JKU

LINZ INSTITUTE
OF TECHNOLOGY

Heinrich Heine Universität Düsseldorf

LINZ INSTITUTE OF TECHNOLOGY

LIT Secure and Correct Systems Lab

ISSE

IVOIRE Project Results

Atif Mashkoor

IVOIRE Workshop
June 25, 2024
Bergamo, Italy

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

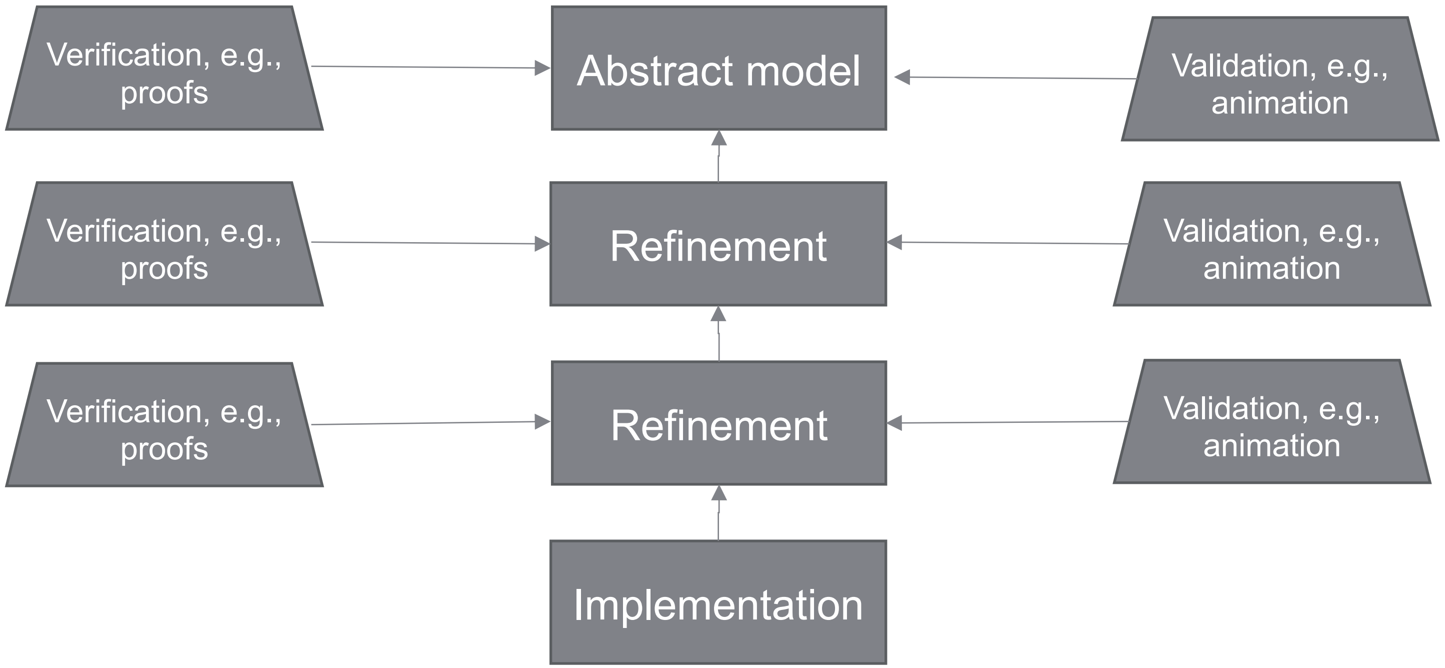
Project team members

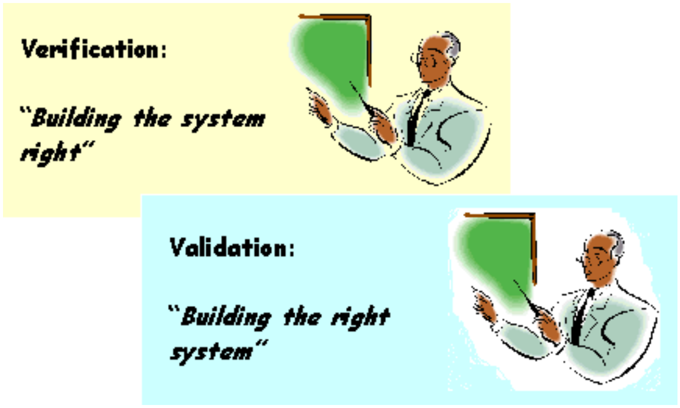
- PIs
  - Atif Mashkoor
  - Michael Leuschel
  - Alexander Egyed
- PhD students
  - Sebastian Stock
  - Fabian Vu
  - David Geleßus
- Time frame: Oct 2020 – Sep 2024

Motivation

- The use of formal methods is highly recommended for quality assurance (QA) of safety-critical systems
- Techniques for programs specification, development and reasoning about their correctness based on mathematics and logic
- Verification often takes the center stage
- Validation is somehow neglected, especially in the stepwise refinement process

Stepwise model development





Proof obligations vs validation obligations

- Proof obligation (PO) is a logical formula associated with the consistency claim of a given verification property
- Verification(specification) = Σ POs(specification)
- Analogous to the idea of PO, we propose to break the overall validation of a specification and associate it with each refinement step
- A validation obligation (VO) is a logical formula associated with the correctness claim of a given validation property
- Validation(specification) = Σ VOs(specification)

Validation obligation

A validation obligation (VO) formally represents the connection between a requirement, a model, and one or more validation tasks.

req/model : tasks

Req1 $\triangleq$ G\{moving = TRUE $\Rightarrow$ door_open = FALSE\}

a lift only moves when its doors are closed

LTL1 $\triangleq$ LTL(G\{moving = TRUE $\Rightarrow$ door_open = FALSE\})

