Formal Modeling of BPEL Workflows Including Fault and Compensation Handling

Máté Kovács, *Dániel Varró, László Gönczy*
kovmate@mit.bme.hu
Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

Research supported by Software Engineering for Service-Oriented Overlay Computers (SENSORIA) EU FET-IST project.

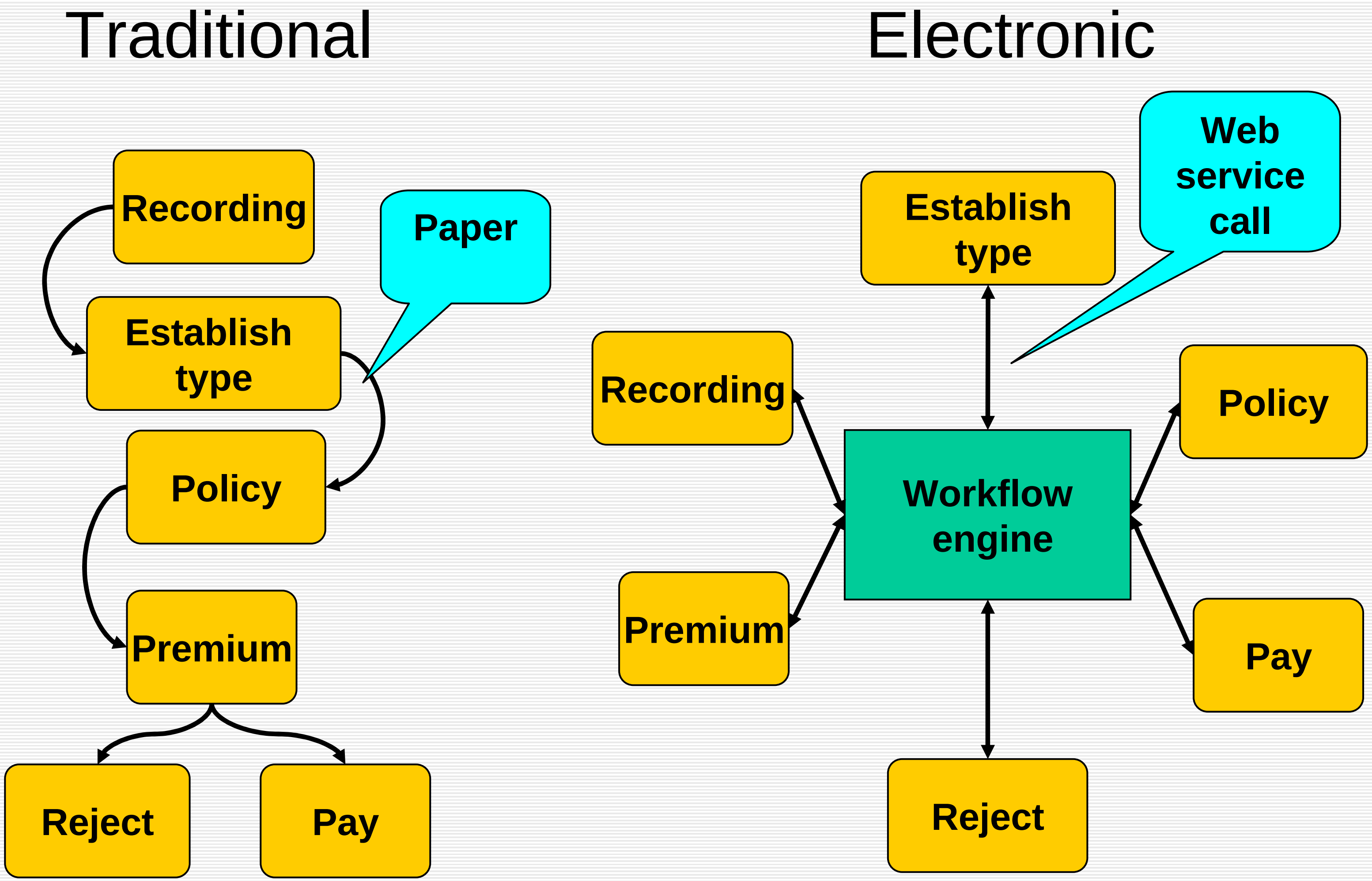
Contents

- Motivation to modeling and verifying business processes
- Short introduction to the BPEL language
- Comparison of existing approaches
- Feature presentation

Motivation

- Web service composition (e.g. BPEL)
  - Widespread tool support
  - Verification techniques still need improvement
    - Design errors of orchestration
- Our aim:
  - Check requirements on workflows formally
  - Derive formal models by model transformations

