Towards Modular Development in Event-B

Thai Son Hoang\textsuperscript{1} \quad Hironobu Kuruma\textsuperscript{2} \quad Michael Butler\textsuperscript{1}

\textsuperscript{1}ECS, University of Southampton, U.K.
\textsuperscript{2}Research and Development Group, Hitachi Ltd., Japan

RODIN Workshop 2016
Linz, Austria
23rd May 2015
Background

Event-B top-down development

- Refinement
- Decomposition

Pros and Cons

- ✔ Details can be introduced \textit{gradually} into the formal model
- ✗ Large models with \textit{monolithic structures}
Existing Composition Approaches

- **Modularisation** (Iliasov et. al., 2010)
- **Feature composition** (Poppleton, 2008)
- **Parallel composition** (Silva and Butler, 2009)
- **Patterns** (Hoang et. al., 2009)
- **Generic instantiation** (Silva and Butler, 2009)

The presented proposal borrows many ideas from these approaches.
The Aims

- Reuse models through composition.
  - Including refinement-chain of the sub-models.

- Integrated smoothly with the Event-B development process.
  - Accomodate changes seamlessly.
Machine Inclusion

Concepts

- Machine A includes machine B

  - machine A includes B
  - A inherits B’s variables
  - A inherits B’s invariants
  - B’s variables can only be modified via event synchronisation

- Multiple instances of B can be included via prefixing.

  - machine A includes p_B
  - Variables, events are renamed accordingly.
Machine Inclusion

Illustration

```plaintext
machine B

variables y
invariants
  J(y)

events
event f
  any u where
    G_B(y, u)
then
  y : | BAP_B(y, u, y')
end
```
Machine Inclusion

Illustration

machine B

variables y

invariants $J(y)$

events

event f

any $u$ where $G_B(y, u)$

then

$y : | BAP_B(y, u, y')$

end

machine A

variables $x$

machine (flatten_)A

variables $x$

events

event $e$ any $t, p_u$ where $G_A(x, t)$ $H_{AB}(x, p_y, t, p_u)$ $G_B(p_y, p_u)$

then

$x : | BAP_A(x, t, x')$

$y : | BAP_B(y, u, y')$

end
Machine Inclusion

Illustration

machine B
variables y
invariants $J(y)$

events
event f
  any $u$ where
    $G_B(y, u)$
  then
    $y : \mid BAP_B(y, u, y')$
end

machine A
includes $p_B$
variables x

machine (flatten_)A
variables $x, p_Y$

machine A
includes $p_B$
variables x
Machine Inclusion

Illustration

machine B
variables $y$
invariants $J(y)$
events
event $f$
  any $u$ where
  $G_B(y, u)$
then
  $y : BAP_B(y, u, y')$
end

machine A
includes $p_B$
variables $x$
invariants $I(x, p_y)$
events
event $e$
  any $t$ where
  $G_A(x, t)$
  $H_{AB}(x, p_y, t, p_u)$
then
  $x : BAP_A(x, t, x')$
end

machine (flatten_)A
variables $x, p_y$
invariants $I(x, p_y)$

$J(y)$
### Machine Inclusion

#### Illustration

<table>
<thead>
<tr>
<th>Machine</th>
<th>Variables</th>
<th>Invariants</th>
<th>Events</th>
</tr>
</thead>
</table>
| **machine B** | $y$ | $J(y)$ | event $f$
|          |          |            | any $u$ where $G_B(y, u)$ then $y : | BAP_B(y, u, y')$ end |
| **machine A** | $x$ | $I(x, \_p_y)$ | event $e$
|          |          |            | synchronises $p_f$
|          |          |            | any $t$ where $G_A(x, t)$ $H_{AB}(x, p_y, t, p_u)$ then $x : | BAP_A(x, t, x')$ end |
| **machine (flatten__)A** | $x$, $p_y$ | $I(x, \_p_y)$, $J(y)$ | |

Where $\_p_B$ and $\_p_u$ are shared variables.

---

The above text appears to describe the inclusion of one machine within another, detailing the variables, invariants, and events involved in this process. The notation $BAP$ likely stands for a specific type of transition or condition, and $G$ and $H$ are transition relations that define the behavior of the machines.

The text also includes a section labeled "Machine Inclusion Illustration," which provides a step-by-step example of how the inclusion is realized through specific events and conditions.