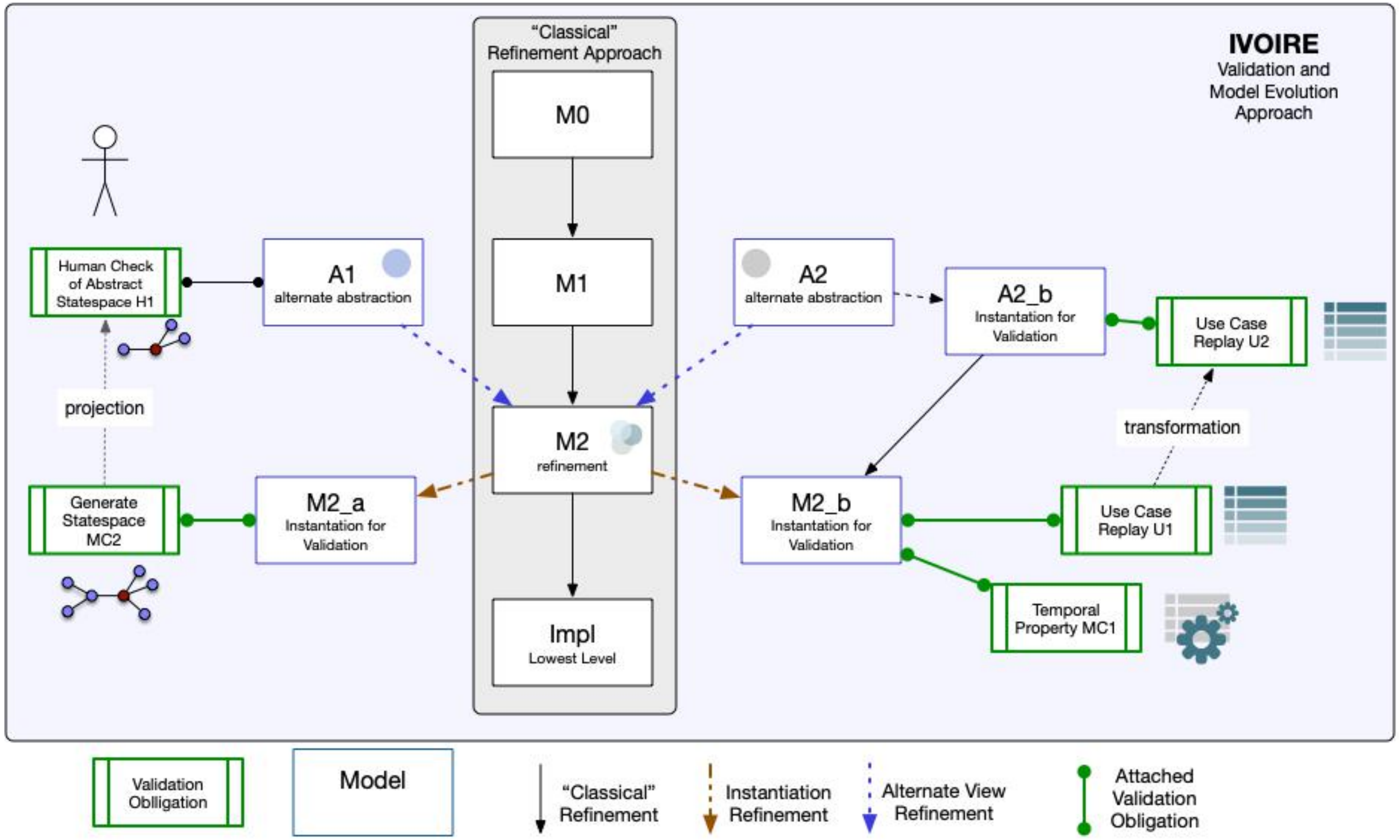
Req1/Lift : LTL1

Atif Mashkoor, Michael Leuschel, Alexander Egyed: Validation Obligations: A Novel Approach to Check Compliance between Requirements and their Formal Specification. ICSE (NIER) 2021: 1-5

Validation tasks

Type	Technique	Parameters	Automation	Result type(s)
TR	Trace replay/animation	Trace T, task(s) *expr* returning trace	Automatic	{status, statespace, trace}
SIM	Simulation	Property P	Automatic	{status, statespace}
MC	Explicit-state model checking	Property P, task(s) *expr* returning trace	Automatic	{status, statespace} or {status, statespace, trace}
LTL	LTL model checking	LTL property P, task(s) *expr* returning trace	Automatic	{status, statespace} or {status, statespace, trace}
CTL	CTL model checking	CTL property P, task(s) *expr* returning trace	Automatic	{status, statespace} or {status, statespace, trace}
SMC	Symbolic model checking	Property P	Automatic	{status, statespace} or {status, statespace, trace}
PO	Proving	Proof P	Automatic	{status}
VIS	Inspection of visualization	Visualization V, task(s) *expr*	Manual	{status}
STAT	Inspection of statistics	Statistics S, task(s) *expr*	Automatic	{status}
TAB	Inspection of table	Table T, task(s) *expr*	Automatic	{status}
root	None (returns initial state)	None	Automatic	{status, statespace, trace}

