Fraunhofer Institute for Open Communication Systems | Kaiserin-Augusta-Allee 31 | 10589 Berlin, Germany

Requirements-driven testing with behavior trees
– Extending behavior engineering for testing purposes

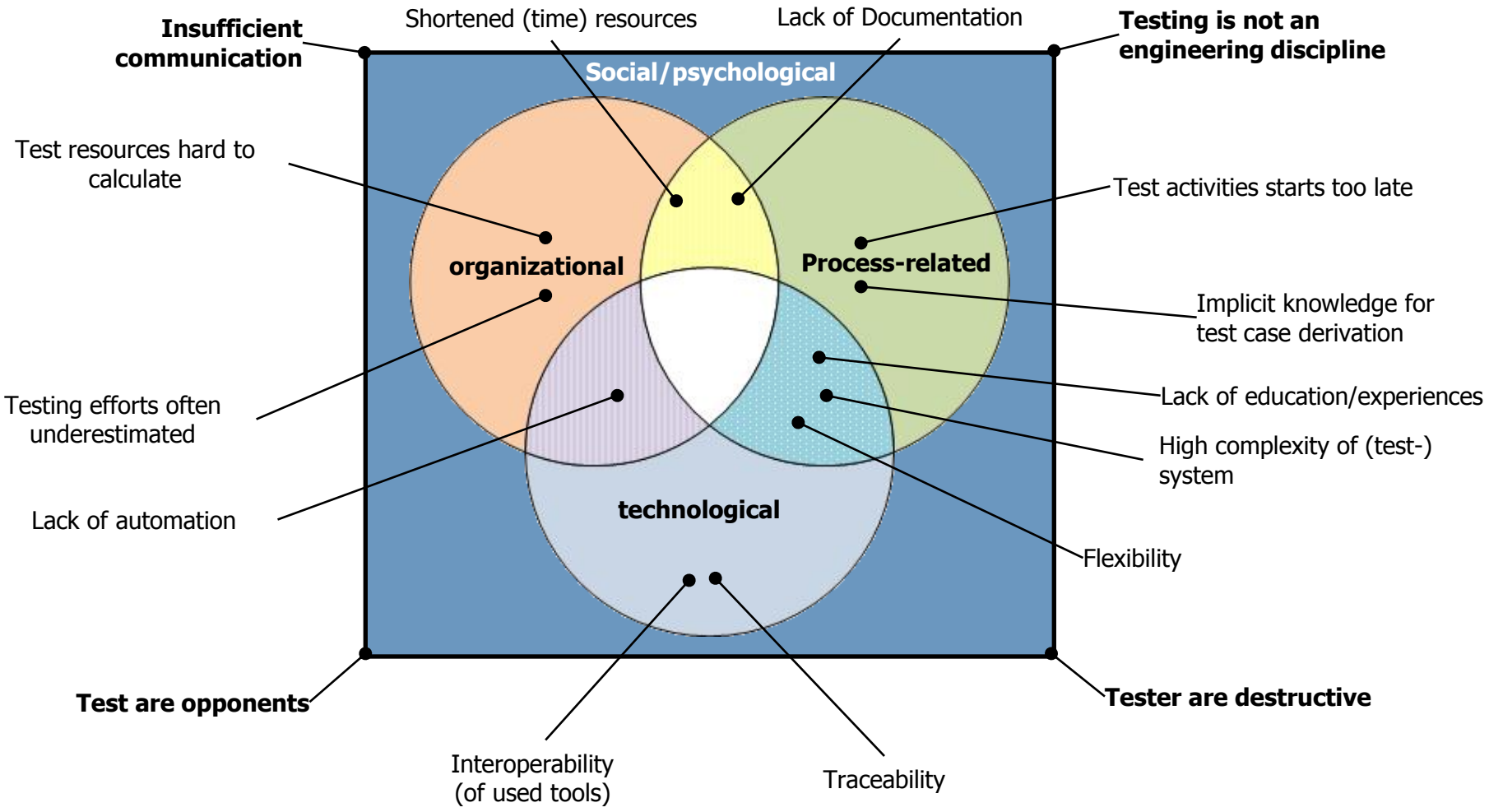
Marc-Florian Wendland, Ina Schieferdecker, Alain Vouffo-Feudjio | ReVVerT 11 | 25th of March, 2011

Agenda

- Introduction
- Challenges of MBT approaches
- Introduction to Behavior Engineering
- Requirements-driven testing with behavior trees
- Challenges and further steps

What‘s wrong with testing?

Introduction.
Challenges of testing.



Introduction
Potential improvement areas for model-based testing

- Formalize test specification
  - Make implicit knowledge explicit

Use (semi-)formal artifacts (test model)

- Increase automation
  - Test case derivation
  - Test case execution
  - Test result analysis

Use (semi-)formal artifacts as the main artifacts to increase automation

- Process transparency
  - Transparent test case derivation
  - Clear documentation

Use well-defined algorithms for test case derivation, and diagrams for documentation

- Improve Communication
  - Tester and developer
  - Test manager and stakeholder

Use a modeling notation, methods and tools appropriate for both tester, developer and stakeholder

What‘s wrong with testing?

