IBM Research

Incremental Development of Model Transformation Chains Using Automated Testing

Jochen Küster, Thomas Gschwind, Olaf Zimmermann

IBM Research - Zurich

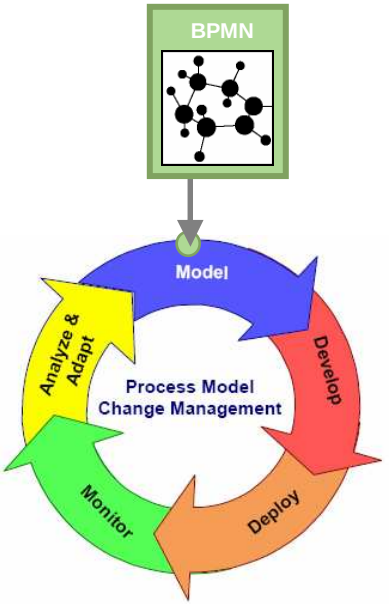
ACM/IEEE 12th International Conference on Model Driven Engineering Languages and Systems (MODELS), Denver, October 2009

Outline

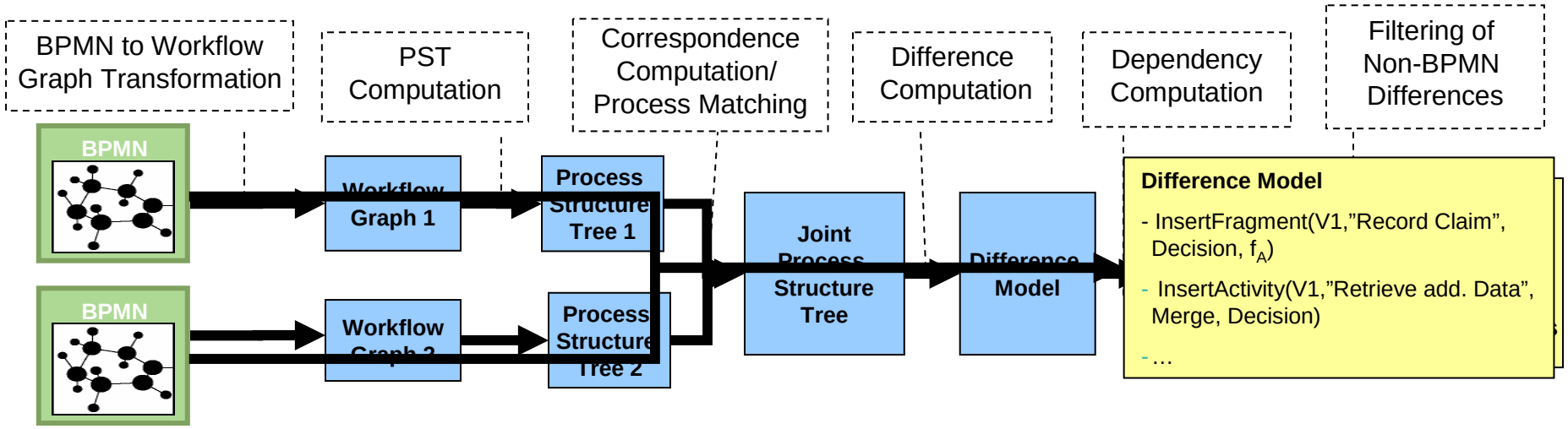
- Motivation and Case Study Overview
- Requirements for Development of Transformation Chains
- Techniques and Approach
  - Test Design Techniques
  - Test Architecture
  - Development Process
- Results
- Conclusion & Outlook

Motivation and Context

- Process Models are key artifact in modern SOA-oriented software development
- Business-Driven Development comprises
  - modeling of processes at **different levels of abstraction**
  - in **different modeling languages** (BPMN,BPEL)
  - in **a multi-user environment** by different people
- Change Management Solution for Process Models
  - see “Language-Independent Change Management of Process Models” by C. Gerth et al
  - framework for change management of process models



