MANAGING ARCHITECTURAL TECHNICAL DEBT - PART 1
Software Center Project, Agile and Architecture

Background
Who is Antonio Martini?

- Italian
  - No kebab pizza! 😊
  - 4 years in Sweden – survived many winters!

- Bachelor in Computer Science
- Master in Software Engineering
- Previous work
  - Back-end development
  - GUI development
  - Contact with the customer (“PO”)
- PhD Licentiate in 2013
- Now PhD Candidate in Software Engineering
  - Finishing my PhD in 1 and ½ months

- Hobbies
  - Board games, strategy computer games, pool, etc.
  - Football, volleyball, beach volley, fencing
  - Piano, Drumset, etc.
  - Travel!
A Software Center Project (1)

Current participants from industry

- AXIS
- GRUNDFOS
- JEPPESEN
- ERICSSON
- VOLVO
- SAAB
A Software Center Project (2)

Current research participants
Chalmers University of Technology
Gothenburg University

- **Antonio Martini**
  - Project Leader
  - antonio.martini@chalmers.se

- **Jan Bosch**
  - jan@janbosch.com
Agile and fast delivery to the customer...

Teams focused on delivering business value

FT = Feature Team

Antonio Martini - PhD Candidate in Software Engineering
...need an architecture "runway"

Agile teams need to be supported by an *architecture runway*

Antonio Martini - PhD Candidate in Software Engineering
Agile and Architecture Runway

**Agile**

- Stakeholder orientation
- Responsiveness
- Frequent deliveries
- Light-weight communication

**Architecture Runway**

- Structure
- Infrastructure
- Tooling
- Automation
- Education
Pre-Agile problem: too much architecture runway

No delivered value!
But what happens with not enough architecture runway?
What is Architectural Technical Debt?

**Architectural Technical Debt**
Horror Story

- Technical debt and Architecture
Optimal architectural decision

Example:
- Standard public API

Let’s put a standard API here… so later we can update the component independently.
During feature development...

No problem, let’s add a component B. The teams will use the standard API!

We need these new features! Our competitor is already delivering them!
...with fast delivery comes...

- Deliver fast!

- We need these new features! Our competitor is already delivering them!

- We have to deliver fast, let’s use the private API… we’ll change it later.

- Fast!

Antonio Martini - PhD Candidate in Software Engineering
The violation is spreading to many components:

- Comp A
- Comp B
- Comp C
- Comp D
- Comp E

...the accumulation of sub-optimal decisions...

We have to deliver fast, let’s use the private API! We’ll change it later...
...until, one day...

- New requirement

- We need these new features! Our competitor is already delivering them!

- Ok, we can replace this component. The teams used the standard API!
...the development is not fast anymore...

- **Costly** to remove the violation and difficult to estimate the impact
...and a crisis starts.

So should we refactor or continuing with other features?

We have to refactor, but we need time...

Impossible to refactor now! We need to deliver the features!
Architecture Technical Debt (ATD)

- Inconsistencies (violations) represent the debt between:
  - Current code
  - Description
  - Desired Architecture

Antonio Martini - PhD Candidate in Software Engineering
So what is Technical Debt in this case?

- Non-allowed dependencies = The Debt
- Cost of refactoring dependencies = Principal
- Extra evolution cost
  - Replacing the component = Interest
- Increased delivery time
So what is Technical Debt in this case?

- Non-allowed dependencies = The Debt
- Cost of refactoring dependencies = Principal
- Extra evolution cost
  - Replacing the component
- Increased delivery time

Important
Problem: a growing interest

ATD interest cost

Time

Cycle 1 Cycle 2 .......................... Cycle n

Principal

Low interest
Problem: a growing interest

ATD interest cost

Cycle 1, Cycle 2, ..., Cycle n

Linear interest

Low interest

Principal
Problem: a **growing interest**

- **Low interest**
- **Linear interest**
- **Non-linear interest**

ATD interest cost vs. Time

Cycle 1 → Cycle 2 ➤ ………. ➤ Cycle n

Principal
Problem: a growing interest

What if it was the interest for the loan?
Problem: a growing interest*

Need to identify this interest early on!

* Martini, Bosch: “The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles,” in accepted for publication at WICSA 2015, Montreal, Canada.
So, what happens in the end?

- Research study in 7 organizations *
- The accumulation of Technical Debt leads to crises

We need to make TD visible

- Invisible accumulation of TD leads to crises

Antonio Martini - PhD Candidate in Software Engineering
Technical Debt is **unavoidable***

- **Main causes**
  - Business pressure
  - Fuzzy or changing requirements
  - Messy architecture/design
  - Lack of documentation
  - Human behavior
  - Technical debt

- **Many more…**

---


Antonio Martini - PhD Candidate in Software Engineering
Good News! Roles, teams and practices in CAFFEA

Improvements
CAFFEA Framework

- Practices
- Roles
- Teams
- Methods
- Tools

* Martini, Pareto, Bosch. *Towards introducing Agile Architecting in Large Companies: the CAFFEA framework*, 2015 – Accepted for publication at XP conference
Architect roles

- CAFFEA framework
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools

- We need some practices to be more and/or better handled:
  - Architecture Consistency
  - Architecture Prioritization
We need some practices to be more and/or better handled:

- Architecture Consistency
- Architecture Prioritization

**CAFFEA framework**
- Architecture Practices
- Architecture Roles
- Architecture Teams
- Methods
- Tools

*Antonio Martini - PhD Candidate in Software Engineering*
Architects and Teams

- CAFFEA framework
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools

Careful, we have Technical Debt!
Governance Team

- CAFFEIA framework
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools

Careful, we have Technical Debt!

What to do next? Refactoring or features?
Architects Team

- CAFFEA framework
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools

What to do next? Refactoring or features?

What do we really need to refactor?

Careful, we have Technical Debt!
ATD method

- **CAFFEA framework**
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools

- Method for ATD management
- Integrated in the process
ATD method

- CAFFEA framework
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools

Prototype to visualize Cost and interest of ATD
How much to allocate to ATD?

- We asked Product Owners and Architects*: 
  - ATD is important when prioritizing based on:
    - Lead Time
    - Risk
    - Maintenance Cost
  - How much?
    - 10-20% suggested resources allocated to ATD management

Need of Architecture Runway, danger of Technical Debt and Improvements

What to Take Away?
TD is dangerous and invisible!

- Invisible accumulation of TD leads to crises

Antonio Martini - PhD Candidate in Software Engineering
2. Prioritize Technical Debt!
3. Improvements are possible!

- **CAFFEA Framework**
  - Practices
  - Roles
  - Teams
  - Methods
  - Tools

- **Holistic approach**

* Developed in this Software Center project by Antonio Martini and Jan Bosch
Questions?

References:
- To know more about this project
- antonio.martini@chalmers.se
- jan.bosch@chalmers.se

Comments?