

Software Architecture and Techniques

Agile Approaches *Scrum, eXtreme Programming, LeSS*

Architect Hats

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

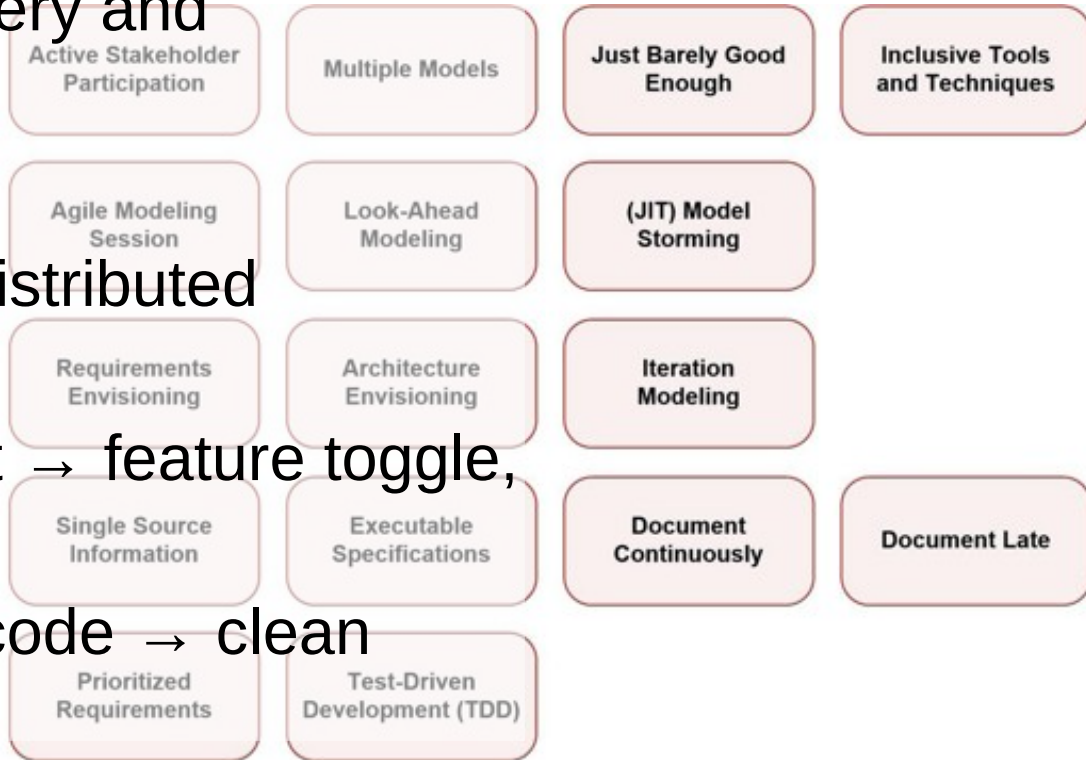
Agile Architecture Principles

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

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

Agile Architecture Techniques

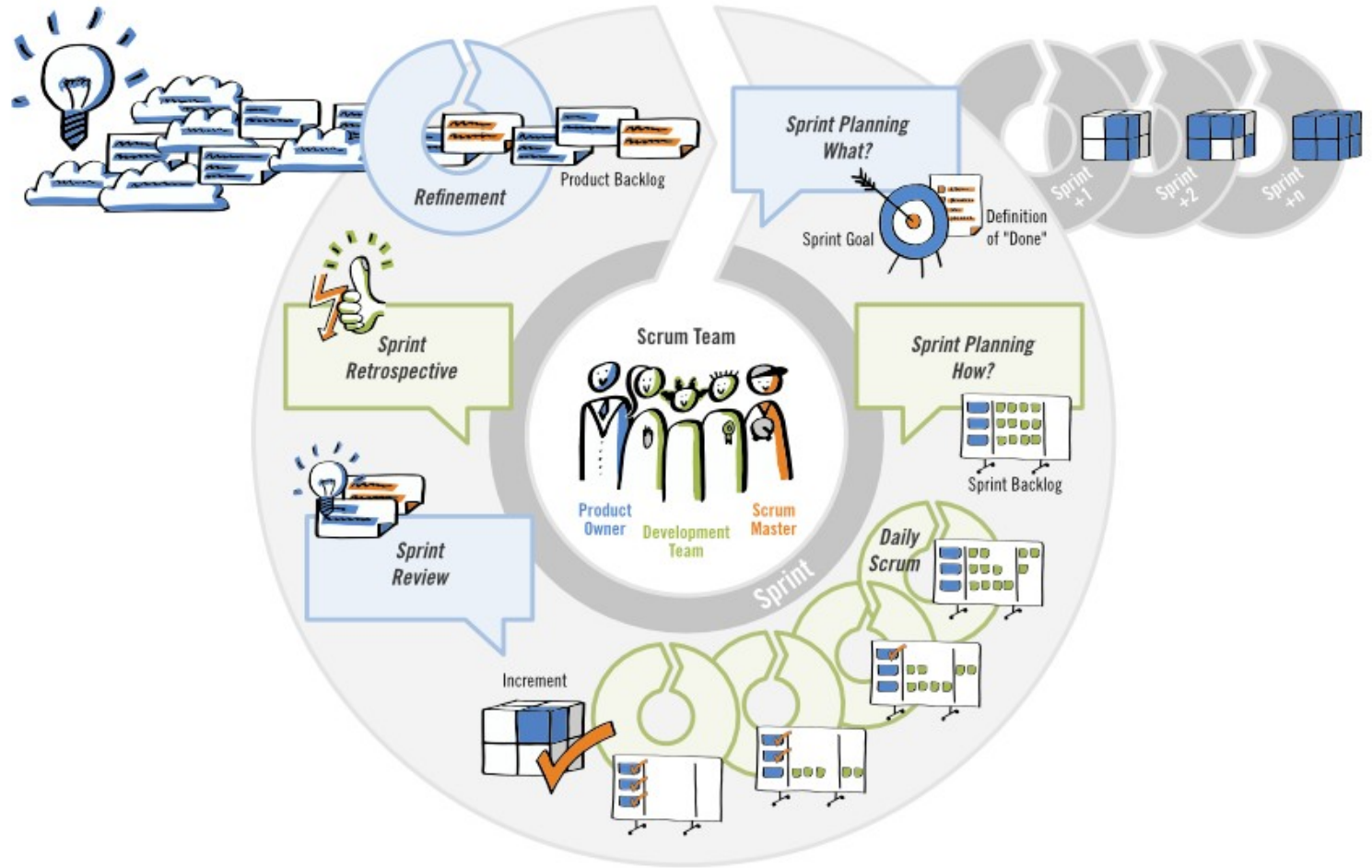
- Continuous integration, delivery and deployment
- Agile implies automation