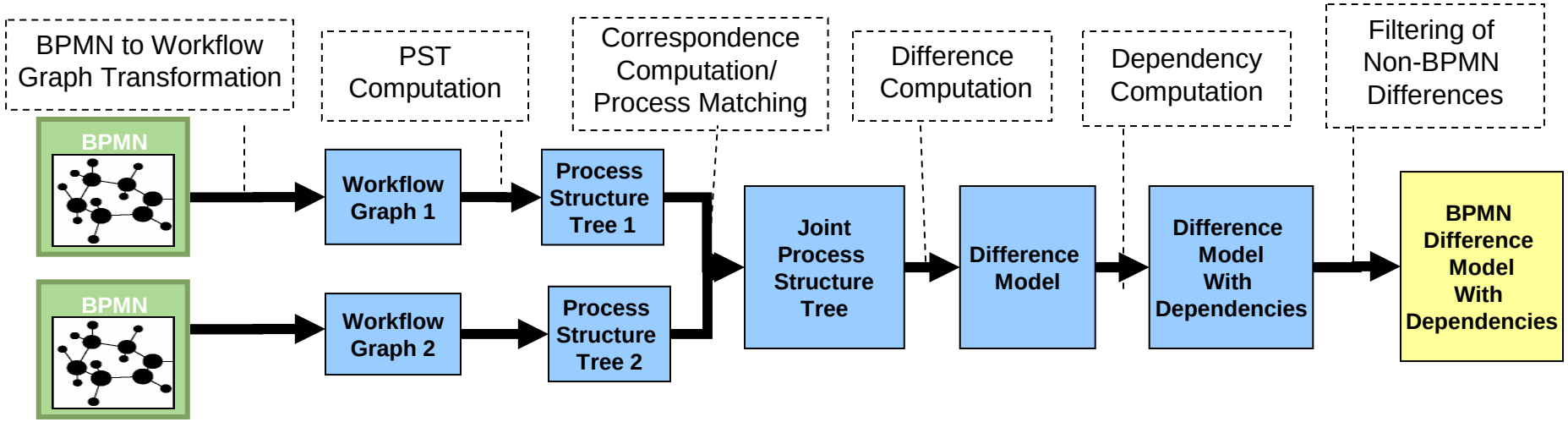
Overview of Change Management Solution



- Transformation for constructing Difference Model
  - obtained by instantiating the framework for change management
- Transformation chain involving 6 individual transformations
- Each transformation has to be implemented in Java (tool integration issue)

How to ensure quality of the transformation chain?

Challenges in Transformation Chain Design and Implementation



- Dependencies between transformations
  - changes in correspondence computation affect Difference Model
  - changes in WFG and PST computation affect Difference Model
  - …
- Dependencies between different input model pairs
  - functionality written for one input model pair may affect other input model pairs
  - infinitely many input model pairs make this difficult
- Diversity of input models
  - transformation chain has to produce a Difference Model for all possible combinations

Outline

- Motivation and Case Study Overview
- Requirements for Development of Transformation Chains
- Techniques and Approach
  - Test Design Techniques
  - Architecture of Test Framework
  - Development Process
- Results
- Conclusion & Outlook

Requirements for Development Approach (1)

How to ensure quality of a transformation chain?

- Development process must be iterative and incremental
  - development will take place in several iterations
  - transformation chain can be developed in teams
  - How to organize the development process for a transformation chain?
- Testing of transformation chain is required
  - important for quality assurance
  - How to design test cases for a transformation chain?
- A fully automated test environment is needed
  - manual testing on large number of input models is impossible in practice
  - How to establish such an automated test environment?

Requirements for Development Approach (2)

How to ensure quality of a transformation chain?

- Specific transformation chain requirements
  - software engineer must be able to change/add functionality of a transformation without breaking transformation chain
  - fixing defects requires a mechanism to ensure that changes of a transformation are side-effect free
  - refactoring for ensuring code quality of the transformations is required, typical refactorings (e.g. [1]) need to be applied
  - How to fulfill these requirements in the development environment and in designing test cases?

