Antonio Martini – Chalmers University of Technology

DevLin 2015


# MANAGING ARCHITECTURAL TECHNICAL DEBT - PART 1



G.U.

CHALMERS

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!



G.U.

CHALMERS

# A Software Center Project (1)

## Current participants from industry










G.U.

CHALMERS

# 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



G.U.

CHALMERS

# Agile and fast delivery to the customer…

Teams focused on deliverying business value


FT = Feature Team



G.U.

CHALMERS

# ...need an architecture ”runway”

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

# 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?




G.U.

CHALMERS

What is Architectural Technical Debt?

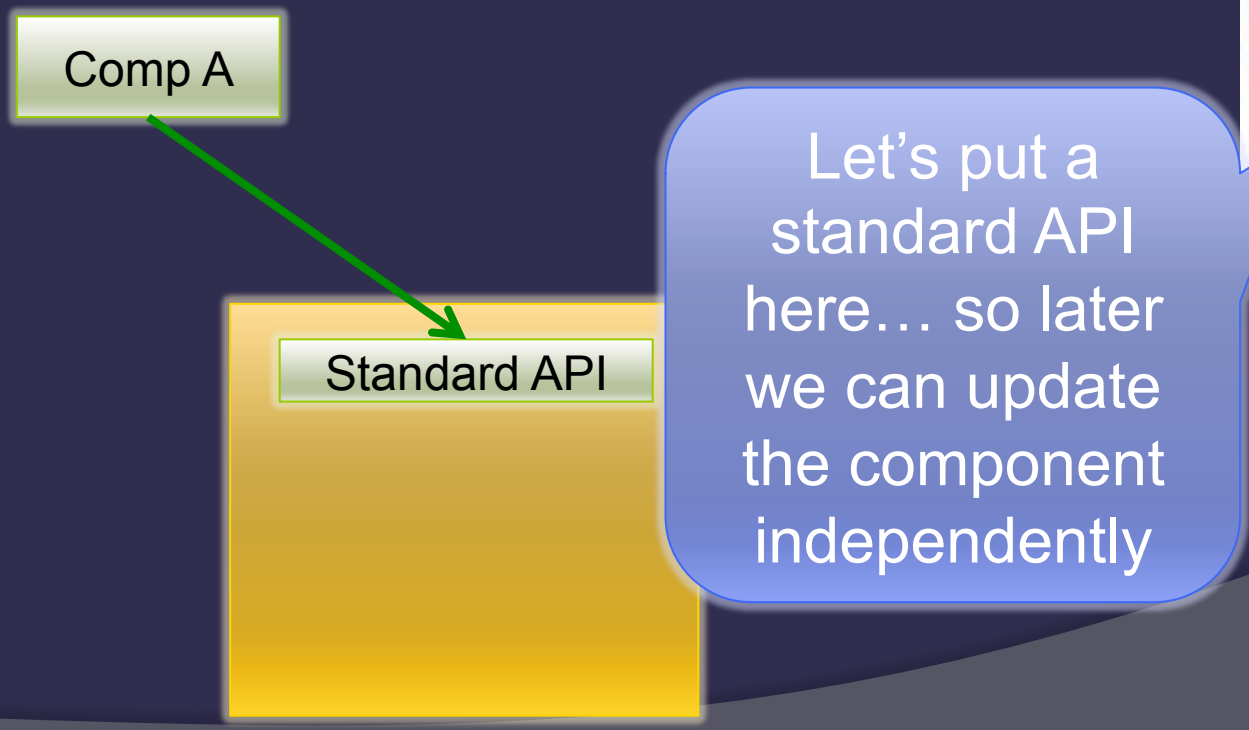
# Architectural Technical Debt


# Horror Story

- Technical debt and Architecture


# Optimal architectural decision

- Example:
  - Standard public API






G.U.

CHALMERS

# 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!

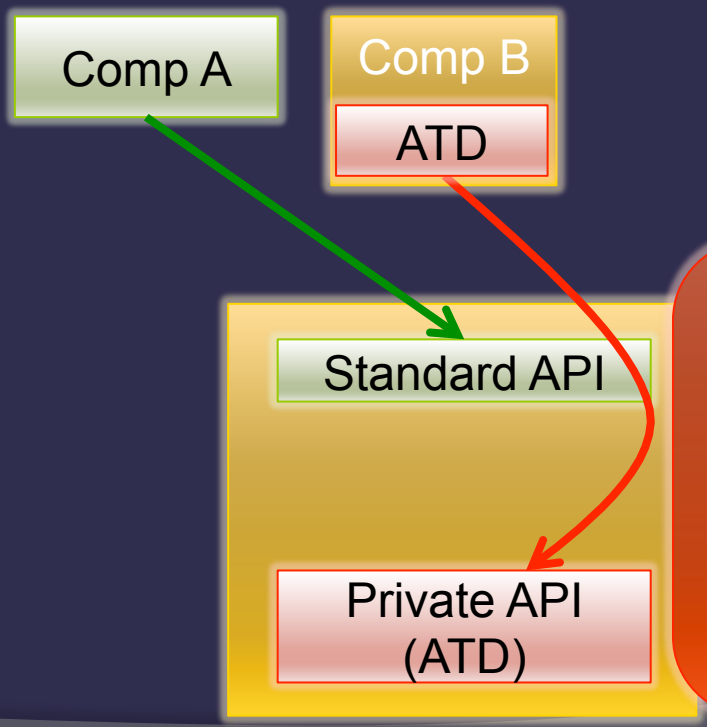
Comp A

Comp B

Standard API

# ...with fast delivery comes...

- Deliver fast!