- Git impact → golden trunk, distributed repository, GitOps
- Potentially shippable product → feature toggle, no *Undone* work
- Clean architecture → clean code → clean coder



Scrum Practices

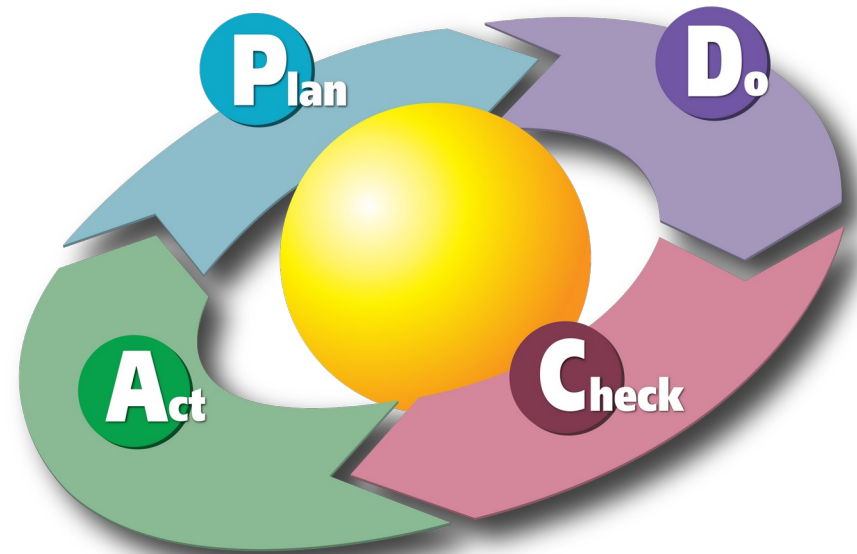
- Scrum does not prescribe any technical practices (see [flaccid Scrum](#))
- Scrum emphasizes vision, context, roadmap
- It should be all about value → *outcome over output*
- Scrum encourages applying eXtreme Programming techniques
- Scrum alliance is working together with [LeSS](#)