---

The text includes mathematical symbols and logical expressions typical in formal verification or system modeling. The use of $BAP$, $G$, and $H$ suggests a formal language used to describe system behavior, possibly related to automata theory or process algebra.
Machine Inclusion
Illustration

machine B
variables y
invariants
\( J(y) \)
events
event f
  any u where
  \( G_B(y, u) \)
then
  \( y : | BAP_B(y, u, y') \)
end

machine A
includes p_B
variables x
invariants
\( I(x, p_y) \)
events
event e
  synchronises p_f
  any t where
  \( G_A(x, t) \)
  \( H_{AB}(x, p_y, t, p_u) \)
then
  \( x : | BAP_A(x, t, x') \)
end

machine (flatten_)A
variables x, p_y
invariants
\( I(x, p_y) \)
\( J(y) \)
events
event e
  any t, p_u where
  \( G_A(x, t) \)
  \( H_{AB}(x, p_y, t, p_u) \)
  \( G_B(p_y, p_u) \)
then
  \( x : | BAP_A(x, t, x') \)
  \( y : | BAP_B(p_y, p_u, p_y') \)
end
Refinement-Chain Inclusion

Concepts

- Machine $A$ includes refinement-chain $B_1 \rightarrow B_m$.
  
  machine $A$
  includes $B_1 \rightarrow B_m$

- $A$ has double interfaces
  
  - To its abstract machine, $A$ includes $B_1$
  - To its refinement, $A$ includes $B_m$
Refinement-Chain Inclusion

Illustration
Refinement-Chain Inclusion

Illustration

\[ B_1 \text{ refines } A + B_1 \]

\[ B_m \text{ includes } A + B_m \]

\[ abs \text{ refines } A \]

\[ cnc \text{ refines } \]
Refinement-Chain Inclusion

Illustration

B_1 \text{ refines } \text{ includes } A + B_1

B_m \text{ includes } A + B_m

\text{ abs refines } \text{ includes } A

\text{ cnc refines }
The refinement is “almost” correct-by-construction.
Refinement-Chain Inclusion

The Two “Interfaces”

machine $A + B_1$

events

event $e$

any $t, u_1$ where

$G_A(x, t)$

then

$x : | BAP_A(x, t, x')$

end

machine $A + B_m$

refines $A + B_1$

events

event $e$

any $t, u_m$ where

$G_A(x, t)$

then

$x : | BAP_A(x, t, x')$

end

The refinement is “almost” correct-by-construction.

- Guard-strengthening for $G_A$ is trivial
- Action-strengthening for $BAP_A$ is trivial
Refinement-Chain Inclusion

The Two “Interfaces”

machine A + B
  events
  event e
  any t, u_1 where

G_{B_1}(y_1, u_1)
then
y_1 :| BAP_{B_1}(y_1, u_1, y')
end

machine A + B_m
refines A + B
  events
  event e
  any t, u_m where

G_{B_m}(y_m, u_m)
then
y_m :| BAP_{B_m}(y_m, u_m, y')
end

The refinement is “almost” correct-by-construction.

- Guard-strengthening for $G_{B_1}$ is guaranteed by $B_m$ refines $B_1$
- Action-strengthening for $BAP_{B_1}$ is guaranteed by $B_m$ refines $B_1$
Refinement-Chain Inclusion

The Two “Interfaces”

machine $A + B_1$

events

event e

any $t, u_1$ where

$H_{AB_1}(x, y_1, t, u_1)$

then

end

machine $A + B_m$

refines $A + B_1$

events

event e

any $t, u_m$ where

$H_{AB_m}(x, y_m, t, u_m)$

then

end

The refinement is “almost” correct-by-construction.

- Guard-strengthening for $H_{AB_1}$ needs to be proved manually.
Some Evaluations

- We have applied the idea to several examples.
- Resulting in **hierarchical development**.
- Modelling effort is reduced.
- Proving effort is reduced.
- Easier to understand specification.
Example. Car on a Bridge

Development hierarchy

Sensor_m0
  \[\text{refines}\]
  Sensor_m2

4 \times \text{includes}

Sensor_m2

4 \times \text{includes}

TrafficLight_m0
  \[\text{refines}\]
  TrafficLight_m3

2 \times \text{includes}

TrafficLight_m0

2 \times \text{includes}

Car_m0

\[\text{refines}\]

Car_m2

\[\text{refines}\]

Car_m3

\[\text{refines}\]

Car_m4

\[\text{refines}\]

Car_m5

\[\text{refines}\]

Car_m6
Machine Car_m6

machine Car_m6
includes
  ML_out_Sensor_m2
  ML_in_Sensor_m2
  IL_out_Sensor_m2
  IL_in_Sensor_m2
  ML_TrafficLight_m3
  IL_TrafficLight_m3
Event ML_in

(flatten_)ML_in

when

ML_in_SNSR = TRUE
ML_in_Snsr_01 = FALSE
ML_in_ctrl_Snsr_01 = FALSE
ML_in_DEP \neq IL_out_DEP

then

ML_in_SNSR := FALSE
ML_in_DEP := ML_in_DEP + 1
ML_in_Snsr_10 := TRUE

end

end event ML_in

synchronises ML_in_SNSR_off

when

ML_in_DEP \neq IL_out_DEP

then

SKIP

end
Summary

- Inclusion of refinement-chain
- Syntactical
- Tool support is needed.
- Reference machines/contexts from another project (see the next talk).