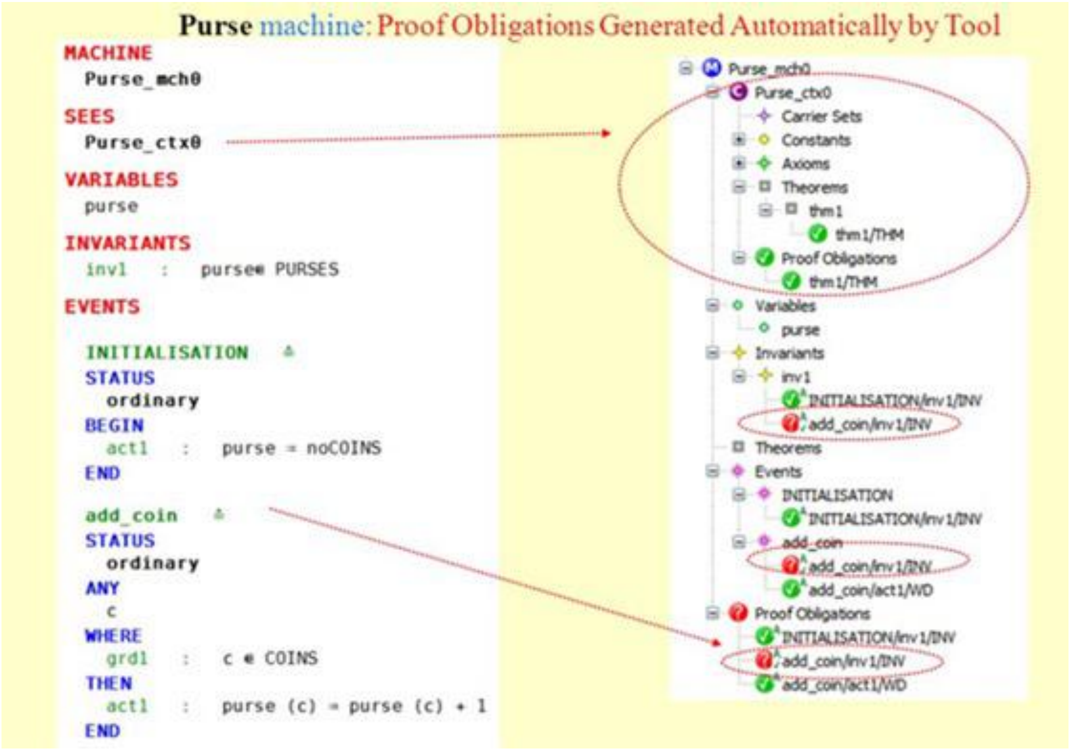
IVOIRE



"Classical"
Refinement Approach
IVOIRE
Validation and
Model Evolution
Approach
M0
M1
M2
refinement
Impl
Lowest Level
A1
alternate abstraction
A2
alternate abstraction
A2_b
Instantiation for
Validation
M2_a
Instantiation for
Validation
M2_b
Instantiation for
Validation
Human Check
of Abstract
Statespace H1
projection
Generate
Statespace
MC2
Use Case
Replay U2
transformation
Use Case
Replay U1
Temporal
Property MC1
Validation
Obligation
Model
"Classical"
Refinement
Instantiation
Refinement
Alternate View
Refinement
Attached
Validation
Obligation

Rigorous method (Event-) B

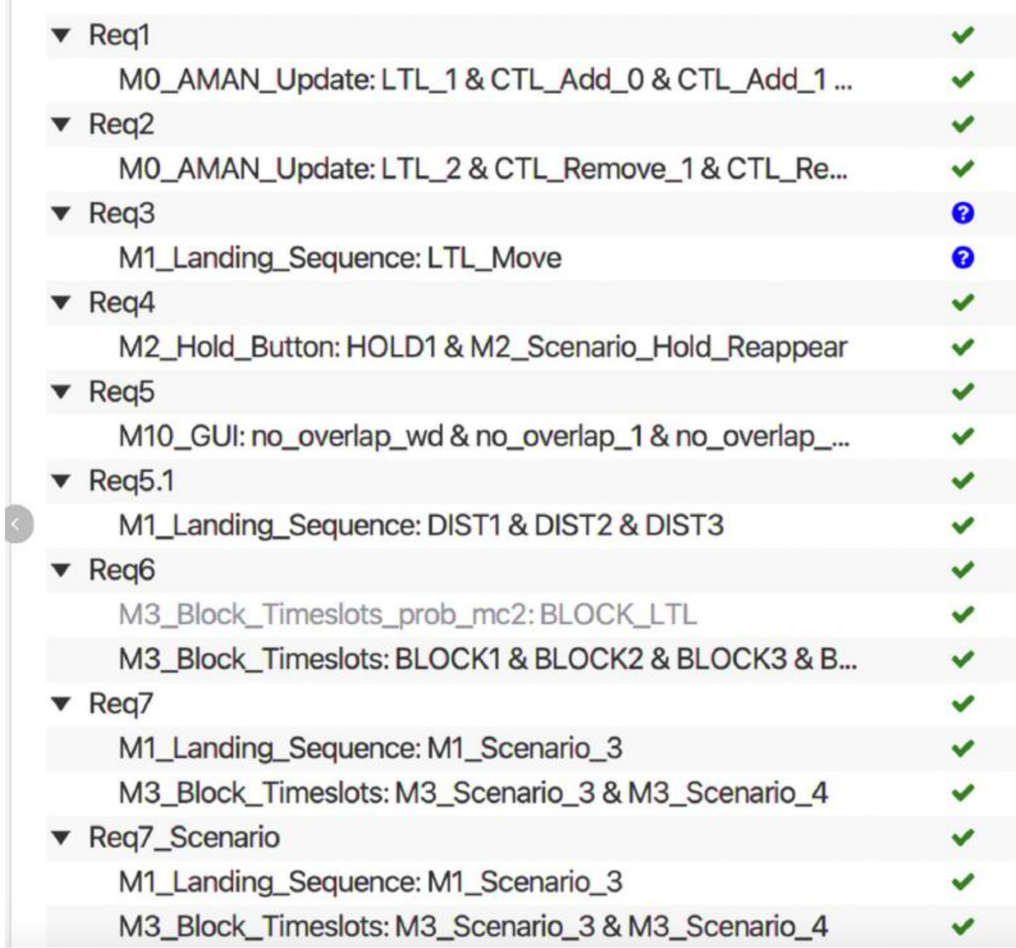
- Set theory and first-order logic
- 1-1 level of refinement, higher degree of automatic proofs
- Correctness by design
- Old and proven, much industrial experience
- Good tool support, esp. for verification
- ProB



Atif Mashkoor, Felix Kossak, Alexander Egyed: Evaluating the suitability of state-based formal methods for industrial deployment. Software: Practice & Experience 48(12): 2350-2379 (2018)