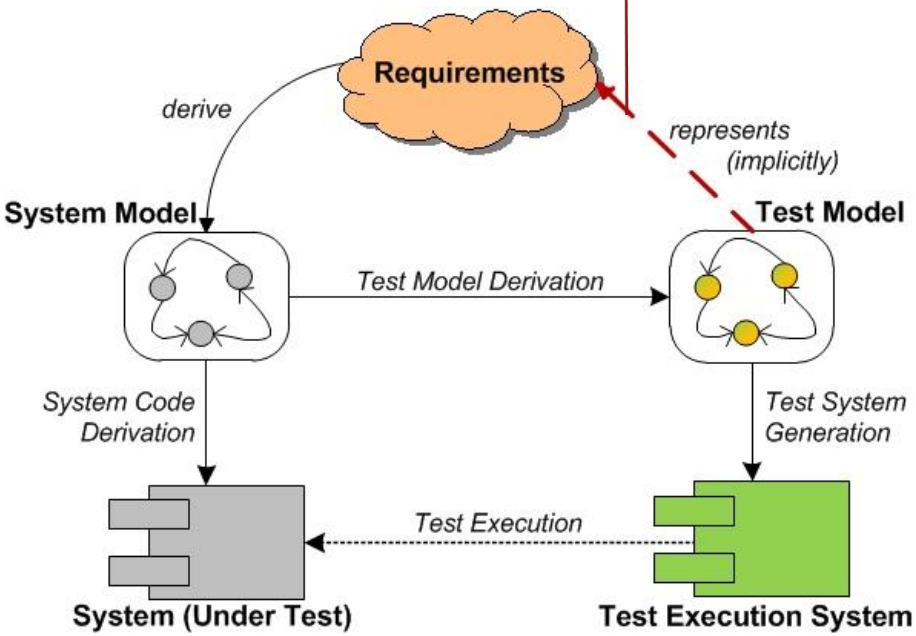
What's wrong with **model-based testing**?

Challenges of model-based testing

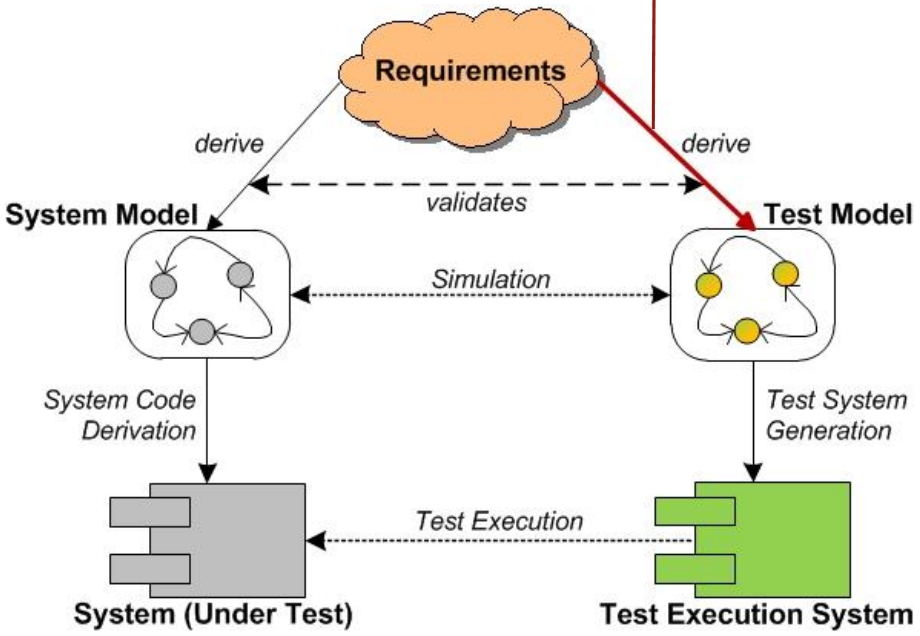
Where to obtain the test model from?

Implicit representation of requirements:
- missing independency
- potential error propagation

Explicit representation of requirements:
- Time and resource consuming
- Manual, error prone activity

