softeng.fe.up.pt

UML Checker – A Toolset for Conformance Testing against UML Sequence Diagrams

https://blogs.fe.up.pt/sdbt/

João Pascoal Faria, FEUP/INESC TEC, jpf@fe.up.pt
(with Ana Paiva, Mário Castro, Zuhanli Yang, Tamara Krasnova, and Bruno Lima)
MAPI, 03 December 2014

Index

Motivation

- The development of detailed UML design models of software intensive systems for documentation only has several problems
  - is time consuming
  - the models are often wrong (no static analysis, compilation and testing)
  - the models soon become outdated and are not maintained
- MDD approaches aim at avoiding such problems by generating executable applications from models
- However, in many cases, the level of detail of the behavioral models needed to generate complete applications may be too high or only effective for specific domains

Approach: hybrid MDE (1)

- For situations where developing full behavioral models is not practical, we propose a lightweight approach:
  - continue to develop structural models from which parts of the application can be generated (e.g., class skeletons)
  - develop partial behavioral models, not sufficient for app generation, but adequate for test generation

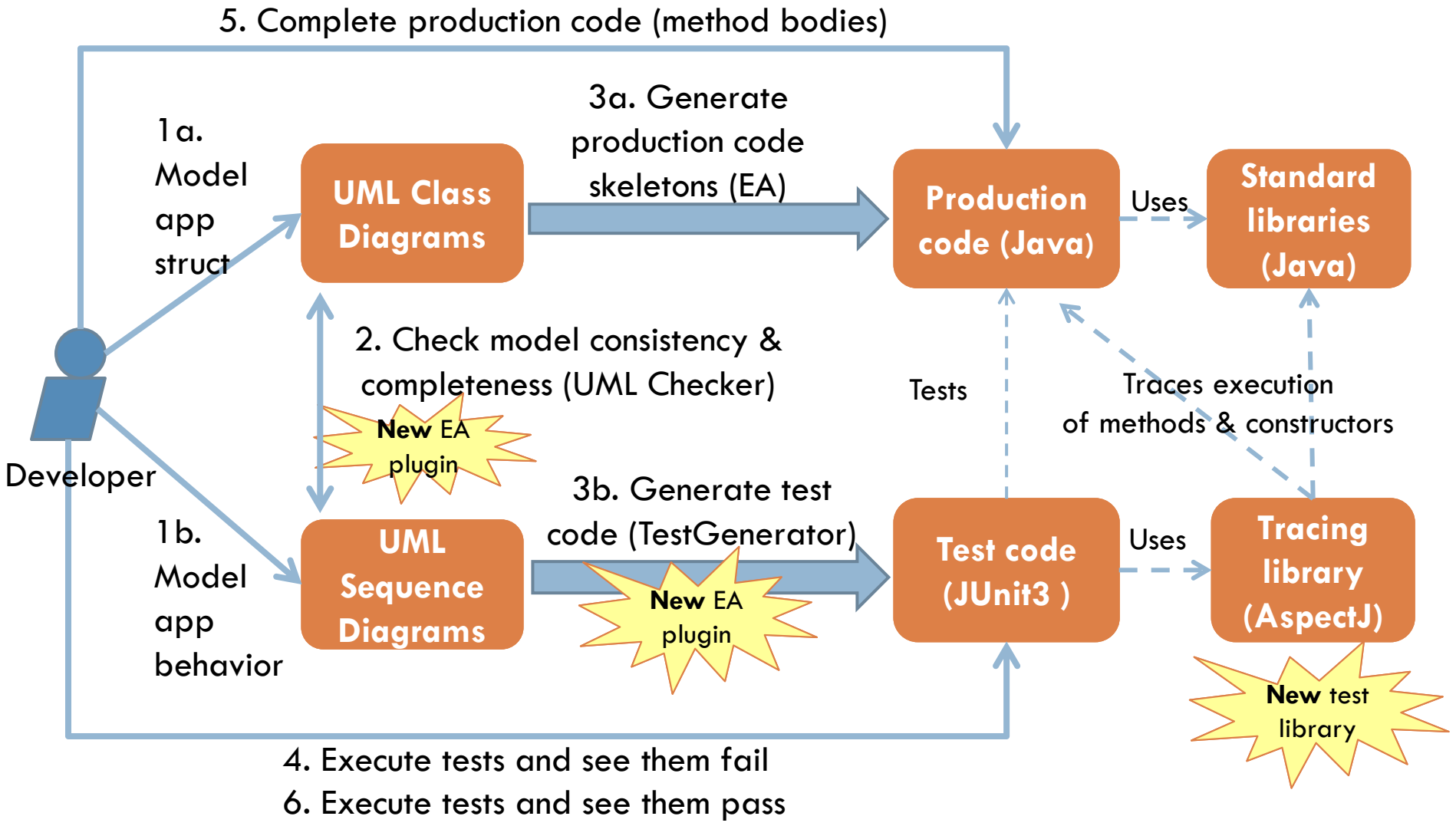
Partial behavior spec = Test spec

- This is also more in line with the agile values

(value more) *Working software over comprehensive documentation*

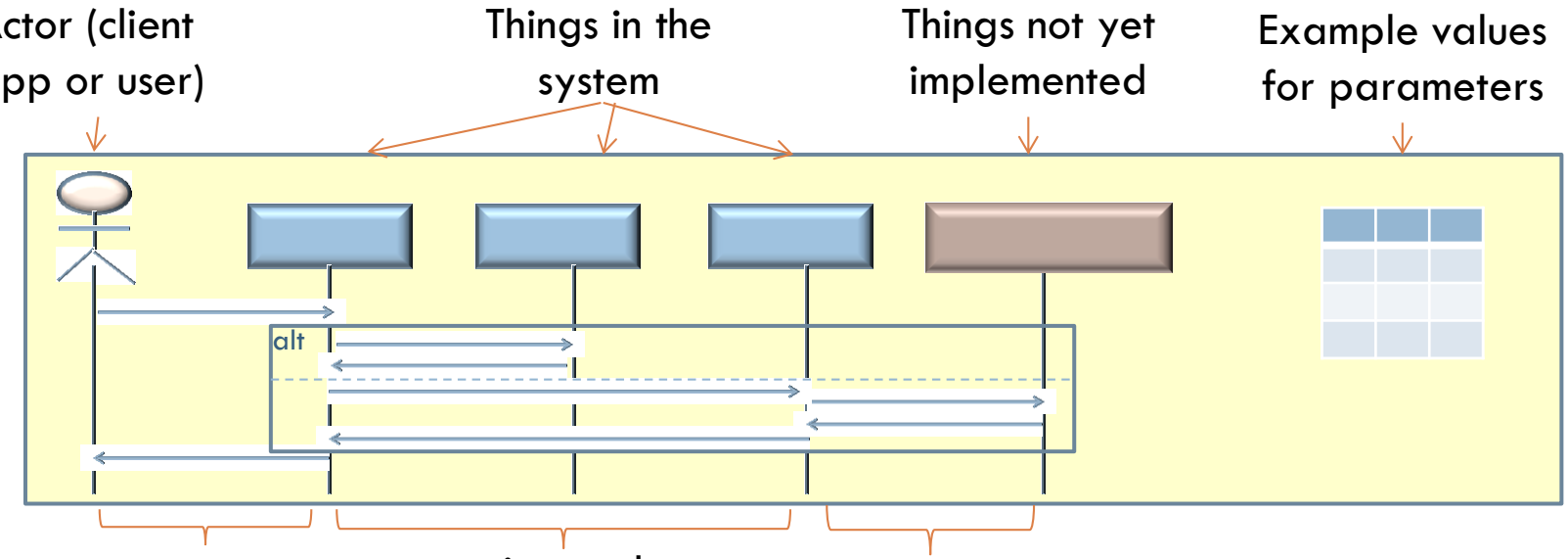
- To demonstrate the approach we developed a tool that generates executable tests from parameterized sequence diagrams acting also as specifications of test scenarios

Approach: hybrid MDE (2)