VO Manager in ProB2-UI

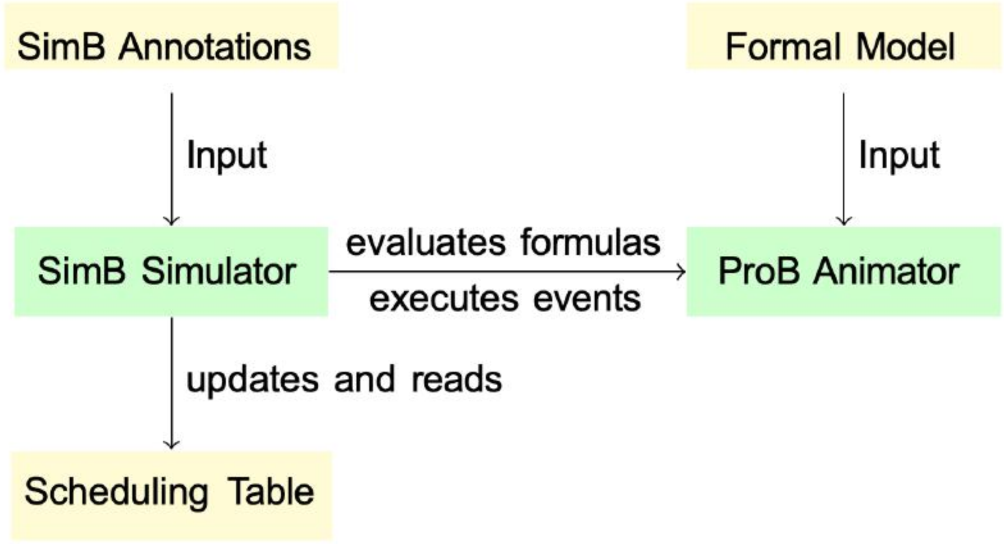
- Tool support for validation obligations
- The user defines VOs to link requirements to formal models and validation tasks
- Supports all verification/validation techniques in ProB2-UI
- Automated checking of entire projects (except tasks that require human validation)



Jens Bendisposto et al. “ProB2-UI: A Java-based User Interface for ProB.” In: Proceedings FMICS. LNCS 12863. 2021, pp. 193–201.

SimB: Timed Probabilistic Simulation

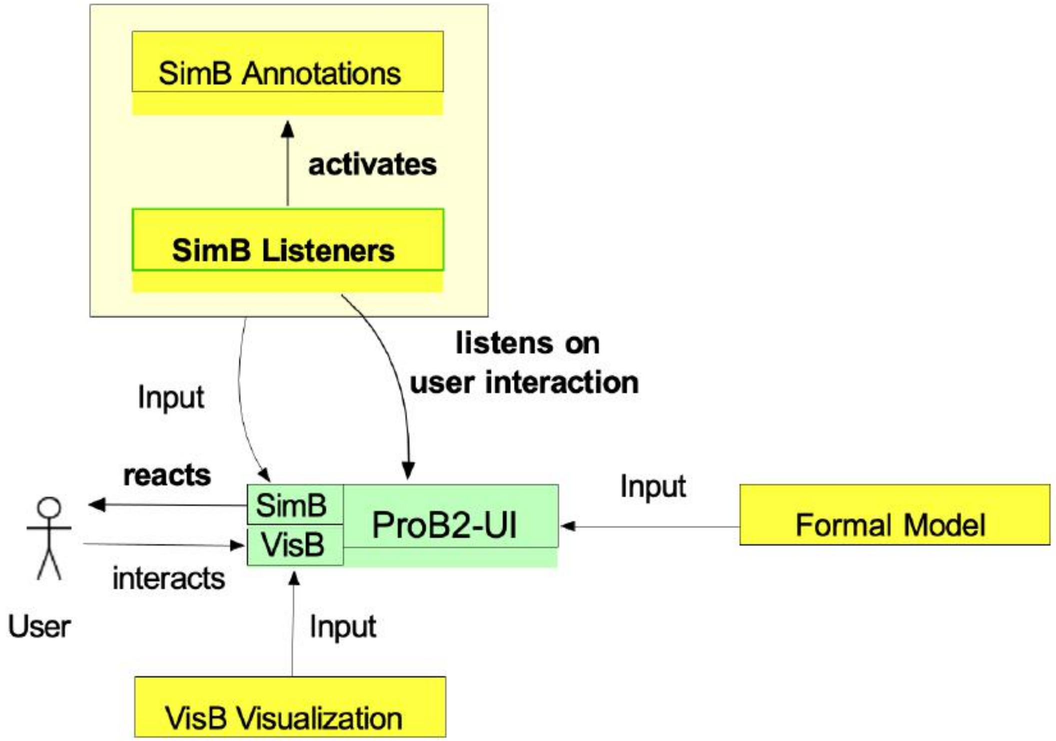
- SimB simulator with timed and probabilistic elements for formal models
- Simulation encoded by Activation Diagram
- Validation: Real-Time Simulation, Monte Carlo Simulation, Hypothesis testing, Estimation of Values and Probabilities
- User Interaction to trigger Simulation; Validation by State Space Projection



Fabian Vu, Michael Leuschel, and Atif Mashkoor. “Validation of Formal Models by Timed Probabilistic Simulation.” In: Proceedings ABZ. LNCS 12709. 2021, pp. 81–96.

