Developing Control Systems in Event-B

Thai Son Hoang

Chair of Information Security, Department of Computer Science
Swiss Federal Institute of Technology Zürich (ETH Zürich)

16th November 2010, McMaster University
Outline

1 Event-B Modelling Method

2 Developing Control Systems
   - A Requirements Document
   - A Modelling Guideline
   - Formal Development

3 Summary
Event-B Modelling Method

- A modelling language for **discrete transition systems**.
- Mathematical language of **first-order logic** and some **typed set theory**.
- Incremental development process using **refinement**.
- Consistency of models: discharging **proof obligations**.
- **Correct-by-construction** systems.
- Supported by the **RODIN Platform**.
Event-B Models

**Context**
- constants
- carrier sets
- axioms

**Static part**

**Machine**
- variables: \( v \)
- invariant: \( I(v) \)
- events: \( \text{evt} \)

**Dynamic part**

- \( t \) – the parameters.
- \( G(t, v) \) – the guard: enable conditions.
- \( v := E(t, v) \) – the action: \( v \) is assigned the value of \( E(t, v) \).
- **Initialisation**: A special event without parameters and guards.

**Consistency**: Invariant establishment and preservation

\[
\text{evt} \\
\text{any } t \text{ where } G(t, v) \\
\text{then } v := E(t, v) \\
\text{end}
\]
Refinement

- A way to **introduce more concrete details** into the formal model.
- The concrete model must be **consistent** with the abstract model.
- Analogies with a **microscope** or a **parachute**.
- The **view** of the system gets **more accurate**.
- Allow to **observe** the system with a **finer time grain**.
Example. File Transfer

Model 0: the file is copied in one-shot.

Model 1: the file is transferred piece-by-piece.

Model 2: each transfer is done via a pair of send/receive actions.
Consistency: The **concrete model** only exhibits **behaviours** allowed by the **abstract model**.

- **Event-wise reasoning:**
  - **Guard strengthening:** concrete guards are **stronger** than abstract guards.
  - **Simulation:** The abstract event can **simulate** the concrete event.
Event-B Refinement

Consistency: The **concrete model** only exhibits **behaviours** allowed by the **abstract model**.

- **Event-wise reasoning:**
  - **Guard strengthening:**
    concrete guards are **stronger** than abstract guards.
  - Simulation: The abstract event can **simulate** the concrete event.
Event-B can be used to model:

- distributed systems,
- concurrent systems,
- sequential programs,
- electronic circuits,
- control systems,
- etc.
Event-B Modelling Method

Developing Control Systems
- A Requirements Document
- A Modelling Guideline
- Formal Development

Summary
Train Control at a Stations

- Joint work with Simon Hudon.
- A station has a single track.
- The track is one way:
  - the train enters the station block via the approaching block.
  - the train exits the station block via the exiting block.
- There are two signals located at the two ends of the station.
- The signals turn to red automatically when a train passes by.
- The system controls when to turn the signals to green.
### Environment

<table>
<thead>
<tr>
<th>ENV 1</th>
<th>A train occupies no more than one block.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENV 2</td>
<td>Each signal is either <strong>green</strong> or <strong>red</strong>.</td>
</tr>
<tr>
<td>ENV 3</td>
<td>Trains are assumed to stop at <strong>red</strong> signals.</td>
</tr>
<tr>
<td>ENV 4</td>
<td>The signals automatically change from <strong>green</strong> to <strong>red</strong> when some train passes by.</td>
</tr>
</tbody>
</table>
The system guarantees that there is no collision between trains.

| SAF 5 | Two trains are not on the same block at the same time. |
Train Schedule

- Each train is associated with a predefined route plan.
- The plan specifies either the train to stop or pass through.

| FUN 6 | Each train either stops or passes through according to a predefined route plan. |
Sensors and Actuators

- There are sensors detecting **if a block is occupied**
- There are sensors detecting the **status of the two signals**.

<table>
<thead>
<tr>
<th>ENV 7</th>
<th>The sensors always reflect the values of the corresponding physical components.</th>
</tr>
</thead>
</table>

- The controller commands the signals via **actuators**.

