Extending Code Generation to Support Platform-Independent Event-B Models

Asieh Salehi, Michael Butler, Colin Snook
University of Southampton, Southampton, United Kingdom

6th Rodin User and Developer Workshop, 23 May, 2016, Linz, Austria

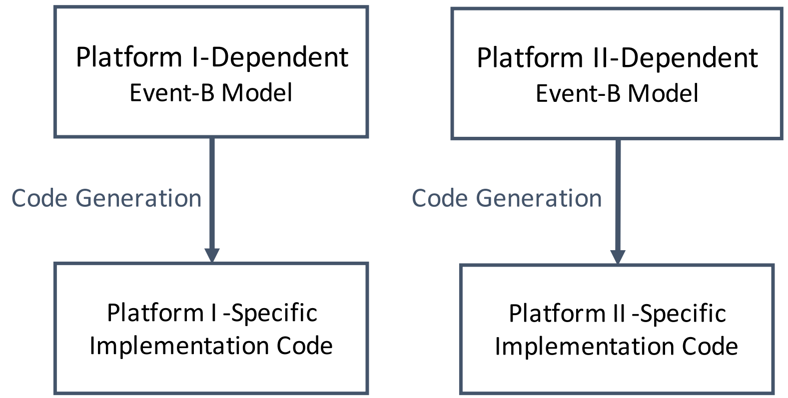
Overview

- Motivation
- Background:
  - Code Generation (CG) with Tasking Event-B
  - Branching Control Flow
- Case Study (PRiME project*):
  - Embedded Run-Time Management (RTM) system
  - Platform Dependencies
- Modelling and Tasking by Restricted CG
- Modelling and Tasking by Extended CG
- Extending Code Generation Plugin
- Case Study Overview
- Summary and Future work

* Power-efficient, Reliable, Many-core Embedded systems. http://www.prime-project.org

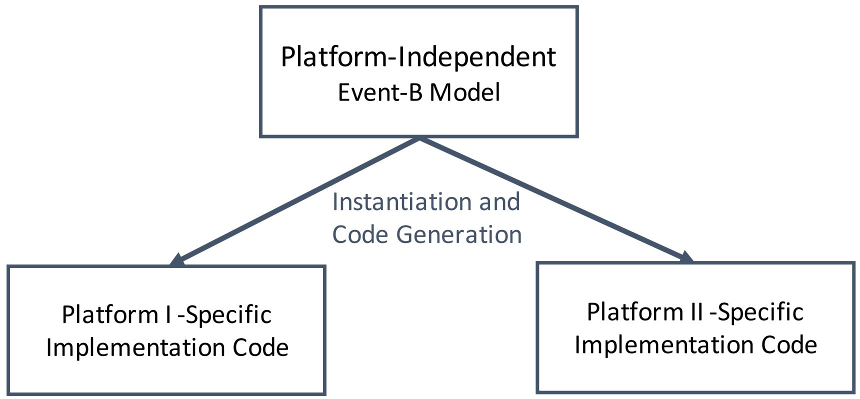
Motivation

Previous CG approach



- a single implementation for the Event-B model
- dependency between the Event-B model and target platform architecture

Extended CG approach

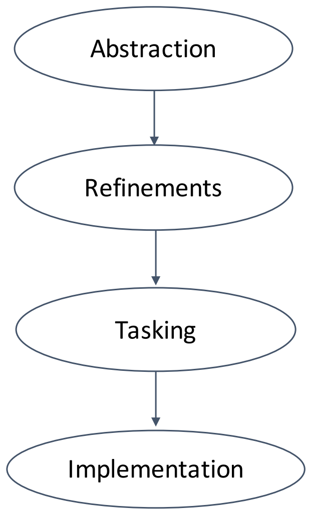


- different platform-specific implementations from the same Event-B model
- platform-independent Event-B model
  platform-specific implementations

Background:
Code Generation with Tasking Event-B

- Code generation in Event-B addresses the gap between the lowest level Event-B refinement and an **implementation**.