VisB: Interactive Simulation

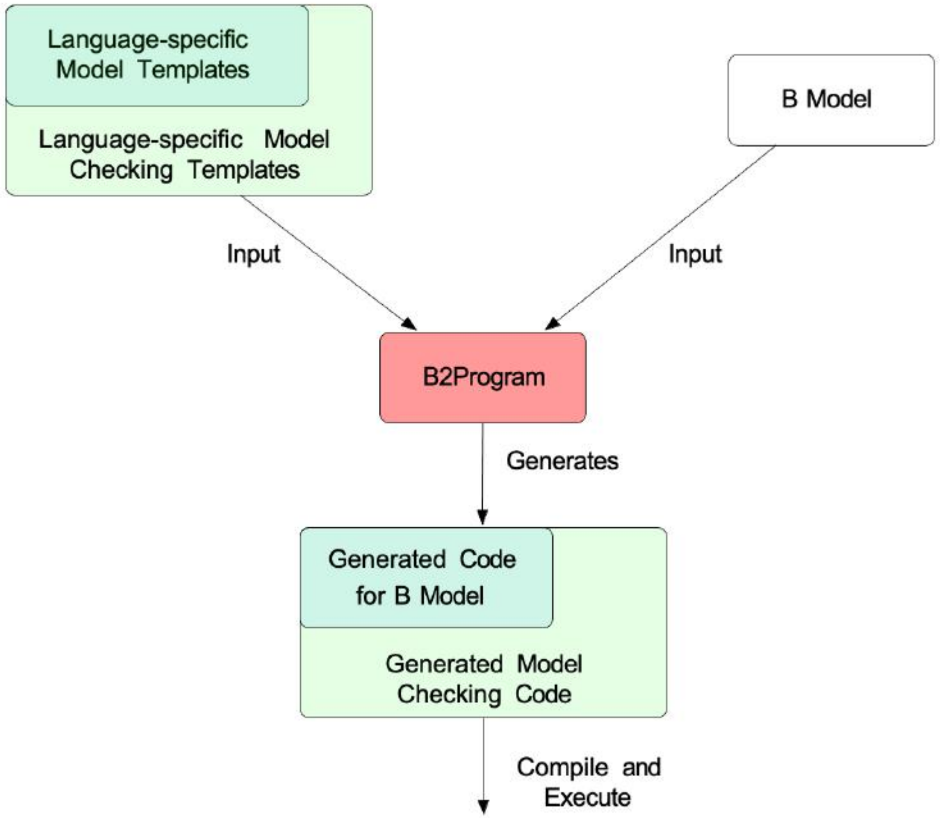
- Extension by interactive elements
- Coordination of User Interaction and System Response
- Validating Requirements of the form “when triggering action, A, then we expected response R”
- Validation by State Space Projection



Fabian Vu and Michael Leuschel. “Validation of Formal Models by Interactive Simulation.” In: Proceedings ABZ. LNCS 14010. 2023, pp. 59–69.

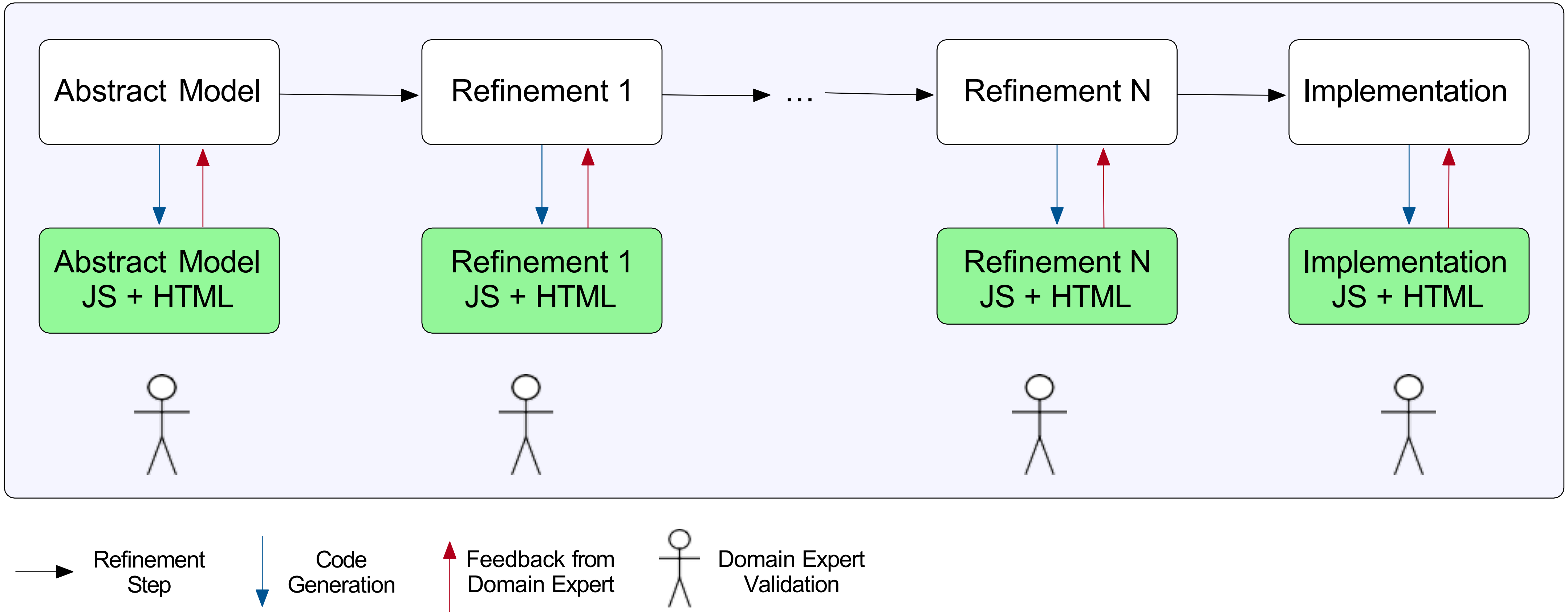
B2Program: Code Generation for Validation

- Domain-specific Visualization for Static/Dynamic Export
- Static Export of Single Execution Trace for a Formal Model
- Dynamic Export of Classical B Model to HTML
- Extension of B2Program by TypeScript/JavaScript for Dynamic Export
- Early-stage validation by Domain Experts without knowledge of formal methods (tools)
- Animation, Simulation, and Sharing of Scenarios between Modelers and Domain Experts with Feedback



- Fabian Vu, Christopher Happe, and Michael Leuschel. “Generating Domain-Specific Interactive Validation Documents.” In: Proceedings FMICS. LNCS 13487. 2022, pp. 32–49.
- Fabian Vu, Christopher Happe, and Michael Leuschel. “Generating interactive documents for domain-specific validation of formal models.” In: International Journal on Software Tools for Technology Transfer 6.2 (2024), pp. 147–168.