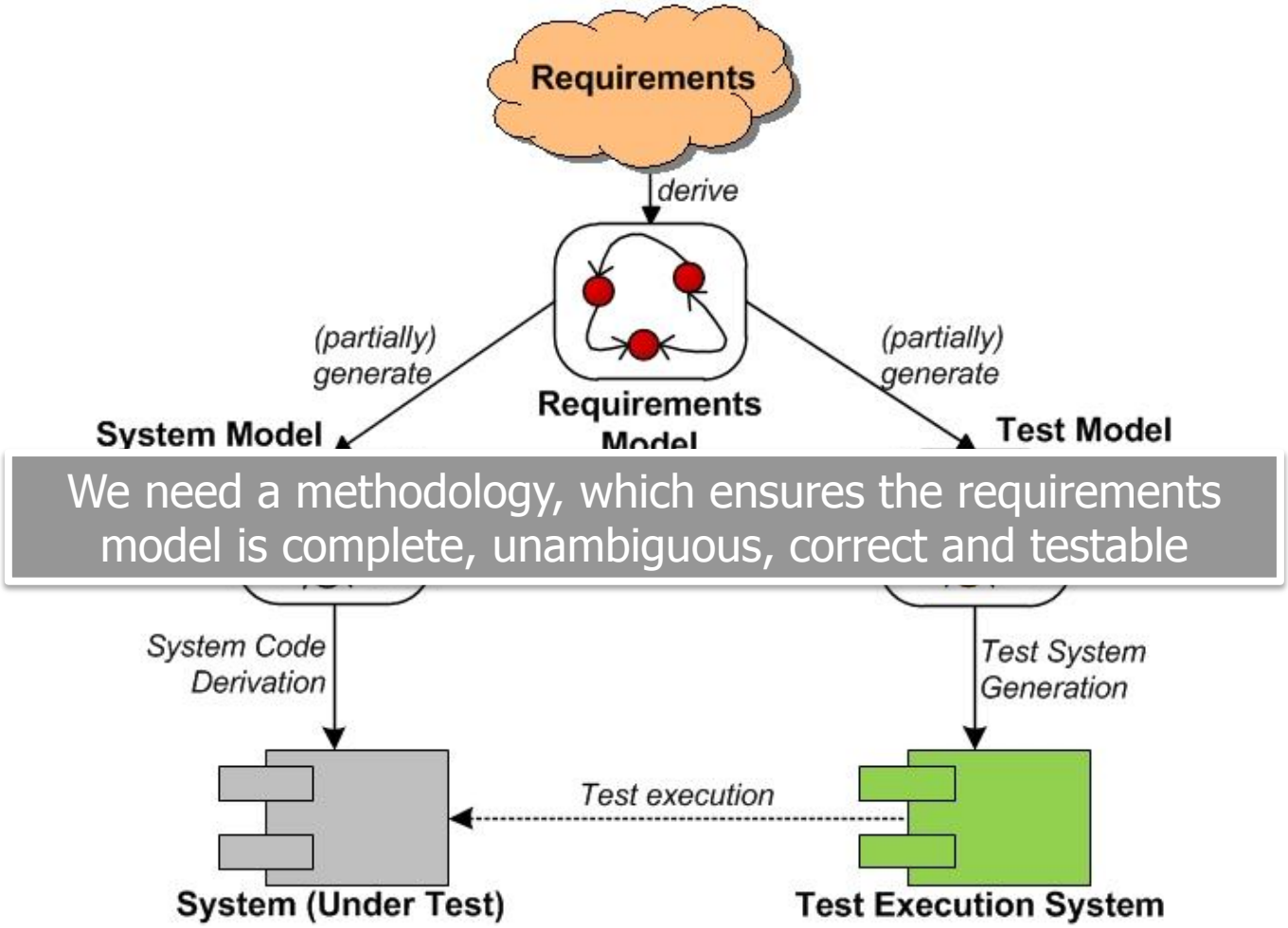


System-model driven approach



Independent models approach

Challenges of model-based testing
Starting from requirements models.



Introduction to Behavior Engineering
Behavior Engineering Methodology (BE)

- Represents a scalable methodology for formalizing requirements out of informal requirements specifications

- Based on a DSL to support the formalization (Behavior Modeling Language (BML))
  - Behavioral aspects of requirements (behavior trees), and
  - Structural information of the system-to-be (composition tree/structural tree)

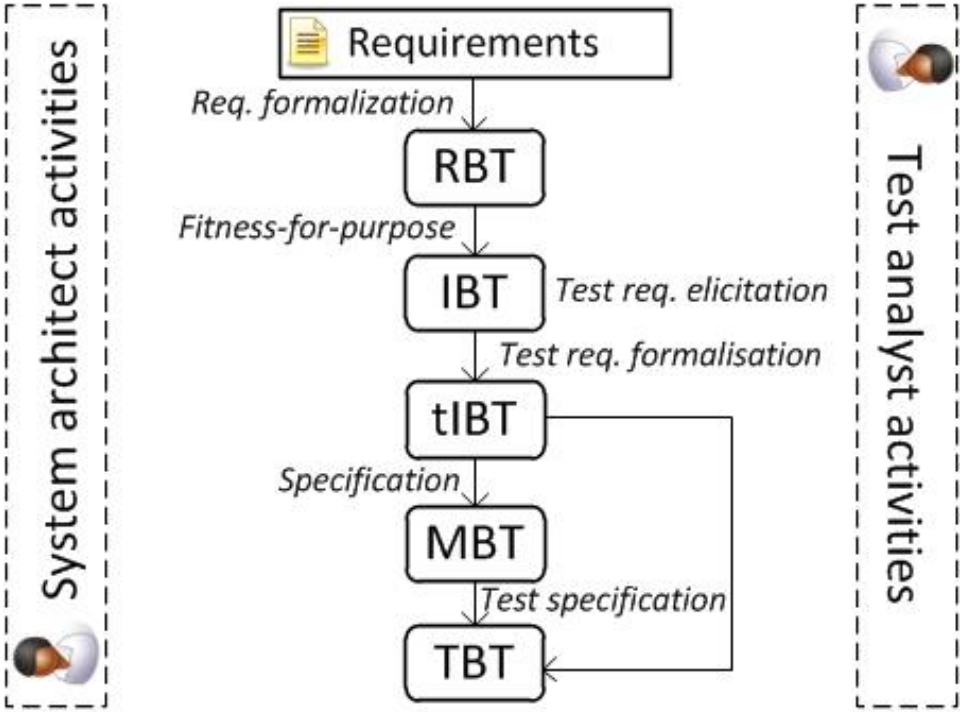
- A BT formalize the behavior of a requirement
  - Expressed as a sequence of behavioral nodes and connecting edges
  - Provides a dynamic view of the behavior of the system
  - Divide the global problem space into a local problem space