[1] M. Fowler: Refactoring. Improving the Design of Existing Code. Addison Wesley,1999

Outline

- Motivation and Case Study Overview
- Requirements for Development of Transformation Chains
- Techniques and Approach
  - Test Design Techniques
  - Architecture of Test Framework
  - Development Process
- Results
- Conclusion & Outlook

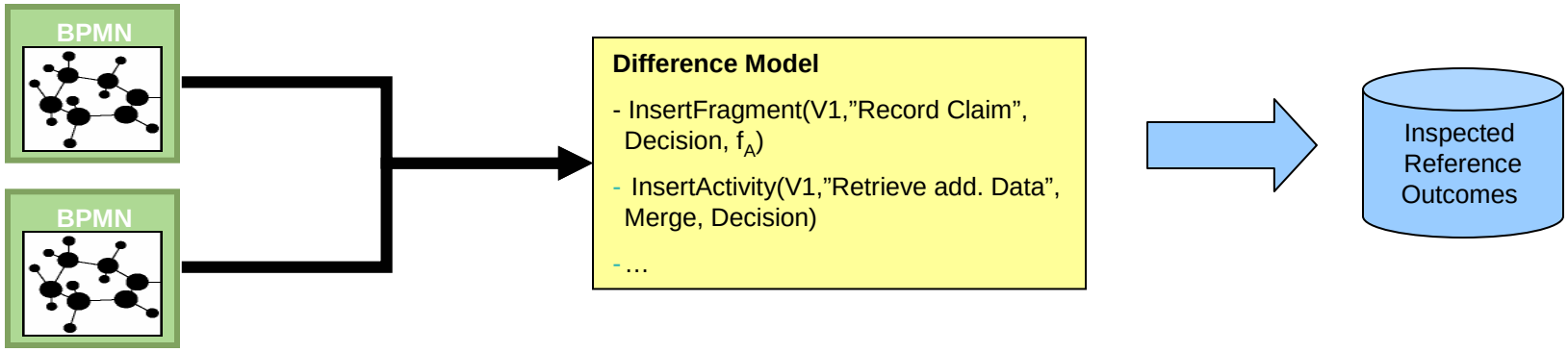
Test Design Techniques for Model Transformation Chains

- Test design techniques for deriving test cases for a transformation chain
  - test cases for transformation chain and each individual transformation
- Three different test design techniques
  - Inspected reference outcomes
  - Invariant validation
  - Deviation test data

Test Design Techniques for Automated Testing (1)

Inspected reference outcomes

- textually specified result of a transformation
- creation: manually inspect the outcome of the transformation once and save it

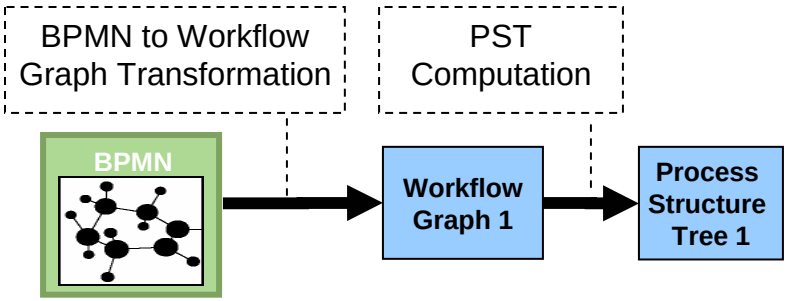


- usage: automatic testing verifies outcome when applying a transformation
- advantage:
  - very valuable during defect removal and refactoring
  - detection of unwanted side effects when changing transformations
- drawback: only limited numbers of test cases possible (manual overhead)

Test Design Techniques for Automated Testing (2)

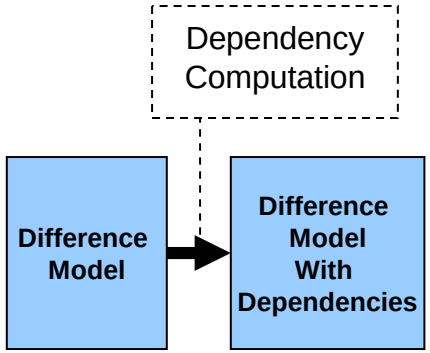
Invariant validation:

- invariants about the outcome of a transformation
- creation: encode invariants in Java or OCL



Invariants: WFG and PST properties

- every node is reachable
- StructuredNode has exactly one entry/exit edge



Invariants:

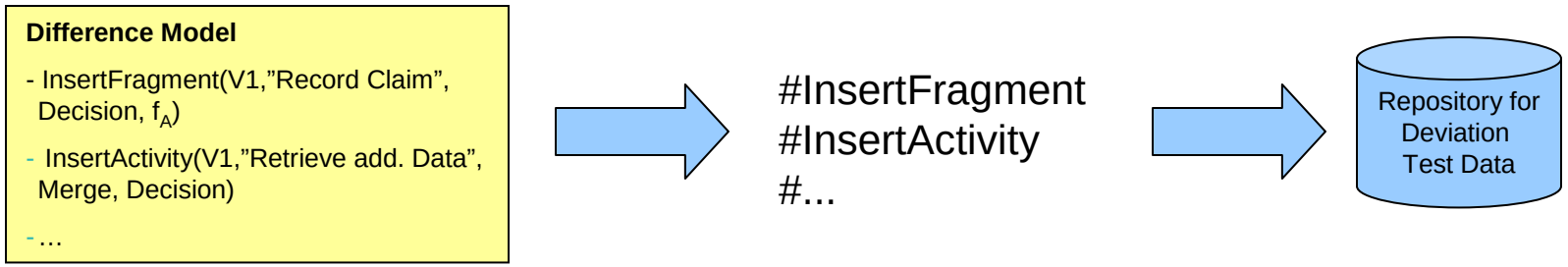
- no cyclic dependencies

- usage: check invariants automatically when applying transformation
- advantage: large number of test cases possible, no manual overhead

Test Design Techniques for Automated Testing (3)