The Execution of Workflows



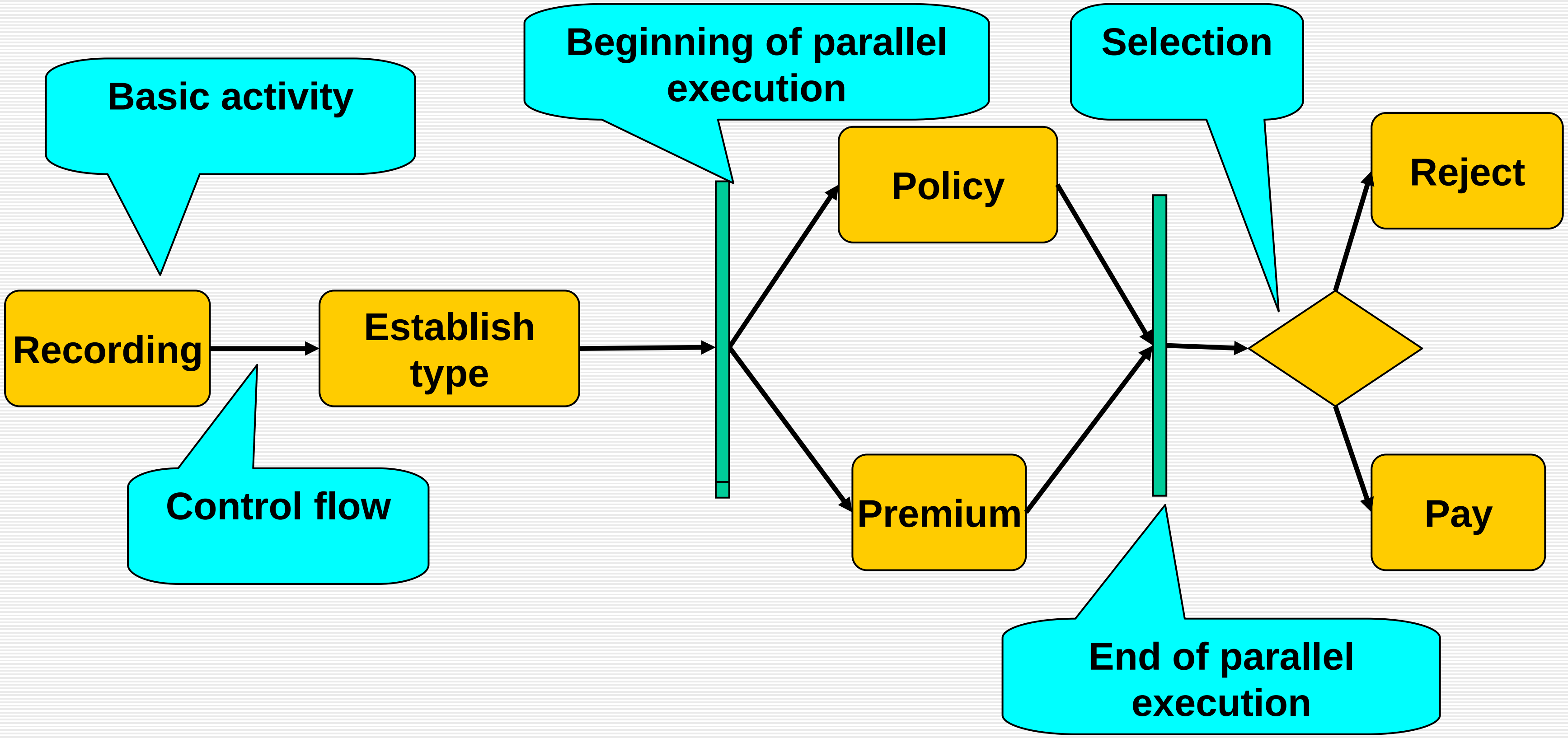
Implementing Workflows

- Languages: BPEL, XPDL
  - Very high level
  - XML based
  - Interpreted
  - No debugger provided
  - Difficult to follow the control flow of a process instance

Testing Workflows

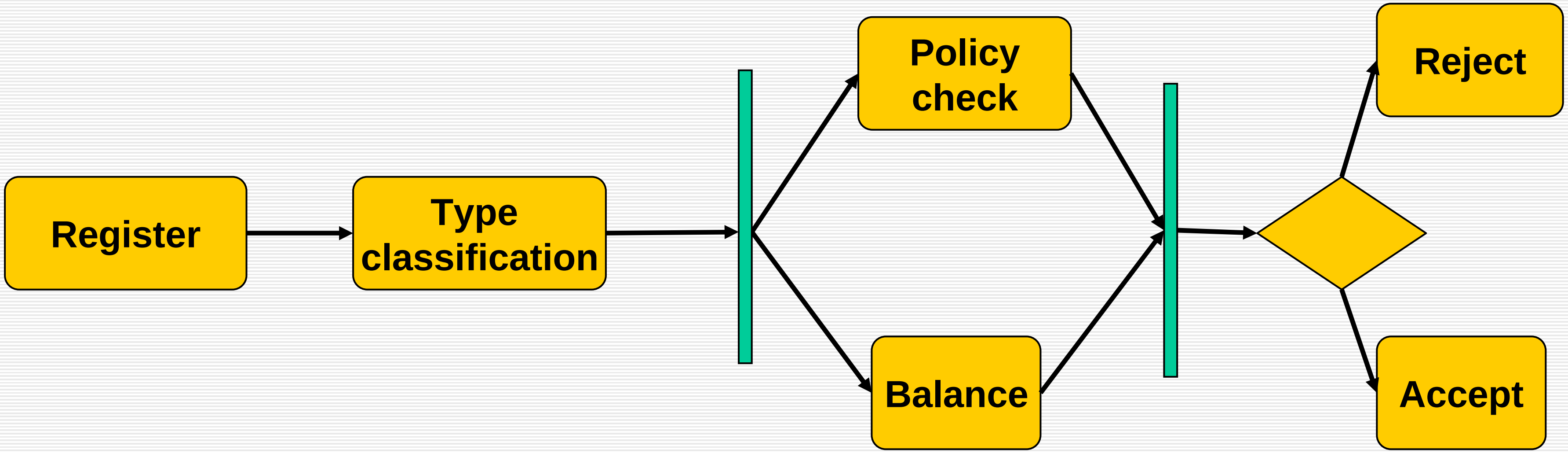
- Problem: the testing of workflows
  - The data is stored in remote databases
  - The effects of test phases have to be rolled back
- Solution: the formal analysis of workflows
  - Formal workflow semantics
  - Formal verification of properties
    - E.g. variable access
  - Fault simulation: assessment of error propagation

A Workflow Example

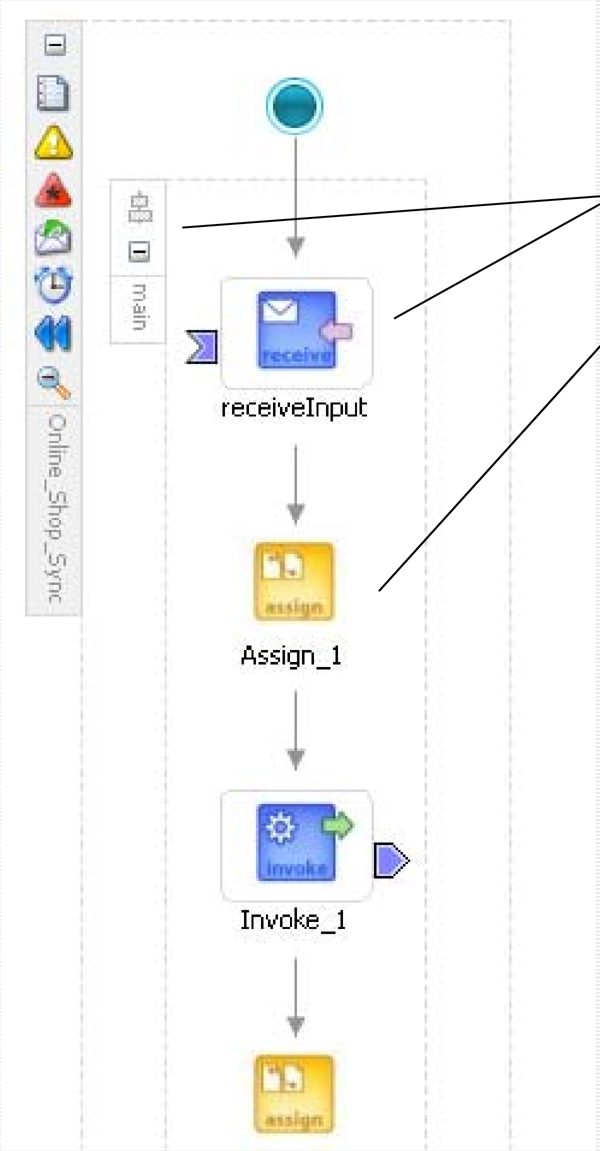


The Concepts of Workflows

- Basic activities
- Structured activities
- Data flow / control flow?