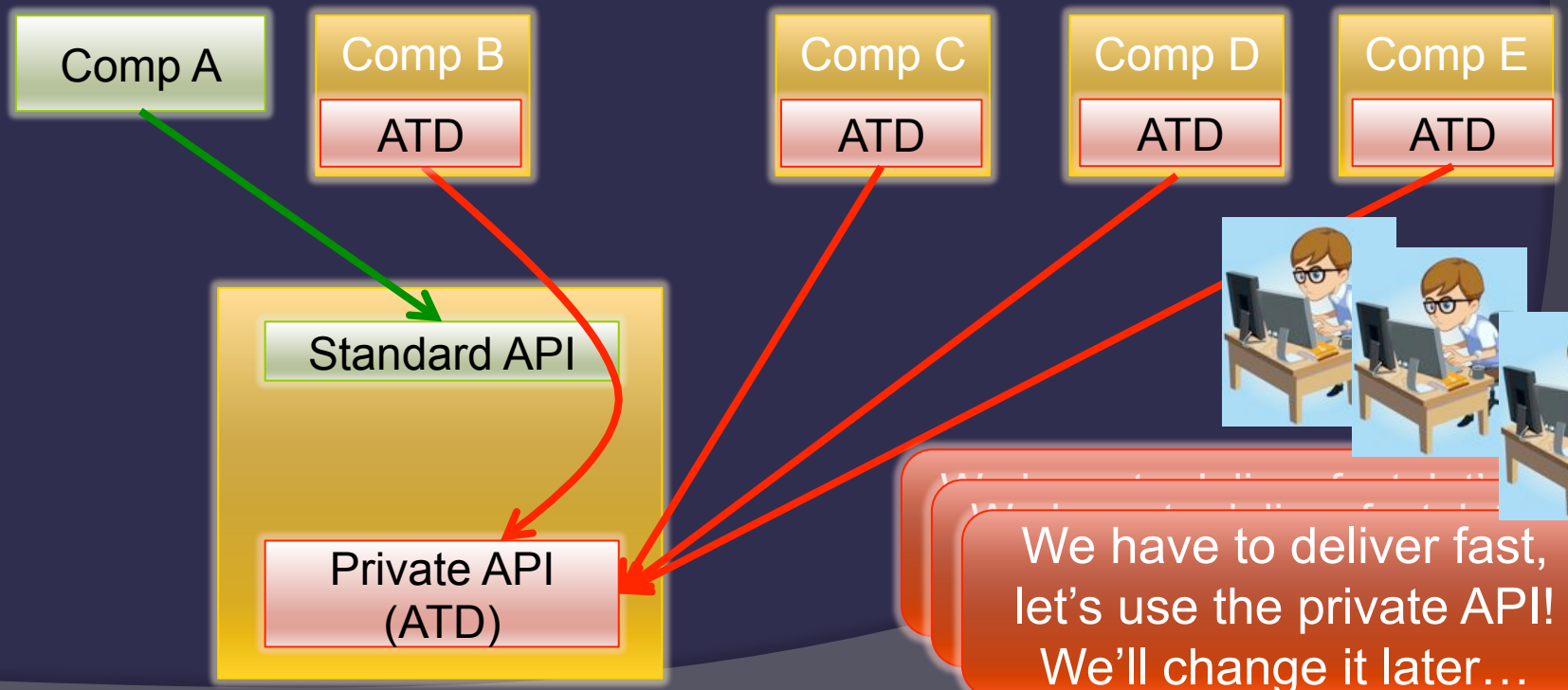

We have to deliver fast, let's use the private API… we'll change it later

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

Fast!

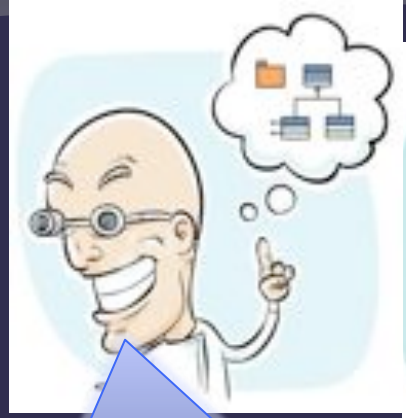
# ...the accumulation of sub-optimal decisions...

- The violation is spreading to many components

# ...until, one day...

- New requirement



Ok, we can replace this component. The teams used the standard API!

Comp A

Com

ATD

ATD

ATD

Com

AT

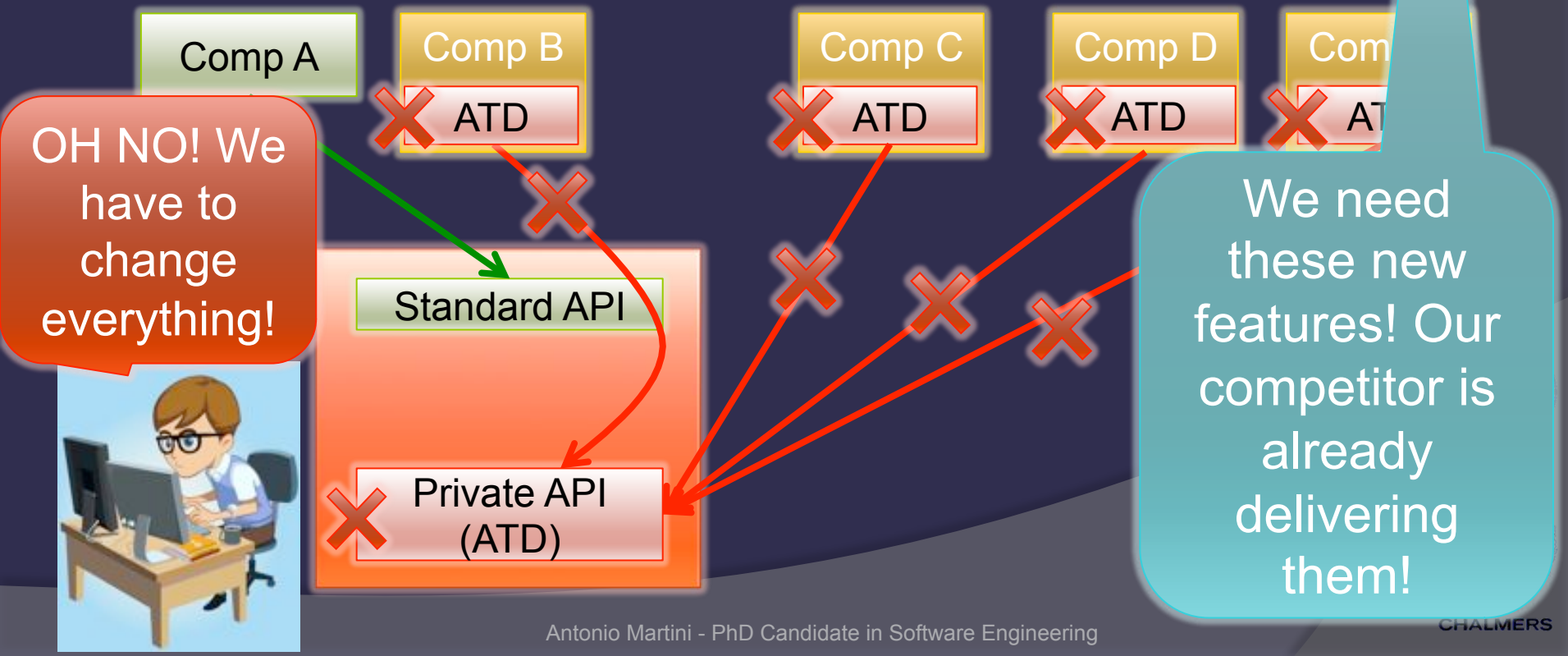
Standard API

Private API (ATD)

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

# ...the development is not fast anymore...

- *Costly* to remove the violation and *difficult to estimate the impact*



# ...and a crisis starts.

We have to refactor, but we need time…

So should we refactor or continuing with other features?