- Tasking Event-B is an extension of Event-B to introduce control flows:
  - Sequence
  - Branch
  - Loop



Background (cont.): Branching Control Flow

- Task body:

```
if evt_1
[ elseif evt_i ]*
else evt_n
```

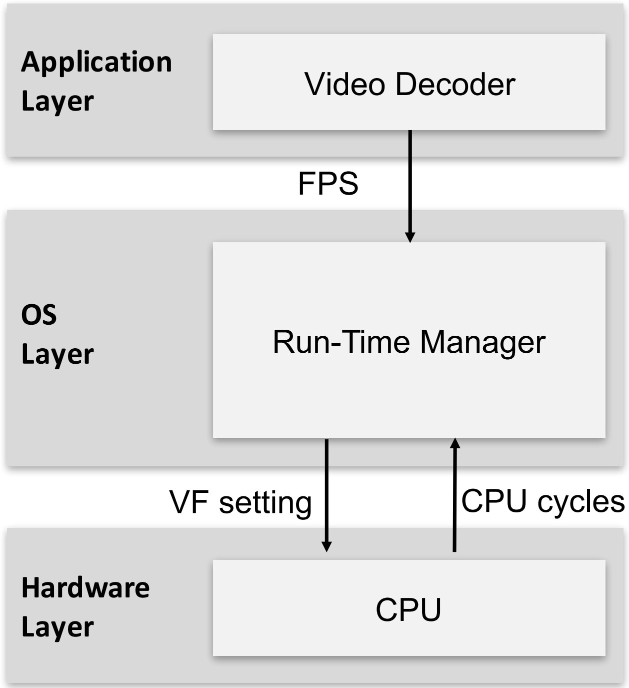
- Translation (pseudocode):

```
if (g_1) then a_1
[else (g_i) then a_i end ]*
else a_n end
```

(where g_i and a_i indicate guard and action of the evt_i respectively)

Case Study:
Embedded Run-Time Management (RTM) system

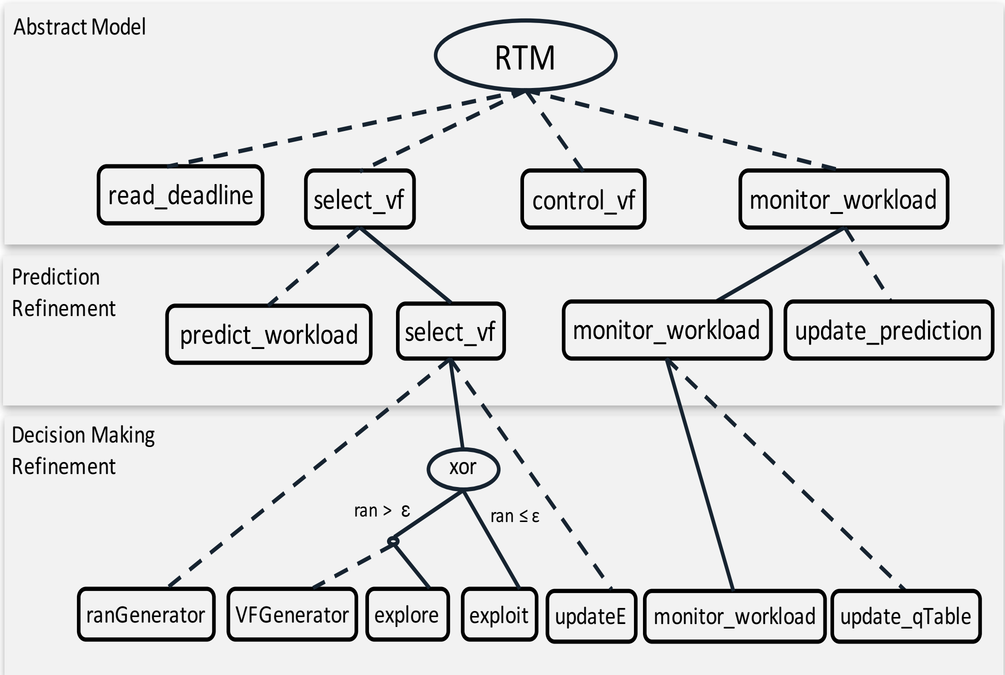
- Application provides the required deadline (frame-per-second (FPS))
- RTM decides about the optimal value of Voltage-Frequency (VF)
- The workload (cpu-cycles) to decode the frame is monitored.
- RTM aims to choose the optimal value of VF which meets deadline while saving power.