Scrum



Scrum Approaches

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

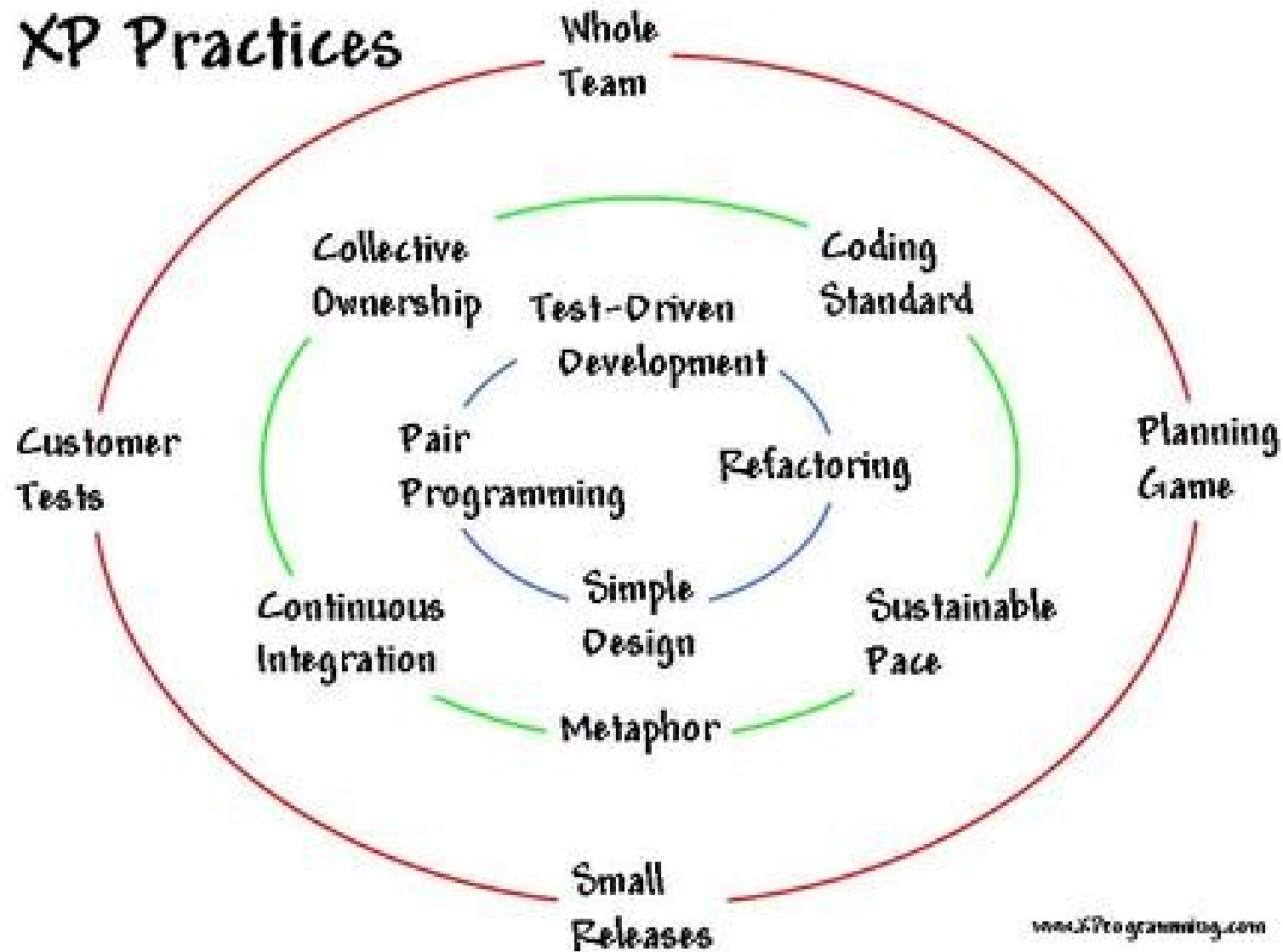


eXtreme Programming Practices

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

XP requires Software Craftsmanship

XP Practices



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 craftsmanship**
- Architect is a **lean leader** – teacher, coach, mentor
- Architect discuss 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 Practices



SPECIFICATION BY EXAMPLE



