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 title*

- Domain expert

- Technology expert

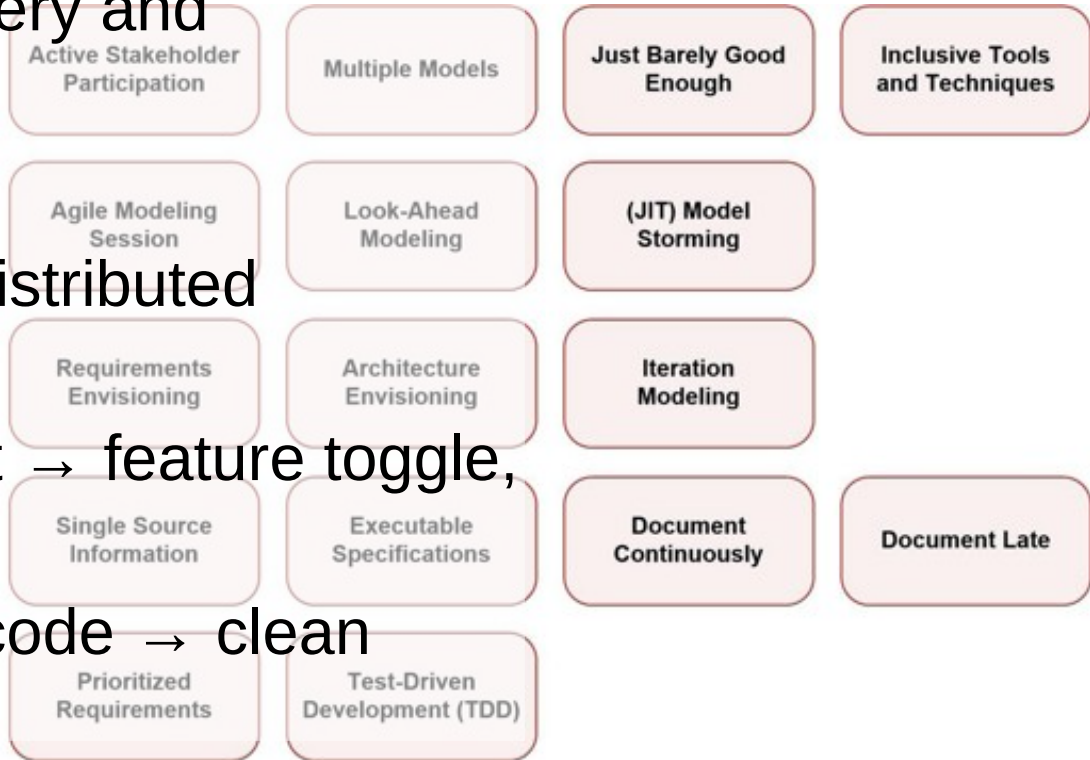- Stakeholder facilitator

- 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

|			
Active Stakeholder Participation	Multiple Models	**Just Barely Good Enough**	**Inclusive Tools and Techniques**
Agile Modeling Session	Look-Ahead Modeling	**(JIT) Model Storming**	
Requirements Envisioning	Architecture Envisioning	**Iteration Modeling**	
Single Source Information	Executable Specifications	**Document Continuously**	**Document Late**
Prioritized Requirements	Test-Driven Development (TDD)		