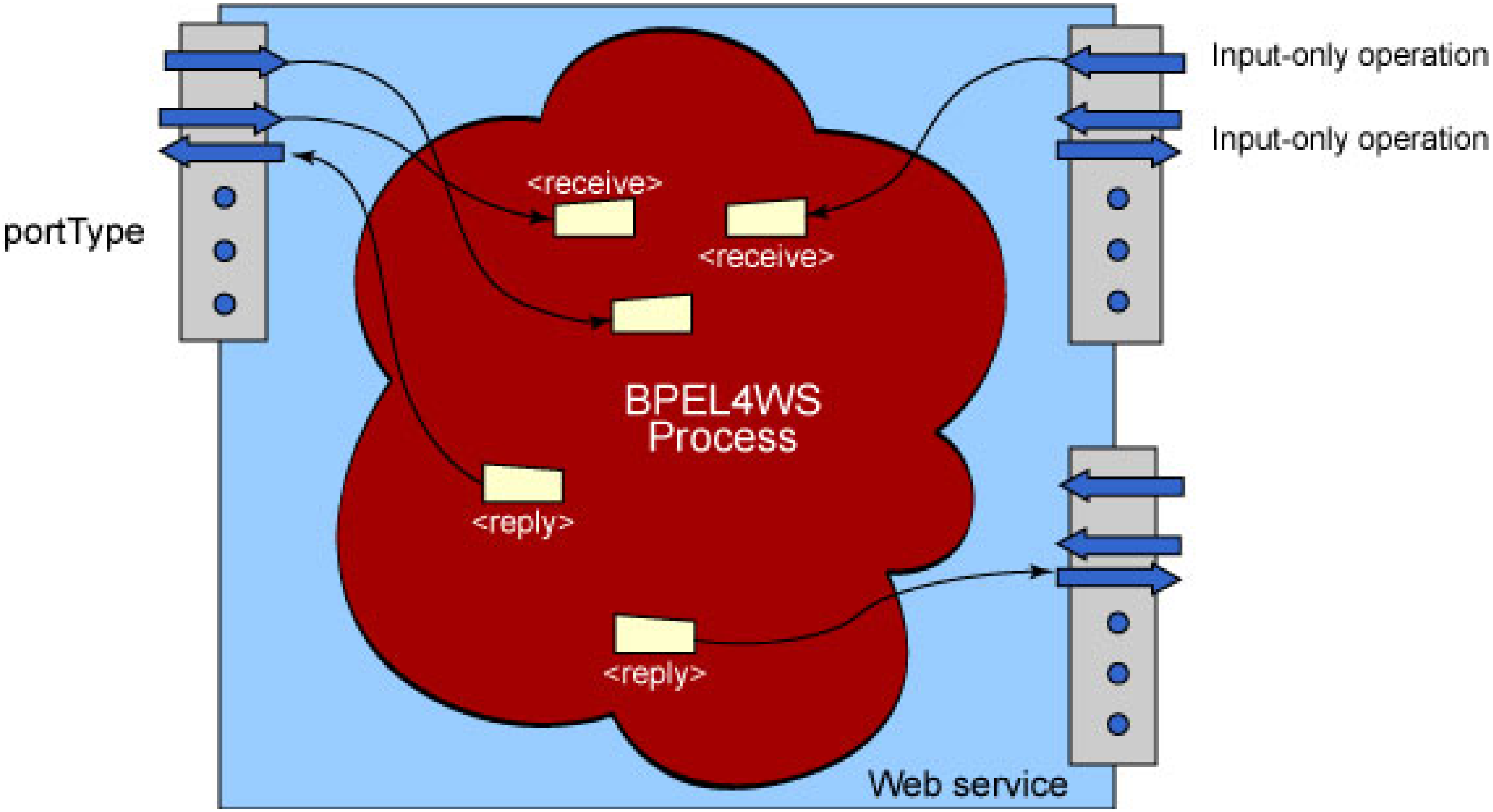


A BPEL Example



- Basic activities
- Structured activity

BPEL: Web Service Orchestration

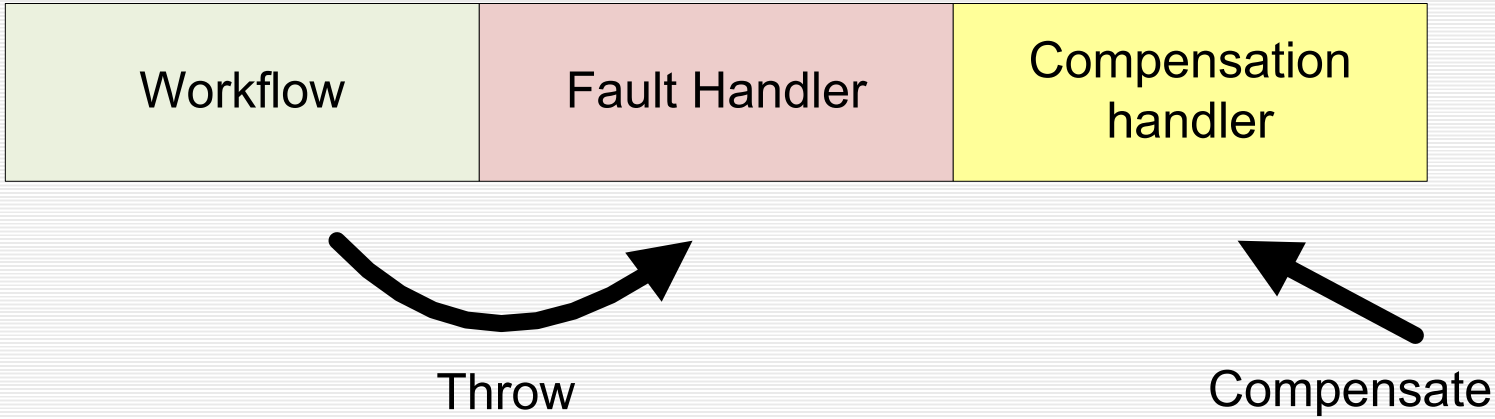


BPEL Instructions

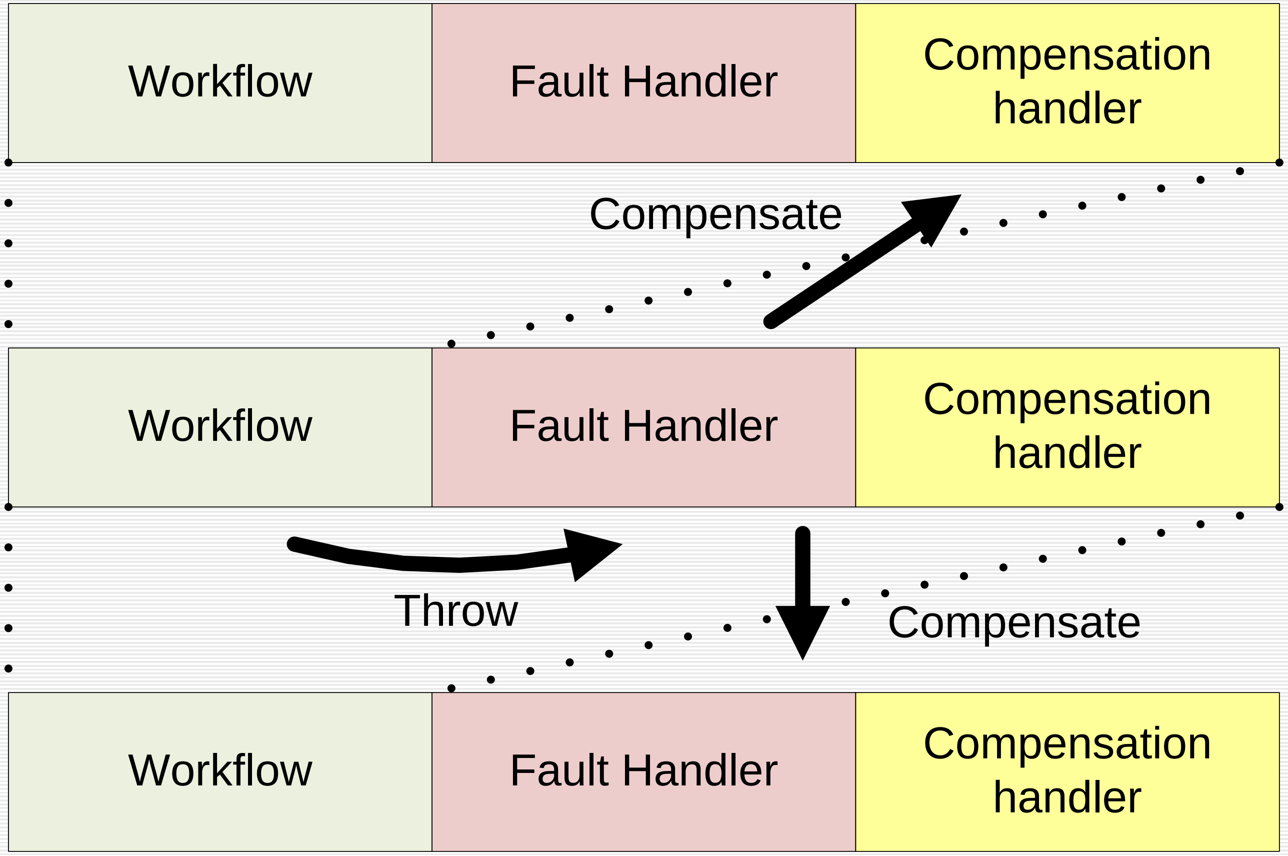
- Basic activities:
  - **`Invoke`**
  - **`Receive`**
  - **`Reply`**
  - **`Empty`**
  - **`Terminate`**
  - **`Throw`**
  - **`Compensate`**
- Structured activities
  - **`Scope`**
  - **`Sequence`**
  - **`Flow`**
  - **`While`**
  - **`Switch`**
  - **`Pick`**

