MODELING TECHNIQUE FOR MODEL BASED ERROR LOCALIZATION AND ERROR REMOVAL BASED ON EXTENDED UML ACTIVITY DIAGRAMS

Wolfgang Fengler, Alexander Pacholik*, Maria Mologina**

* Ilmenau Technical University
  wolfgang.fengler / alexander.pacholik@tu-ilmenau.de

** Moscow Power Engineering Institute (Technical University)
Topics

1. INTRODUCTION
2. FORMAL DEFINITION OF ACTIVITY DIAGRAM SEMANTICS
3. DUAL BEHAVIOR OF ACTIVITY DIAGRAMS
4. ERROR HANDLING BY DYNAMIC RESTRUCTURING
5. SUMMARY AND OUTLOOK
INTRODUCTION(1)

- Typically control systems can be modeled by discrete event models.
- Failure events and failure states of a discrete event system are often not externally visible.
- Thus failure handling should be integrated in the system design.
- In more complex systems or unknown environments we cannot catch all combinations of causes and system states by design.
- If there are too many combinations, manual intervention is impossible or the internal state is not observable by default.
- We need a mechanism for error localization and algorithms for an appropriate error removal strategy.
INTRODUCTION(2)

- We use a high level specification language, namely UML activity diagrams, to invite the method.
- The error localization strategy is based on the work in (Lautenbach, 2003)
- Extended by a mechanism to define exceptional activities which can contain error removal methods.
FORMAL DEFINITION OF ACTIVITY DIAGRAM SEMANTICS

- The UML is defined within a meta model, which is a class diagram.
- The semantics of behavioral diagrams is described by text and thus informal.
- Mostly UML diagrams are formalized by specifying transformation rules from UML diagrams into other formal descriptions.
- The formal semantics given here is analogous to Petri nets.
Basic Activity Diagrams

AD = (S, T, F)

S: is a finite set of states, consists of two disjoint subsets AS (action states) and PS (pseudo states)

T: is a finite set of transitions:
  - Normal transition (NT)
  - Decision transition (DT)
  - Merge transition (MT)
  - Fork transition (BT)
  - Join transition (ST)

F: is the flow relation: Every diagram element is connected to at least with one or more elements of the other element type.
Basic Activity Diagrams: Example

AD = (S, T, F) with
S = {{I, E}, {A1, A2, A3, A4, A5, A6, A7}},
T = {NT1, BT1, DT1(b11, b12), MT1, ST1, NT2},
F = {{I, NT1}, (NT1, A1), (A1, BT1), (BT1, A2), (BT1, A3), (A2, DT1), (DT11, A4), (DT12, A5), (A4, MT1), (A5, MT1), (MT1, A6), (A3, ST1), (A6, ST1), (ST1, A7), (A7, NT2), (NT2, E)}. 
S-marking and Firing Rule

Decision transition: pre
S-marking and Firing Rule

Decision transition: post
S-marking and Firing Rule

Join transition: pre
S-marking and Firing Rule

Join transition: post
DUAL BEHAVIOR OF ACTIVITY DIAGRAMS

- Dual behavior has been defined for Petri nets (Müller, 2004) (Lautenbach, 2003).
- T-Marking: $M_T: T \rightarrow N$
  - $T$: is a finite set of transitions,
  - $N$: is the set of natural numbers.
- T-marked activity diagram:
  $DM = (S, T, F, M_{T0})$ such that:
  - $S, T, F$: is a simple activity diagram,
  - $M_{T0}$: is the initial transition marking.
Firing Rules of the States

A state $s_0$, leading to transition $T_0$ can fire, if for all states leading to $T_0$, including $s_0$, all resulting states are marked.
Firing Rules of the States

c)
Complete Activity Diagram Marking

\[ M = M_T \cup M_S \]

- Transition \( T \) can fire, if it can fire in the s-marked activity diagram and \( T \) doesn’t contain a T-Marking: \( M_T(t) = 0 \)
- The state \( s \) can fire, if it can fire in the t-marked activity diagram and \( s \) doesn’t contain an s-marking.
- If the states need to fire simultaneously and one of these states is marked, none of them can fire.
Example of a Complete Marked Activity Diagram

The transitions NT1 and NT4 can fire in the s-marked diagram and the states s2₀ and s₂₁ for the T-marked.

In the dual marked diagram M(NT4) = 1, so s₂₁ can’t fire. Also M(s₂₁) = 1, thus NT4 can’t fire as well.
ERROR HANDLING BY DYNAMIC RESTRUCTURING

Dependent transition AT:

Transition \( AT(T) \) is dependent on transition \( T \), if \( AT \) is only active in the case of \( M_T(t) = 1 \). This states, the transition \( AT(T) \) can only fire, if \( T \) contains a T-token.
Example of a Dependent Transition

S$_2^0$ and ANT2(MT1) can fire
Example of a Dependent Transition

After switching of S2₀ and ANT2 we get this marking
Summary

- Formal, Petri net related definition of UML activity diagrams
- Formal model of dual behavior in terms of activity diagrams
- New transition type ‘dependent transition’
- Method to use dual behavior for error localization and error correction by dependent transitions
- Methodology based on UML activity diagrams to derive an implicit fault tree automatically, based on the system model.
- This implicit fault tree can then be used to add fault removal strategies.
Outlook

- There is a vision for technical environments to improve fault tolerance and robustness by establishing self organization.
- This means to use self organization methods with deterministic results.
- Dealing with self organization in technical systems we also need to make the features to be self organized explicit.
- We expect this can be applied by extending our method.
- For the future we plan to examine different selection strategies as well as strategies for permanent model restructuring.
Thank you.