Patterns of Co-Evolution: a Learning Perspective?

Riccardo Scandariato

Koen Yskout



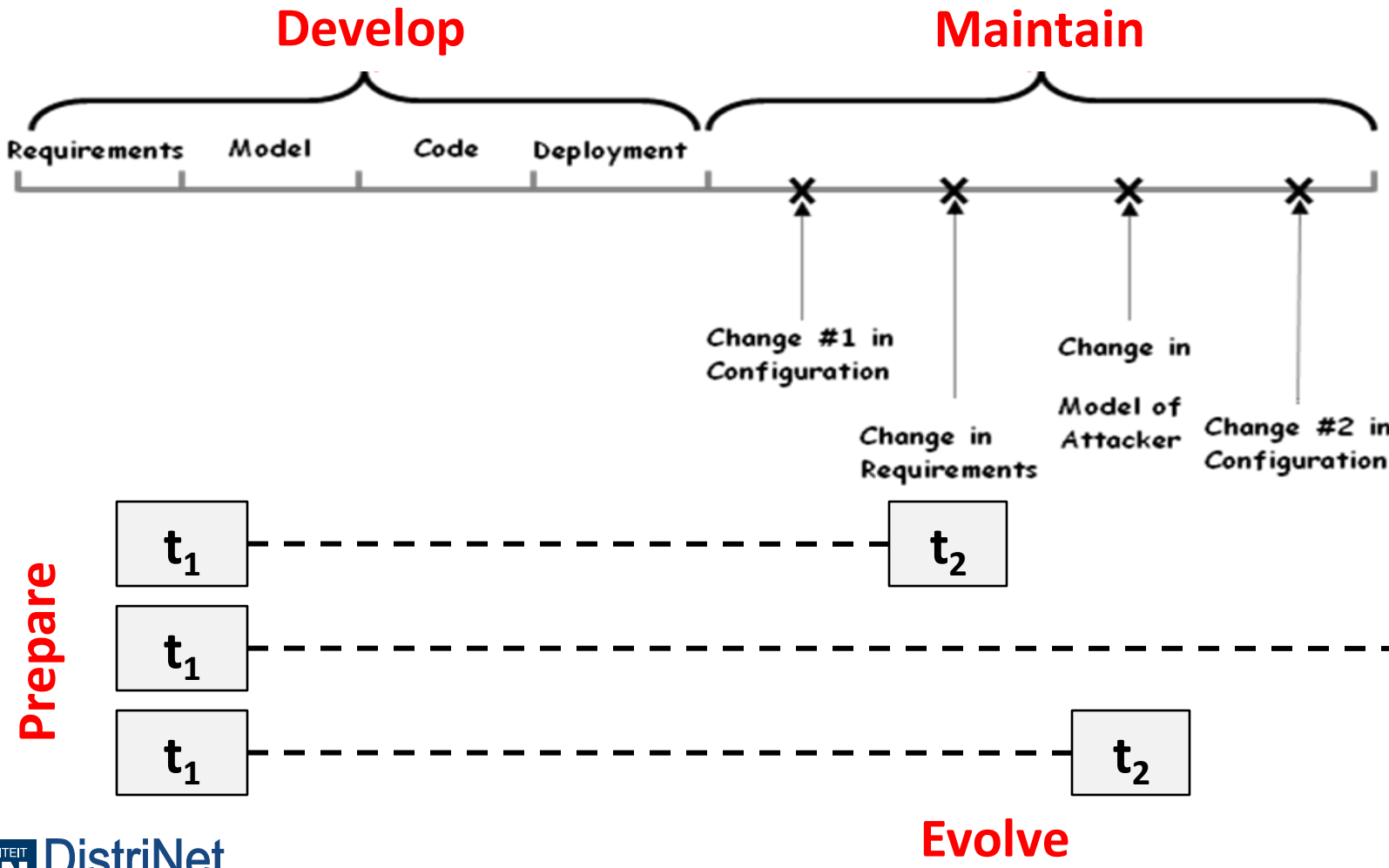



KATHOLIEKE UNIVERSITEIT LEUVEN DistriNet RESEARCH GROUP

Acknowledgement

This research is partially funded by the SecureChange EU FP7 project, the Interuniversity Attraction Poles Programme of the Belgian Science Policy, and by the Research Fund K.U.Leuven