Structure of BPEL

- Workflow: the main business process
- Fault handler:
  - Faults thrown in workflow
  - User defined or default handler
- Compensation handler:
  - Initiated from outside
  - User defined or default handler



Scope hierarchy



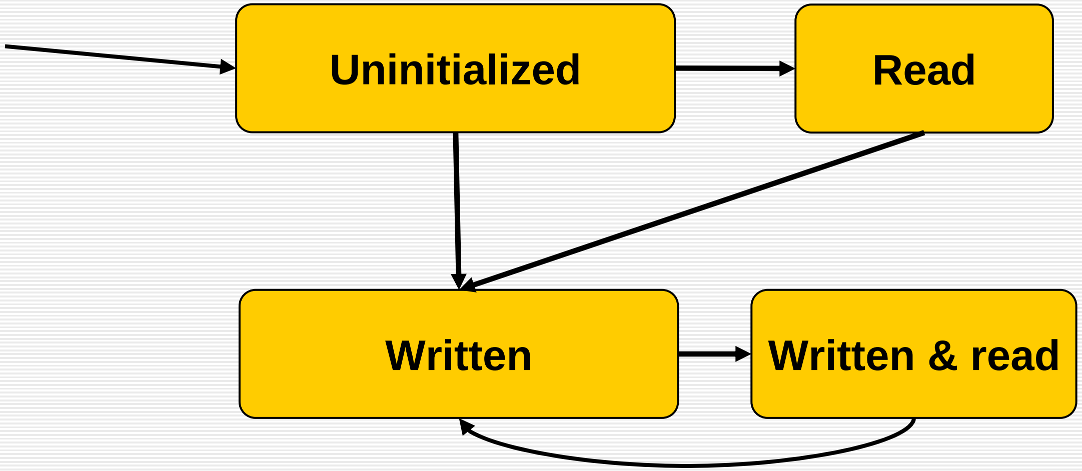
Some Existing Approaches

- W.van der Aalst and K. van Hee. Workow Management Models, Methods, and Systems. The MIT Press, 2002.
  - Regular Petri net modeling basic workflows
- S. Nakajima. Model-checking behavioral specication of BPEL applications. Electr. Notes Theor. Comput. Sci.,151(2):89105,2006.
  - BPEL modeling with Extended Finite-State Automata: event and fault handling is not considered

Some Existing Approaches

- M.Kovacs and L.Gonczy. Simulation and formal analysis of workflow models. In GT-VMT, pages 215-224, 2006.
  - Modeling formalism: dataflow networks
  - Limited success w.r.t. the covering of event handling
- S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri Nets. In W.M.P.v.d.Aalst, B. Benatallah, F.Casati ,and F.Curbera, editors, Proceedings of the Third International Conference on Business Process Management (BPM2005), volume 3649 of Lecture Notes in Computer Science, pages 220-235, Nancy, France, Sept. 2005. Springer-Verlag.
  - Petri net model covering the entire BPEL semantics
  - One of the most extensive modeling approach w.r.t. BPEL features
  - The semantics of compensation handling is over approximated / generalized