Case Study (cont.): Prediction and Machine Learning at Run-time

RTM uses **prediction** and **machine learning** algorithms to decide about the optimal value of VF.

- First Refinement: Prediction
- Second Refinement: Machine Learning (using a learning table)



Case Study(Cont.): Platform Dependencies

- An **RTM model** can require **adjustments** across different hardware platforms due to the **diversity of architecture characteristics**.

- The number of supported **frequencies** by each platform determines the number of **columns** in the **learning table** used to record the rewards and penalties for each VF setting decision.

		Frequencies			
		freq1	freq2	freq3	freq4
workload	1	0.8	0.2	0.2	0.1
	2	-0.1	0.6	0.4	0.2
	3	-0.4	-0.2	0.7	0.3
	4	-1.0	-0.8	-0.1	0.8

Modelling and Tasking by Restricted CG

Event-B actions to modify the learning table
(dependent on the number of frequencies)

update_col1_qTable ≙
WHERE freq = FREQ1
THEN
qTable ≔ updateArray(qTable, row, 0, value)

update_col2_qTable ≙
WHERE freq = FREQ2
THEN
qTable ≔ updateArray(qTable, row, 1, value)

…

update_coln_qTable ≙
WHERE freq = FREQn
THEN
qTable ≔ updateArray(qTable, row, N-1, value)

tasking

tasking body

…
if update_col1_qTable
[**elseif** update_coli_qTable]*
else update_coln_qTable
…

code generation

generated C code for a
platform with n number of frequencies

```
…
if ( freq = FREQ1 )
{
        qTable[row][0] = value;
}
…
else if ( freq = FREQi )
{
        qTable[row][i] = value;
}
…
else
{
        qTable[row][N-1] = value;
}
…
```

Modelling and Tasking by Extended CG

Event-B action to modify the learning table (independent on the number of frequencies)

```
update_qTable ≙
ANY i
WHERE i ∈ 1 ‥ N    expanding
        &  freq = F(i)
THEN
qTable ≔ updateArray( qTable, row, i, value )
```

tasking

tasking body

```
…
update_qTable
…
```

code generation

N = 4

generated C code for platform ARM8 with 4 frequencies

```
…
if ( freq = FREQ1 )
            qTable[row][0] = value;
else if ( freq = FREQ2 )
            qTable[row][1] = value ;
else if ( freq = FREQ3 )
            qTable[row][2] = value ;
else
            qTable[row][3] = value ;
…
```

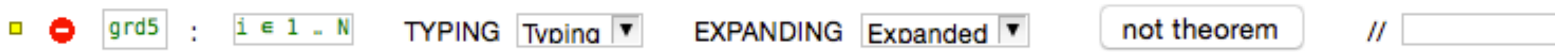
N = 13

generated C code for platform ARM7 with 13 frequencies

```
…
if ( freq = FREQ1 )
            qTable[row][0] = value;
.
.
else if ( freq = FREQ12 )
            qTable[row][11] = value ;
else
            qTable[row][12] = value ;
…
```