Test ready sequence diagrams

Behavioral Model/Spec



Generated Test Code

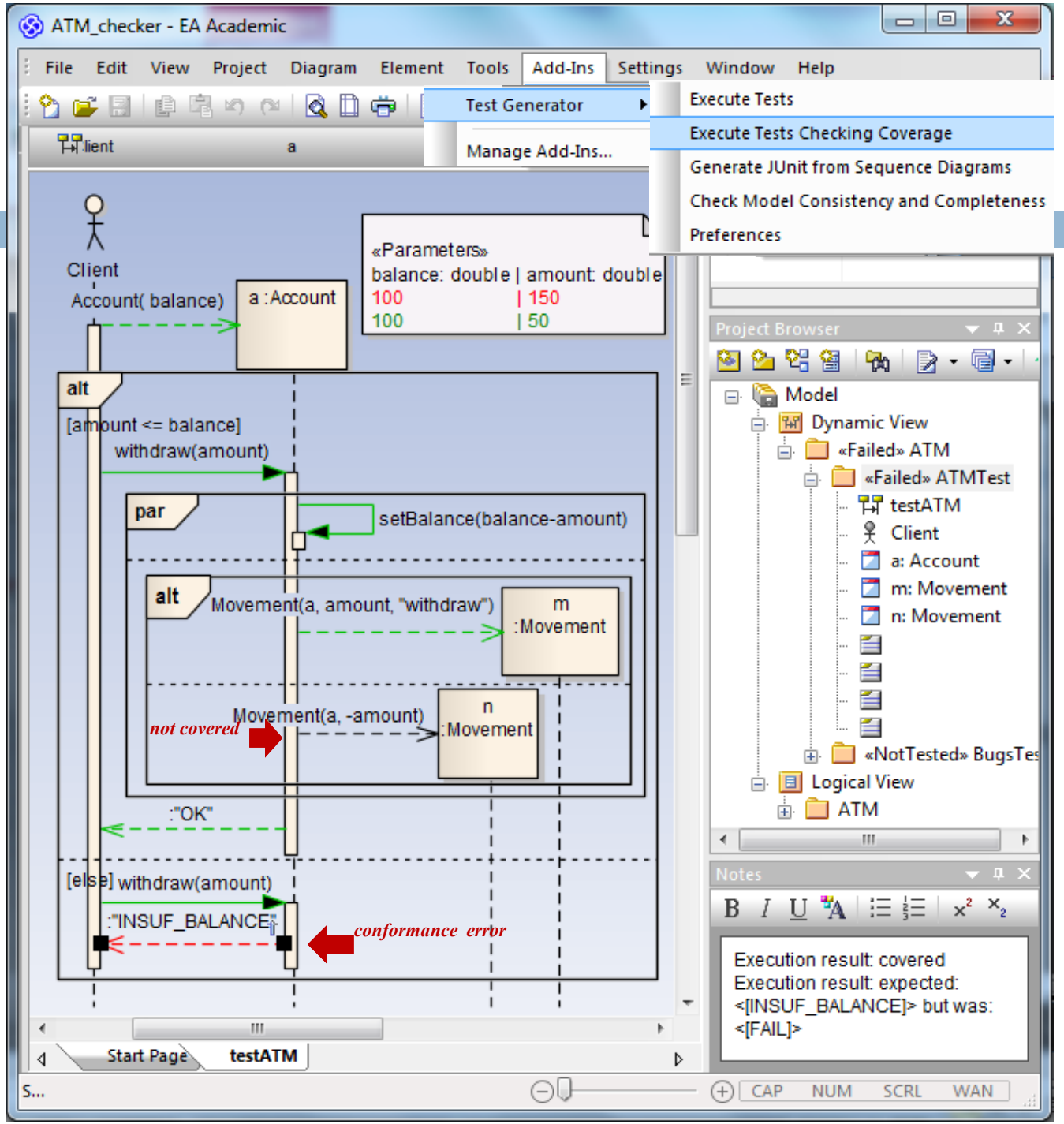
(Driver) Generate inputs as in spec and check responses against spec