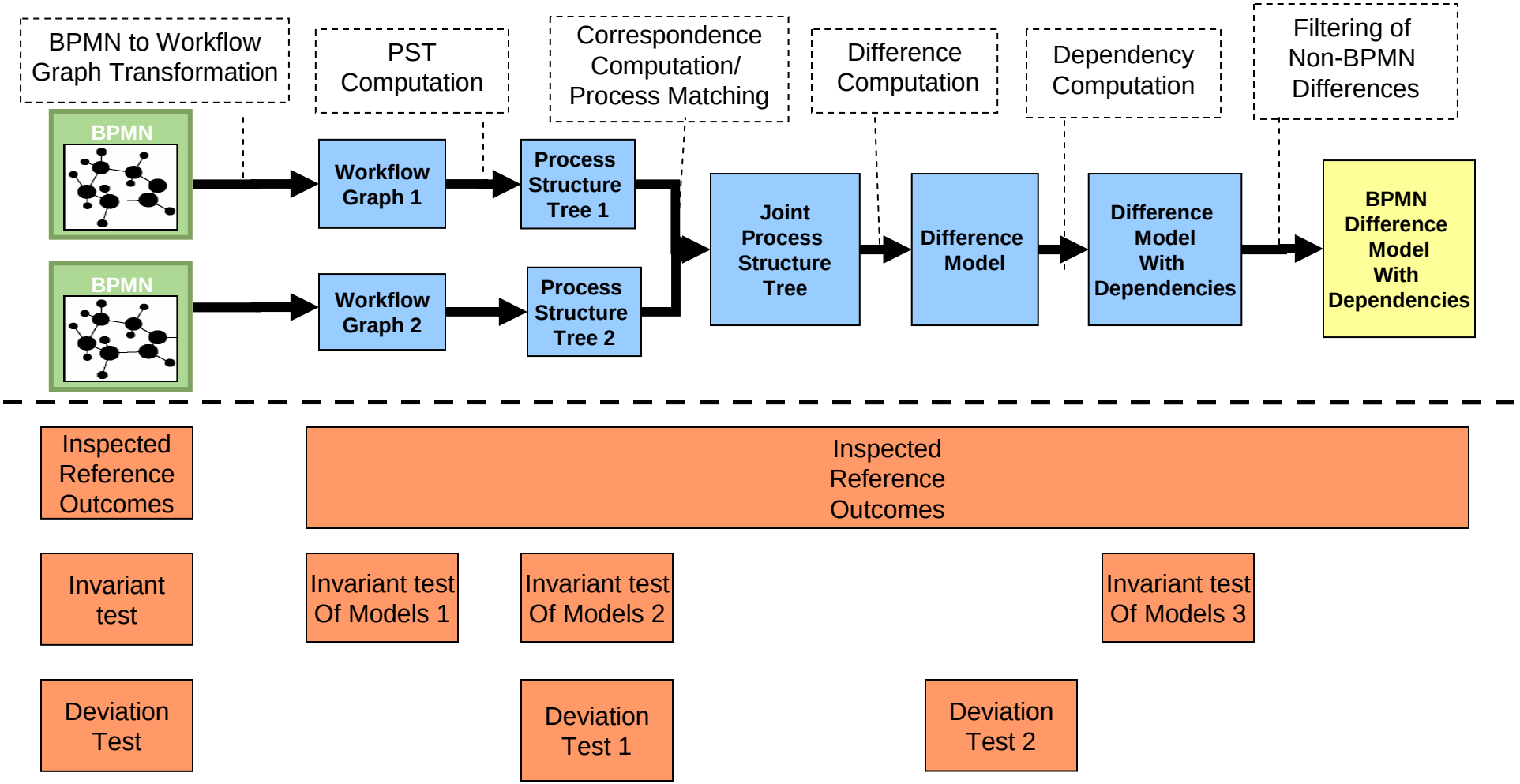
Deviation test data

- test data about created model structures
- creation: determine test data to be computed, compute it automatically for each test case



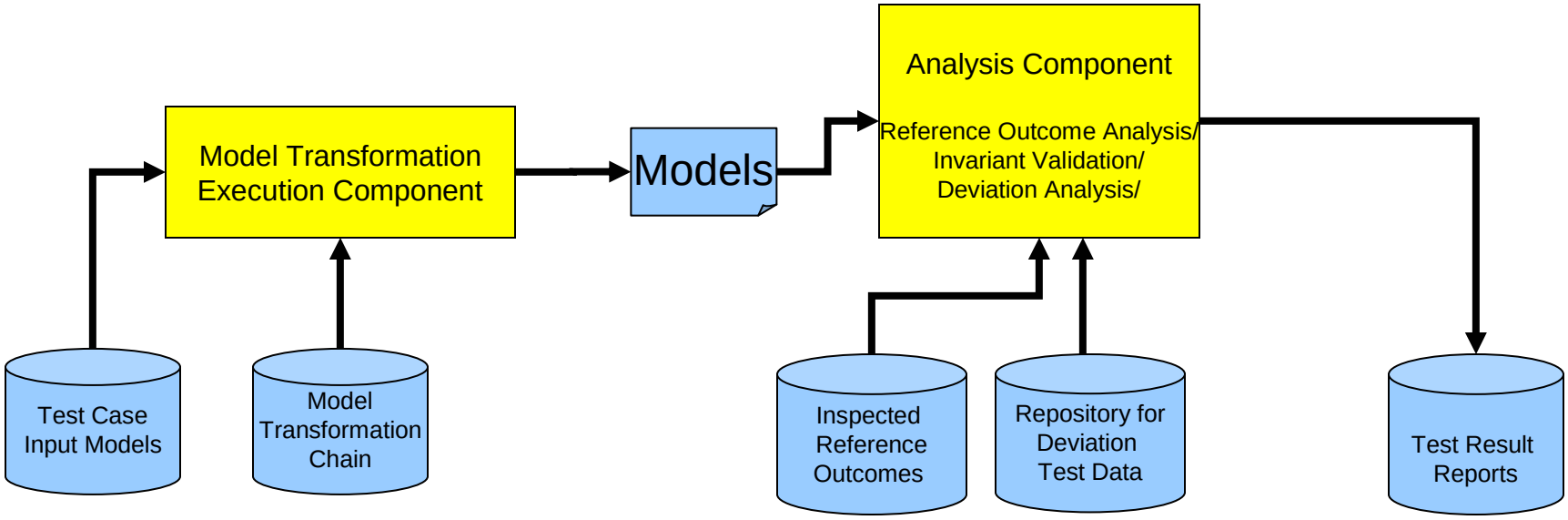
- usage: in subsequent test runs, compare data automatically and report deviations
- advantage:
  - valuable while refactoring and defect removal for detecting side effects
  - identification of input models where deviation occurs