<table>
<thead>
<tr>
<th>ENV 8</th>
<th>For each signal, there is an <strong>actuator</strong> for the controller to turn it from red to green.</th>
</tr>
</thead>
</table>
A Modelling Guideline for Developing Control Systems

Phase 1 Model the environment.
Phase 2 Model the actuators.
Phase 3 Model the sensors and the controller.
Phase 4 Model the schedule.
A Modelling Guideline for Developing Control Systems

Phase 1  Model the environment.
Phase 2  Model the actuators.
Phase 3  Model the sensors and the controller.
Phase 4  Model the schedule.
Phase 1  Model the environment.
Phase 2  Model the actuators.
Phase 3  Model the sensors and the controller.
Phase 4  Model the schedule.
A Modelling Guideline for Developing Control Systems

Phase 1  Model the **environment**.
Phase 2  Model the **actuators**.
Phase 3  Model the **sensors** and the **controller**.
Phase 4  Model the **schedule**.
Phase 1  Model the environment.
Phase 2  Model the actuators.
Phase 3  Model the sensors and the controller.
Phase 4  Model the schedule.
Phase 1. Environment
First Model. Trains and Tracks (1/3): The Context

carrier sets:  \textit{BLOCK, TRAIN}  

constants:  \textit{APP, STN, EXT}  

axioms:
\begin{align*}
\text{axm0}_1 & : \quad \text{partition}(\textit{BLOCK}, \{\textit{APP}\}, \{\textit{STN}\}, \{\textit{EXT}\})
\end{align*}

\begin{itemize}
  \item \textbf{axm0}_1: \textit{APP, STN, EXT} are distinct blocks.
\end{itemize}
Phase 1. Environment
First Model. Trains and Tracks (2/3): The State

variables: $\text{Trns}$, $\text{Loc}$, $\text{Occ}$

init
begin
    $\text{Trns}$, $\text{Loc}$, $\text{Occ}$ := $\emptyset$, $\emptyset$, $\emptyset$
end

invariants:
inv0_1 : $\text{Trns} \subseteq \text{TRAIN}$
inv0_2 : $\text{Loc} \in \text{Trns} \rightarrow \text{BLOCK}$
inv0_3 : $\text{Occ} = \text{ran} (\text{Loc})$
inv0_4 : $\forall t_1, t_2 \cdot t_1 \in \text{Trns} \land t_2 \in \text{Trains} \land t_1 \neq t_2 \Rightarrow \text{Loc}(t_1) \neq \text{Loc}(t_2)$

- **inv0_1**: $\text{Trns}$ is the set of “monitored” trains.
- **inv0_2–3**: Each monitored train occupies one block (ENV 1).
- **inv0_4**: No two trains are on the same block (SAF 5).
There are 4 events modelling the movements of trains.

- **APPROACH**
  - any \( t \) where
    - \( t \notin \text{Trns} \)
    - \( \text{APP} \notin \text{Occ} \)
  - then
    - \( \text{Trns} := \text{Trns} \cup \{t\} \)
    - \( \text{Loc}(t) := \text{APP} \)
    - \( \text{Occ} := \text{Occ} \cup \{\text{APP}\} \)
  - end

- **ENTER**
  - any \( t \) where
    - \( t \in \text{Trns} \)
    - \( \text{Loc}(t) = \text{APP} \)
    - \( \text{STN} \notin \text{Occ} \)
  - then
    - \( \text{Loc}(t) := \text{STN} \)
    - \( \text{Occ} := (\text{Occ} \cup \{\text{STN}\}) \setminus \{\text{APP}\} \)
  - end

Guards guarantee safety properties.

Events **EXIT** and **LEAVE** are similar.
Phase 1. Environment
First Model. Trains and Tracks (3/3): The Events

- There are 4 events modelling the movements of trains.

Events:
- **APPROACH**
  - `any t where t \notin Trens
  \quad APP \notin Occ
  then
  \quad Trens := Trens \cup \{t\}
  \quad Loc(t) := APP
  \quad Occ := Occ \cup \{APP\}
  end

- **ENTER**
  - `any t where t \in Trens
  \quad Loc(t) = APP
  \quad STN \notin Occ
  then
  \quad Loc(t) := STN
  \quad Occ := (Occ \cup \{STN\}) \setminus \{APP\}
  end

- Guards guarantee safety properties.
- Events **EXIT** and **LEAVE** are similar.
There are 4 events modelling the movements of trains.

**APPROACH**

\[
\text{any } t \text{ where } \\
t \notin Trns \\
\text{APP } \notin \text{Occ} \\
\text{then } \\
\text{Trns} := \text{Trns} \cup \{t\} \\
\text{Loc}(t) := \text{APP} \\
\text{Occ} := \text{Occ} \cup \{\text{APP}\} \\
\text{end}
\]

**ENTER**

\[
\text{any } t \text{ where } \\
t \in \text{Trns} \\
\text{Loc}(t) = \text{APP} \\
\text{STN } \notin \text{Occ} \\
\text{then } \\
\text{Loc}(t) := \text{STN} \\
\text{Occ} := (\text{Occ} \cup \{\text{STN}\}) \setminus \{\text{APP}\} \\
\text{end}
\]

 Guards guarantee safety properties.