Comp B

ATD

Comp C

ATD

Comp D

ATD

Comp

AT

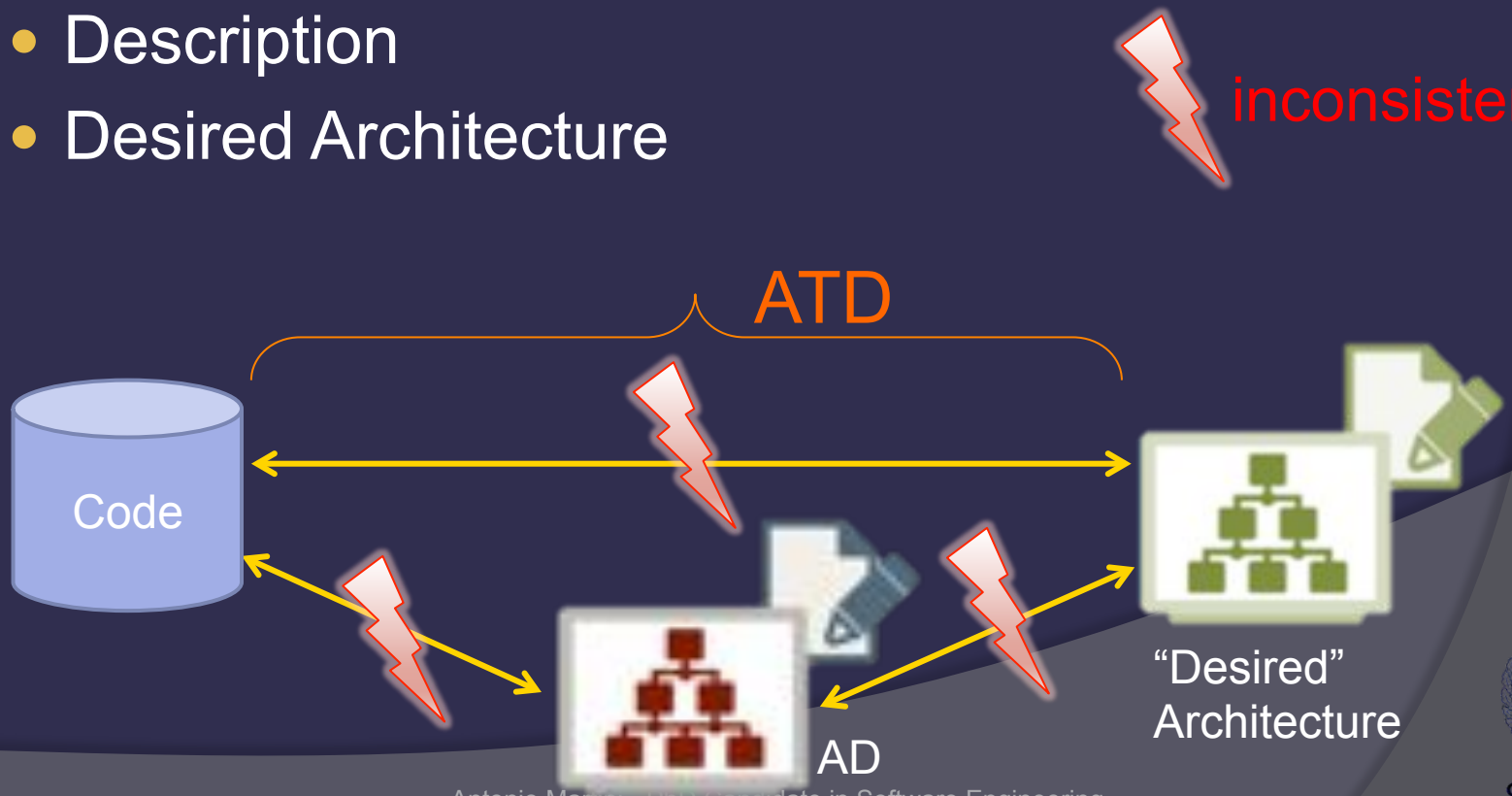
Standard API

Private API (ATD)

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






G.U.

CHALMERS

# 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

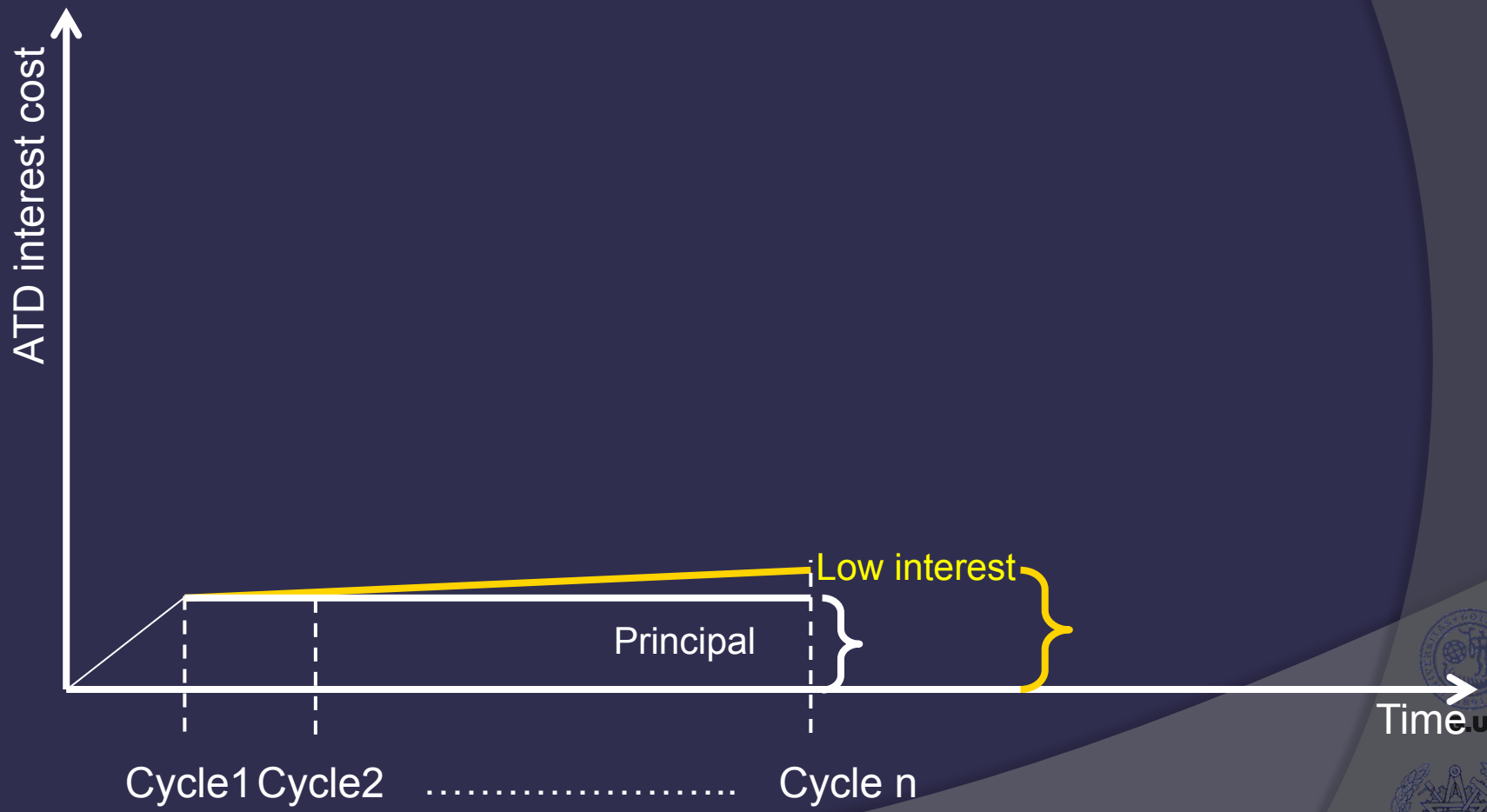
# So what is Technical Debt in this case?