Tests and Transformation Overview



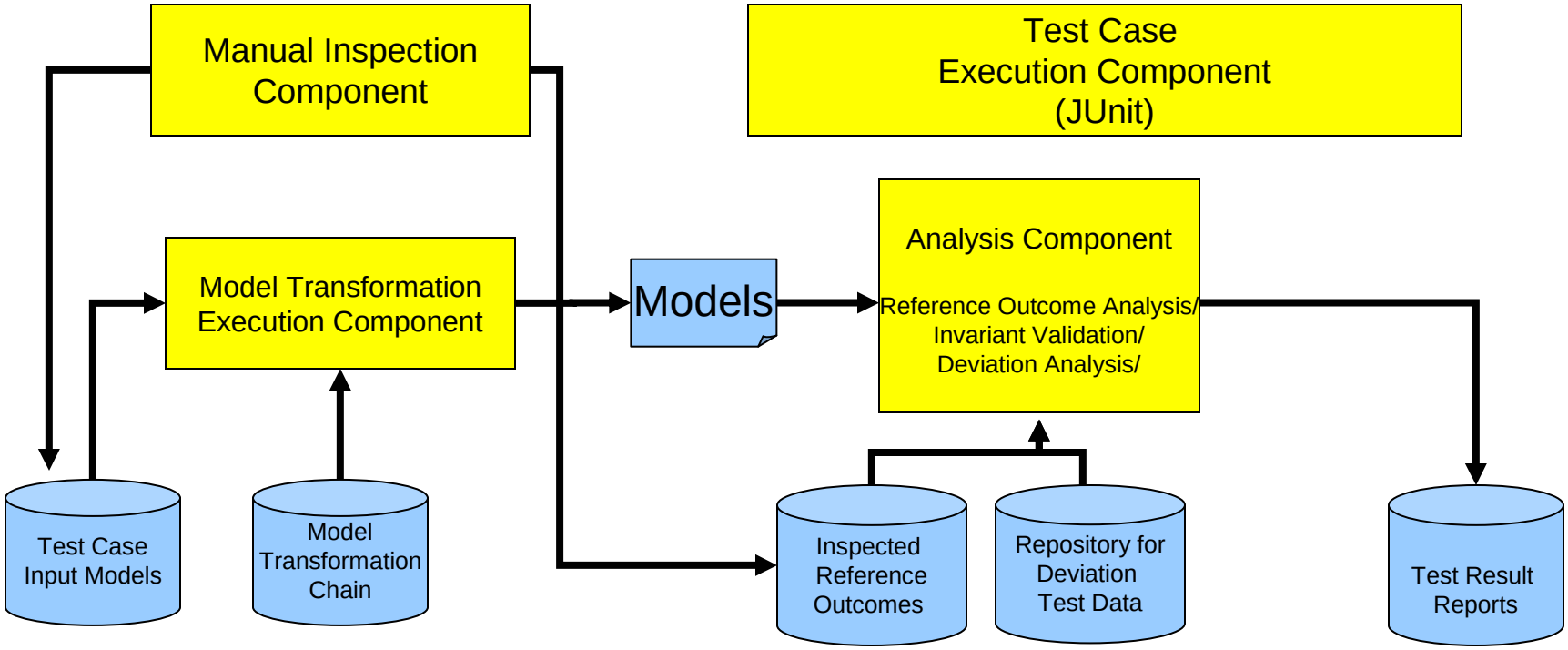
Architecture of Test Framework

- Test case input models are created manually
- Analysis Component is added for evaluating test results
  - based on test design techniques presented
  - produces detailed test result report