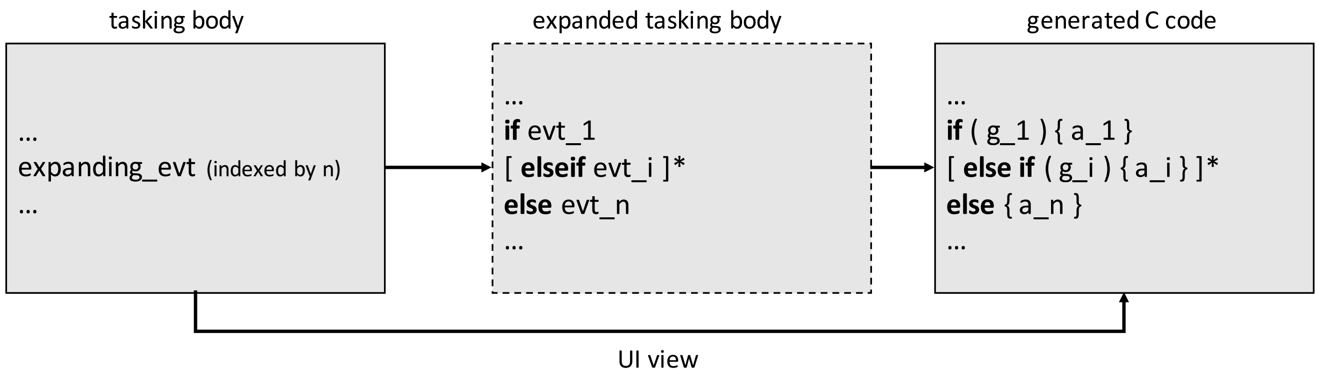
Extending Code Generation Plugin

- Expanding Guard:



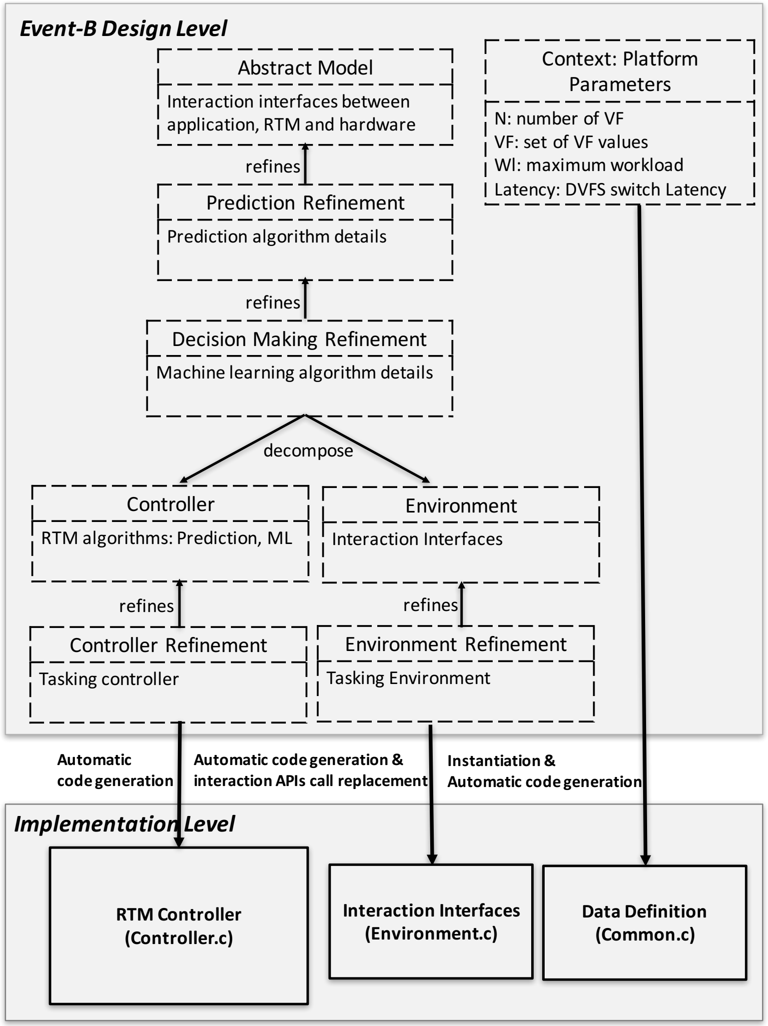
  - indexed by a constant (N)
  - N is instantiated during translation