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

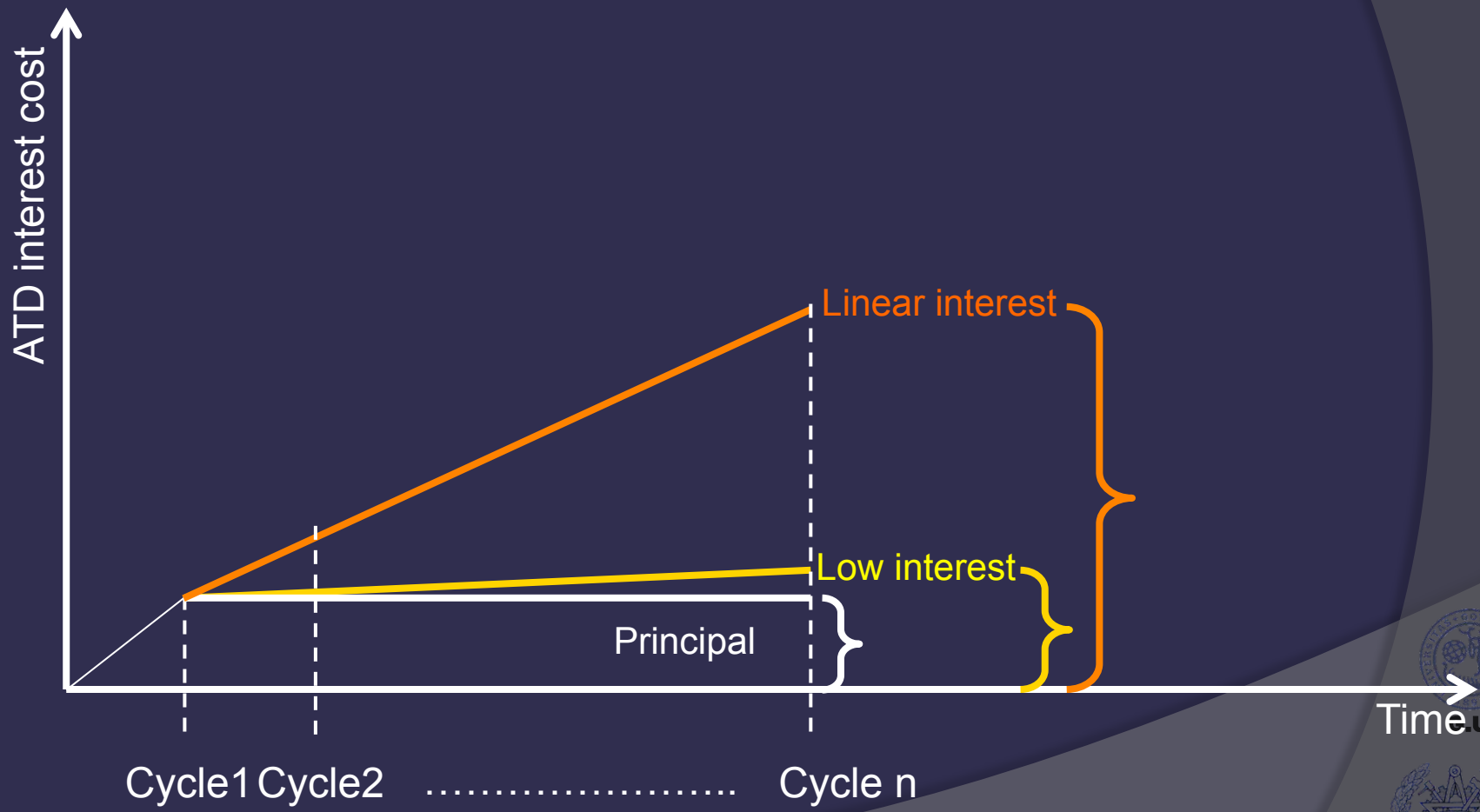
Important

- Increased delivery time

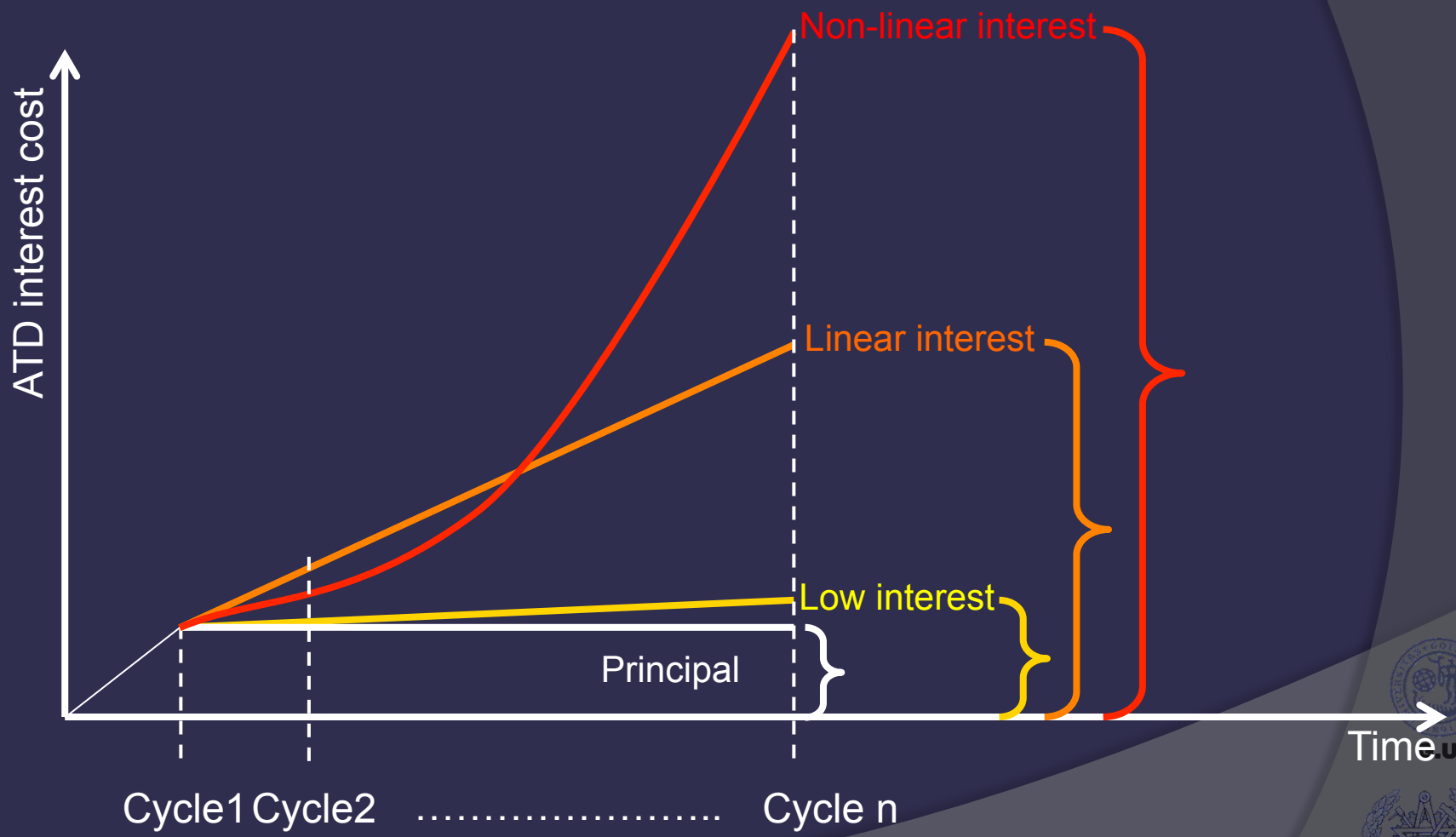
# Problem: a growing interest



# Problem: a growing interest



# Problem: a growing