Context

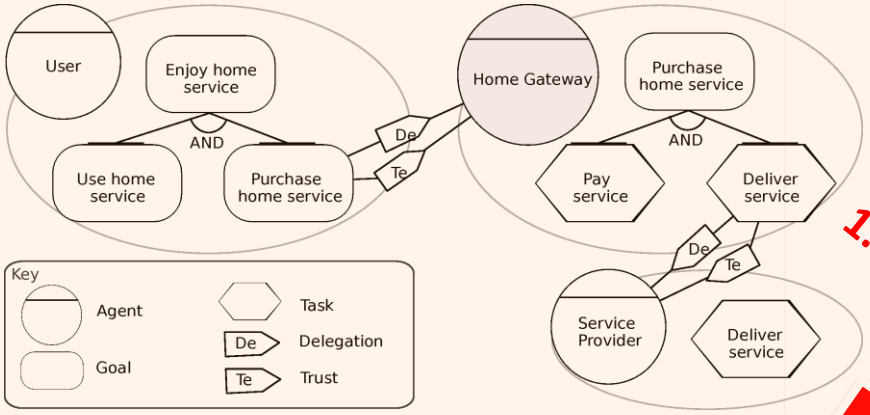

Develop
Maintain
Requirements
Model
Code
Deployment
Change #1 in Configuration
Change in Requirements
Change in Model of Attacker
Change #2 in Configuration
Prepare
t_1
t_1
t_1
t_2
t_2
Evolve

Idea: Change Patterns

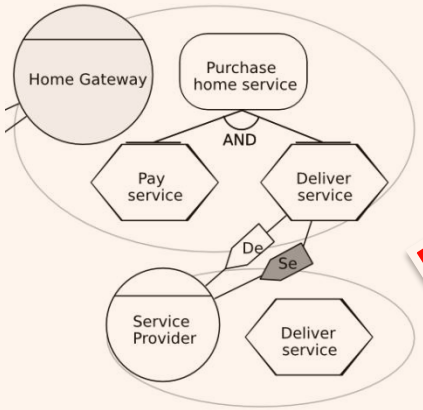
- Prepare a stable architecture at dev-time
  - Based on foreseen scenarios
  - Assuming limited development resources
  - According to a model-based approach
- Evolve when necessary
  - Avoid high-impact changes
  - Supported by automation
- Trade-off ‘now’ vs. ‘later’ (necessary vs. too much)

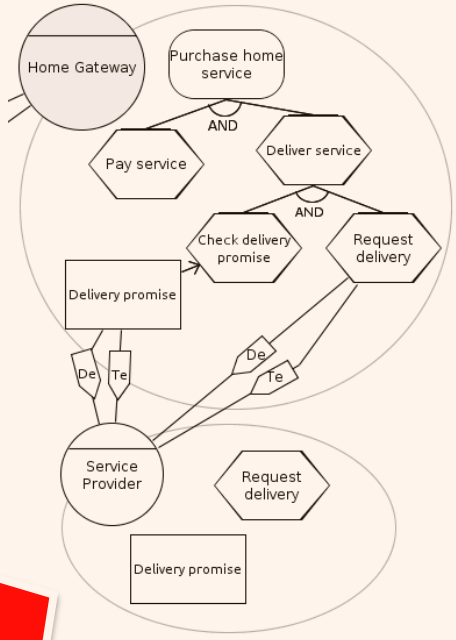
Outline

- Change Patterns explained
- Empirical validation
  - ➔ Has shown that CP bring value to the table
- Problem: do CP exists in nature?
  - ➔ Learning from “in vivo” experiment

