**MOTIVATION**

Modeling allows to formally capture the syntactical structure of application domains

Validation of (static) semantics is still a challenge

OCL often not powerful enough while formal methods require much effort to learn

**OBJECTIVES**

- Enable definition and calculation of the information flow in arbitrary models
- Adaptability to meta model refactorings through local calculation and propagation
- Support fixed-point semantics to account for cyclic paths in the model’s information flow

**APPROACH**

Data-flow analysis (DFA) is commonly used to derive a program’s static properties by applying a fixed-point analysis to its control flow graph.

Since models comprise a graph structure, the DFA concept is applicable to the modeling domain.

Introduce the notion of model-based DFA

**Tasks**

- Alignment and Definition
  - Compare application domains (compiler construction, modeling) to devise a suitable methodology
- Fixed-point Analysis
  - Develop algorithm for efficiently solving model-based DFA equation systems
- Evaluation, Proof of Feasibility
  - Compare complexity, performance and implementation effort to other approaches
- Tooling and Use Cases
  - Provide tooling (DFA-Solver, IDE) and example use cases

**USE CASES**

Control Flow Analysis

- Attribution for determining reachability (textual representation):
  - `ActivityNode::isReachable` (recursive) semantic rule at node superclass
  - `InitialNode::isReachable` (overwrite ActivityNode rule at InitialNode)

Business Process Analysis

- DFA-based decomposition into Single-Entry-Single-Exit components
- Detection and completion of quasi-structured components
- Heuristic-based validation and transformation into block structure

**TOOLING**

Model Analysis Framework (MAF)

- Based on the Eclipse Modeling Framework (EMF)

**REFERENCES**

1. Christian Saad and Bernhard Bauer: Data-flow based Model Analysis, Second NASA Formal Methods Symposium, April 2010