interest

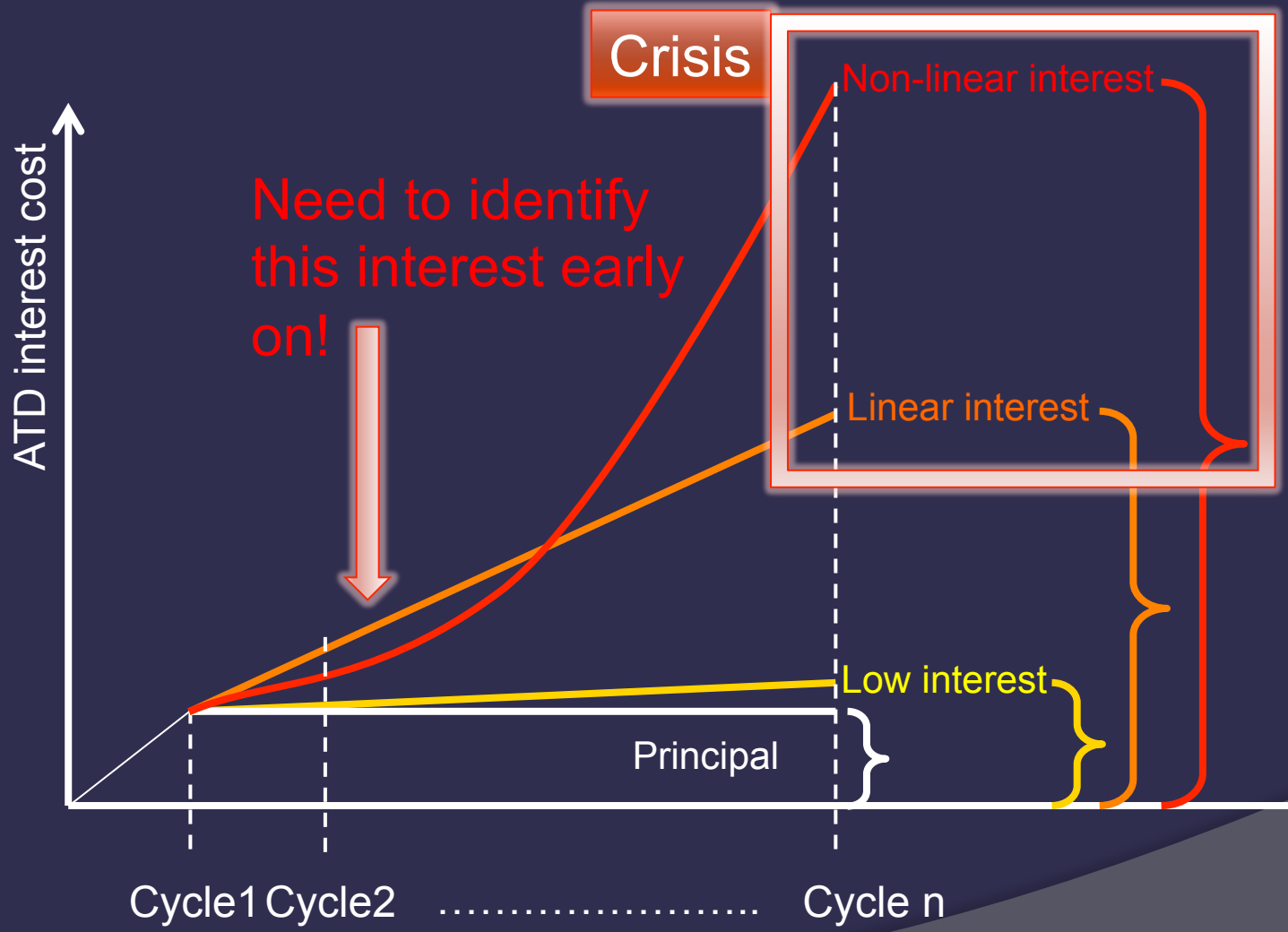


# Problem: a growing interest




What if it was the interest for the loan?

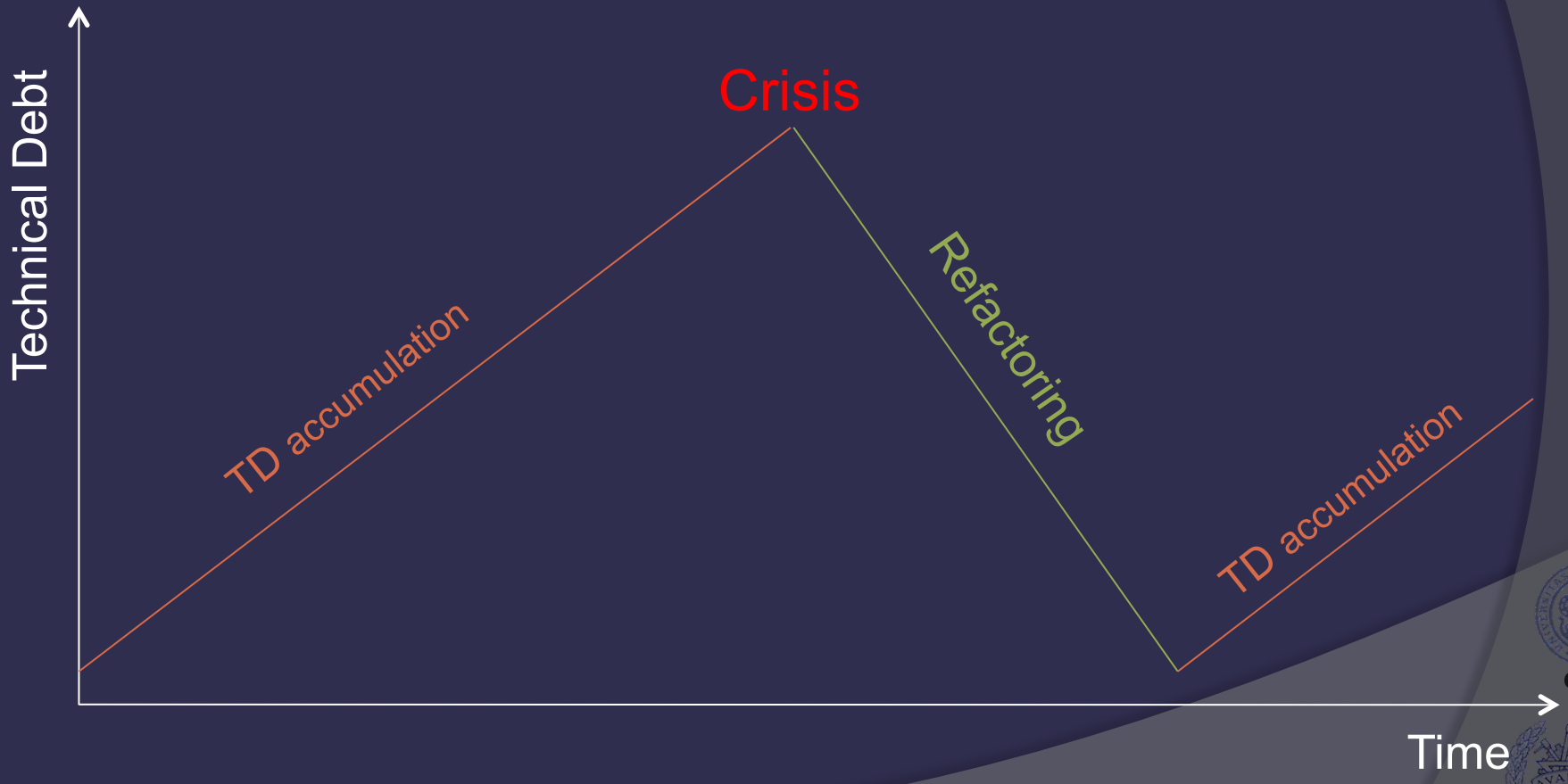
# Problem: a growing interest*




* 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?