Events EXIT and LEAVE are similar.
Phase 1. Environment
Second Model. Signals

carrier sets: LIGHT

constants: RED, GREEN

axioms:
axm1_1 : partition(LIGHT, {RED}, {GREEN})

variables: \ldots, Ent\_sgn, Ext\_sgn

init
begin
\ldots
Ent\_sgn, Ext\_sgn := RED, RED
end

invariants:
inv1_1 : Ent\_sgn \in LIGHT
inv1_2 : Ext\_sgn \in LIGHT

ENV 2: Signals are either red or green.
Phase 1. Environment
Second Model. Signals

**ENV 3:** Trains suppose to obey the signals.

**ENV 4:** Signal changes from green to red automatically.

**Addition invariant (due to guard strengthening)**

\[ \text{inv1}_3 : \ Ent\_sgn = \text{GREEN} \Rightarrow STN \notin \text{Occ} \]
Phase 1. Environment
Second Model. Signals

```
CHANGE_ENTER_SIGNAL
  when
  Ent_sgn = RED
  STN \notin Occ
  then
  Ent_sgn := GREEN
end
```

- Recall: invariant \textbf{inv1\_3}

```
inv1\_3 : Ent_sgn = GREEN \Rightarrow STN \notin Occ
```

- Similar for events EXIT and CHANGE_EXIT_SIGNAL.
Phase 2. Actuators

variables: \ldots, act\_ent\_sgn, act\_ext\_sgn

invariants:
\begin{align*}
\text{inv2}\_1 & : act\_ent\_sgn \in \text{BOOL} \\
\text{inv2}\_2 & : act\_ext\_sgn \in \text{BOOL}
\end{align*}

init
begin
\ldots
act\_ent\_sgn, act\_ext\_sgn := \text{FALSE}, \text{FALSE}
end
Phase 2. Actuators

(\text{abs}_\text{c})\text{CHANGE\_ENTER\_SIGNAL}
\begin{align*}
\text{when} & \quad \text{Ent\_sgn} = \text{RED} \\
& \quad \text{STN} \notin \text{Occ} \\
\text{then} & \quad \text{Ent\_sgn} := \text{GREEN} \\
\text{end}
\end{align*}

(\text{cnc}_\text{c})\text{CHANGE\_ENTER\_SIGNAL}
\begin{align*}
\text{when} & \quad \text{act\_ent\_sgn} = \text{TRUE} \\
\text{then} & \quad \text{Ent\_sgn} := \text{GREEN} \\
& \quad \text{act\_ent\_sgn} := \text{FALSE} \\
\text{end}
\end{align*}

- Additional invariant

\text{inv2\_3} : \quad \text{act\_ent\_sgn} = \text{TRUE} \Rightarrow \\
\quad \text{Ent\_sgn} = \text{RED} \land \text{STN} \notin \text{Occ}

- ENV 8: The signals is changed accordingly to the actuators.

- Similar for event \text{CHANGE\_EXIT\_SIGNAL}. 
Phase 2. Actuators

```plaintext
ctrl_change_enter_signal
  when
    act_ent_sgn = FALSE
    Ent_sgn = RED
    STN ∉ Occ
  then
    act_ent_sgn := TRUE
  end
```

- Take into account the following invariant

  ```plaintext
  inv2_3 : act_ent_sgn = TRUE⇒ Ent_sgn = RED ∧ STN ∉ Occ
  ```

- Similar for events `ctrl_change_exit_signal` and `ctrl_change_both_signal`. 

Thai Son Hoang (ETH-Zürich)
Phase 3. Sensors and Controller

variables: \ldots, sen_blk, sen_ent_sgn, sen_ext_sgn

invariants:
\begin{align*}
\text{inv3}_1 & : \quad \text{sen}._{blk} = \text{Occ} \\
\text{inv3}_2 & : \quad \text{sen}._{ent}._{sgn} = \text{Ent}._{sgn} \\
\text{inv3}_3 & : \quad \text{sen}._{ext}._{sgn} = \text{Ext}._{sgn}
\end{align*}

init
begin
\ldots
\text{sen}._{blk} := \emptyset \\
\text{sen}._{ent}._{sgn}, \text{sen}._{ext}._{sgn} := \text{RED}, \text{RED}
end

- \text{sen}_blk: Sensors detecting if a block is occupied.
- \text{sen}_ent_sgn, \text{sen}_ext_sgn: Sensors detecting status of signals.
- Invariants: Sensors reflect the status of components (ENV 7).
Phase 3. Sensors and Controller

- Additional assignment(s) in physical events set the value of the sensor appropriately.