CONTINUOUS
INTEGRATION



CONTINUOUS DELIVERY

Architecture and Design
Technical Excellence
Continuous Integration



TEST AUTOMATION



TECHNICAL
EXCELLENCE



ARCHITECTURE
& DESIGN



ACCEPTANCE
TESTING



THINKING ABOUT TESTING



CLEAN CODE



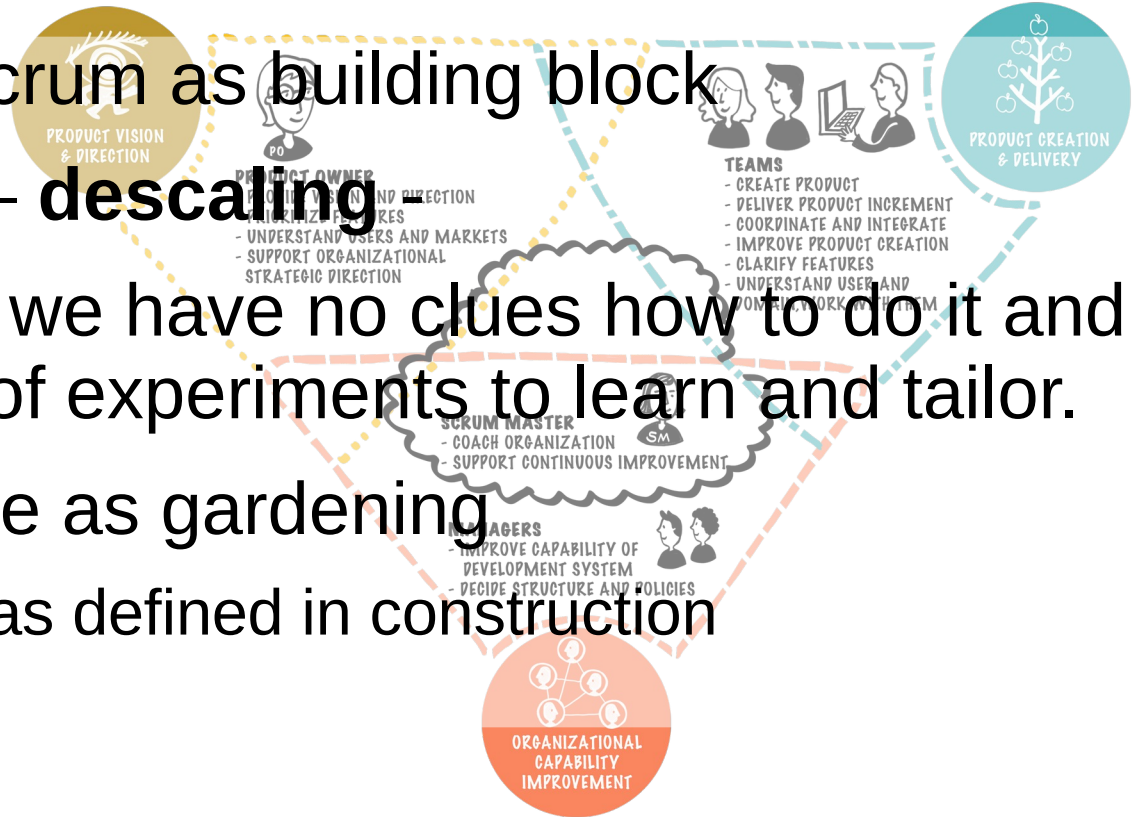
TEST-DRIVEN DEVELOPMENT



UNIT TESTING

LeSS Practices

- LeSS emphasizes Scrum as building block
- Do more with LeSS – **downscaling** -
- 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