(Monitor) Trace execution and check against spec

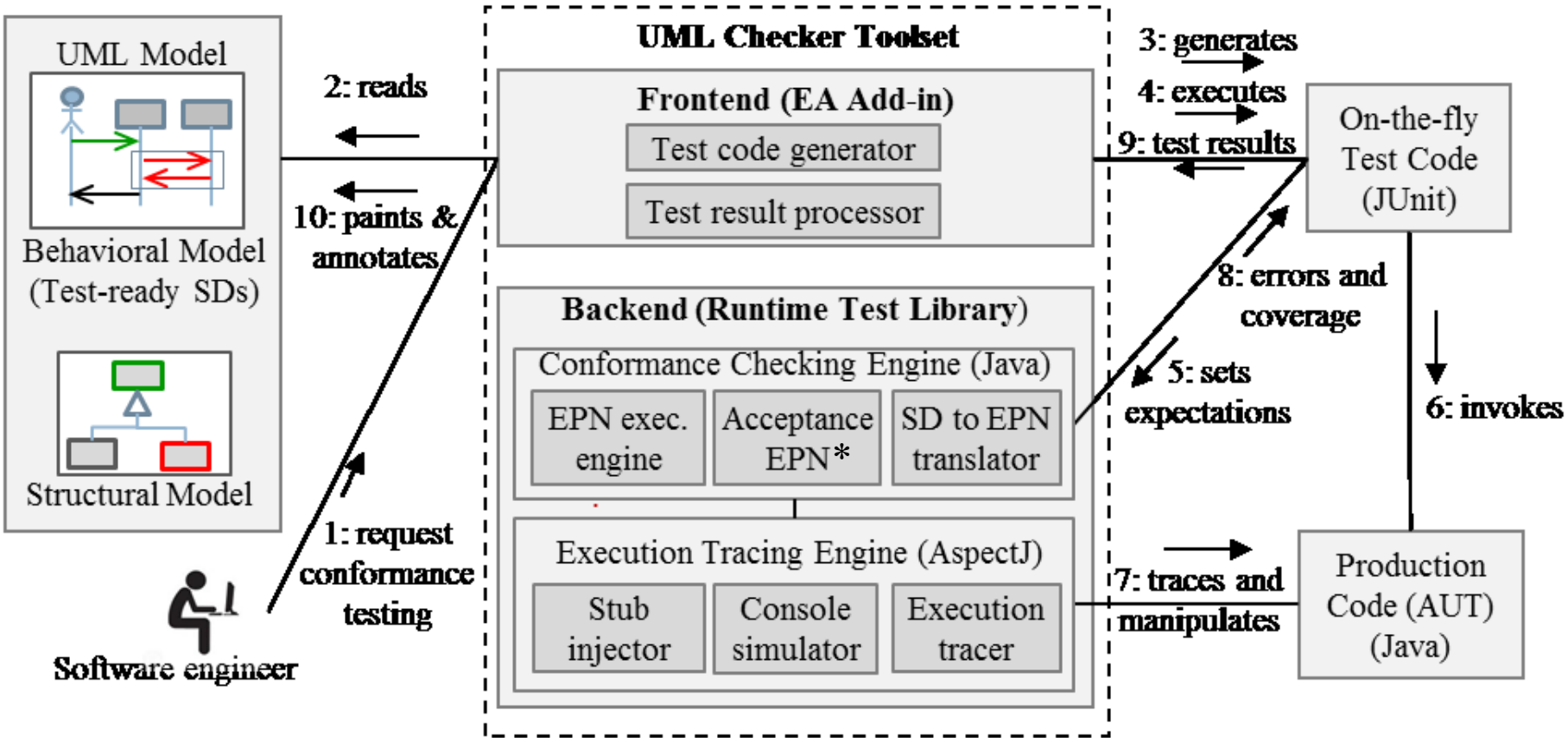
(Stub) Generate the responses as in spec

Exercise the scenario for each example

Tool user interface



Tool architecture (v3)



*EPN=Extended Petri Nets

Live demonstration

- Example UML model
- Test generation
- Test code generated
- Test execution and reporting (including coverage information)
- Bug fixing
- Stubs
- Loose conformance
- Combined fragments
- User interaction testing
- Model consistency and completeness checking