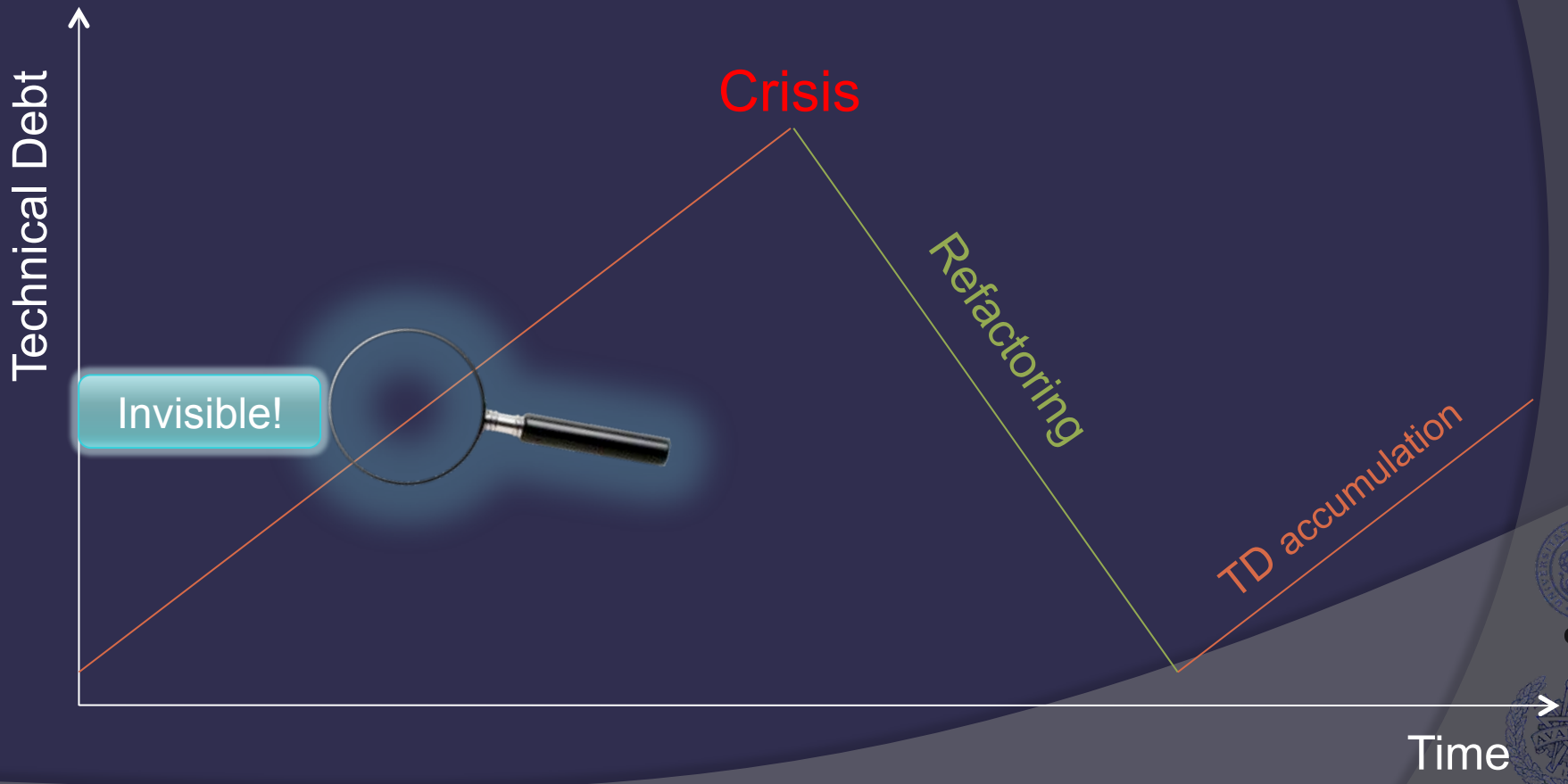
- Research study in 7 organizations *
- The accumulation of Technical Debt leads to crises




* Martini, A., Bosch, J., Chaudron, M., 2015. “Investigating Architectural Technical Debt Accumulation and Refactoring over Time: a Multiple-Case Study,” *Information and Software Technology*.

# We need to make TD visible

- Invisible accumulation of TD leads to crises






G.U.

CHALMERS

# Technical Debt is unavoidable*

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

* Martini, A., Bosch, J., Chaudron, M., 2015. “Investigating Architectural Technical Debt Accumulation and Refactoring over Time: a Multiple-Case Study,” *Information and Software Technology*.

Good News! Roles, teams and practices in CAFFEA

# Improvements


# Research Results*

## 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



G.U.