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
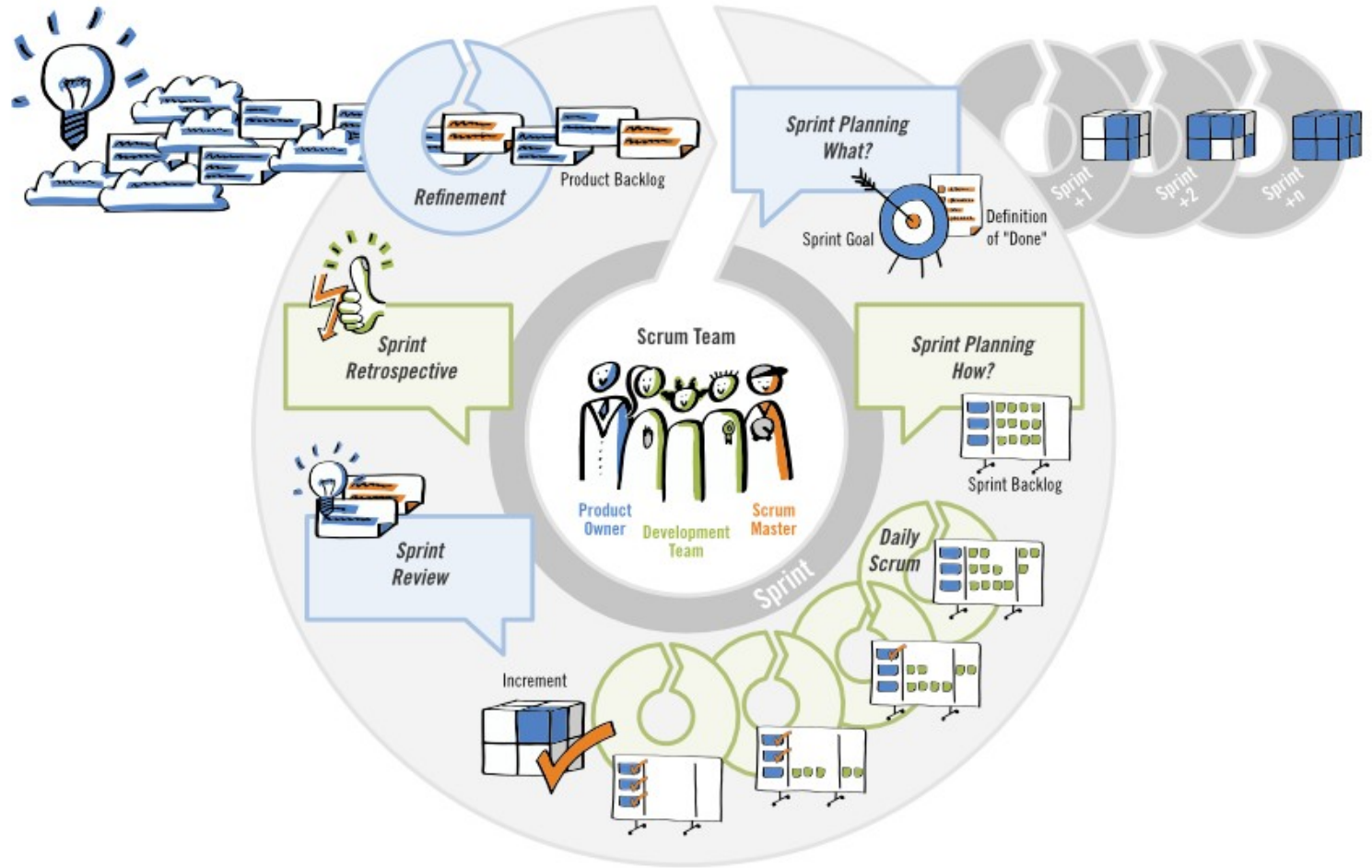
Scrum Approaches

