LARGE SOFTWARE DEVELOPMENT AT ERICSSON

Diarmuid Corcoran: Expert Software Architecture / DURA System & Technology
OVERVIEW

• Part 1: Setting the Context

• Part 2: Large Scale Development Practices at Ericsson

• Part 3: Some Tools and Techniques Used to Manage Complexity

• Part 4: Some Research Challenges
WHO AM I?

Worked at Ericsson since 1992. Primarily in software and system roles.

Worked on all the major radio standards: GSM, WCDMA, LTE, 5G

Hold a position as “Expert in Software Architecture”

Primarily working on software research activities to support next generation radio networks.
LEADING COMMUNICATION
FROM THE START

Impact on society

1800 1850 1900 1950 2000 2020

Morse code
Signal lamps
Ericsson was founded
Automatic call switching system
First mobile designed
Mobile text and data system
Bluetooth
3G
Mobile video call
Wide spread
Adaption of mobile communications
THE OBLIGATORY ERICSSON MARKETING VIDEO

Video: This is Ericsson

› https://www.youtube.com/watch?v=c0TMimgBdsU
WIRELESS SUBSCRIBER GROWTH:

- ~10 Years for a new RAN technology to become mainstream
- RAN technology lifespan 20-30 years.
- 40% of all mobile traffic passes through Ericsson systems.

Source: Ericsson Mobility Report Nov 2016
Software for system management and services

HW platforms and software for core network

Diverse radio access platforms: HW and software

IPR and standards for system access
RADIO ACCESS NETWORK ARCHITECTURE EXAMPLE

4G (3G LTE/SAE)

- Many system nodes
- Many standardized protocols

IMS

5G (BEING DEFINED)

• Many system nodes
• Many standardized protocols
### Different Generations of Mobile Networks

<table>
<thead>
<tr>
<th>Technology Standard</th>
<th>Example systems</th>
<th>Year of introduction</th>
<th>Application area</th>
<th>Approximate maximum data rate</th>
<th>System Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1G</td>
<td>NMT, AMPS</td>
<td>1980</td>
<td>Voice</td>
<td>~10 kbps</td>
<td>Highly Centralized</td>
</tr>
<tr>
<td>2G</td>
<td>GSM, PDC</td>
<td>1992</td>
<td>Voice + text</td>
<td>10-400 kbps</td>
<td>Highly Centralized</td>
</tr>
<tr>
<td>3G</td>
<td>WCDMA, CDMA 2000</td>
<td>2001</td>
<td>Voice + text + data</td>
<td>2 – 80 Mbps</td>
<td>Medium Centralised</td>
</tr>
<tr>
<td>4G</td>
<td>LTE</td>
<td>2009</td>
<td>Capacity</td>
<td>50 – 1000 Mbps</td>
<td>Highly Distributed</td>
</tr>
<tr>
<td>5G</td>
<td>RAN in Cloud</td>
<td>2020</td>
<td>- X Capacity</td>
<td>&gt;10,000 Mbps</td>
<td>Flexible-Centralized/Distributed</td>
</tr>
</tbody>
</table>

- **1G**: Highly Centralized
- **2G**: Highly Centralized
- **3G**: Medium Centralised
- **4G**: Highly Distributed
- **5G**: Flexible-Centralized/Distributed
LTE SYSTEM ARCHITECTURE

Evolved Core

Evolved Radio Access

Evolved Node B (base station)

Layer 1: How to transmit bits & packets over radio
Layer 2: How to get the data into the right packets, and to get the right packet to the right receiver
Layer 3: How to keep track of network and user states and where to find each user (in which cell they are)
SO WHERE DO THE “SYSTEM OF SYSTEM” REQUIREMENTS COME FROM

3gpp

Non-Functional

Technology Oriented

External Requirement Source

HW Capability

Customer Specific

Verification Driven

Legal Compliance

Internal Requirement Source

System

eNB System

System Level

Product Level

Subsystem Level

Design Level

System Node

Product Node

Sub-System Node

Design Node

Services

Apps

Middleware

OS

Silicon

External Requirement Source

Non-Functional

Technology Oriented
The 3rd Generation Partnership Project (3GPP) is a collaboration between groups of telecommunications associations and has become the de facto standards body for developing worldwide cellular standards.