CHALMERS

# 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

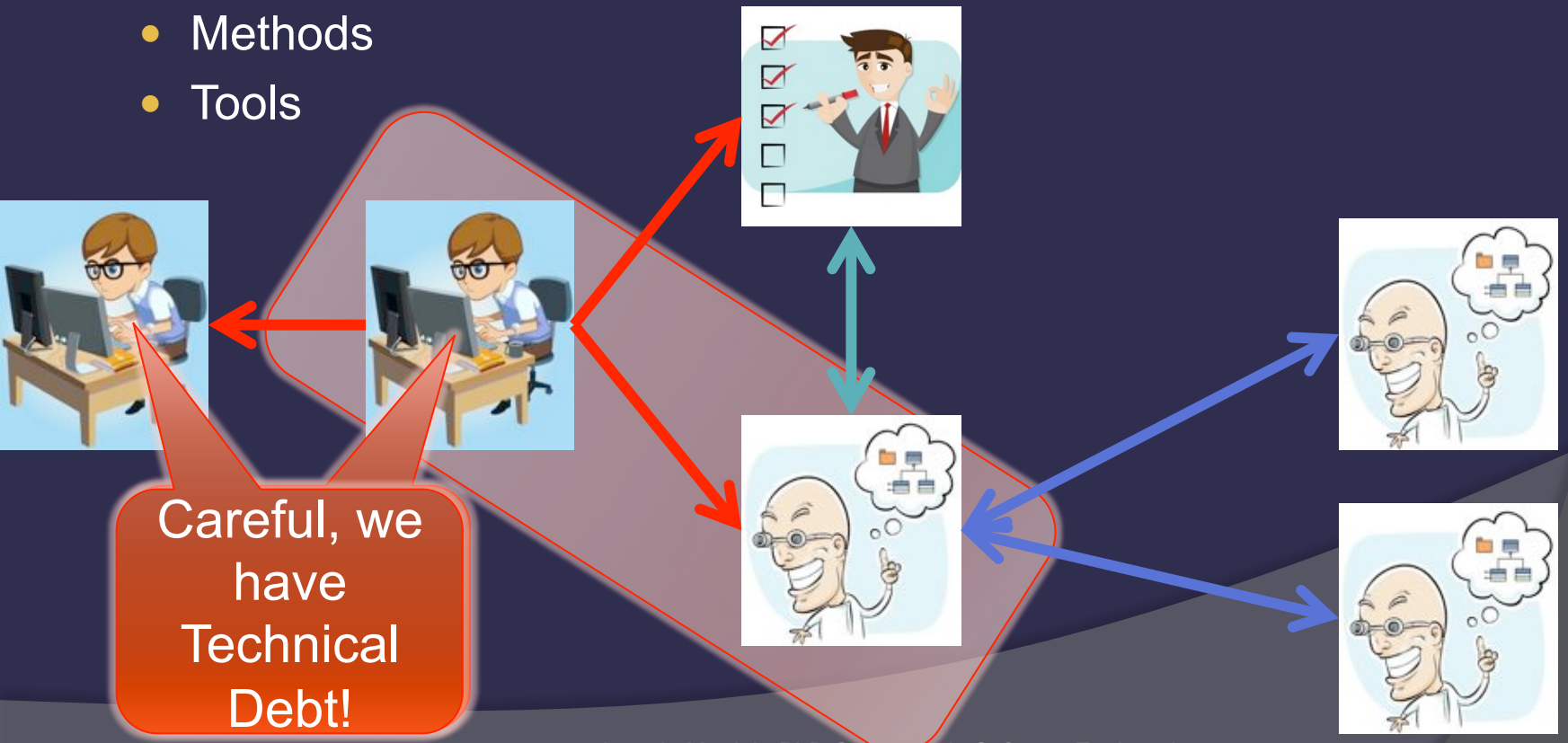
# Architect roles

- 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

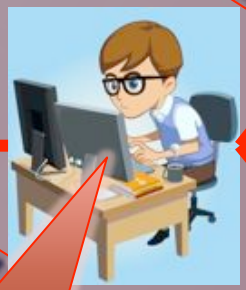
# Architects and Teams

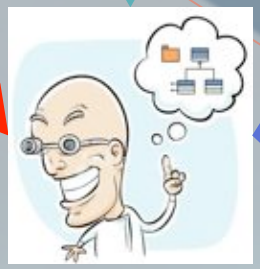
- CAFFEA framework
  - Architecture Practices
  - Architecture Roles
  - Architecture Teams
  - Methods
  - Tools



# Governance Team

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



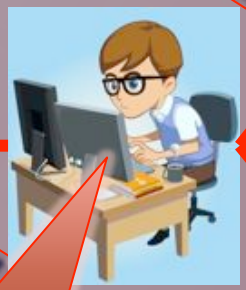


What to do next? Refactoring or features?

Careful, we have Technical Debt!

# Architects Team

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



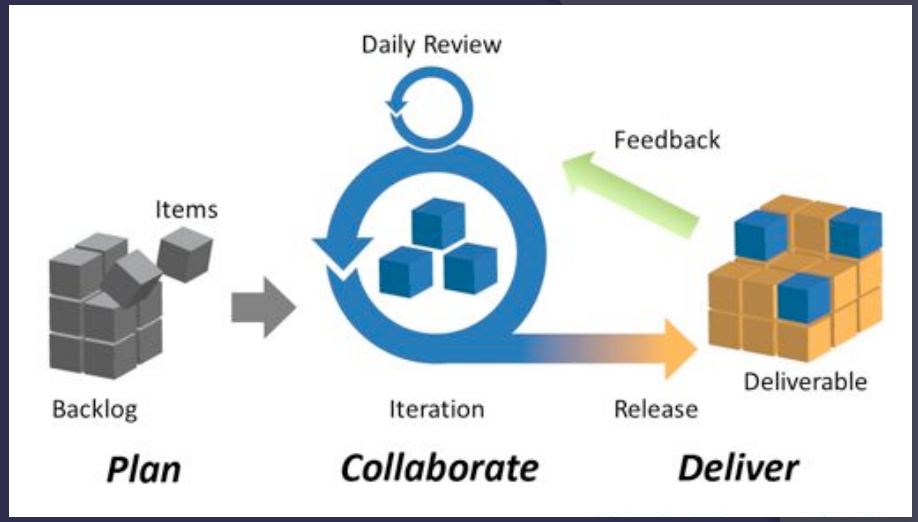
What to do next? Refactoring or features?

What do we really need to refactor?

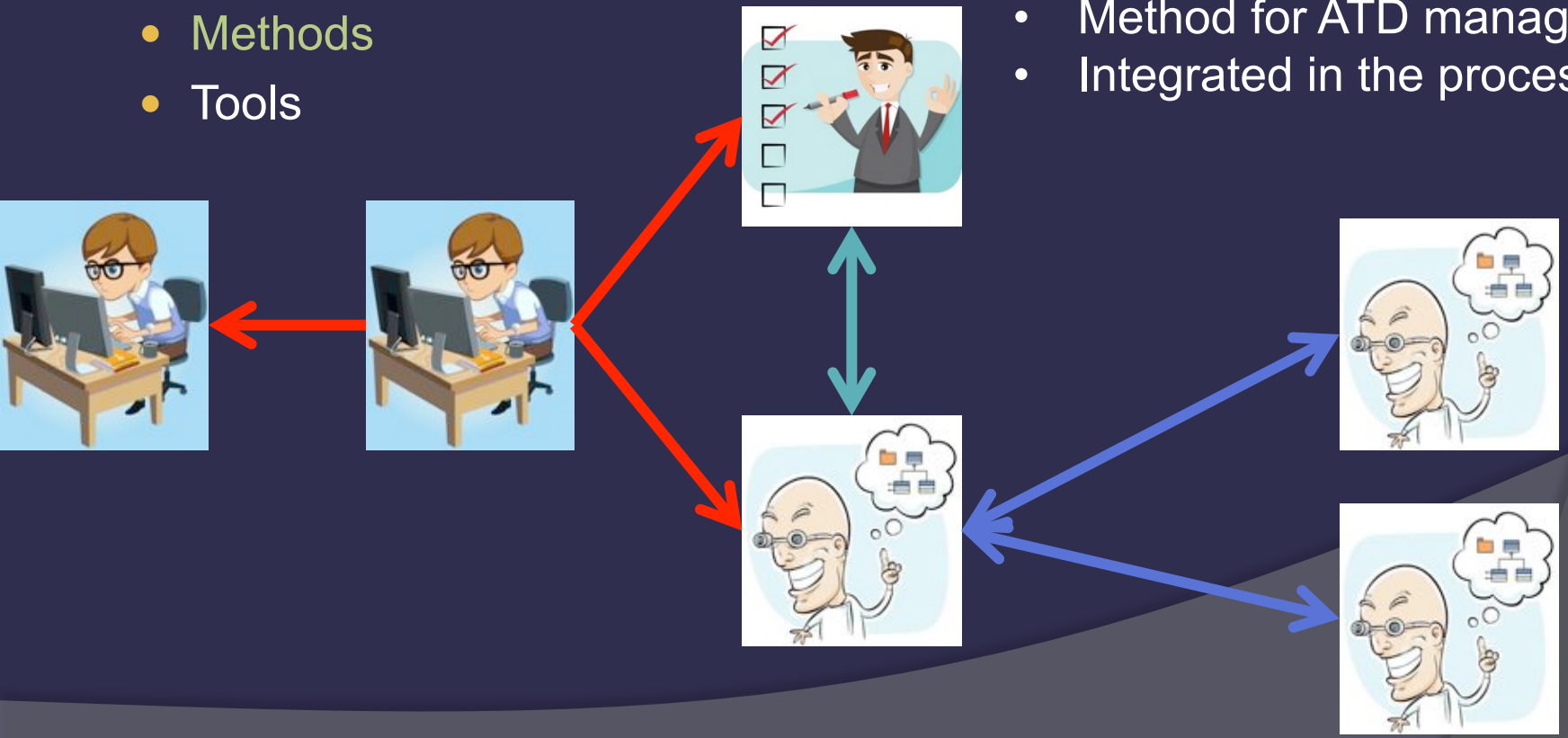
Careful, we have Technical Debt!