- A CT models the structural foundation on which behavior is carried out
  - Identifies components of the system and their relationships
  - Captures the vocabulary of the requirements specification

Introduction to Behavior Engineering
Behavior Engineering Phases

State of the art behavior engineering
- Formalization of requirements (RBT)
- Integration of requirements (IBT)
- *Specification (MBT)*

Testing activities for behavior engineering
- Identification of test requirements
- Test requirements formalization
- Test specification
- *Test execution*



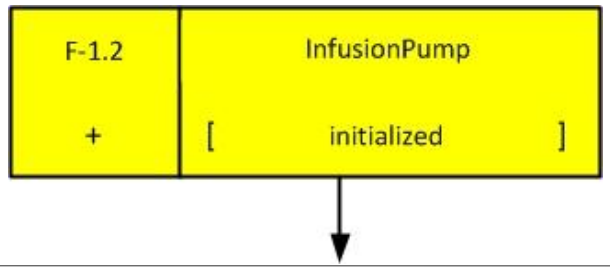
RBT = Requirements Behavior Tree
IBT = Integrated Behavior Tree
tIBT = testable Integrated Behavior Tree
MBT = Modeling Behavior Tree
TBT = Testing Behavior Tree

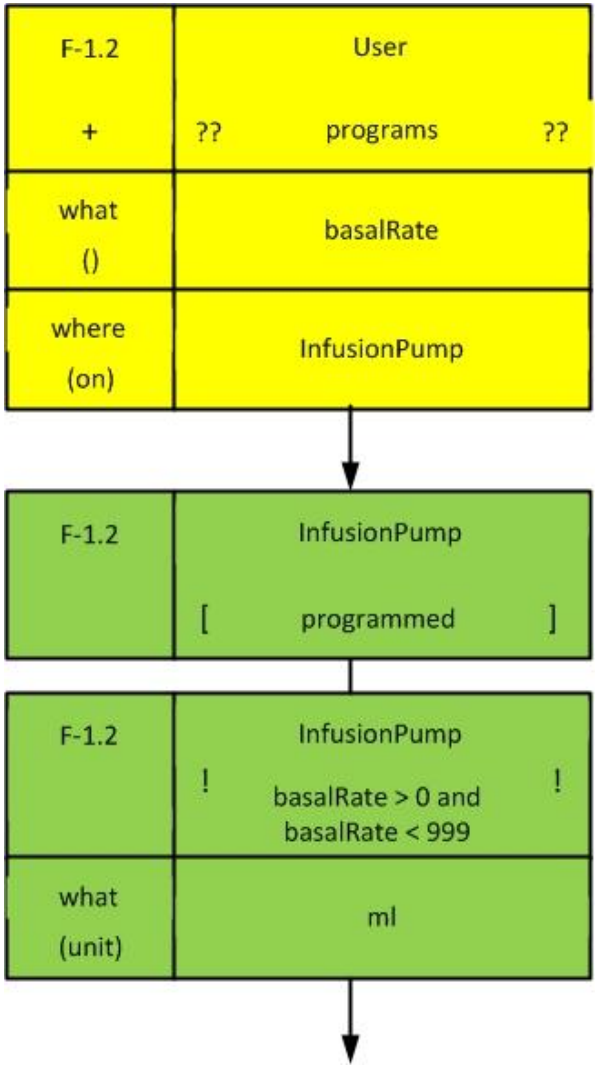
Introduction to Behavior Engineering
From informal requirements to RBTs

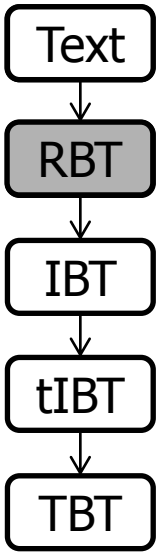
F-1.2: The infusion pump is programmed with a basal rate. The programmable basal rate shall be between 1 ml and 999 ml.

Issue #1: Who is programming the basal rate? We assume the user.