- Example

```plaintext
ENTER
   any  t  where
   
   then
   Loc(t) := STN
   Occ := (Occ \cup \{STN\}) \setminus \{APP\}
   Ent_sgn := RED
   sen_blk := (sen_blk \cup \{STN\}) \setminus \{APP\}
   sen_ent_sgn := RED
end
```
Phase 3. Sensors and Controller

\[(\text{abs\_})\text{ctrl\_change\_enter\_signal}\]
\[
\text{when}
\begin{align*}
act\_ent\_sgn &= \text{FALSE} \\
Ent\_sgn &= \text{RED} \\
\text{STN} &\notin \text{Occ}
\end{align*}
\[
\text{then}
\begin{align*}
act\_ent\_sgn &:= \text{TRUE}
\end{align*}
\]
\[
\text{end}
\]

\[(\text{cnc\_})\text{ctrl\_change\_enter\_signal}\]
\[
\text{when}
\begin{align*}
act\_ent\_sgn &= \text{FALSE} \\
\text{sen\_ent\_sgn} &= \text{RED} \\
\text{STN} &\notin \text{sen\_blk}
\end{align*}
\[
\text{then}
\begin{align*}
act\_ent\_sgn &:= \text{TRUE}
\end{align*}
\]
\[
\text{end}
\]

- Refinement is trivial with the invariants

\[
\text{inv3\_1} : \quad \text{sen\_blk} = \text{Occ}
\]

\[
\text{inv3\_2} : \quad \text{sen\_ent\_sgn} = \text{Ent\_sgn}
\]
Phase 4. Schedule

- FUN 6: Every train has some predefined route plan.

  constants: \( plan \)

  \textbf{axioms:}
  \[
  \text{axm4}_1 : \quad plan \in TRAIN \rightarrow BOOL
  \]

- Plan of the train at the approaching block.

  variables: \( a_{\text{plan}} \)

  \textbf{invariants:}
  \[
  \text{inv4}_1 : \quad \forall t . t \in Trns \land \Loc(t) = \text{APP} \Rightarrow a_{\text{plan}} = plan(t)
  \]

```plaintext
APPROACH
  any \ t \ where
  \ldots
  then
  \ldots
  a_{\text{plan}} := plan(t)
end
```


\( \text{Thai Son Hoang (ETH-Zürich)} \)

\( \text{Signal Control using Event-B} \)

\( \text{McMaster University, 16/11/10} \)
Phase 4. Schedule

- **ctrl_change_enter_signal**
  
  ```
  when
  ... 
  a_plan = TRUE
  APP ∈ sen_blk
  then
  ... 
  end
  ```

- **ctrl_change_both_signal**
  
  ```
  when
  ... 
  a_plan = FALSE
  APP ∈ sen_blk
  then
  ... 
  end
  ```

- **ctrl_change_exit_signal**
  
  ```
  when
  ... 
  STN ∈ sen_blk
  then
  ... 
  end
  ```

- Schedule appropriately using the plan.
- Change the signals only when it is necessary.
Phase 4. Schedule

- ctrl_change_enter_signal
  
  ```
  when
  ...
  a_plan = TRUE
  APP ∈ sen_blk
  then
  ...
  end
  ```

- ctrl_change_both_signal
  
  ```
  when
  ...
  a_plan = FALSE
  APP ∈ sen_blk
  then
  ...
  end
  ```

- ctrl_change_exit_signal
  
  ```
  when
  ...
  STN ∈ sen_blk
  then
  ...
  end
  ```

- Schedule appropriately using the plan.
- Change the signals only when it is necessary.
## Development Summary

<table>
<thead>
<tr>
<th>Phase</th>
<th>Model</th>
<th>Requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>Model 0</td>
<td>ENV 1, SAF 5</td>
</tr>
<tr>
<td></td>
<td>Model 1</td>
<td>ENV 2, ENV 3, ENV 4</td>
</tr>
<tr>
<td>Phase 2</td>
<td>Model 2</td>
<td>ENV 8</td>
</tr>
<tr>
<td>Phase 3</td>
<td>Model 3</td>
<td>ENV 7</td>
</tr>
<tr>
<td>Phase 4</td>
<td>Model 4</td>
<td>FUN 6</td>
</tr>
</tbody>
</table>
A modelling method for **discrete transition systems**.

**Mathematical language** of first-order logic and set theory.

**Step-wise refinement** to reduce development complexity.

**Correct by construction**.

Can be used to model a **wide range of applications**.
Summary. Developing Control System

1. Start with model of the problem: the environment with various constraints.

2. Step-by-step introduce:
   - Actuators (output of the controller).
   - Sensors (input of the controller) and the controller.

3. Schedule the controller appropriately.

4. Important features of the approach:
   - Safety properties are introduced early in terms of the environment: Safety properties are maintained by refinement.
   - Scheduling details in later phase of the development: Separation of concerns between safety properties and schedule.