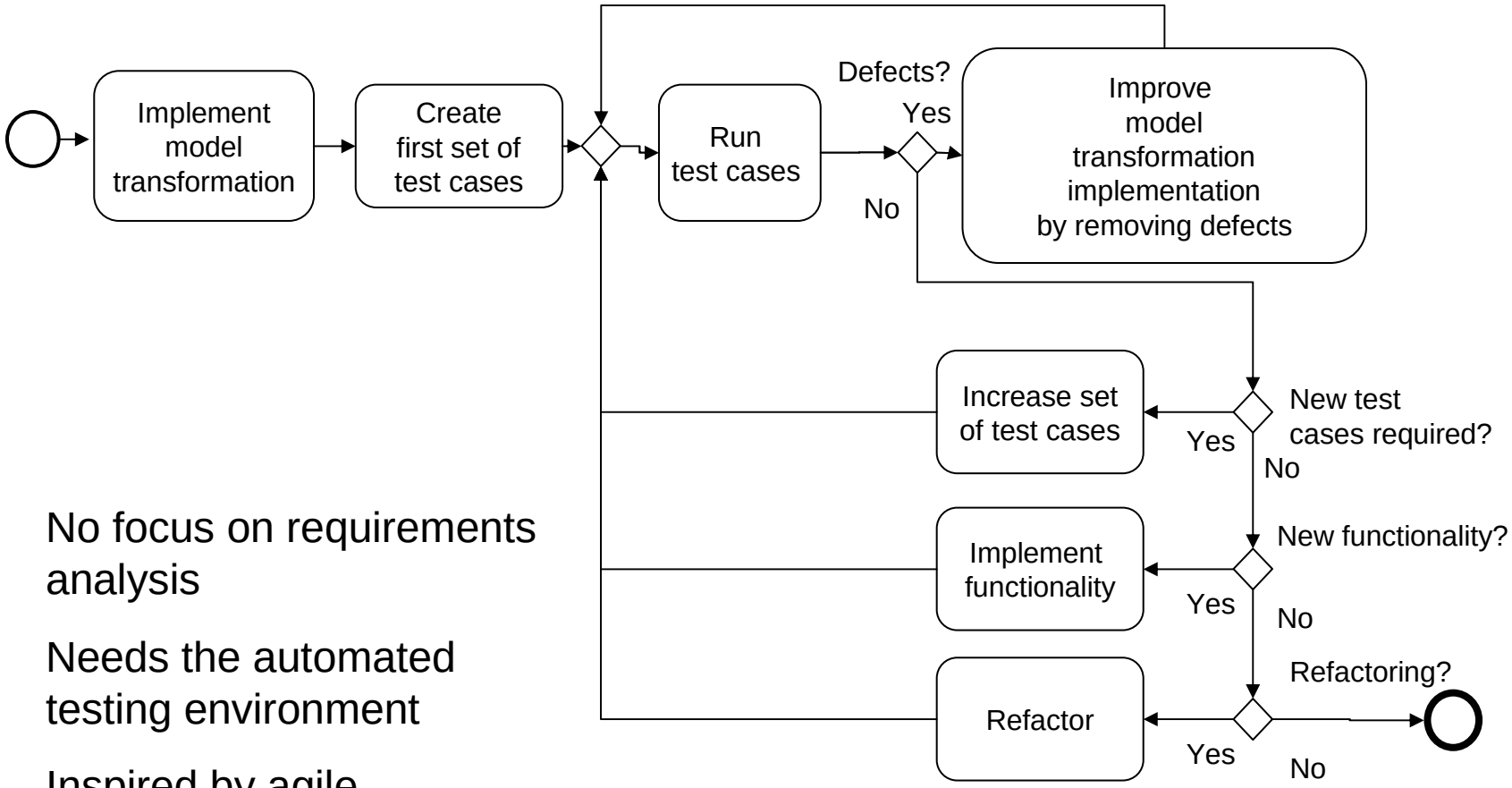


Architecture of Test Framework

- Manual Inspection Component simplifies creation of test case reference outcomes
- Execution component reuses JUnit framework for test case execution