- ~10,000 pages of standard documents describe the LTE eNB.
- Very difficult to interpret the standards if you are not part of the process.
- Patents from standards compliance is a very important source of income.
KEY TECHNICAL SOFTWARE CHALLENGES IN SYSTEM IMPLEMENTATION

- Scaling up/down number of concurrent instances
- Handling software robustness in a sensible way
- Powerful abstractions for handling concurrency and parallelism
- Building portable software that can live over many HW generations
- Finding and defining good component interfaces
- Run-time frame-works that allow many languages to integrate seamlessly.
- Good test and observability strategies.
- Keeping up with the technology curve
How do we build complex software systems at scale.
THE STORY LINE: LARGE SCALE SOFTWARE DEVELOPMENT

“The Early Days” : Waterfall and “V” Model

Rational Unified Model

Agile, Lean and Scrum

Service Based Model?

• Sequential development process.
• Highly specification focused. Less focus on software.
• Very little opportunity to re-visit requirements after entry to design phase.
• Each group had very limited interaction. “Talk-Through-Spec”.
• “Big Bang” delivery model. You don’t find problems until late.
• Super nerve racking from a project management perspective.
• The rational method was a “framework” that could be applied on-top of the “V” model.
• Each phase can iteratively incorporate various disciplines such as model, test.
• Attempted to make “V” model more iterative and responsive to requirements flux.
• Still very engineering focused. But software is also important.
• Came with a lot of sophisticated tool support.
• Model driven in our Ericsson version
CURRENT: LEAN, AGILE AND SCRUM

› Lean:
  – Eliminate unnecessary artifacts in SW development process.
  – Delivery rapid increments of value.
  – Optimize the end-to-end flow of information.

› Agile:
  – A reaction to heavy-weight methodologies.
  – A set of principals or practices to adhere to.

› Scrum:
  – A management framework for handling agile ways-of-working.
MOVE TO AGILE
TRANSFORMATION CHANGE DRIVER

1. CREATE MORE VALUE
2. REDUCE LEADTIME
3. BUILD QUALITY IN
4. EMPOWER PEOPLE
THE AGILE MANIFESTO

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:"

• **Individuals and interactions over processes and tools**

• **Working software over comprehensive documentation**

• **Customer collaboration over contract negotiation**

• **Responding to change over following a plan**

http://agilemanifesto.org/
• Many information artefact handover points
AGILE DEVELOPMENT

› Use Cross Functional Teams (XFT) with end-to-end responsibility
› Work organized according to Scrum or Kanban
AGILE WAY OF WORKING: ROLES

› **Product Owner: ”Keeper of Requirements”**
  - Keeps track of and prioritizes team’s backlog of Work Items to do
  - Is responsible for the **value** of the work done

› **Scrum / Kanban Master: ”Part-Time Team/Project Lead”**
  - Facilitates the daily scrum / kanban and other meetings for the XFT

› **XFT members: ”Implementation Task Force”**
  - Responsible for delivering good quality code and documentation
  - Strive for a broad competence regarding both overall system knowledge and different disciplines (system, design and test)
  - Responsible for the code of the whole product

› **Product Guardian: ”Minimize Technical Debt”**
  - Secures that product architecture doesn’t deteriorate over time
  - Coaches teams in his/her area of expertise

› **Line Manager: ”Vacation, Sick-leave, Salary”**
  - Organization & staff handling
  - Team coaching
TYPICAL CROSS-FUNCTIONAL TEAM (XFT)

- 1 Scrum / Kanban master (part time)
- 1 System competence (short supply)
- 4 SW developers
- 1 – 2 testers
  - Test equipment
SPRINT

› Planning horizon: One Sprint
  – Team commitment only valid for ongoing Sprint

› Regular meetings in a Sprint
  – Daily Scrum
  – Sprint planning
  – Backlog handling
  – Sprint Review / Demo
  – Sprint retrospective
XFT’S ORGANIZED IN PROGRAMS

This is in contrast to the projects we used to run
SCAILING UP “OUT-OF-THE-BOX” AGILE IS A MASSIVE CHALLENGE

Number of teams
- Typical: Smaller number of teams
- PDU’s@Ericsson: Large number of teams