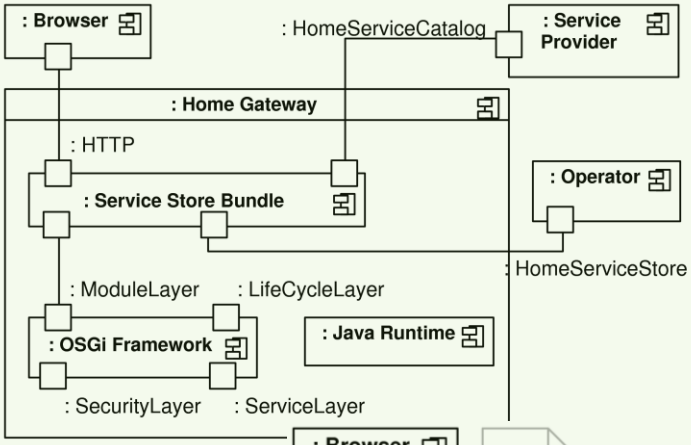


1. Scenario

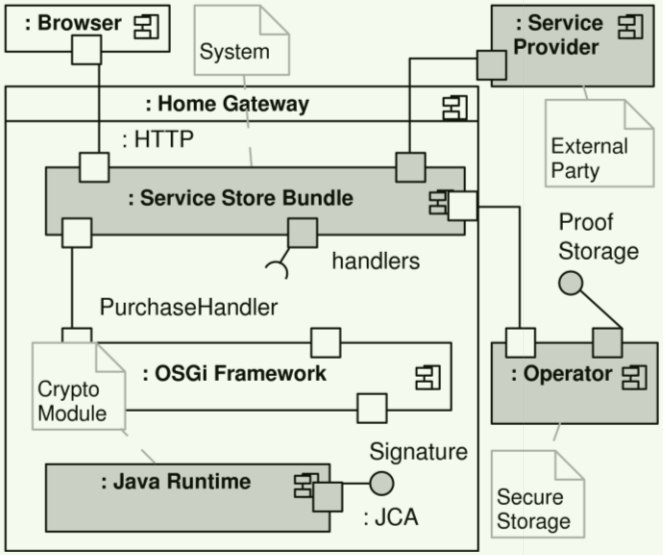




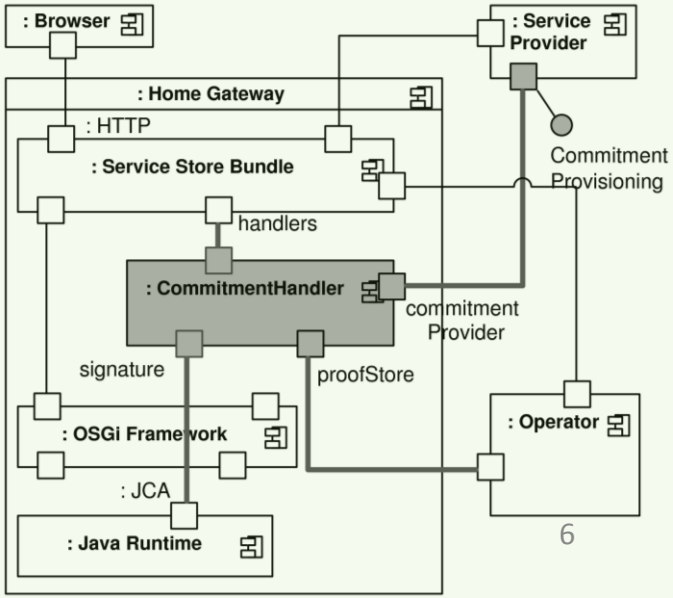
4. Feedback



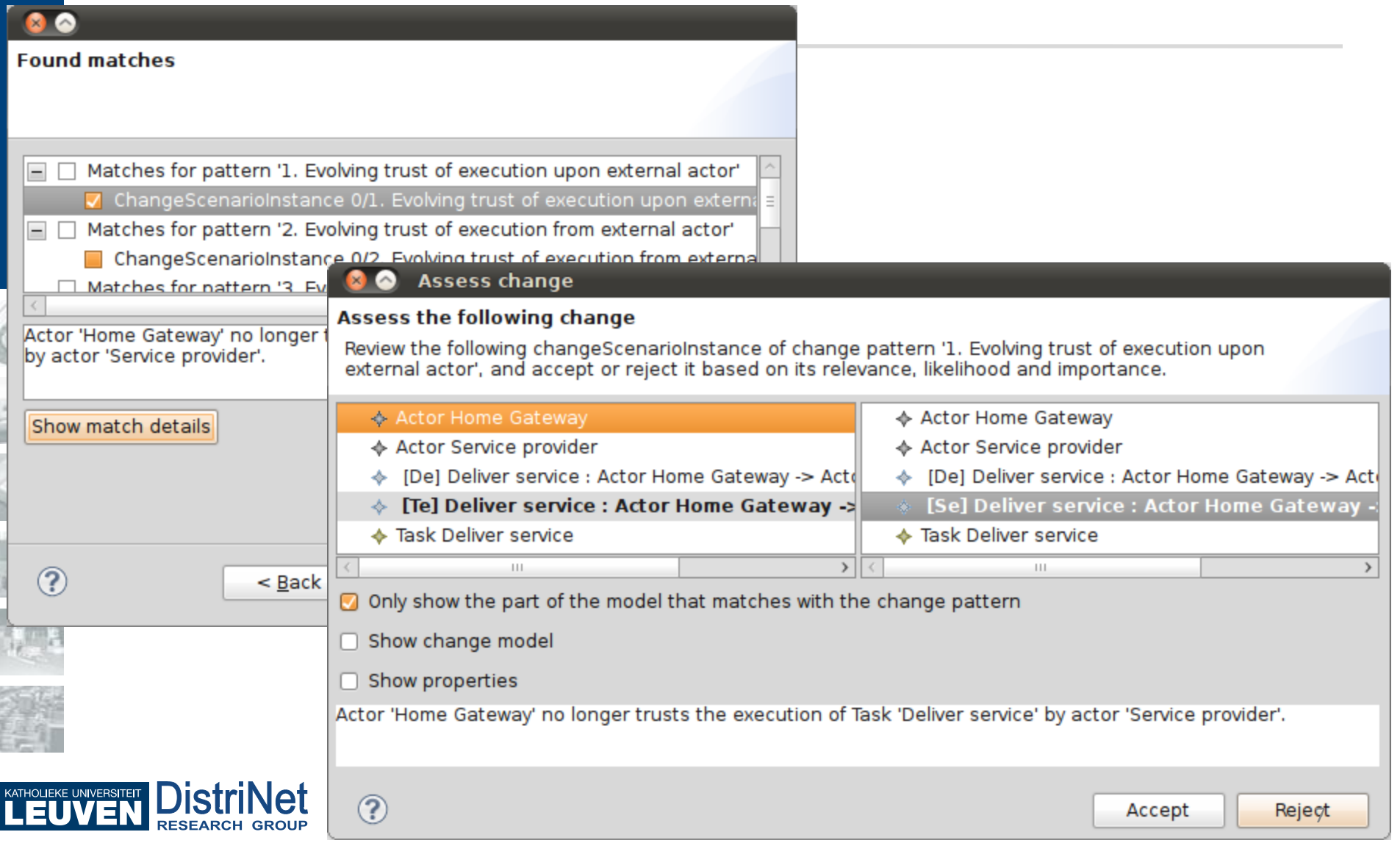
2. Preparation



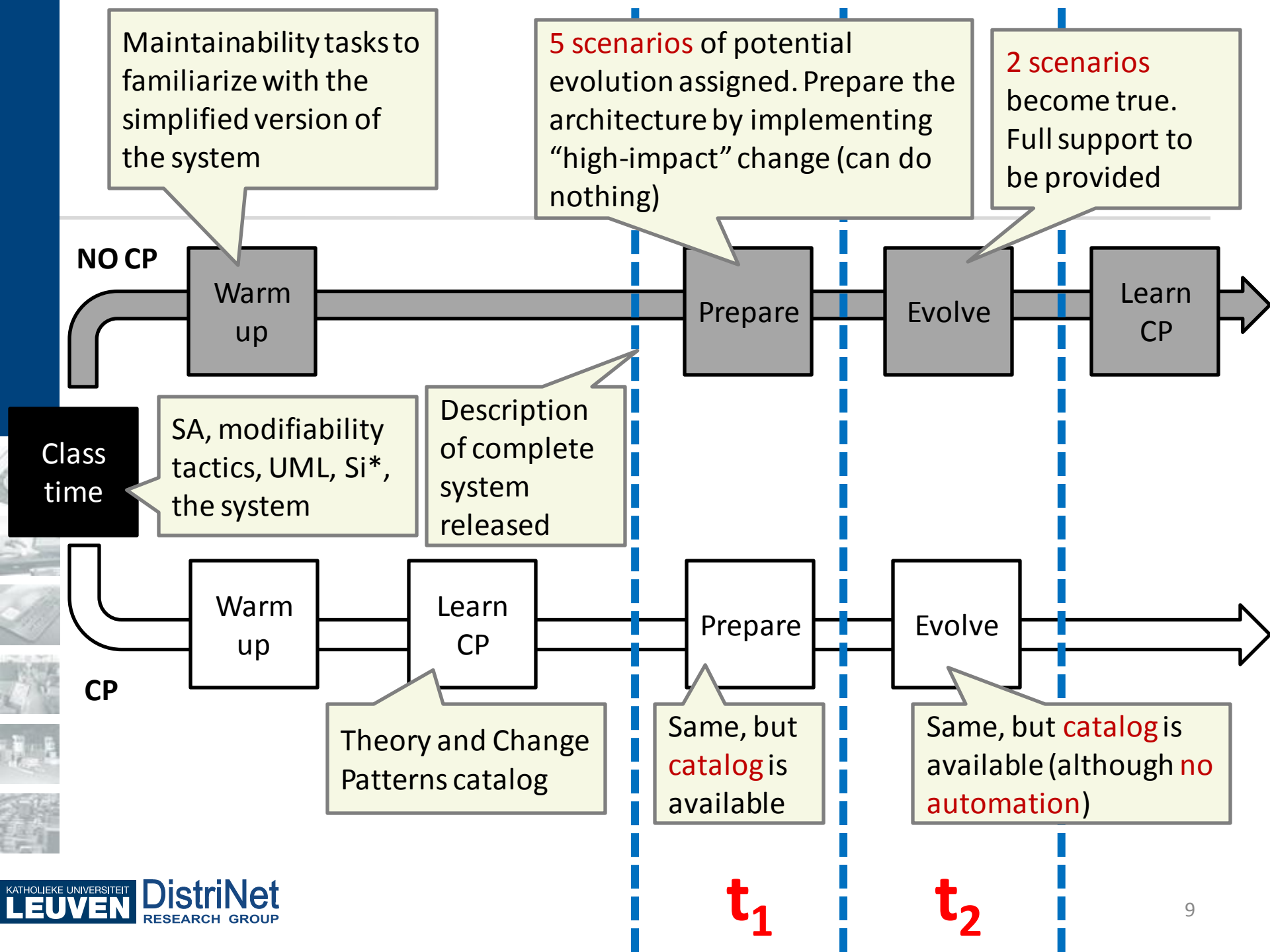
3. Evolution



Tool support (Eclipse plugin)



CONTROLLED EXPERIMENT


Maintainability tasks to familiarize with the simplified version of the system