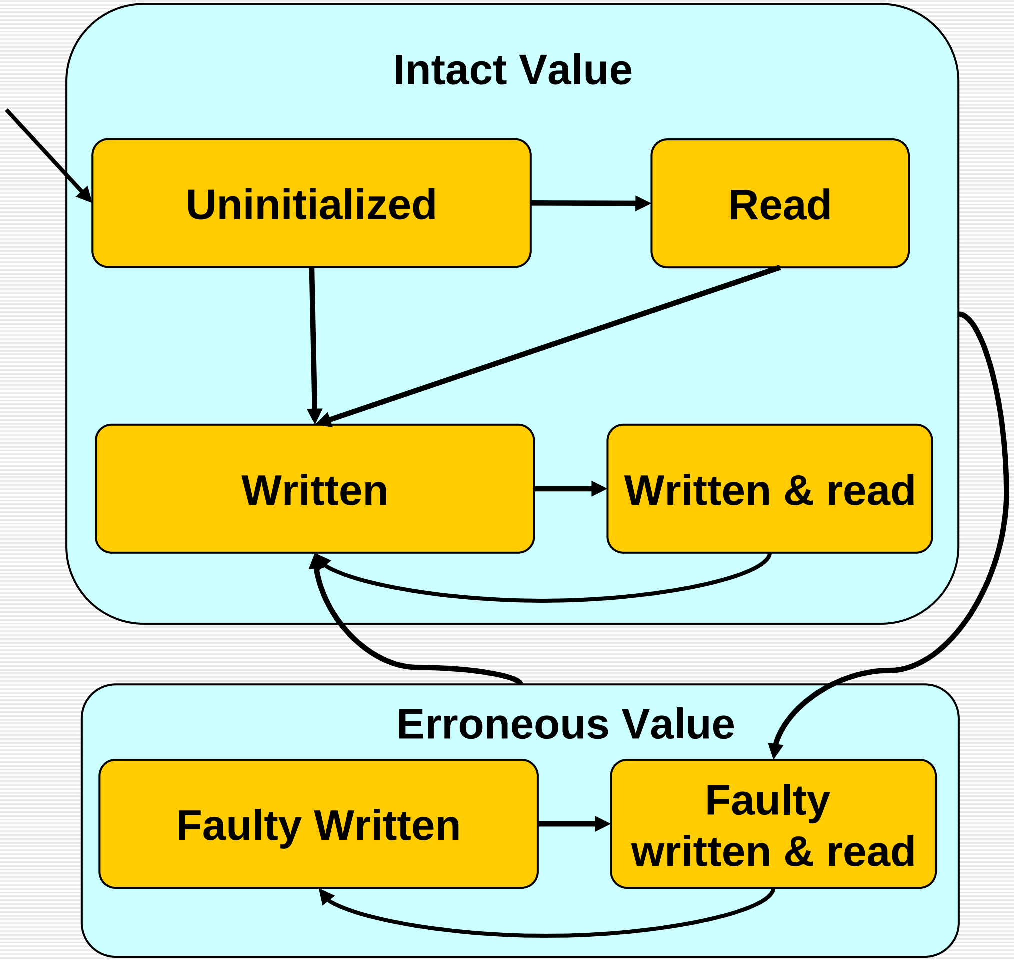
Modeling the Behaviour of Variables



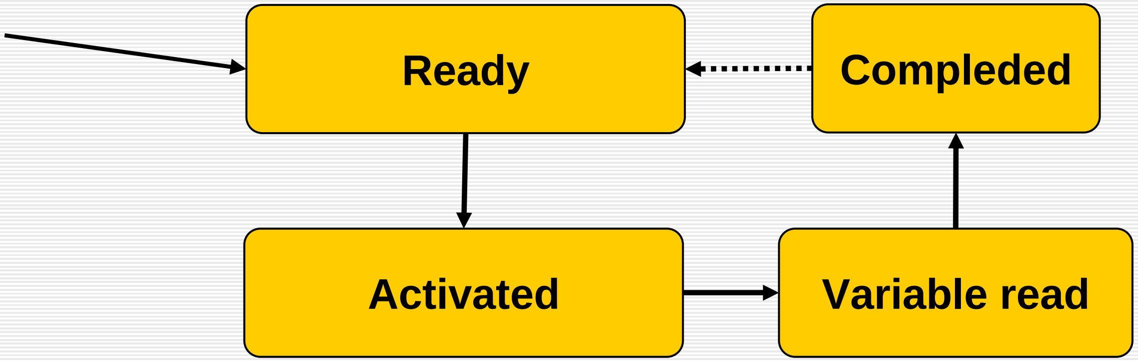
- Information carried:
  - If the variable contains data
  - If it has already been used

Fault Model and Error Propagation

- Error is probagated by basic activities (read-write)



Modeling Basic Activities

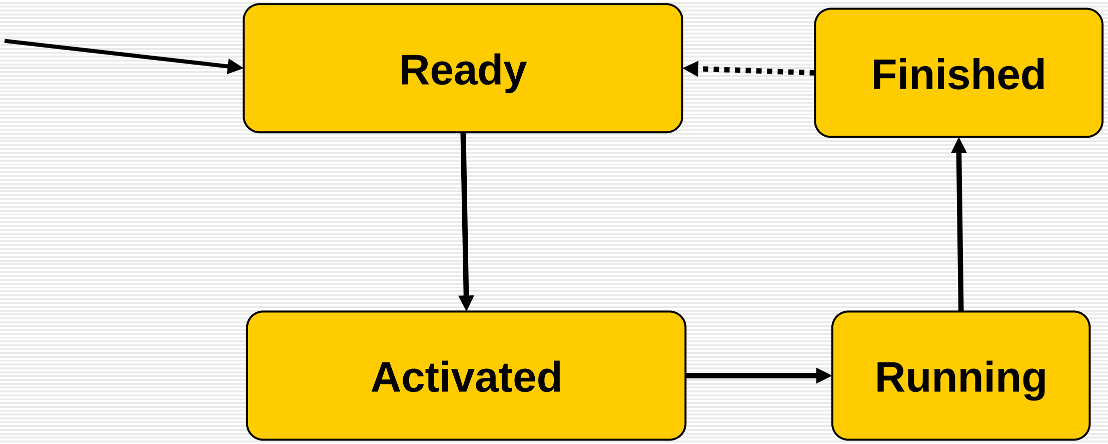


- Activated: the control reached the activity
- Dotted arrow: tiggers when the containing activity finishes

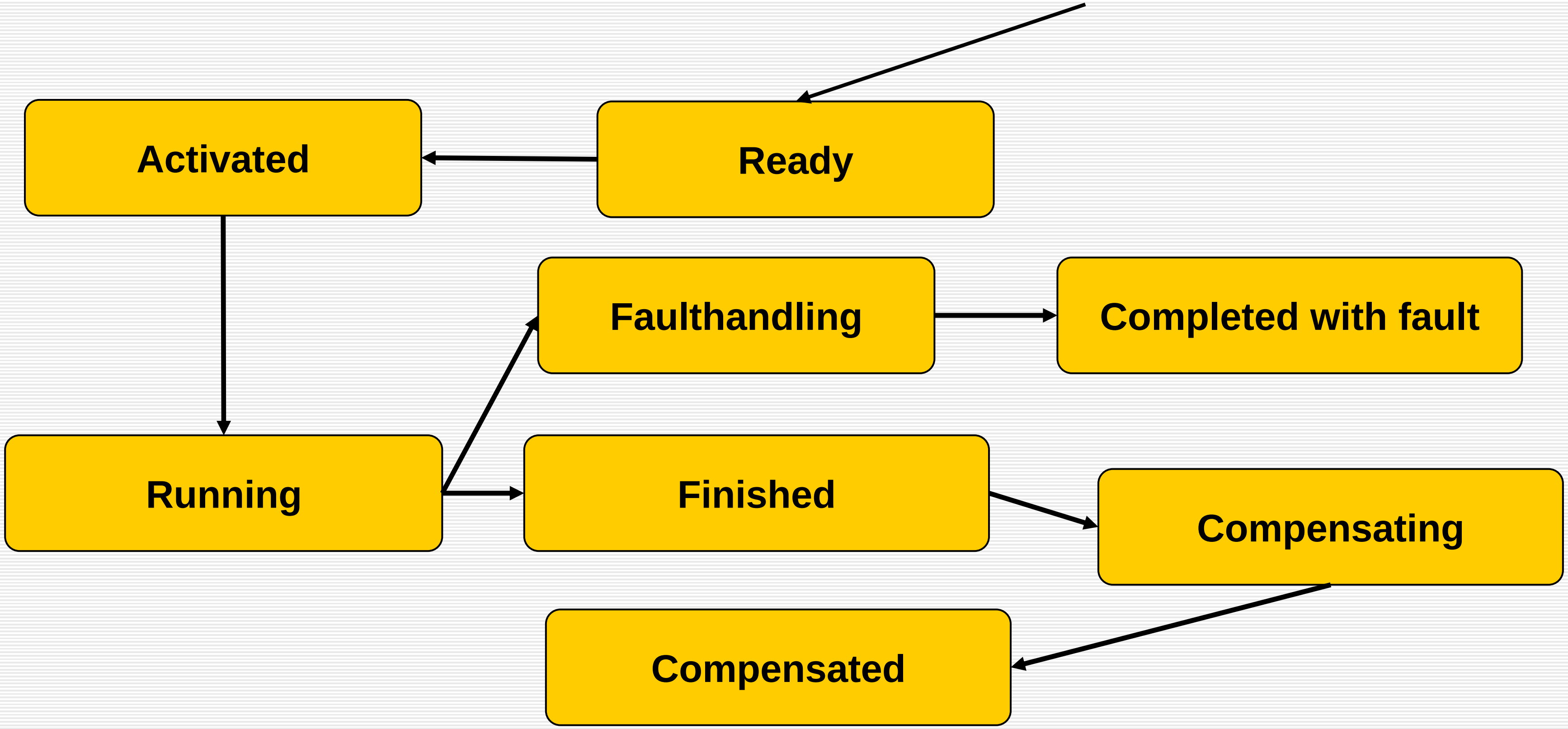
Modeling Structured Activities

```
…
<sequence>
 <invoke name="a"/>
 <invoke name="b"/>
</sequence>
```

```
…
sequence=running AND invoke_a=ready →
   invoke_a=actviated;
sequence=running AND invoke_b=ready AND
   invoke_a=finished → invoke_b=activated;
sequnece=running AND invoke_b=finished →
   sequence=finished;
```