- Task body Translation



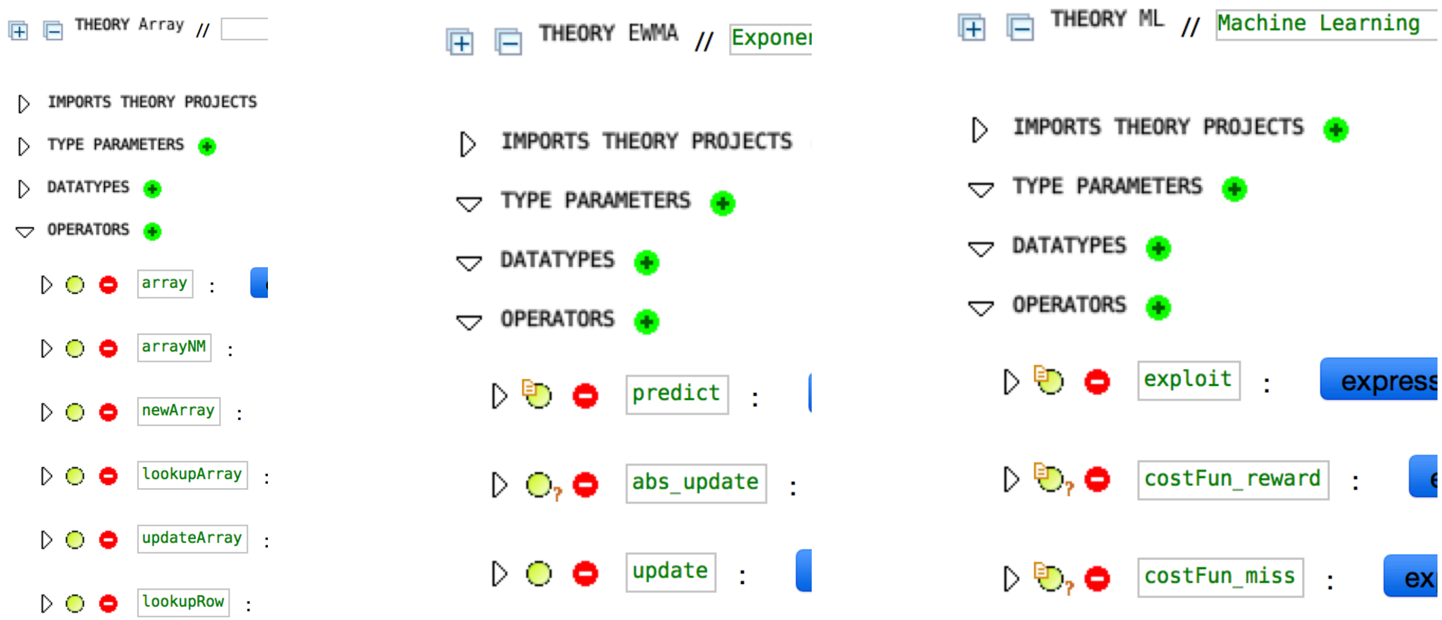
Case Study Overview

- One abstract, two refinements
- Model Decomposition:
  - Controller: RTM algorithms
  - Environment: Interaction Interfaces
- Code Generation
- Theory definitions



Case Study Overview: Theories

- Array theory to specify the learning table
- EWMA* theory to specify the prediction operators and translations
- ML theory to specify the machine learning operators and translations



* Exponentially Weighted Moving Average

UNIVERSITY OF Southampton

Case Study Overview: Proving

- Abstract specification of the prediction*:

Theory EWMA

operator $predict\ (l \in \mathbb{Z},\ index \in \mathbb{Z},\ w \in \mathbb{Z} \mathrel{+\!\!\!\rightarrow} \mathbb{Z})\ \widehat{=}$

