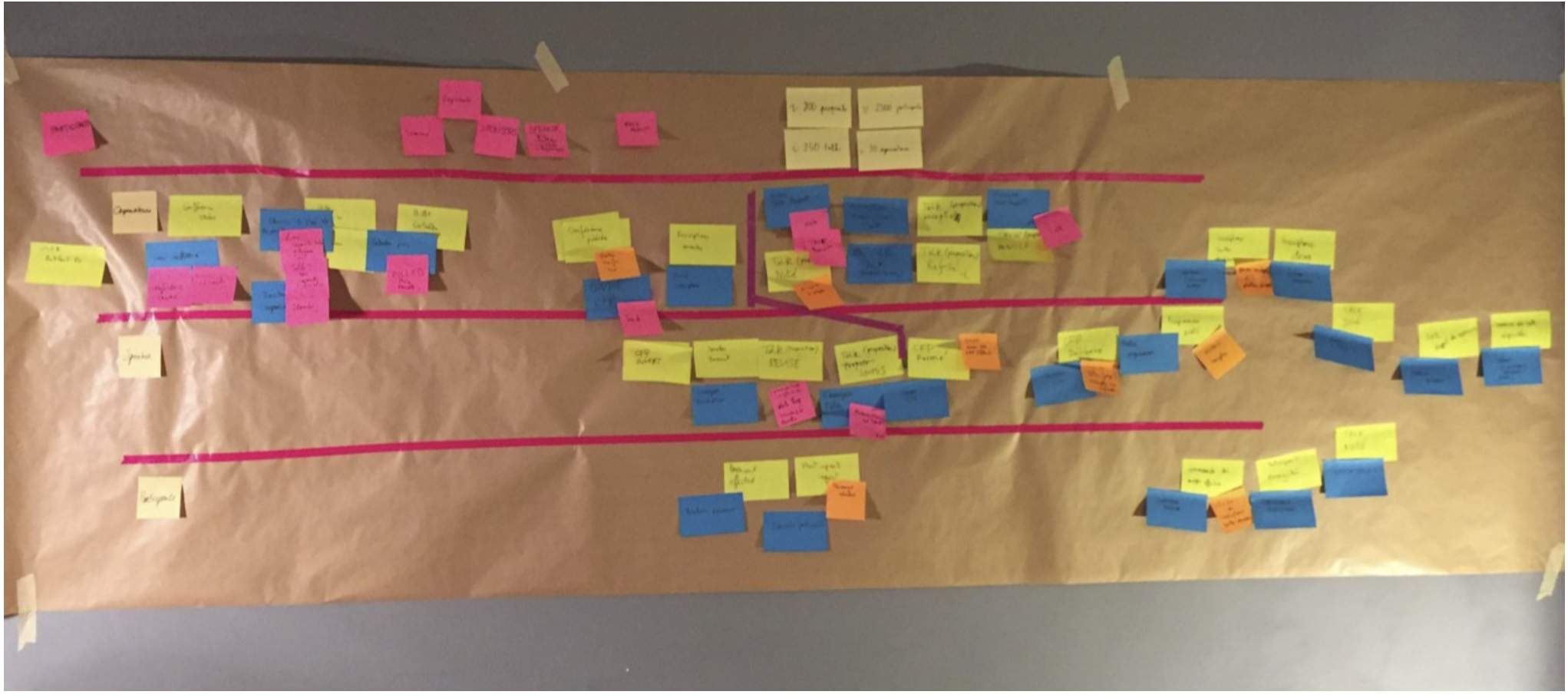
DDD – Domain Knowledge



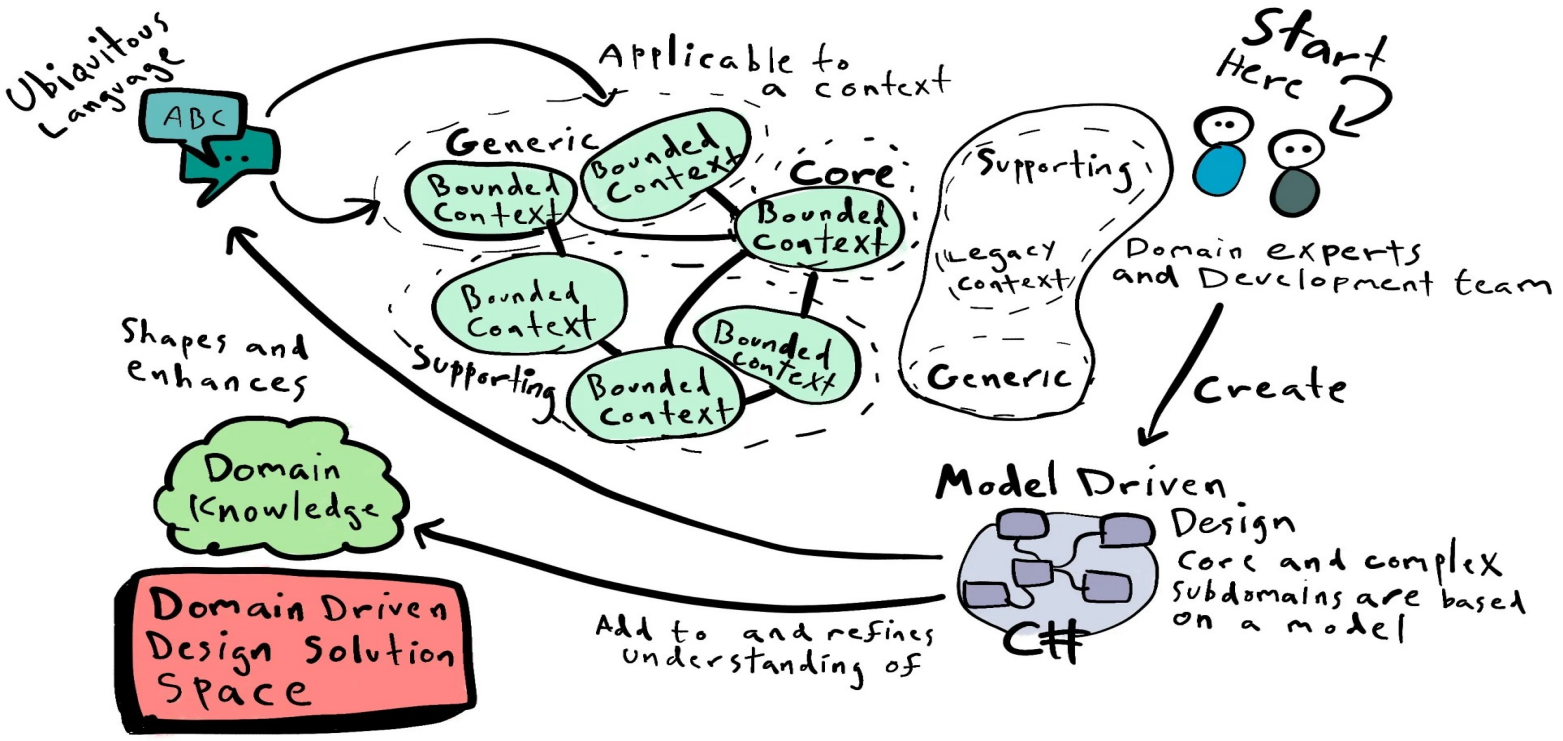
DDD – Event Storming



DDD – Event Storming Workshop



DDD – Bounded Domain



Your Project Status

- Java project with some classes and packages under Git
  - Use current JDK
- Gradle or Maven build script
- Development environment setup
- Run Sonar on the project – perhaps also PMD, Checkstyle or SpotBugs
- Have a set of unit tests and coverage information
- Have refactoring experience
- Have component architecture improvements experience

Links (1/2)

- Henrik Kniberg
  Agile Product Ownership in a Nutshell
- Henrik Knieberg Spotify Engineering Culture - I
- Henrik Knieberg Spotify Engineering Culture - II
- Introduction to LeSS
- Michael James *MJ* Introduction to LeSS

Links (2/2)

- Blog Scrum Developer
- Blog Agile Architecture with Scrum
- Blog Scrum Master
- Blog Product Owner
- You Must Be Crazy To Do Pair Programming, Dave Farley, GOTO 2022

Exercises (1/2)

- Read the LeSS “Large Scale Agile Design And Architecture Ways Of Working” article
- Read the “Scrum Guide”
- Identify the bounded domains of your product
- Evaluate golden trunk approach – *Death to long living branches (meaning more than a few hours)*
- Work on your product

Exercises (2/2)

- Coding Dojos
  - Logging in your components using **log4j2** or slf4j (a performance comparison can be found on the Apache log4j2 website)
  - Coding guidelines
  - Have your project under git, gradle and CI pipeline
- Quality Attributes
  - Naming quality → legible code
  - Which quality attributes of source code do you use during your coding activities? Discuss with your colleagues