# ATD method

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




- Method for ATD management
- Integrated in the process





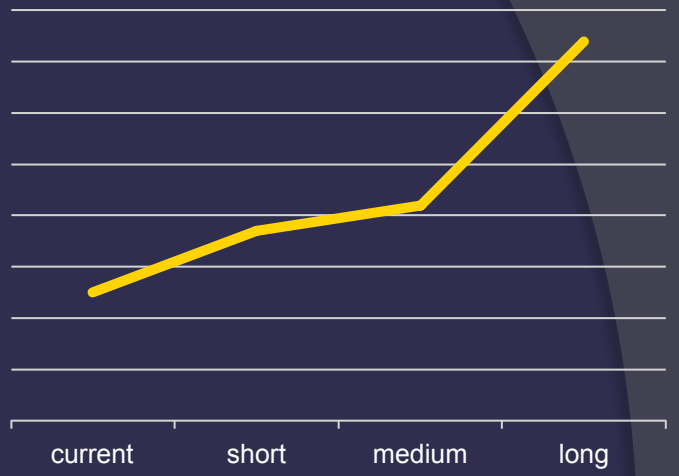
G.U.

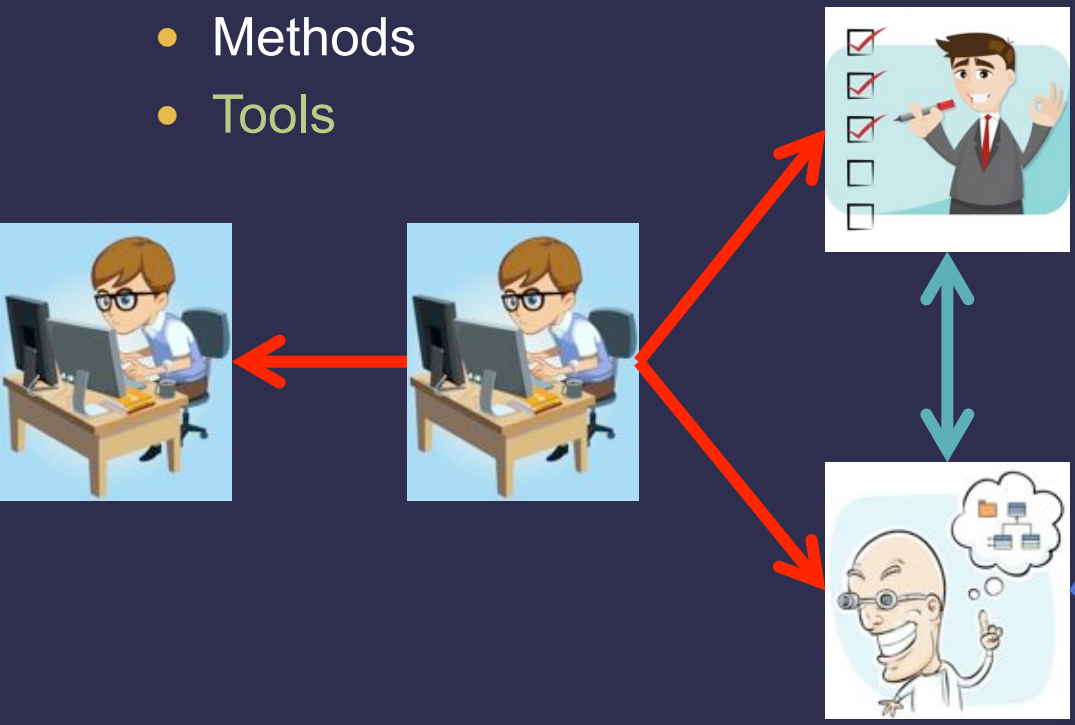
CHALMERS

# ATD method

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

Prototype to visualize Cost and interest of ATD





G.U.

CHALMERS

# 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

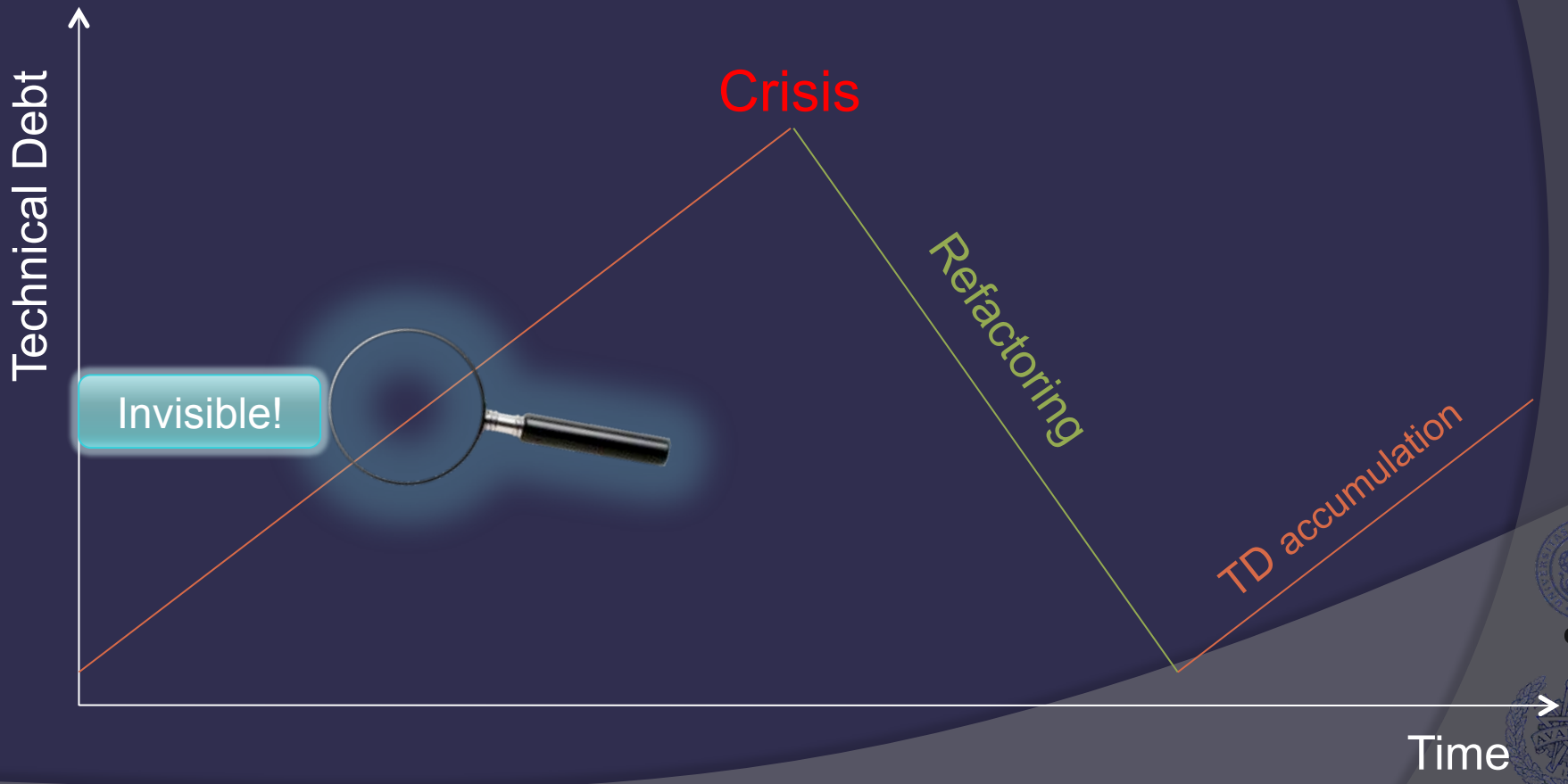
* Martini, Bosch – "Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners" SEAA 2015

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






G.U.

CHALMERS

# 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?

# Comments?

- References:
  - To know more about this project


  - antonio.martini@chalmers.se
  - jan.bosch@chalmers.se




G.U.

CHALMERS