- Scrum emphasizes **learning in the team**

- Scrum builds on continuous improvement
  - Retrospective
  - Review
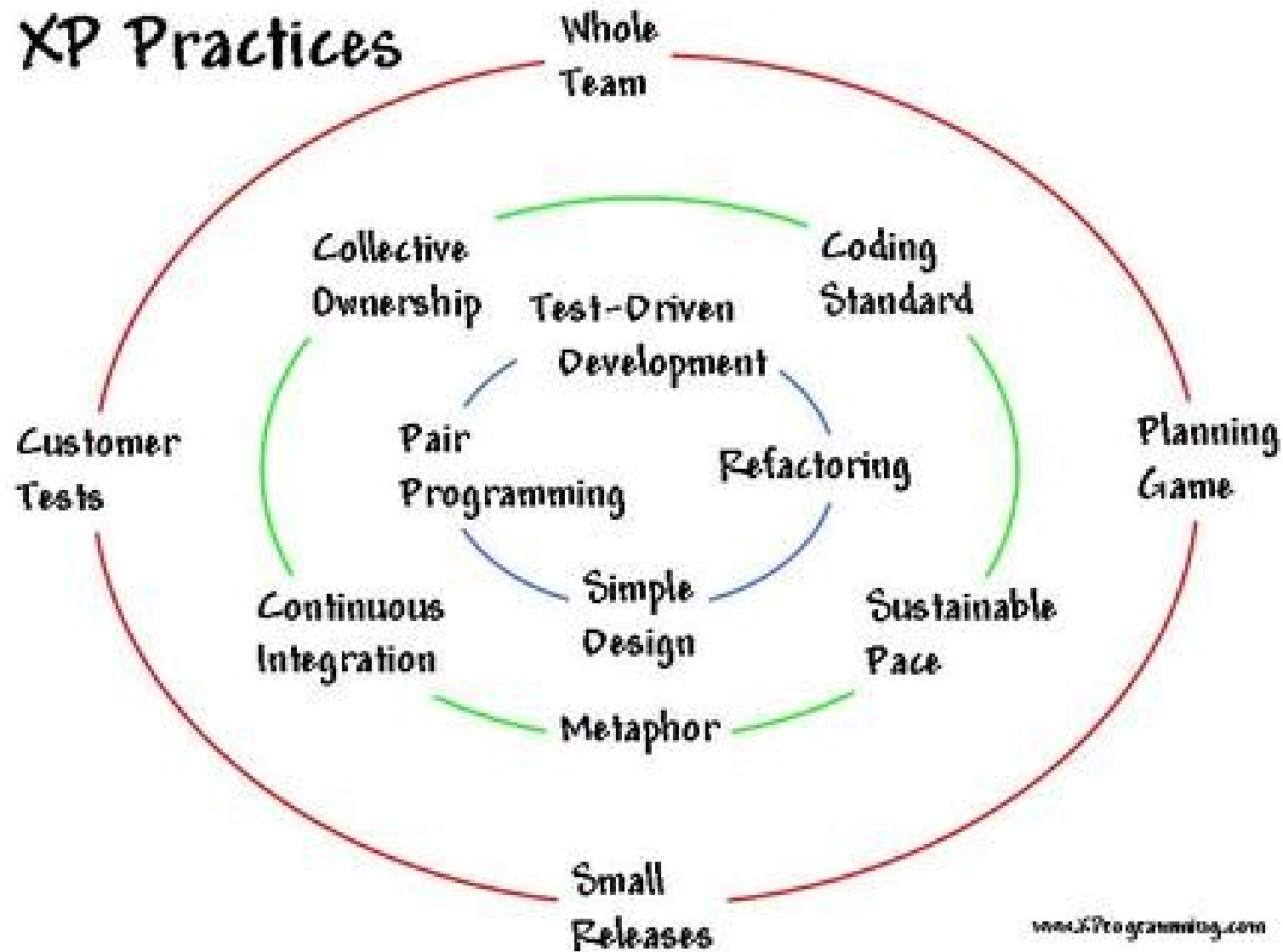 - Daily meeting
  - *Always*
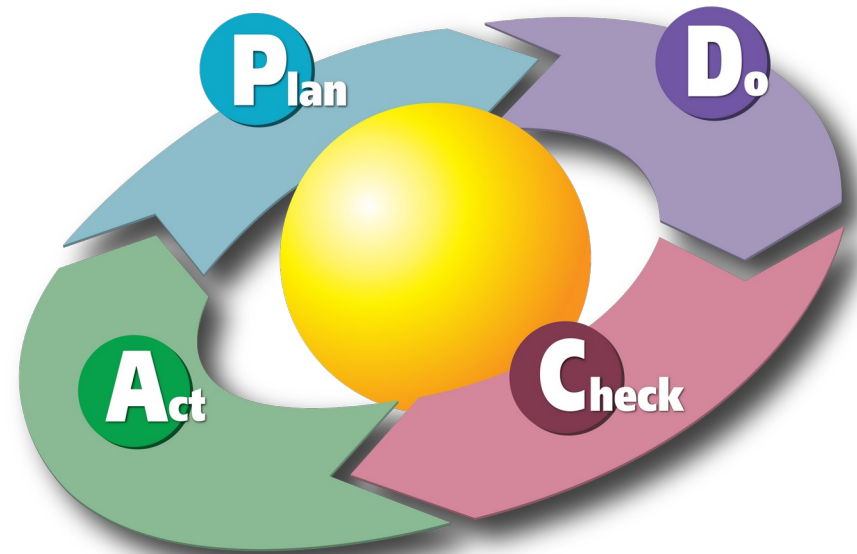
eXtreme Programming Practices