Key features and benefits (1)

Feature

- Support the modeling & automatic testing of
  - External interactions with users (UI)
  - External interactions with client applications (API)
  - Internal interactions among objects in the program

Benefits

- Covers 4 design views (w/ structural model)
- Assures higher conformance with spec
- Improves fault localization
- Accelerates test phase

	Dynamic	Static
Ext.	Sequence diagrams (external interactions)	Class diagrams (public/external interfaces)
Int.	Sequence diagrams (internal interactions)	Class diagrams (private/internal interfaces)

Key features and benefits (2)

Feature	*Benefits*
Parameterization Combined fragments (alt, opt, loop, par)	Keep behavioral specs as generic as desired
Loose conformance checking additional or intermediate calls are allowed in implementation	Keep behavioral specs as simple as desired (focus on relevant interactions)
Automatic checking of model consistency & completeness	Verifiable completeness criteria Higher quality assurance
"Stubs" inject the specified response messages for things marked as not yet implemented	Iterative implemention & testing Independence of external components

Related work

	SeDi-TeC [30]	Javed et al. [32]	SCEN-TOR [31]	Test conductor [33]	UML checker
Interaction parameters	Partly[a]	Partly[a]	Partly[a]	Yes	Yes
Keyword-based UI testing	No	No	No	No	Partly[b]
Internal interaction checking	Yes[c]	Partly[d]	No	Partly[e]	Yes[f]
Loose conformance checking	Yes	No[g]	No	Partly[h]	Yes
Test stub injection	Yes[i]	No	No	Partly[j]	Yes[k]
Interaction operators	No[l]	No	No	Partly[m]	Yes
Complex value specifications	No	No	No	No	Yes[n]
Test code generation	No[o]	Yes[p]	Yes	Yes	Yes
Test results in the model	Partly[q]	No	No	Yes	Yes
Model coverage analysis	No	No	No	Yes	Yes

Conclusions

- Presented a lightweight MDE approach
  - Based on lightweight behavioral and structural models
  - (Partial) production code and (full) test code generation from models
- That is “PSP friendly” (PSP – Personal Software Process)
  - Promotes complete (in a sense), precise and reviewable designs
  - Embeds test specification in the design phase (as behavior specs)
  - Is designed to bring short term productivity and quality benefits
- And “agile friendly”
  - Compilable models are not just documentation
  - TDD/BDD [create a test = create an (external + internal) behavior spec]

Ongoing work

- Extend UI modeling and testing features for GUIs
- Automatically generate test data (i.e., actual values for scenario parameters) through constraint satisfaction
- Conduct more extensive experimentation and process performance and usability analysis
- Support the testing of time constrained, concurrent and distributed systems, particularly for integration testing

References and further reading

- See: https://blogs.fe.up.pt/sdbt/
- Automating Interaction Testing with UML Sequence Diagrams: Where TDD and UML meet, João Pascoal Faria, Agile Portugal 2010, Porto, Portugal
- Integrating Model-Driven Engineering Techniques in the Personal Software Process, João Pascoal Faria, TSP Symposium 2012, St. Petersburg, Florida, USA
- Test Generation from UML Sequence Diagrams, João Pascoal Faria, Ana C. R. Paiva and Z. Yang, Proceedings of the 8th International Conference on the Quality of Information and Communications Technology (QUATIC'12), IEEE Computer Society Press, 2012
- Automating Scenario Based Testing with UML and Aspect-Oriented Programming, Mário Ventura de Castro, MSc thesis, FEUP, January 2013 (in portuguese)
- Techniques and Toolset for Conformance Testing against UML Sequence Diagrams, João Pascoal Faria, Ana C. R. Paiva, Mário Ventura de Castro, The 25th IFIP International Conference on Testing Software and Systems, LNCS 8254, pp. 180–195, 2013
- A Toolset for Conformance Testing against UML Sequence Diagrams based on Event-Driven Colored Petri Nets, João Pascoal Faria, Ana Paiva, International Journal on Software Tools for Technology Transfer, 2014 (to appear)