Iterative and Incremental Development Process



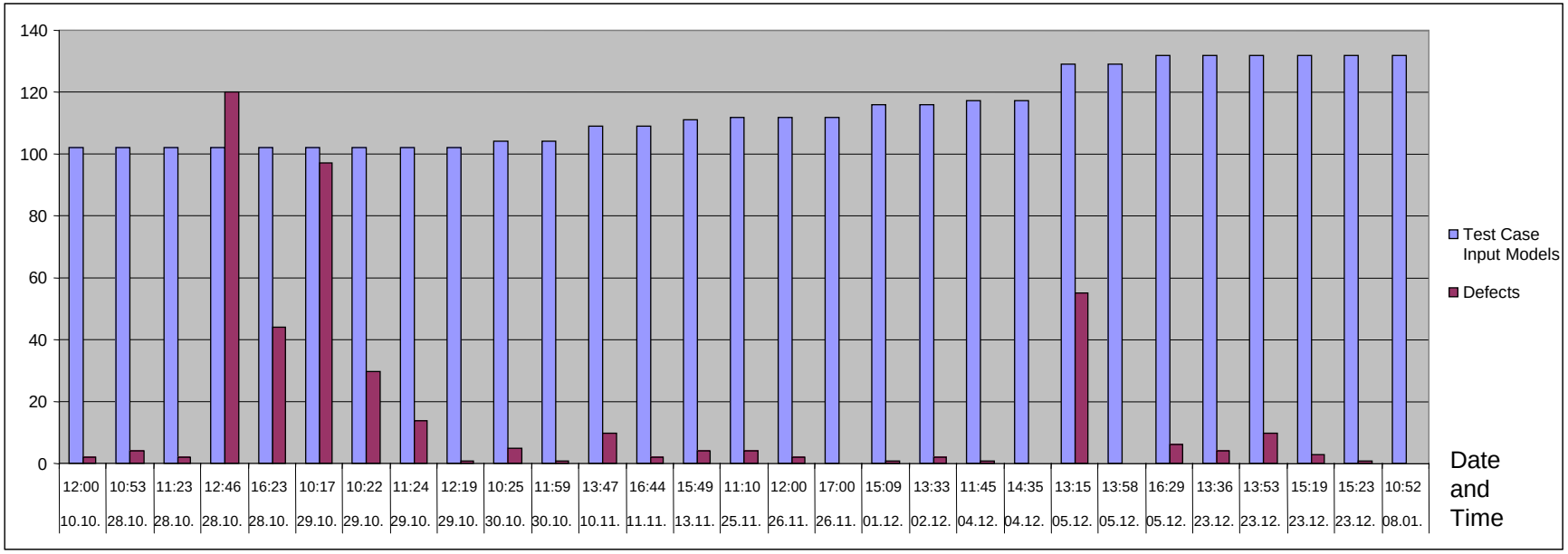
- No focus on requirements analysis
- Needs the automated testing environment
- Inspired by agile development

Outline

- Motivation and Case Study Overview
- Requirements for Development of Transformation Chains
- Techniques and Approach
  - Test Design Techniques
  - Architecture of Test Framework
  - Development Process
- Results
- Conclusion & Outlook

Defects versus Test Case Input Models

- ~300 test case input model pairs with inspected reference outcomes
- fully automatic test environment based on the test framework architecture (completion in ~ 20 sec)
- refactoring needs automated testing



Lessons Learnt

- Incremental development for model transformation chains is required in practice
- Iterative improvement of the transformations is the reality
- Systematic quality improvement is not possible without automated testing
- Code quality is an important issue when implementing model transformations in Java
- One has to choose the encoding of the inspected reference outcomes carefully

Conclusion & Outlook

- Incremental and iterative development process for developing high-quality transformation chains
- Test design techniques and test framework for automated testing of transformation chains
- Further research challenges include:
  - Model coverage of test case input models
  - Automated generation of test case input models