5 scenarios of potential evolution assigned. Prepare the architecture by implementing "high-impact" change (can do nothing)
2 scenarios become true. Full support to be provided
NO CP
Warm up
Prepare
Evolve
Learn CP
Class time
SA, modifiability tactics, UML, Si*, the system
Description of complete system released
Warm up
Learn CP
Prepare
Evolve
CP
Theory and Change Patterns catalog
Same, but catalog is available
Same, but catalog is available (although no automation)
t1
t2

Test hypotheses

- H0' The average combined effort of preparing and evolving the architecture (μ_{t1+t2}) is the same in the control group (NO CP) and the treatment group (CP)

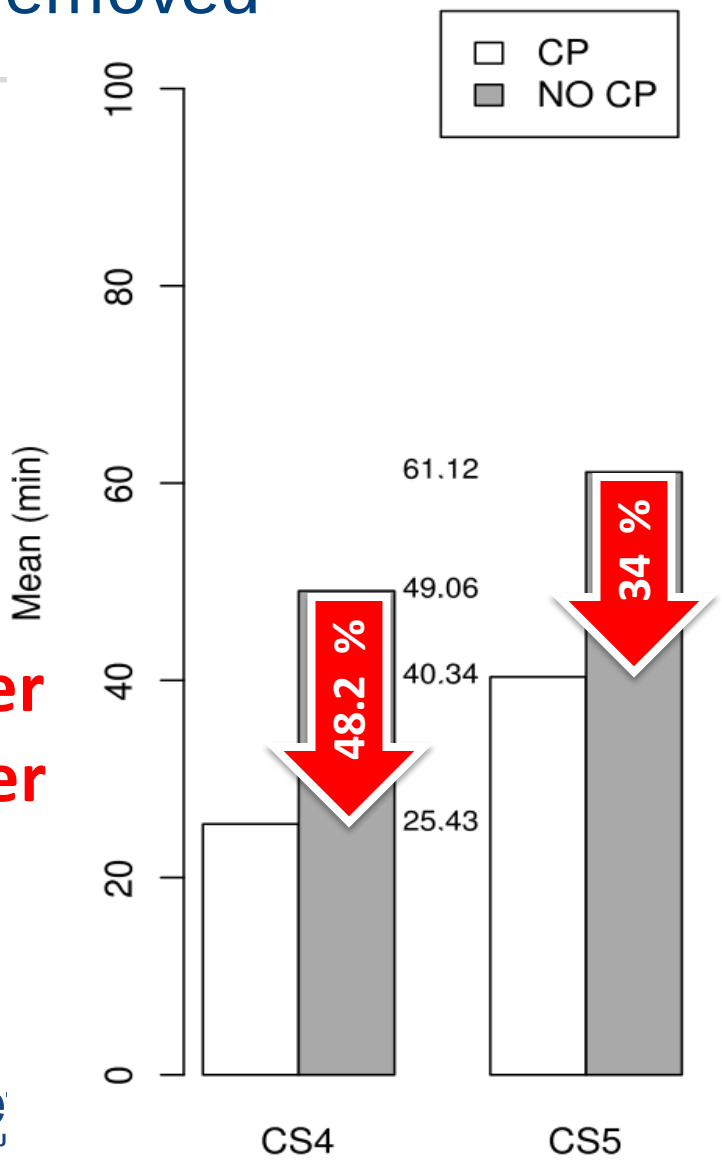
- H0'' The average effort to evolve the architecture (μ_{t2}) is the same in the control group (NO CP) and the treatment group (CP)