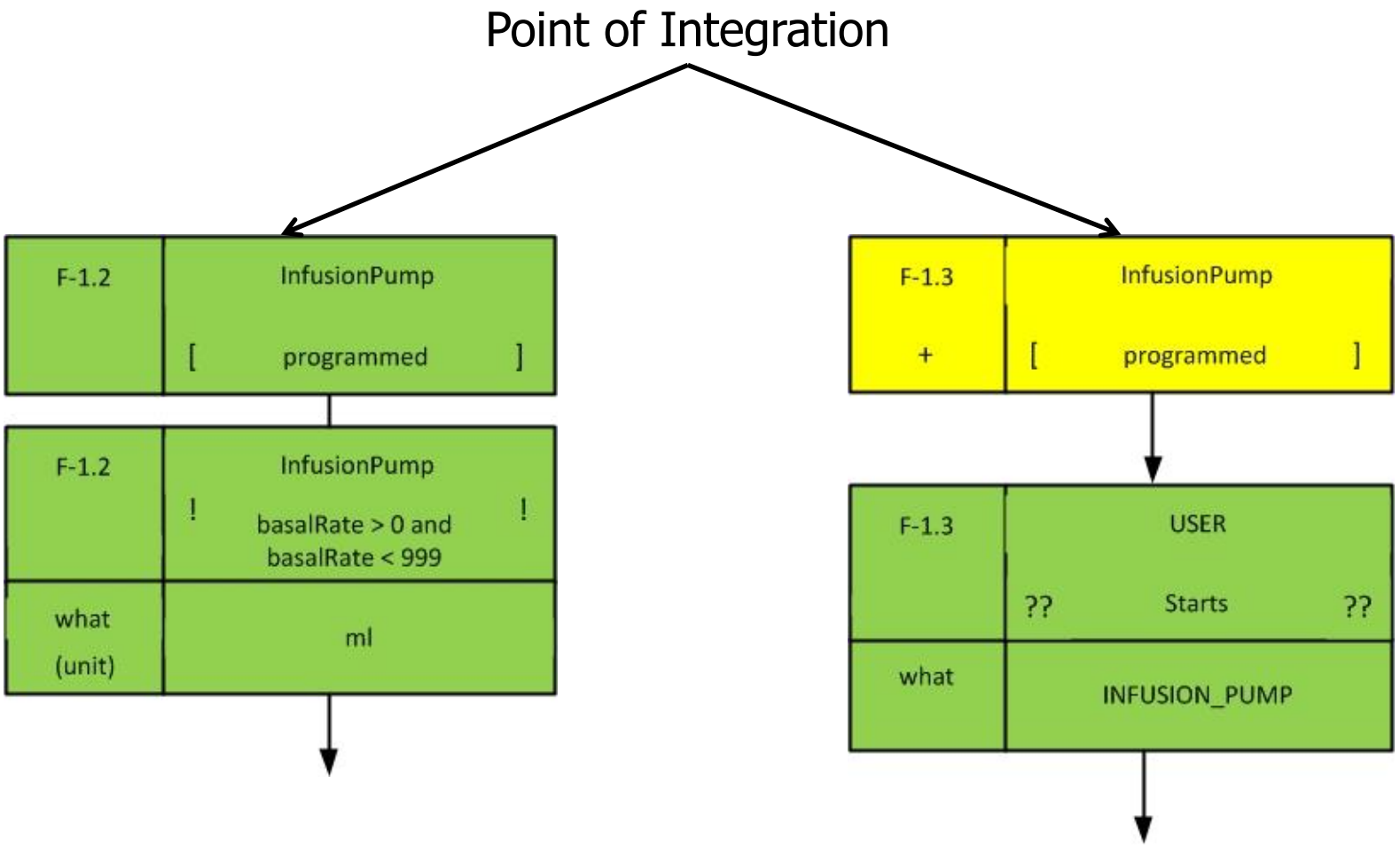
Issue #2: Infusion Pump must be initialized for starting the programming progress.