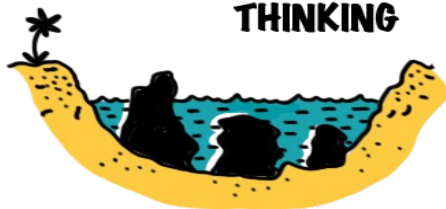
QUEUEING THEORY



**EMPIRICAL
PROCESS CONTROL**



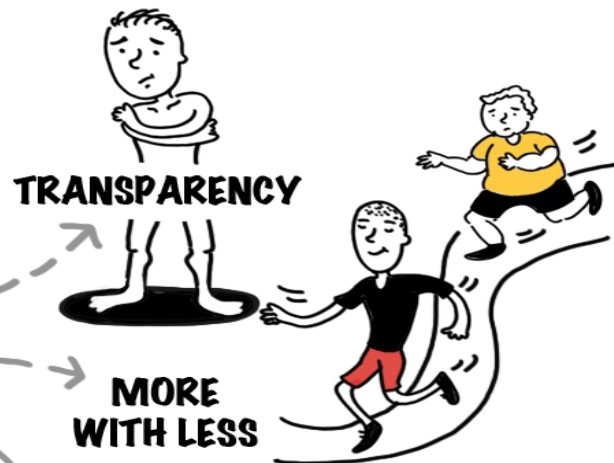
**SYSTEMS
THINKING**



**LEAN
THINKING**



**LARGE-SCALE
SCRUM IS SCRUM**



TRANSPARENCY

**MORE
WITH LESS**



**WHOLE
PRODUCT
FOCUS**

360°

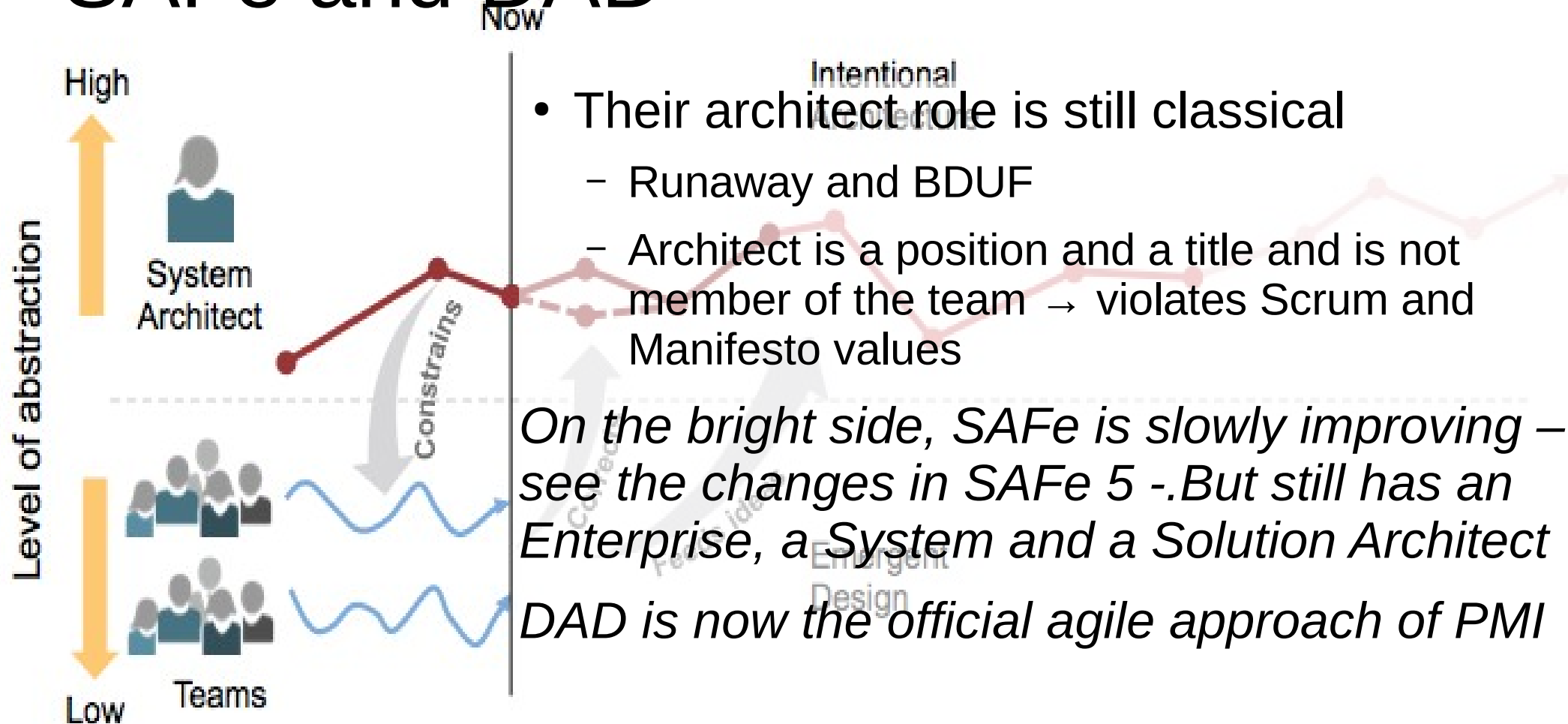
**CUSTOMER
CENTRIC**