Size of products
- Typical: Size of code is smaller
- PDU’s@Ericsson: Huge code sizes
Scaling of the Product Owner (PO) Role

Total PO (TPO)

Requirement Area PO (APO)

Operative PO (OPO)

TPO Prioritizes between Requirement Areas
TPO defines no of XFTs per Requirement Area

APO prioritizes features in its Requirement Area

OPO prioritizes “Work Items” within its allocated features

Operative PO (OPO)

1

XFT

1

XFT

1

m

n

k

Total PO (TPO)

1

m

...
**XFT TEAM COMPETENCES**

System Area A
System Area B
System Area Z
Java
C++
Erlang
Integration
Test

What skills does and XFT need?

**Option 1:**
- Allow the teams to be completely generic.
- Any XFT can implement any feature.
- The XFT move freely through the system doing what is needed to implement a feature
- Specialization disappears.

**Option 2:**
- XFT can only operate within a system area zone.
- Specialization keep within the active area.
AGILE AND LEAN ARCHITECTURES

› System and software architectures tend to damage when subjected to long term agile XFT's

› Technical debt tends to accumulate.

› Product Guardian role tends to be be weak in the face of a [T,A,O]PO pushing feature velocity.

› Software and system architectures need to be specifically designed with agile in mind.

› Less tight cohesion which tend to be common in embedded systems.

› More “micro-service” based architectures that allow component decoupling.
KEY FACTOR: CONTINUOUS INTEGRATION

Before CI

- System Design
- SW Design
- Test

With CI

- Continuous Integration
- Early Phase
- Continuous Legacy Test
  - Hourly cycle
  - Daily cycle
  - Weekly cycle
  - Sprint cycle

- Product Development by XFTs

Release

- Quality
  - Slow feedback
- Quality
  - Fast feedback & constantly high quality

Constantly high quality
AGILE: KEY POSITIVE ASPECTS

› Some groups show significant productivity (37%) and quality increase. Measured by McKinsey.

› Very strong XFT dynamics. Teams form very tight social groups.

› Much better relationship between requirements source and implementation of these.

› Very strong test focus. XFT’s tend to use a test-driven development approach.

› Empowerment: Teams seem to really enjoy the feeling of being in charge of their backlog.
Breaking teams up and re-deploying members can be difficult.

Agile in the large can have a lot of management like roles. And few “doers”.

Some areas show a decrease in productivity and feature velocity.

Long term impact on architecture and technical debt.

Competence and specialized skill erosion.

We might have gone a bit too far with “the code is the documentation” concept.

Competence and specialized skill erosion.

Some areas show a decrease in productivity and feature velocity.

We might have gone a bit too far with “the code is the documentation” concept.
Design Tools to Aid in Large Scale System Development
Many views to capture various system aspects.
- Requirements
- System Architecture
- Design Implementation
- System Deployment
- Detailed Algorithms
- Test and Verification

Many flavors suited to specialized systems domains
MODEL DRIVEN DEVELOPMENT

› We build complex systems with lots of dependencies.

› Complexity has a technological aspects and a human aspect.

› MDD is a tool to help fight this complexity
MODEL BASED SYSTEM ENGINEERING – WUM (WCDMA UNIFIED MODELLING)

MDD systems approach using a process called “MBSE” (Model Based System Engineering).

Move from highly document driven process to repository “single source of information” approach.

Objective: Significant reduction in specifications faults.

UML-RT based with UML2 + SysML like extensions.
**Objective:** Battle design implementation complexity using a subset of UML called UML-RT.

Capsule and state-machine abstractions provide very powerful abstractions for dealing with reactive embedded systems.

C++ action language. Run-time system abstraction to hide the underlying platform.
Research Challenges in Large Scale Software Development
CHALLENGE #1: AUTONOMOUS CONFIGURATION OF SYSTEMS

- Autonomous communication systems
- Deriving optimal operation based on massive data awareness.

- Challenge:
  - New architectures
  - Modelling these systems.
  - Understanding emergent behaviour.
Much focus on new data driven architectures.

New machine learning technologies make unravelling complex relationships in data feasible.

This leads to new insights and possibilities.

Challenge: How can we use model driven approaches to empower data driven architectures.