- Pair Programming

- Test Driven Development

- CI

- Refactoring

- Coding Standards

- Collective Code Ownership

- Simple Design

- System Metaphor

XP requires Software Craftsmanship

XP Practices

Whole Team

Coding Standard

Collective Ownership

Test-Driven Development

Customer Tests

Pair Programming

Refactoring

Planning Game

Continuous Integration

Simple Design

Sustainable Pace

Metaphor

Small Releases

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

Architecture and Design

Technical Excellence

Continuous Integration

SPECIFICATION BY EXAMPLE

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

TEST AUTOMATION

ACCEPTANCE TESTING

TECHNICAL EXCELLENCE

ARCHITECTURE & DESIGN

CLEAN CODE

THINKING ABOUT TESTING

TEST-DRIVEN DEVELOPMENT

UNIT TESTING

LeSS Practices

- 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
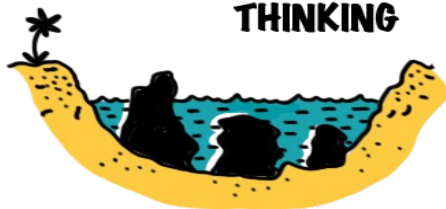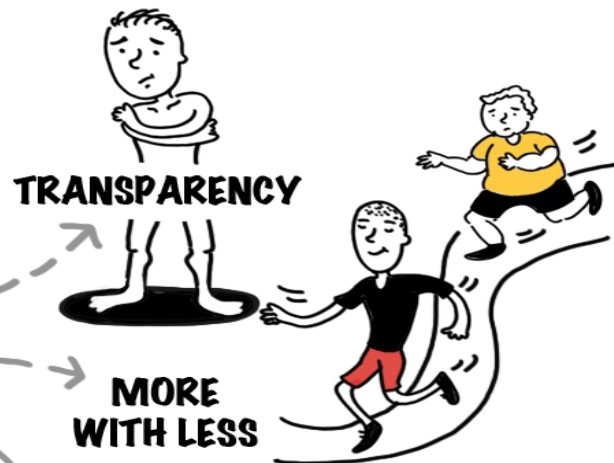 - Discard architecture as defined in construction