**CONTINUOUS IMPROVEMENT
TOWARDS PERFECTION**



SAFe and DAD



Organizational
Agility



Lean Portfolio
Management



Enterprise
Solution
Delivery



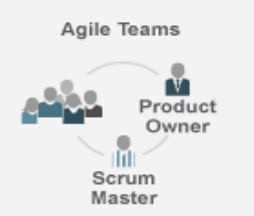
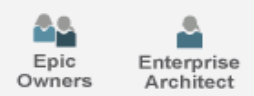
Agile
Product
Delivery



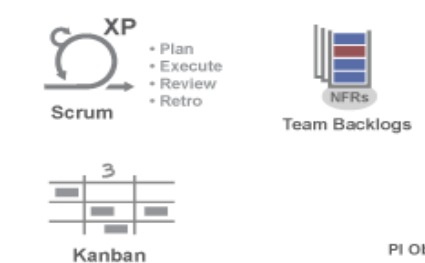
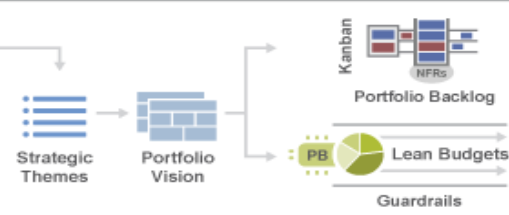
Team and
Technical
Agility



