A Meta-Metamodel for Dynamic Constraint Feedback in Modeling Languages

MATT BUNTING, JONATHAN SPRINKLE
UNIVERSITY OF ARIZONA
Cyber Physical System Design

- CPS domain: networking, control, and computation
- Design complexity managed through model based design, DSMLs
- Modeling languages enforce structural constraints
- Best known practices translate models into low level artifacts
- Models may be valid in syntax, but not in dynamic behavior
Model Verification for Dynamic

- Model verification is critical for CPS design
- Verification tools check dynamic constraints
- If dynamic constraints are violated, a model may be modified
- Modification may be performed automatically based on known design practices in the domain
Dynamic Constraint Feedback

- **x**: Model
- **b**: Behavior
- **r**: Requirements (dynamic constraints)
- **e**: Constraint violations
- **u**: Model modification method
Design of a DSML with verification tools is time consuming, considering DCF adds to development time.

DCF, as presented, has a structure.

Thinking like a DSML designer, let’s make a modeling language for DCF.
WebGME based DCFML
Metamodel and Transformations

Metamodel

Model Transformations
Case Study: Hybrid Controller Design

- Domain: Hybrid LTI controllers for vehicle with simple transitions
- Constraints: Reach a location, have particular controller performance
- Expert blocks: Tune controllers, Add control modes
- Verification: MATLAB’s stepinfo, HyCreate
Implementation: DCF Design

DCFML

Hybrid Controller ML with DCF

DCFML

HybridMetaModel

+ ATTRIBUTES
+ CONSTRAINTS
+ ASPECTS

ReachGoal

AddModeExpert

AddModeAndTransition

NoOvershoot

TuneGainExpert

ModifyControllerValue

RiseTime

HyCreateTool

MatlabStepinfo
Implementation: Metamodel

Hybrid Controller Metamodel

Metamodell
Implementation: Transformation

Model Transformation

addModeRule
This simple use case lets a user define a sequence of controllers with a destination goal.
Verification: Stepinfo

- An interpreter converts the controllers to parameters needed for MATLAB’s stepinfo.
- Stepinfo produces verification feedback.

\[
\begin{align*}
K_p &= 100; \\
s &= \text{tf('s')}; \\
C &= \text{pid}(K_p); \\
P &= 1/(s^2 + 20*s + 40); \\
T &= \text{feedback}(C*P,1); \\
\text{output} &= \text{stepinfo}(T);
\end{align*}
\]
Model Correction: TuneGain

- The TuneGainExpert provides new parameters for the ModifyControlValue transformation

### Overshoot Correction

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Gain</th>
<th>Overshoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>0.6962</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td>0.3228</td>
</tr>
<tr>
<td>3</td>
<td>81</td>
<td>0.1054</td>
</tr>
<tr>
<td>4</td>
<td>72.9</td>
<td>0.0127</td>
</tr>
<tr>
<td>5</td>
<td>65.61</td>
<td>0</td>
</tr>
</tbody>
</table>

### Rise Time Correction

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Gain</th>
<th>RiseTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0.7693</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>0.7045</td>
</tr>
<tr>
<td>3</td>
<td>19.6</td>
<td>0.6284</td>
</tr>
<tr>
<td>4</td>
<td>38.416</td>
<td>0.4531</td>
</tr>
<tr>
<td>5</td>
<td>53.7824</td>
<td>0.3636</td>
</tr>
</tbody>
</table>
Verification: HyCreate

- HyCreate configured to simulate a bicycle model for the car
- If the goal is not reached, a new mode with new set points are added with the AddModeAndTransition transformation
Conclusions

- The DCFML can greatly aid in the design of CPS DSMLs
- Adds great utility from model verification integration
- The DCF can be potentially closed and automated, always ensuring models that meet dynamic constraints
Future Work

- The DCFML is in a proof of concept stage
- Requires a lot of post development to truly implement
- Missing features
  - Expert block lacking
  - Metamodel missing WebGME mixins
- Verification Tool Integration with CPS-VO verification tools
Thank You!