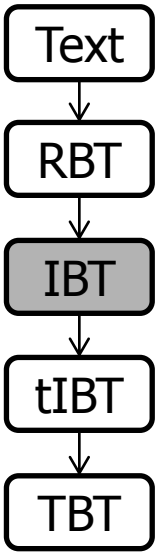




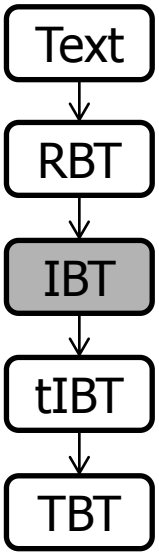


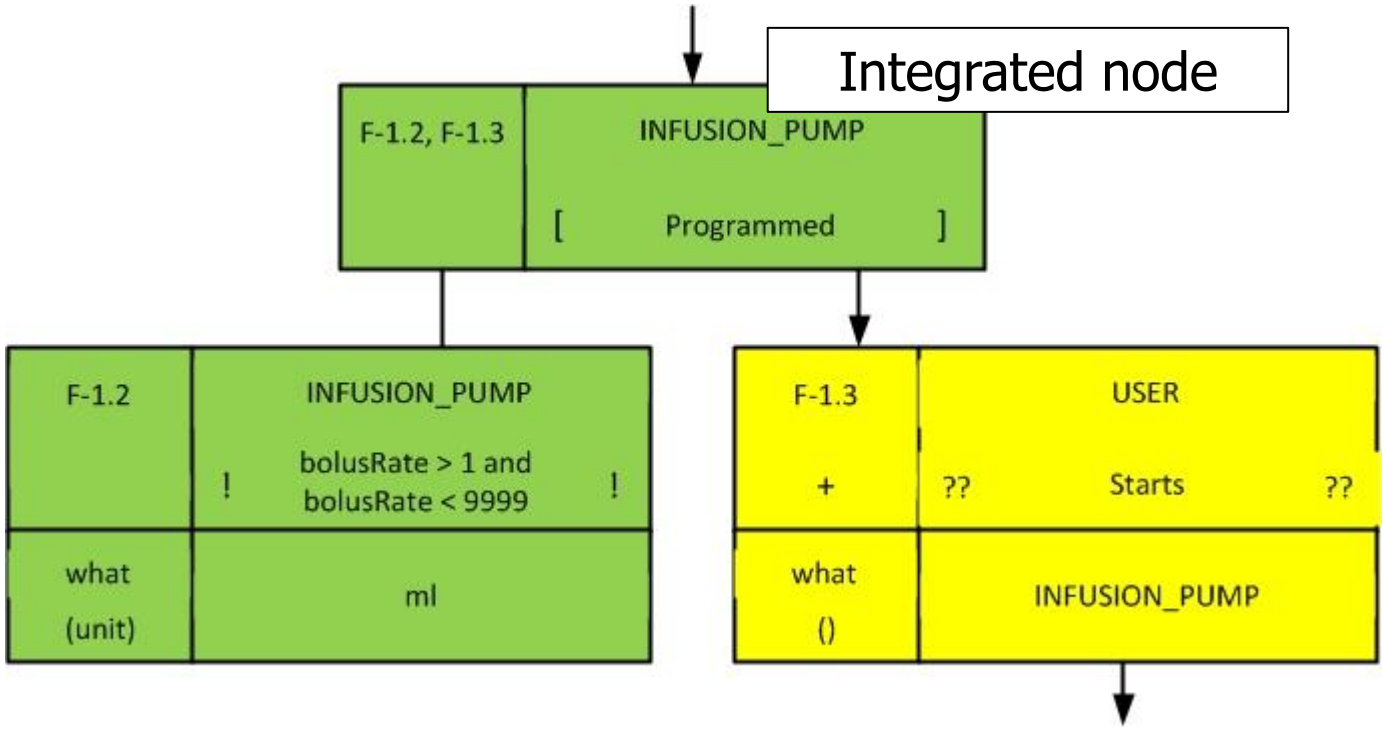
Introduction to Behavior Engineering
Integration of RBTs

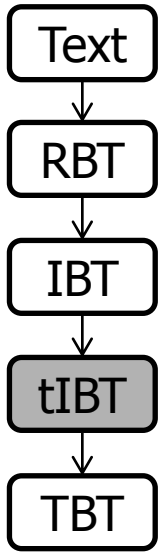




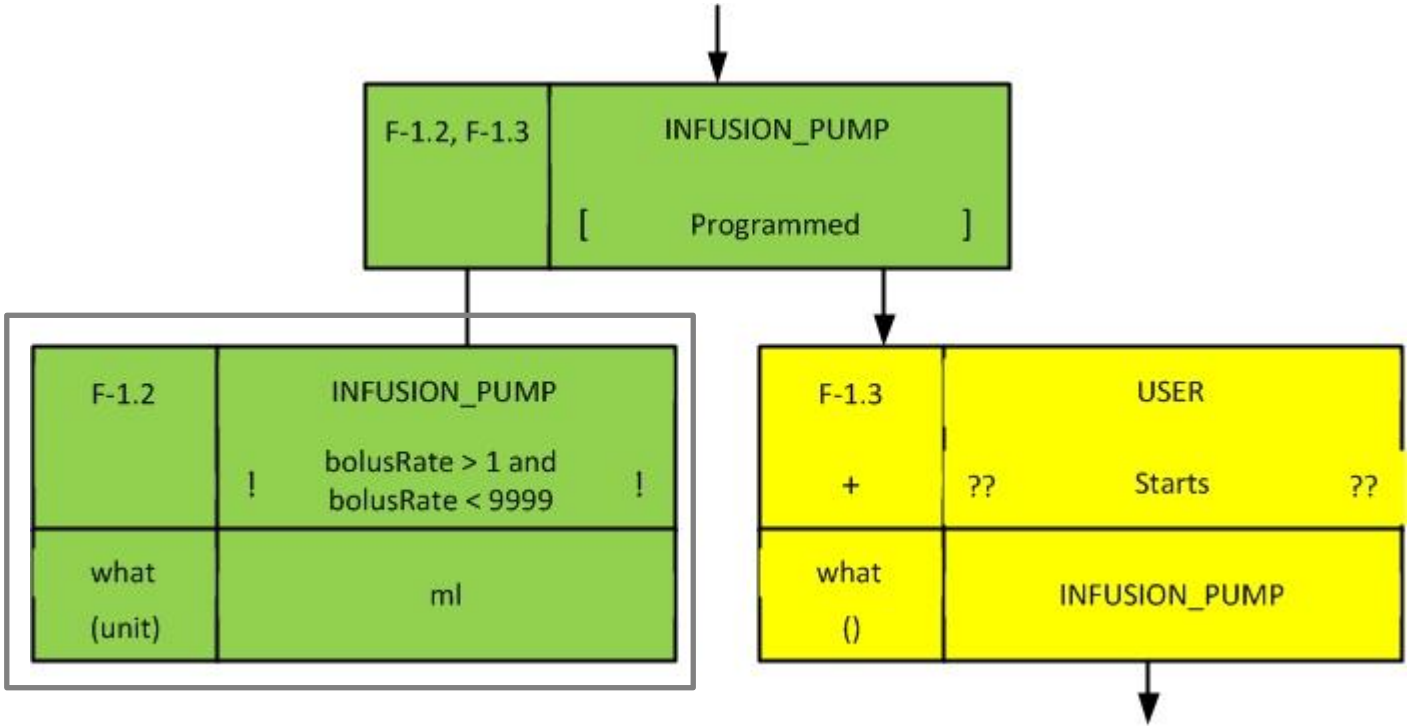
Introduction to Behavior Engineering
Integrated Behavior Tree (IBT)