B2Program: Code Generation for Validation



Trace refinement for result adaptation

- Preserve desirable traces during refinement
- Deal with renaming, stuttering and skip
- Tool support
- Findings
  - Helps to find counterparts
  - May point out counterexamples

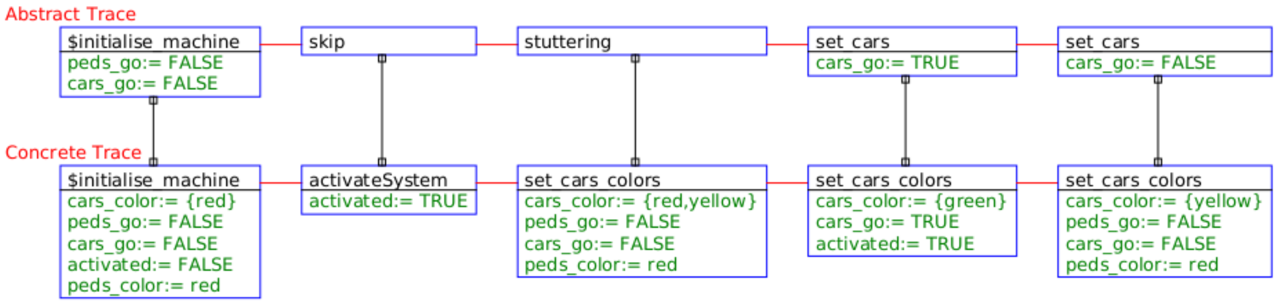


Fig. 1: Example output of the tool

Sebastian Stock, Atif Mashkoor, Michael Leuschel, Alexander Egyed: Trace Refinement in B and Event-B. ICFEM 2022: 316-333
Sebastian Stock, Atif Mashkoor, Michael Leuschel, Alexander Egyed: Trace preservation in B and Event-B refinements. J. Log. Algebraic Methods Program. 137: 100943 (2024)

Failure divergence refinement for result preservation

- Failure-divergence refinement for Event-B
- Proof that failure-divergence refinement preserves trace properties
- Implement tool support
- Less work for validation
  - Results can be kept

```
ALL OPERATIONS COVERED
machine2_additional_parameter_mch.eventb is a failure_divergences refinement of machine1_mch_refine_spec.P
% Refinement Check [FD=] CPU Time: 50 ms
Runtime for refinement_check: 51 ms
==> Refinement Check Successful
```

Fig. 2: Successful failure divergence refinement

```
MAbs_helper_prob_mc_mch.eventb is *not* a failure_divergences refinement of MAbs_prob_mc_mch_refine_spec.P
% Refinement Check [FD=] CPU Time: 60 ms
finding_trace_from_to(root)

Runtime for refinement_check: 64 ms
*** Refinement Check Counter-Example: ***
[$initialise_machine,DIVERGES,however at this position the specification could do:,[Move_Mouse_Hold,Move_M
```

Fig. 3: Unsuccessful failure divergence refinement with counter example

Sebastian Stock, Michael Leuschel, Atif Mashkoor, and Alexander Egyed, Failure divergence refinement for Event-B, submitted to iFM 2024

Validation-driven development

- Making validation the objective
  - How can we show the presence of the requirements in the model?