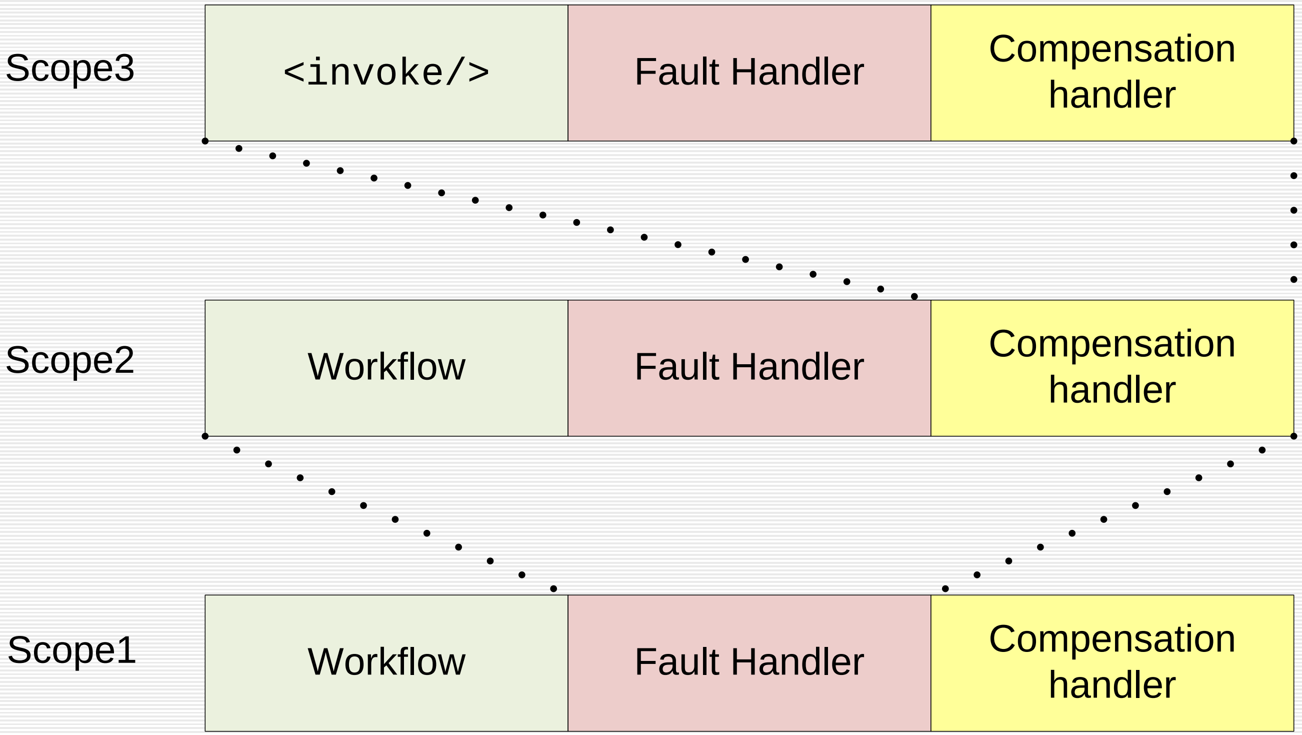


Modeling Scopes



- Restriction: scopes may not be executed in an iterative manner.

Constraints of Activity Triggering



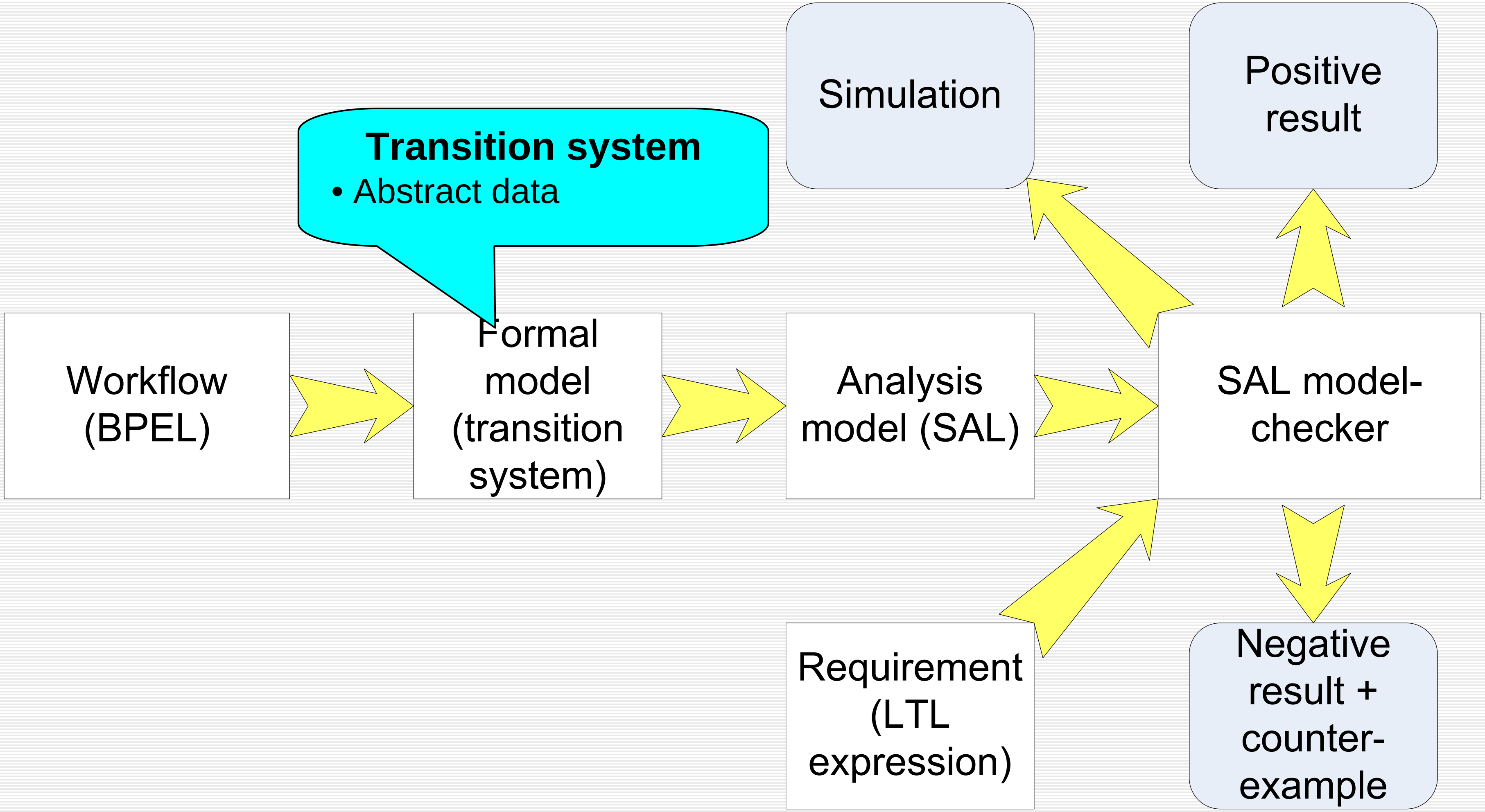
Scope3 <invoke/> Fault Handler Compensation handler