$l * SUM(\lambda i.i \in 0..index - 1 \mid w(i) * (i - l)\ exp\ (index - i))$

Event $predict\_workload\ \widehat{=}$
begin
act1 : $pwl := predict(l, n, wl\_hst)$
end

Event $update\_prediction\ \widehat{=}$
begin
act1 : $wl\_hst := wl\_hst \cup \{n \mapsto w\}$
act2 : $n := n + 1$
end

* $l \cdot \sum_{i=0}^{n-1} w(i) \cdot (i - l)^{n-i}$ where $0 \leq l \leq 1$

- Refined specification of the prediction:

Theory EWMA

operator $update(l \in \mathbb{Z},\ w \in \mathbb{Z},\ avgwl \in \mathbb{Z})\ \widehat{=}$

$l * w + (1 - l) * avgwl$

Event $predict\_workload\ \widehat{=}$
refines $predict\_workload$
begin
act1 : $pwl := avgwl$
end

Event $update\_prediction\ \widehat{=}$
refines $update\_prediction$
begin
act1 : $avgwl := update(l, w, avgwl)$
end

Case Study Overview: Proving (cont.)

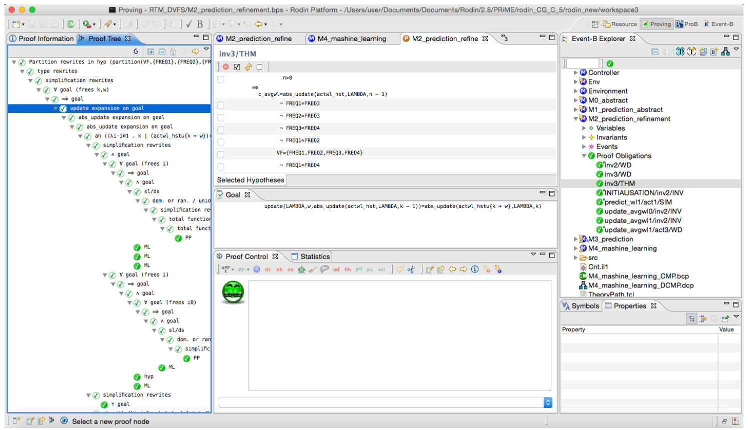
- Abstract specification of the prediction*:

$$\textbf{Theory}\quad \text{EWMA}$$
$$\textbf{operator}\ \ predict\ (l \in \mathbb{Z},\ index \in \mathbb{Z},\ w \in \mathbb{Z} \mathrel{+\!\!\!\rightarrow} \mathbb{Z})\ \widehat{=}$$
$$l * SUM(\lambda i.i \in 0..index - 1 \mid w(i) * (i - l)\ exp\ (index - i))$$

- Refined specification of the prediction:

$$\textbf{Theory}\quad \text{EWMA}$$
$$\textbf{operator}\ \ update(l \in \mathbb{Z},\ w \in \mathbb{Z},\ avgwl \in \mathbb{Z})\ \widehat{=}$$
$$l * w + (1 - l) * avgwl$$

$$\texttt{thm1}: \forall n, w \cdot n > 0 \land w \in \mathbb{Z} \Rightarrow$$
$$update(l, w, predict(l, n, wl\_hst)) = predict(l, n + 1, wl\_hst \cup \{n \mapsto w\})$$



* $l \cdot \sum_{i=0}^{n-1} w(i) \cdot (i - l)^{n-i}$ where $0 \leq l \leq 1$

Summary and Future Work

Extending Code Generation technique in the Event-B formal method provides an approach to manage *platform diversity* by shifting the focus away from *hand-written platform-specific* code to *platform-independent verified models* from which platform-specific implementations are *automatically generated*.

- In the PRiME project, we are extending our RTM modelling, translation to support more run-time algorithms:
  - Extend the code generation in-line
- New plugin release

UNIVERSITY OF Southampton

PRiME
www.prime-project.org

Thank you
Any Question?