Continuous
Learning
Culture

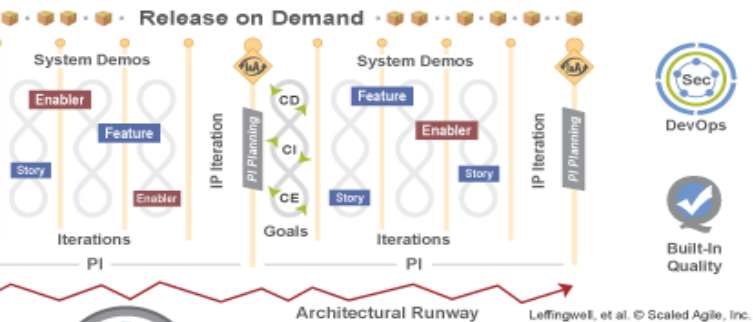
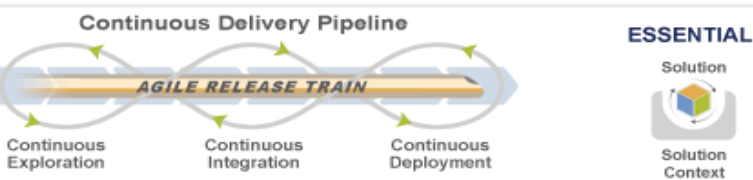
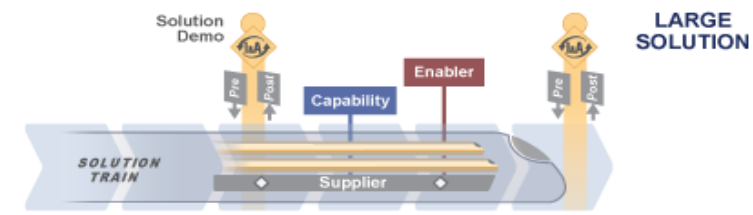
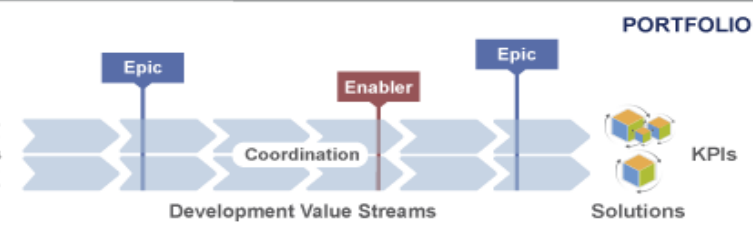


Operational Value Streams



Business Agility

Measure & Grow



- Vision
- Roadmap
- Milestones
- Shared Services
- CoP
- System Team
- Lean UX
- Metrics

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

- 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

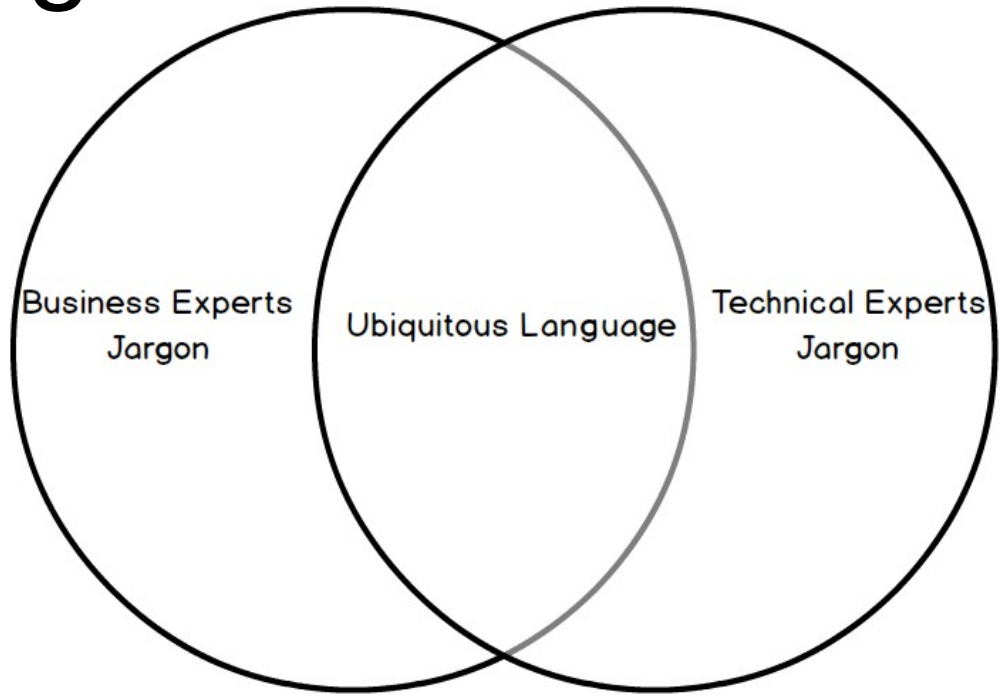
Ubiquitous Language

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's so contrary to the basic idea of object-oriented designing. The anemic domain model is just a procedural style design [...]. What's 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

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 Knieberg
Agile Product Ownership in a Nutshell
- Henrik Knieberg Spotify Engineering Culture - I
- Henrik Knieberg Spotify Engineering Culture - II
- 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
 - 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