- Tool instrumented for activity monitoring
  - → The tool logs a timestamp whenever an editor is opened/closed

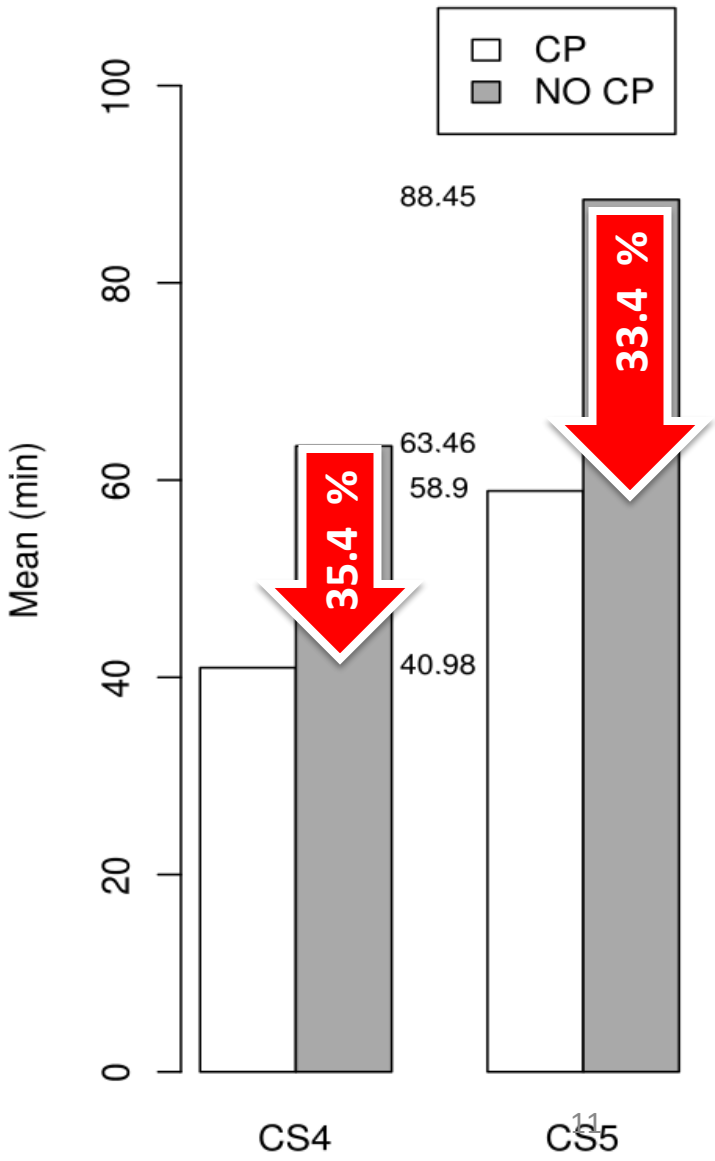
Results

Outliers removed

(effort to evolve)