Scope2 Workflow Fault Handler Compensation handler

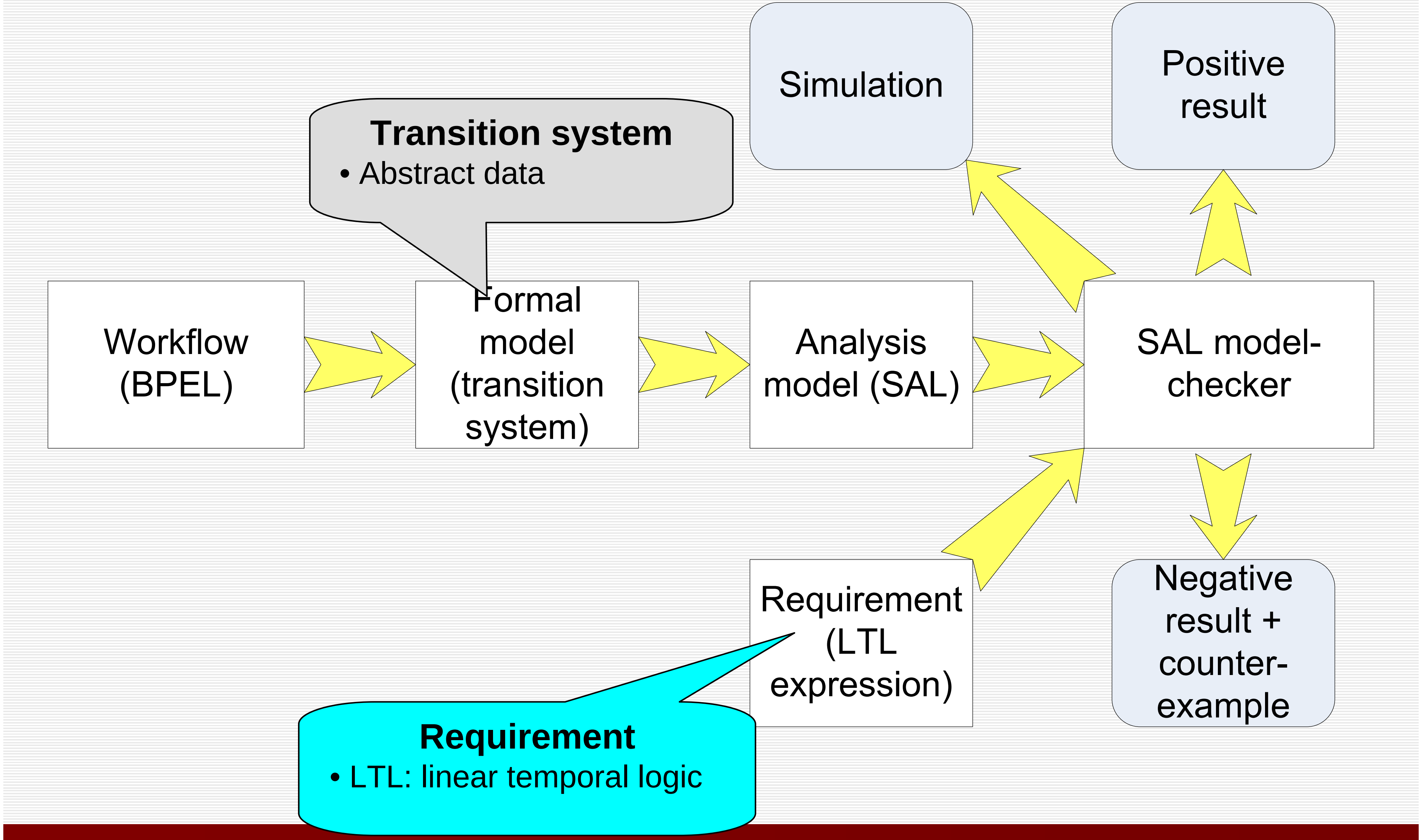
Scope1 Workflow Fault Handler Compensation handler

```
… AND scope_1=faulthandling AND
  scope_2=compensating AND
  scope_3=running AND …
```