QUEUEING THEORY

LARGE-SCALE SCRUM IS SCRUM
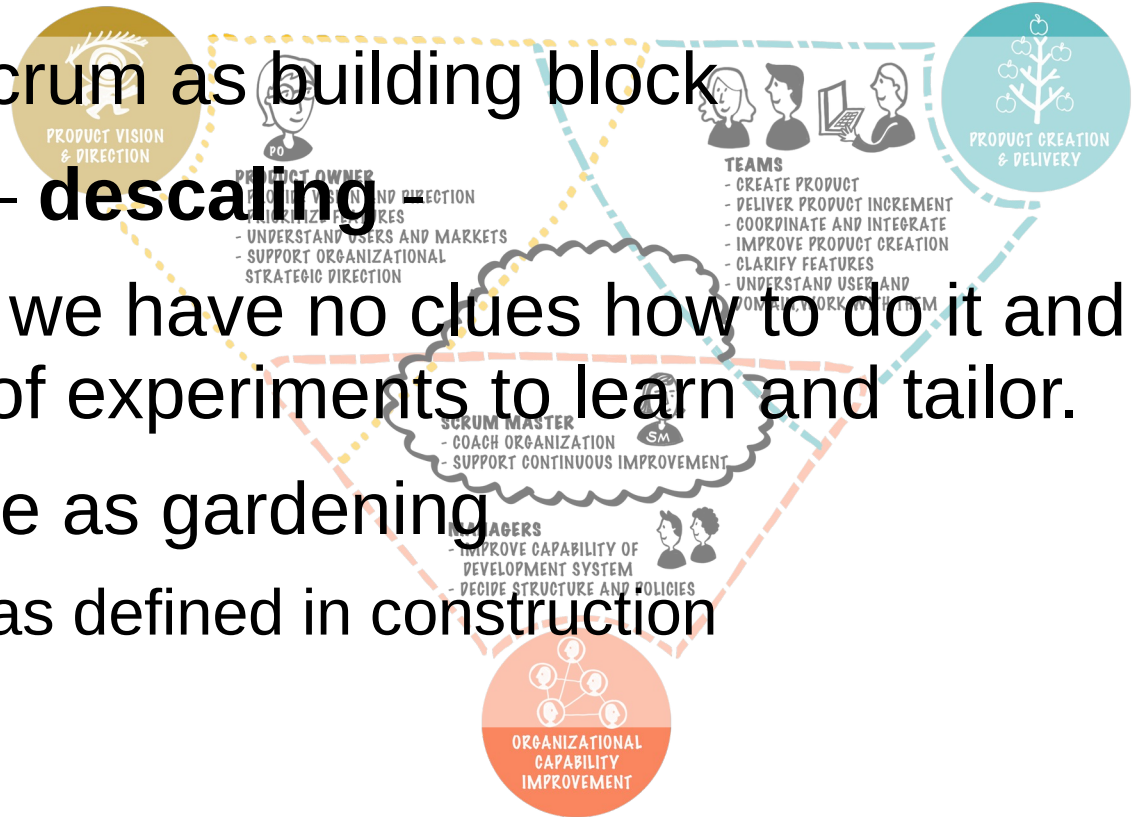
TRANSPARENCY

MORE WITH LESS

EMPIRICAL PROCESS CONTROL

SYSTEMS THINKING

WHOLE PRODUCT FOCUS

360°

LEAN THINKING

CONTINUOUS IMPROVEMENT TOWARDS PERFECTION

CUSTOMER CENTRIC

CUSTOMER

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 -.But still has an Enterprise, a System and a Solution Architect

DAD is now the official agile approach of PMI

Business Agility

Operational Value Streams

Measure & Grow

Enterprise | Government

Organizational Agility

Lean Portfolio Management

Enterprise Solution Delivery
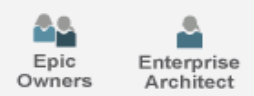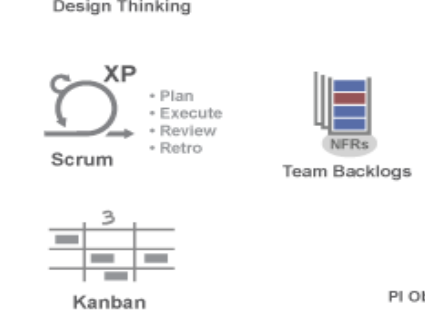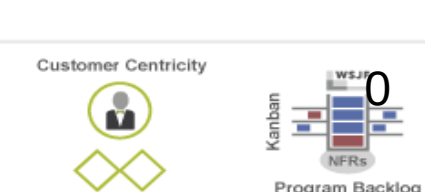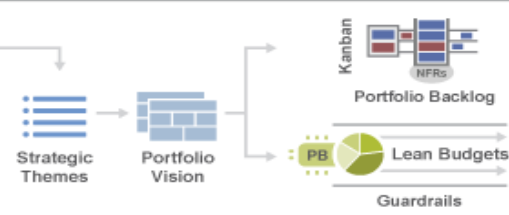
Agile Product Delivery
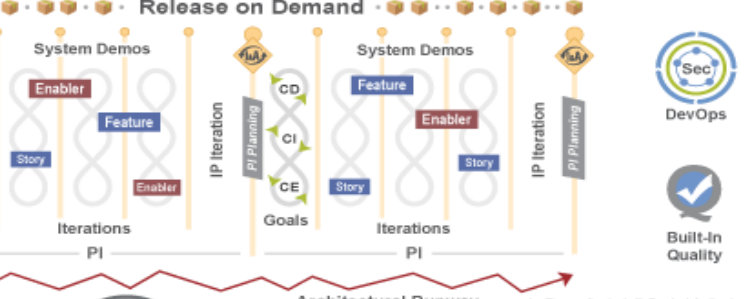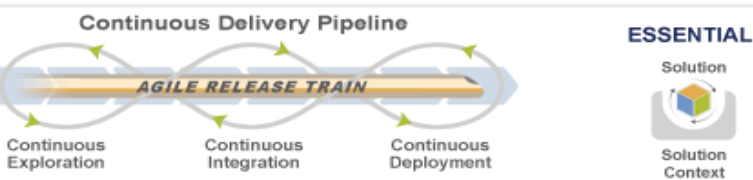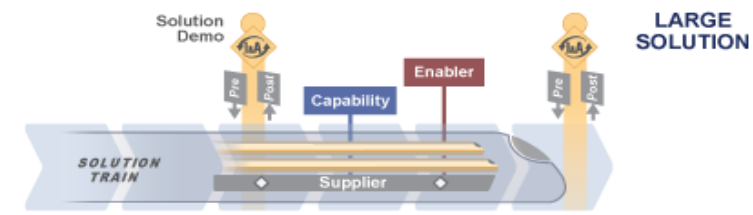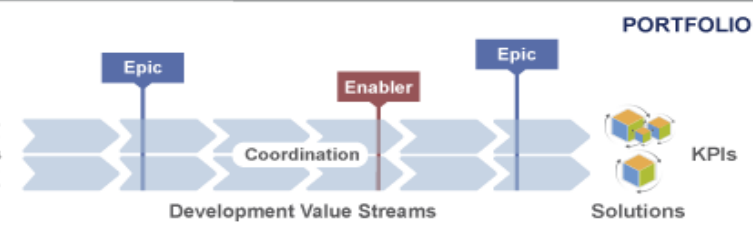
Team and Technical Agility