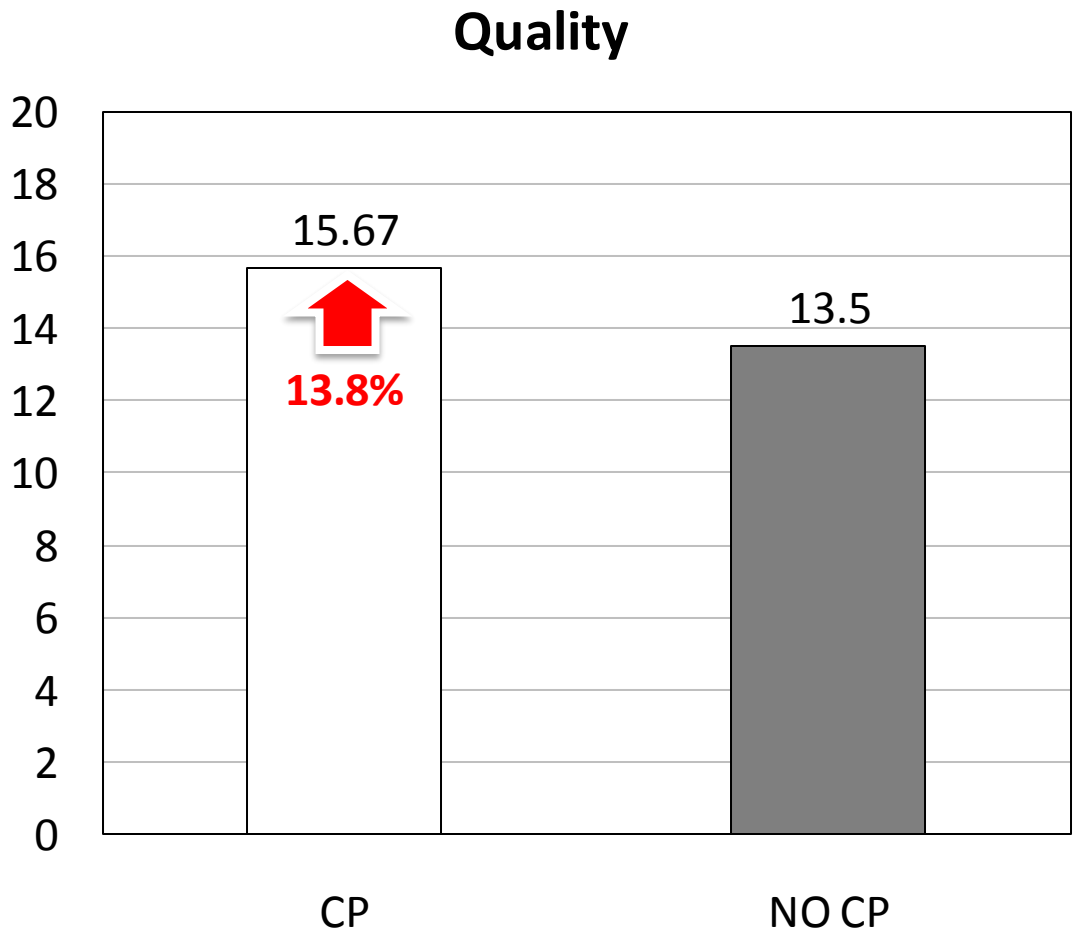
(combined effort)



The lower the better

Overall quality

The higher the better



LEARNING

Do CPs exist in nature?

- Patterns are nor invented, they are discovered
- Mining (open) software repositories not feasible
  - Documentation of SW architecture non-existing
  - Extraction is witchcraft (been there, done that!)
  - We need both reqs and archi deltas

- Alternative: put together a pool of designers and observe how they react to change requests, in a controlled experiment

Experiment design (tentative)

- Give a UML description of an architecture to N participants
- Assign a number of change requests (same to each)
- Collect the resulting artifacts (updated UML diagrams)
  - ➔ In a nutshell, the artifacts are graphs
- Cluster the artifacts according to a distance function (*similarity*)
- Label clusters (wrong, naïve/intuitive, pattern candidate)

- Some initial data already available (the control group of previous experiment)
  - ➔ Only 5 data points… good for a pilot?

Open questions

- What techniques for clustering? Ontology matching?
- Similarity function? Structure + lexicographic + semantic?
- What size of experiment do I need for the techniques to work?
  - → N=16 PhD students vs. N=100 Master students