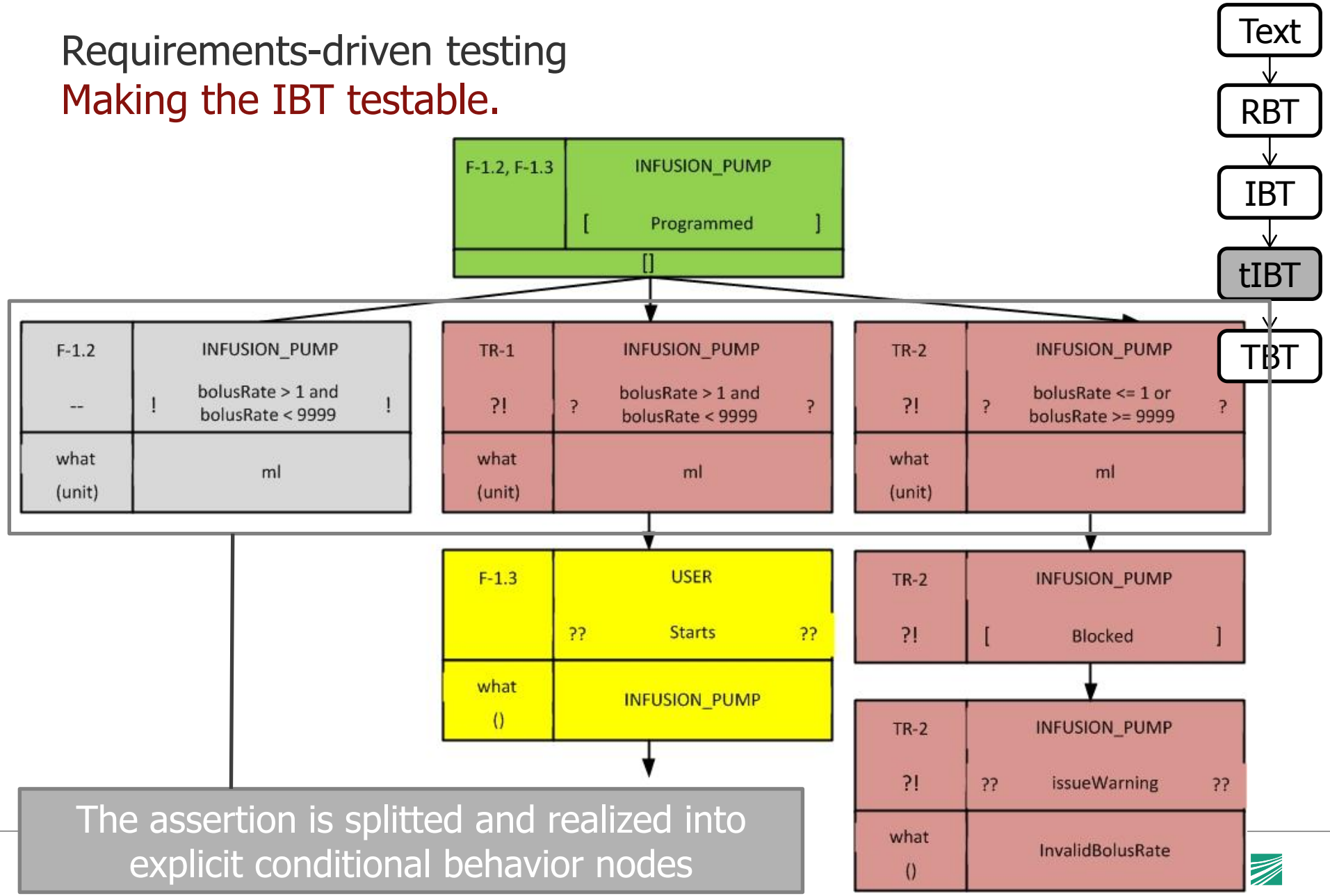


Requirements-driven testing
Identifying test requirements



Assertions are potential conditions to verify

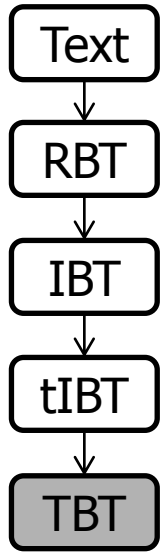
Requirements-driven testing
Making the IBT testable.