- “A priori” workflow
  - Formulate a VO
  - Implement
  - Verify
  - Validate

- Validation becomes the driving force of modeling process

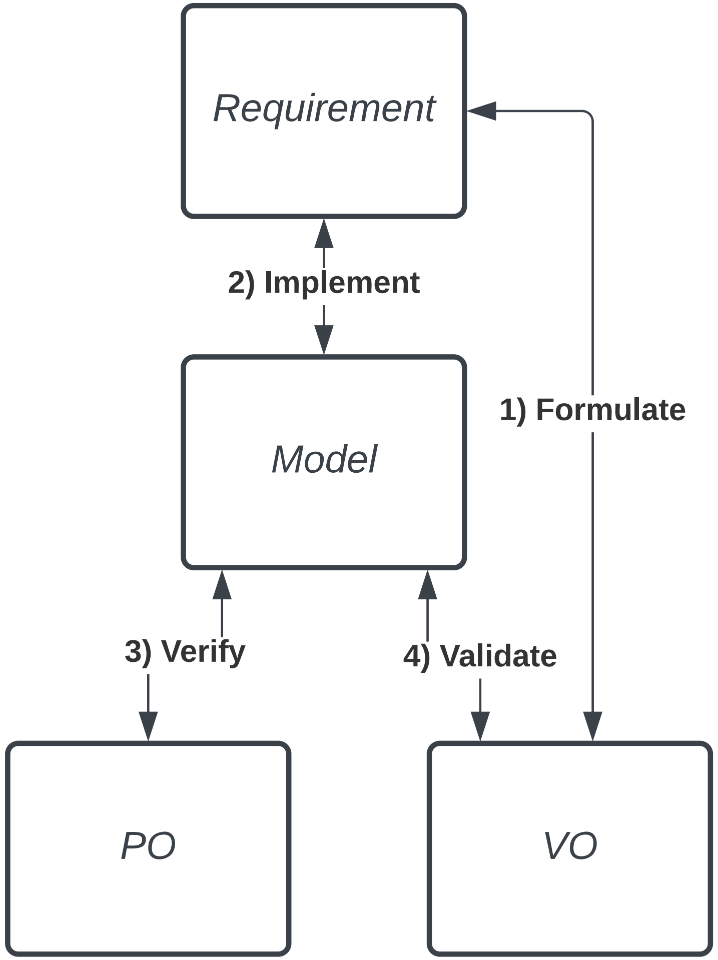


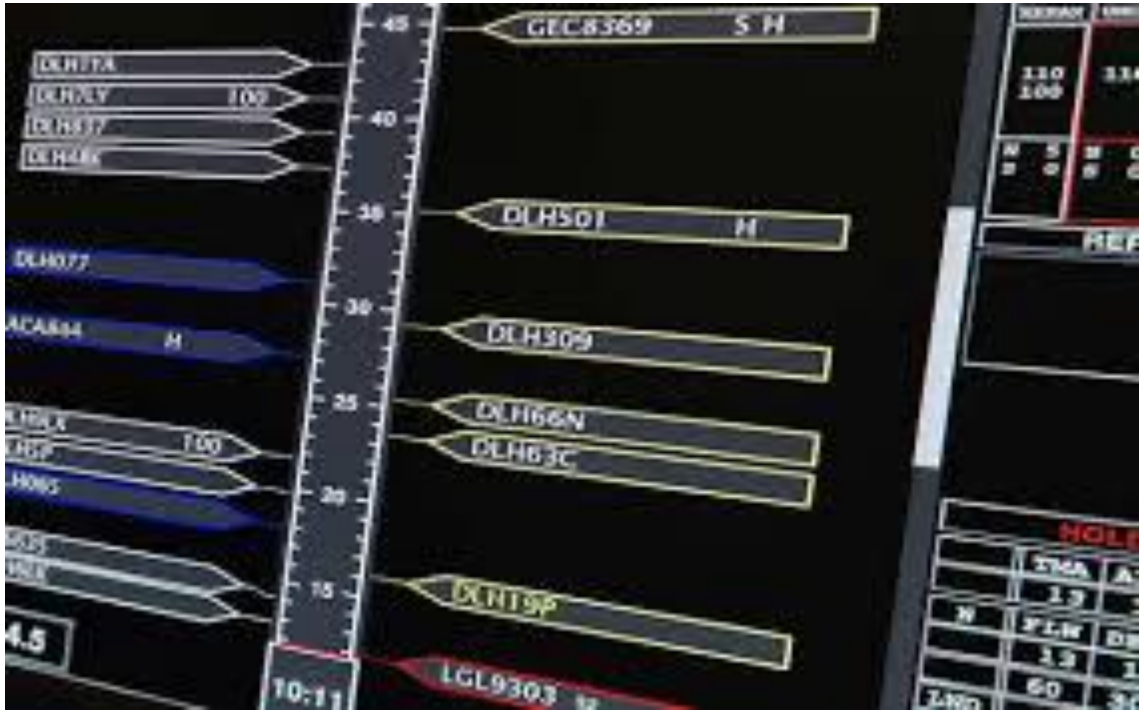
Fig. 4: Validation focused workflow

VDD - workflow

1. Finding a good model structure
   a. Problem Frames to sort knowledge
   b. Create refinement strategy
   c. Plan VOs
2. A priori strategy
   a) Implement model
   b) Verify
   c) Validate
3. Refine the model
   a) Adapt VOs
   b) Repeat 2)

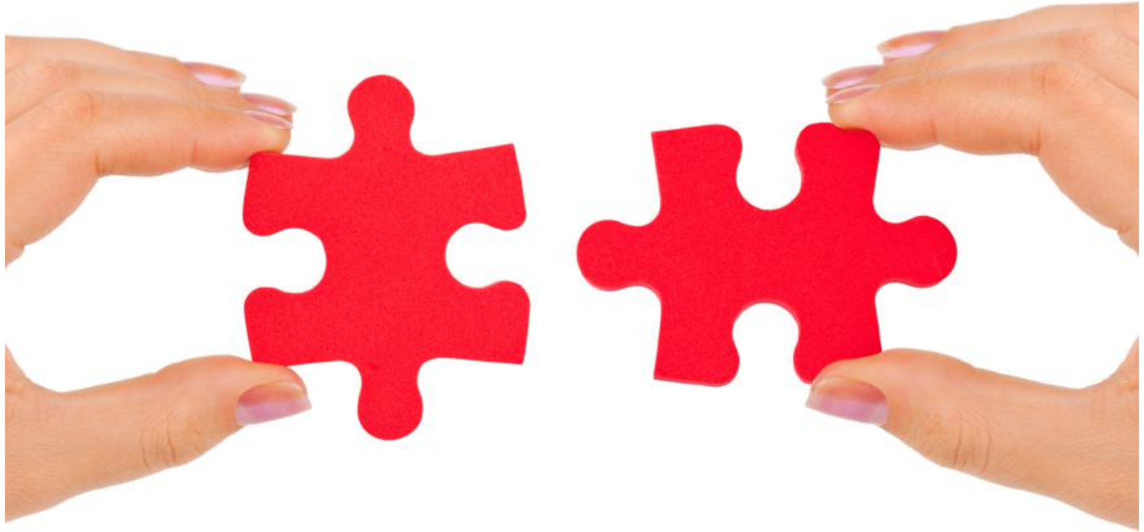
AMAN Case Study (ABZ 2023)

- First application of VOs during the development of a new, large formal model
- Comparison of a priori vs. a posteriori VO development
- Validation using both automatic validation tasks (model checking, trace replay, proof) and manual ones (visualization)
- Use of VOs during modeling uncovered unclear/ambiguous requirements