Prototype Implementation



Prototype Implementation

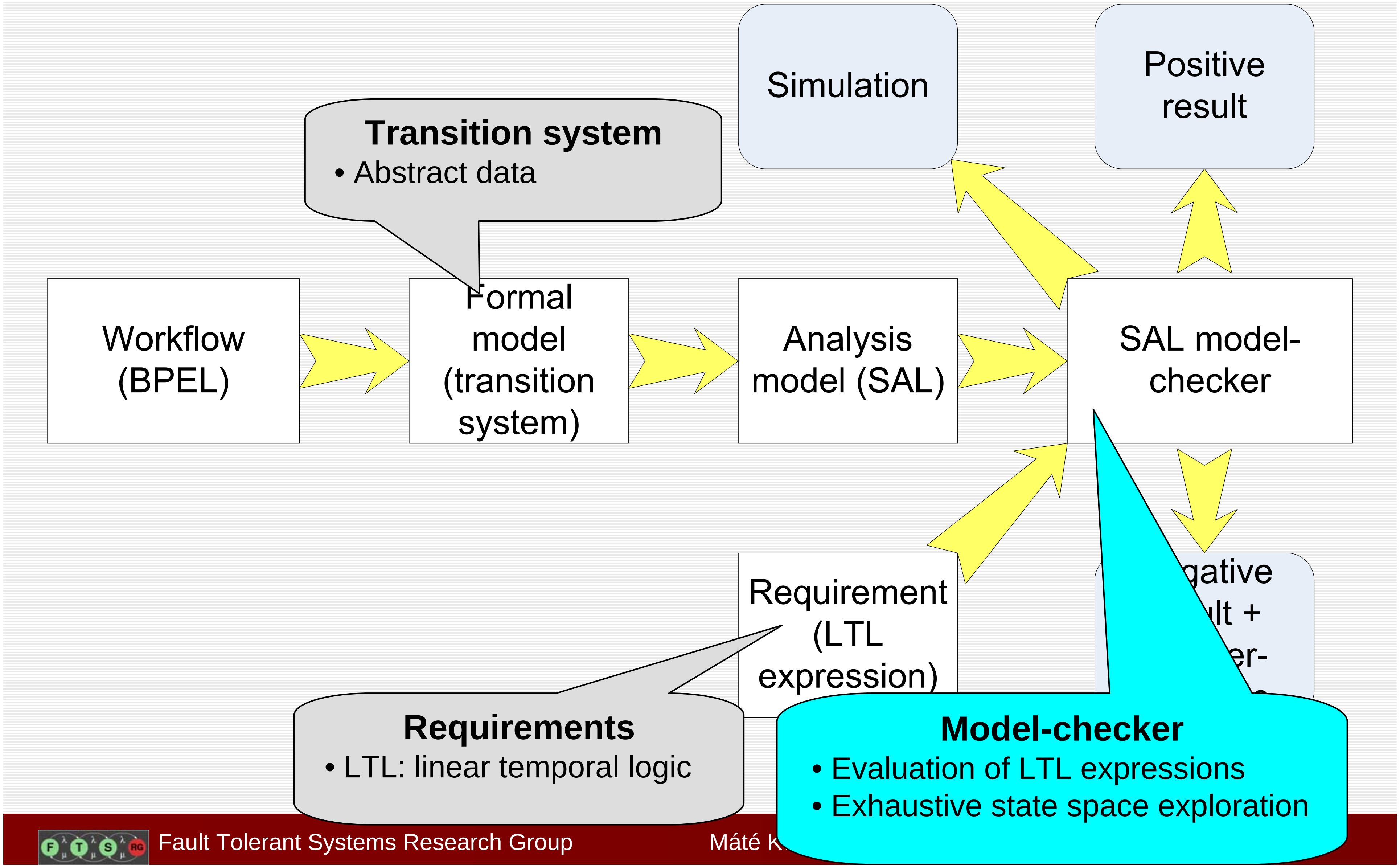


Transition system
- Abstract data

Requirement
- LTL: linear temporal logic

Simulation
Positive result
Workflow (BPEL)
Formal model (transition system)
Analysis model (SAL)
SAL model-checker
Requirement (LTL expression)
Negative result + counter-example

Prototype Implementation



Simulation

Positive result

Transition system
- Abstract data

Workflow (BPEL)

Formal model (transition system)

Analysis model (SAL)

SAL model-checker

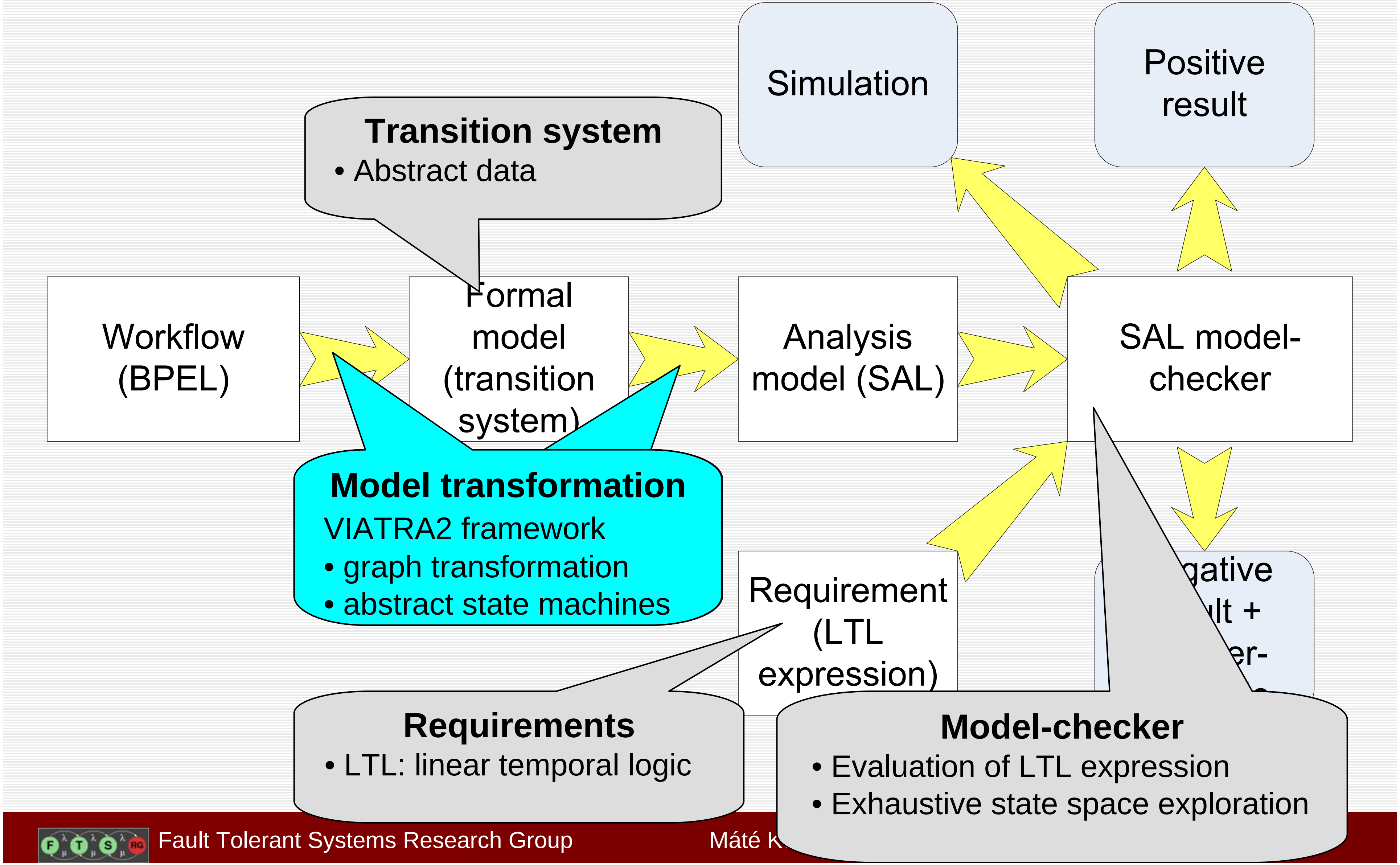
Requirement (LTL expression)

gative ult + er-

Requirements
- LTL: linear temporal logic

Model-checker
- Evaluation of LTL expressions
- Exhaustive state space exploration

Prototype Implementation



Simulation

Positive result

Transition system

- Abstract data

Workflow (BPEL)

Formal model (transition system)

Analysis model (SAL)

SAL model-checker

Model transformation

VIATRA2 framework

- graph transformation
- abstract state machines

Requirement (LTL expression)

gative ult + er- e

Requirements

- LTL: linear temporal logic

Model-checker

- Evaluation of LTL expression
- Exhaustive state space exploration

Preliminary Results

- Verification of a Online Shop process:
  - 10 structured activities
  - 27 basic activities
- Results
  - Negative results within 3-5 minutes
  - The proof of positive cases takes n*10 minutes

Plans for the Future

- Algorithmical generation of common requirements:
  - Uninitialized variables are never read
  - Synchronous processes never end without an answer
- Modeling the composition of multiple BPEL workflows
- Back annotation to workflow editors

Thank you for your attention!