A Model Checkable UML Soccer Player

Third Workshop on Model-Driven Engineering Tools (MDETools ’19) in Munich, Germany

Valentin BESNARD  
Ciprian TEODOROV  
Matthias BRUN

1 ERIS, ESEO-TECH, Angers, France

Frédéric JOUAULT
Philippe DHAUSSY

2 Lab-STICC UMR CNRS 6285, ENSTA Bretagne, Brest, France

This work has been partially funded by Davidson.
Model Design: System Component

Modular UML Model

Different environment models can be connected to the system at different times:

- A concrete environment model for actual execution with the soccer simulator
- Or an abstract environment model for model analysis

System Component

- **Controller**: Manage the high-level strategy of the robot
- **TrajectoryManager**: Manage trajectories of the robot

---

1Our github repository: https://github.com/ValentinBesnard/mdetools19-emi
Abstract Environment Model

**Goal**
Closes the system for the verification step (executable model)

**Specificities**
- Relies on some abstractions of the physical environment
- Focuses on the verification of control flows

**Usage**
For verification and validation activities
# Model Design: Environment Component

## Abstract Environment Model

**Goal**
Closes the system for the verification step (executable model)

**Specificities**
- Relies on some abstractions of the physical environment
- Focuses on the verification of control flows

**Usage**
For verification and validation activities

## Concrete Environment Model

**Goal**
Links the system with the actual soccer player

**Specificities**
- Implements a TCP client to interact with the player
- Implements a TCP client to be notified by the referee

**Usage**
For actual model execution
Tool Presentation: Classical UML-based Approaches

- **Metamodel**
  - conformsTo

- **Model**
  - transformation
  - code generation

- **Models for Analysis**

- **Environment**
  - I/O

- **Code**
  - Execution Environment

- **High Level Analysis Toolbox**
  - Simulator
  - Debugger
  - Model-checker
  - Profiler

- **Low Level Analysis Toolbox**
  - Debugger
  - Profiler
  - Monitor
Tool Presentation: Some Problems

Metamodel

conformsTo

Model

transformation

Models for Analysis

Semantic gap

High Level Analysis Toolbox

- Simulator
- Debugger
- Model-checker
- Profiler

Semantic Gap

code generation

Environment

I/O

Code

Execution Environment

Equivalence Problem

Low Level Analysis Toolbox

- Debugger
- Profiler
- Monitor
A unique implementation of the language semantics for all activities: simulation, verification, and execution.
EMI: an implementation of this approach for UML [Besnard et al., 2018]

Use the OBP2 tool [Teodorov et al., 2017] for:
- Trace-based simulation
- Model-checking

Perform runtime monitoring using UML observer automata [Besnard et al., 2019]
Results

Connect the OBP2 (https://plug-obp.github.io/) model-checker to EMI to verify safety and liveness LTL properties, for instance:

1. The player finally goes to the shooting position or aborts its action after having taken the ball.
   
   
   "\[\] ((playerHasBall && goToBall) -> <> (goToGoal || listenReferee))"

2. The player is never in the wrong direction when shooting.
   
   "\[\] !(inShootPos && !goalDirection)"

Model-checking results

- 2 safety properties and 6 liveness properties successfully verified on the system
- Composed of 16,844 configurations linked together with 31,370 transitions
- 4.3 seconds and 28 MB of memory (on a laptop with 8 CPU cores 4 GHz, 16 GB RAM, running a Linux OS)
Results

Connect the OBP2 (https://plug-obp.github.io/) model-checker to EMI to verify safety and liveness LTL properties, for instance:

1. The player finally goes to the shooting position or aborts its action after having taken the ball.
   "[] ((playerHasBall && goToBall) -> <> (goToGoal || listenReferee))"

2. The player is never in the wrong direction when shooting.
   "[] ! (inShootPos && !goalDirection)"

Execution results

- Scores on average 10 goals per match (when playing alone)
- Monitors execution of the model at runtime
Conclusion and Feedbacks

A solution to the MDETools’19 challenge

- Design of a modular UML model with two different environment models
- Analysis of the model with different V&V tools

⇒ Implementation of a real case study from the community
Conclusion and Feedbacks

A solution to the MDETools’19 challenge

- Design of a modular UML model with two different environment models
- Analysis of the model with different V&V tools
⇒ Implementation of a real case study from the community

Design

- Two versions of the model:
  - First simple prototype for design assessment (a couple of hours)
  - Second version as modular UML model (one week)
A solution to the MDETools’19 challenge

- Design of a modular UML model with two different environment models
- Analysis of the model with different V&V tools
- Implementation of a real case study from the community

Design

- Two versions of the model:
  - First simple prototype for design assessment (a couple of hours)
  - Second version as modular UML model (one week)

Analysis activities

- Simulation: Helps to identify design mistakes at early design stages
- Model-checking: Detects full event pools that block model execution
Conclusion and Feedbacks

A solution to the MDETools’19 challenge

- Design of a modular UML model with two different environment models
- Analysis of the model with different V&V tools

⇒ Implementation of a real case study from the community

EMI limitations

- Only a subset of UML is supported
- No support for time or real-time constraints
Conclusion and Feedbacks

A solution to the MDETools’19 challenge

- Design of a modular UML model with two different environment models
- Analysis of the model with different V&V tools

⇒ Implementation of a real case study from the community

EMI limitations

- Only a subset of UML is supported
- No support for time or real-time constraints

Possible UML soccer player improvements

- Implement a pathfinding algorithm to optimize trajectories
- Support several shooting positions