Continuous Learning Culture

PORTFOLIO

Epic Owners | Enterprise Architect

Strategic Themes → Portfolio Vision

Kanban — Portfolio Backlog — NFRs

PB — Lean Budgets

Guardrails

Epic | Enabler | Epic

Coordination

KPIs

Development Value Streams

Solutions

LARGE SOLUTION

Solution Arch/Eng | Solution Mgmt

STE

Variable / Fixed — SOLUTION INTENT
- Compliance
- MBSE
- Set-Based

Kanban — Solution Backlog — WSJF — NFRs

Solution Demo
Pre / Post

Capability | Enabler

SOLUTION TRAIN

Supplier

ESSENTIAL

Business Owners

System Arch/Eng | Product Mgmt

RTE

Agile Teams

Product Owner

Scrum Master

Customer Centricity

Design Thinking

XP — Scrum
- Plan
- Execute
- Review
- Retro

Kanban

Kanban — Program Backlog — WSJF — NFRs — 0

Continuous Delivery Pipeline

AGILE RELEASE TRAIN

Continuous Exploration | Continuous Integration | Continuous Deployment

Release on Demand

Team Backlogs — NFRs

System Demos
CD / CI / CE
Enabler | Feature | Story
Goals
Iterations
IP Iteration
PI Planning
PI Objectives
PI
Architectural Runway

Solution

Solution Context

CoP

System Team

Lean UX

DevOps

Built-In Quality

Vision | Roadmap | Milestones | Shared Services

Leffingwell, et al. © Scaled Agile, Inc.

Business | Technology

Core Values | Lean-Agile Mindset | SAFe Principles | Implementation Roadmap | SPC

Lean-Agile Leadership

5.1

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
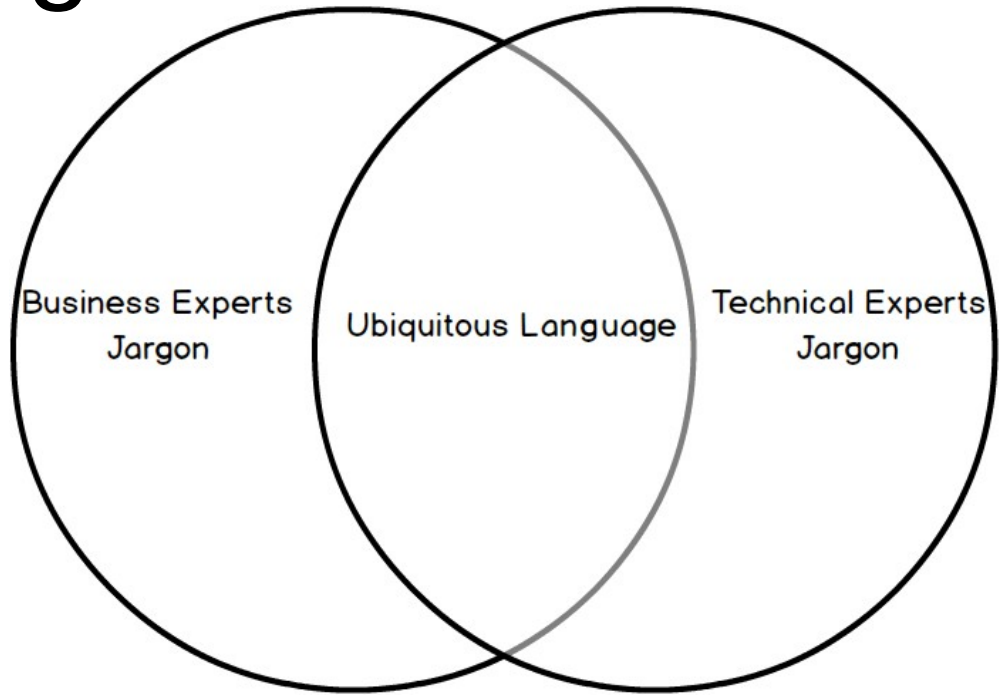 - Look at Google 40'000 developers, 8 technology stacks

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

Business Experts Jargon — Ubiquitous Language — Technical Experts Jargon

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 Agile Architecture Principles
- Blog Agile Architecture with Scrum
- Blog Scrum Developer

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