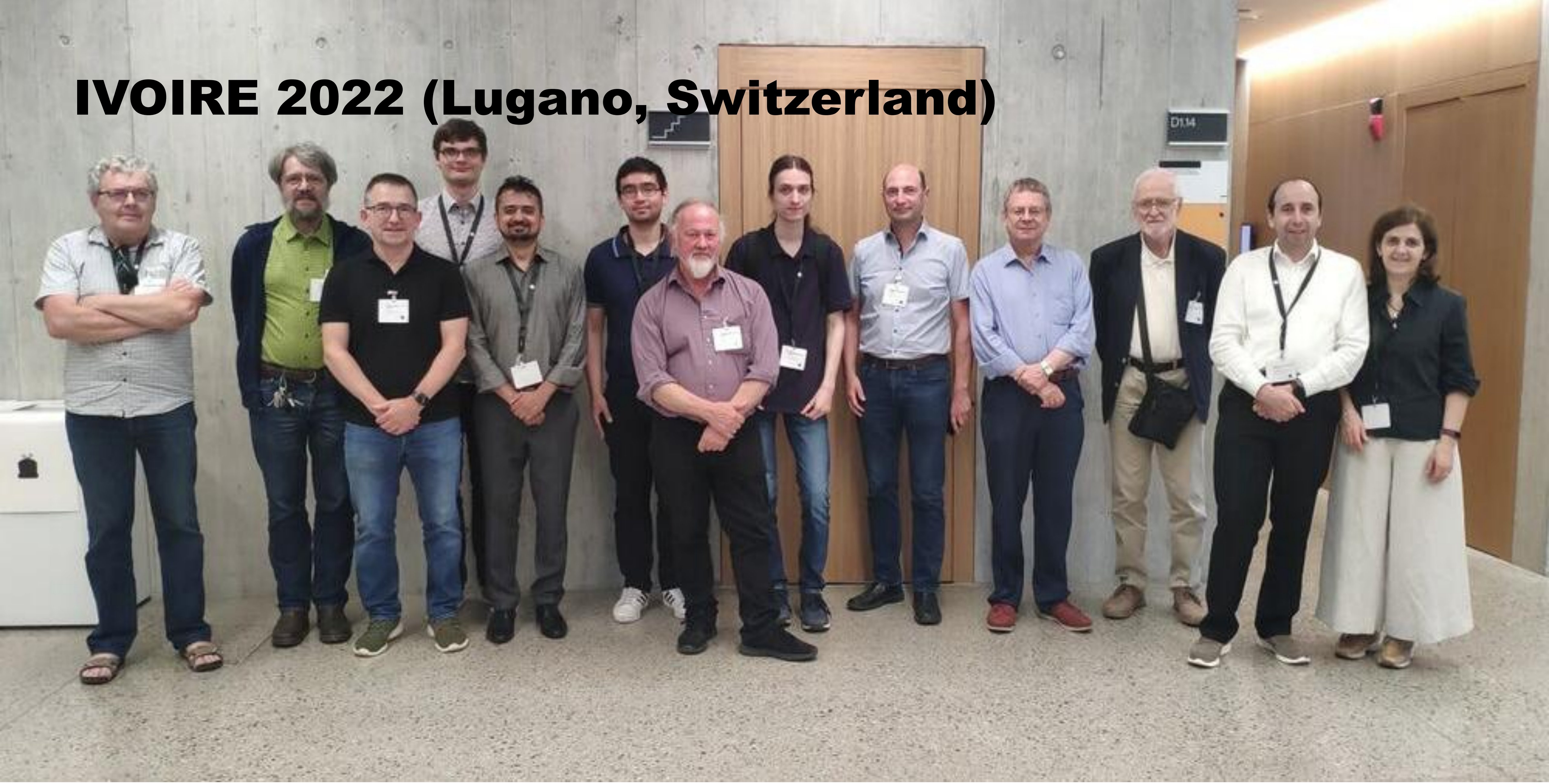
D. Geleßus et al. “Modeling and Analysis of a Safety-critical Interactive System through Validation Obligations.” In: Rigorous State-Based Methods. ABZ 2023. LNCS 14010. June 2023, pp. 284–302.

Conclusion



- Verification and validation are equally important activities and, hence, merit equal attention
- The IVOIRE methodology puts validation at the center of refinement-based development
- VOs can provide POs like semantics to the concept of formal validation

IVOIRE 2022 (Lugano, Switzerland)



IVOIRE 2023 (Nancy, France)

References

- Atif Mashkoor, Michael Leuschel, Alexander Egyed: Validation Obligations: A Novel Approach to Check Compliance between Requirements and their Formal Specification. ICSE (NIER) 2021: 1-5
- Jens Bendisposto, David Geleßus, Yumiko Jansing, Michael Leuschel, Antonia Pütz, Fabian Vu, Michelle Werth: ProB2-UI: A Java-Based User Interface for ProB. FMICS 2021: 193-201 Fabian Vu, Michael Leuschel, Atif Mashkoor: Validation of Formal Models by Timed Probabilistic Simulation. ABZ 2021: 81-96
- Fabian Vu, Dominik Brandt, and Michael Leuschel. "Model Checking B Models via High-level Code Generation." In: Proceedings ICFEM. LNCS 13478. 2022, pp. 334–351.
- Sebastian Stock, Atif Mashkoor, Michael Leuschel, Alexander Egyed: Trace Refinement in B and Event-B. ICFEM 2022: 316-333
- Fabian Vu, Michael Leuschel: Validation of Formal Models by Interactive Simulation. ABZ 2023: 59-69
- Sebastian Stock, Atif Mashkoor, Alexander Egyed: Validation-Driven Development. ICFEM 2023: 191-207
- David Geleßus, Sebastian Stock, Fabian Vu, Michael Leuschel, Atif Mashkoor: Modeling and Analysis of a Safety-Critical Interactive System Through Validation Obligations. ABZ 2023: 284-302
- Sebastian Stock, Fabian Vu, David Geleßus, Michael Leuschel, Atif Mashkoor, Alexander Egyed: Validation by Abstraction and Refinement. ABZ 2023: 160-178
- Fabian Vu, Jannik Dunkelau, and Michael Leuschel. "Validation of Reinforcement Learning Agents and Safety Shields with ProB." In: Proceedings NFM. LNCS 14627. 2024, pp. 279–297.
- Sebastian Stock, Atif Mashkoor, Michael Leuschel, Alexander Egyed: Trace preservation in B and Event-B refinements. J. Log. Algebraic Methods Program. 137: 100943 (2024)

JKU

LINZ INSTITUTE
OF TECHNOLOGY