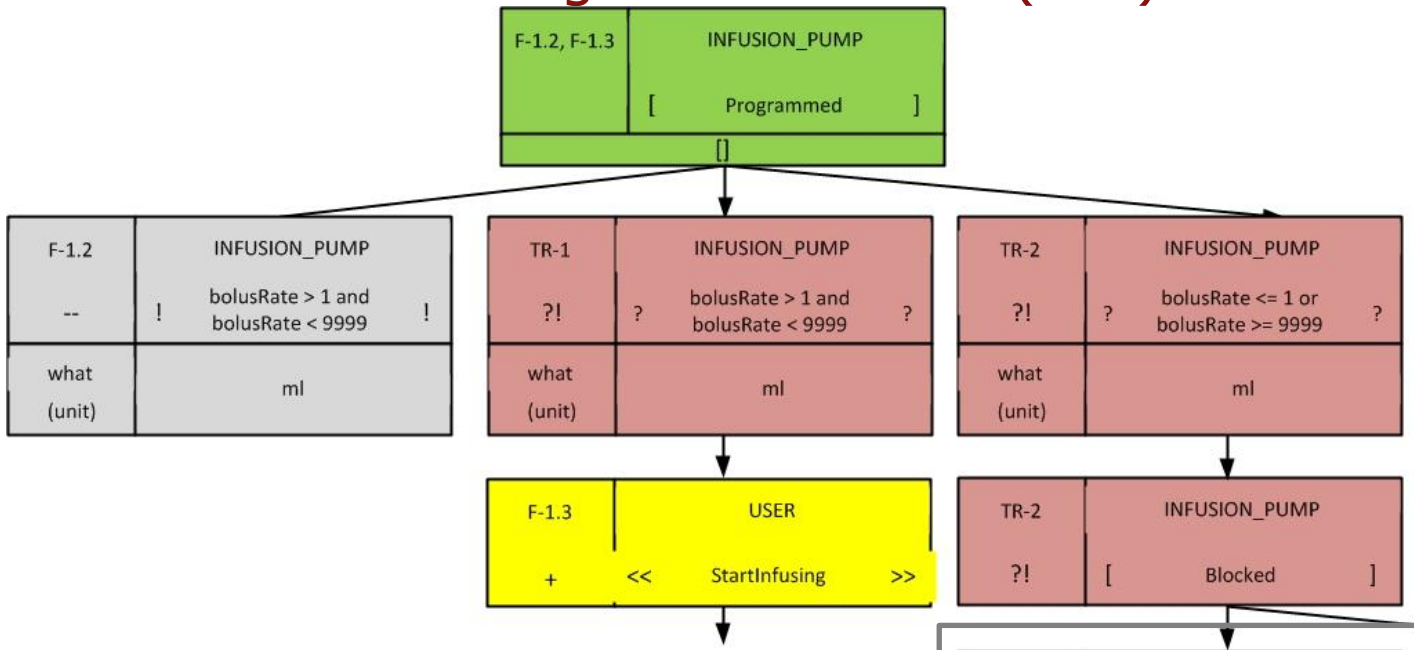


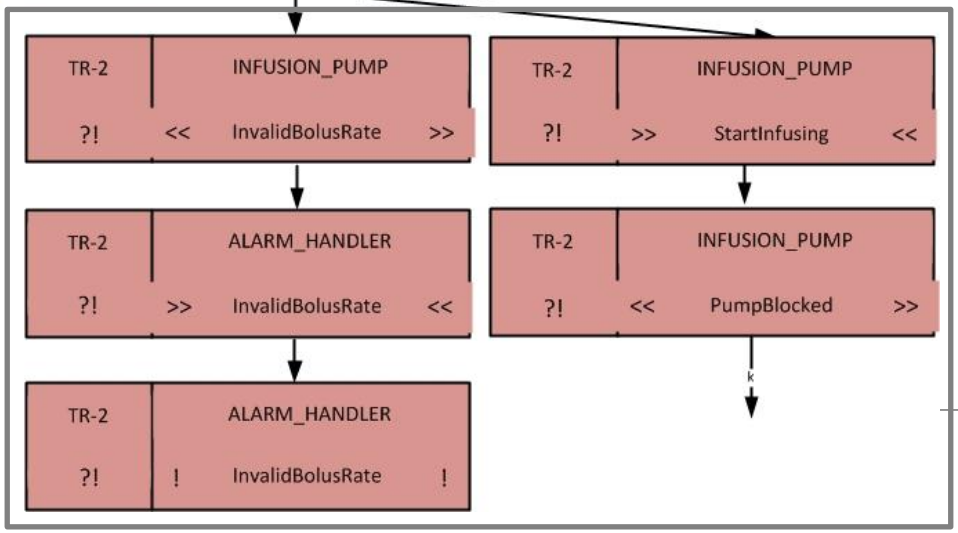
The assertion is splitted and realized into explicit conditional behavior nodes

Requirements-driven testing with behavior trees

From tIBT to Testing Behavior Trees (TBT)







Concretization of high level events to obtain a precise test specification

Requirements-driven testing with behavior trees
Exploitation of Testing Behavior Trees

- Manual exploitation
  - Since a TBT is precise, unambiguous and verifiable, a tester can easily interpret it as a test case specification for designing and/or executing test cases manually
- Automated exploitation
  - Partial system model derivation (into a GPL/DSL)
  - Test model derivation (into a test concepts-supporting GPL/DSL)
  - Test code generation
- C

Challenges and further steps
Encountered challenges

- Behavior engineering methodology is not that popular
- BML <> UML (just another undefined modeling notation?)
- Behavior trees require a different way of thinking
- Time consuming at the beginning
- Variying notations and concepts complicate first steps
- Lack of adequate, stable and freely-avaiblabe tool support

Challenges and further steps
Further steps

- Establish a UML profile for the BML
- Implement a stable tooling for our needs
- Bringing UML and BML together to benefit from each other
- Define transformation rules from BML to UML state machines

Any Questions