4.2.4. Extension

Definition: Extension.

- Domain extension is a domain requirements facet.
- It is an operation performed on a domain description or a requirements prescription.
- It effectively extends a domain description by entities, functions, events and/or behaviours conceptually possible, but not necessarily humanly or technologically feasible in the domain (as it was).
- Figure 5 on the facing page abstracts some of the extensions to nets: the plaza entry and exit booths.

The following is a prolonged example.
- It contains three kinds of formalisations:
  - a RAISE/CSP model,
  - a Duration Calculus model [zcc+mrh2002,olderogdirks2008] and
  - a Timed Automata model [AluDil:94,olderogdirks2008].
- The narrative for all three models are given when narrating the RAISE/CSP model.
4.2.4.1. Intuition

- A toll road system is delimited by toll plazas with entry and exit booths with their gates.
- To get access, from outside, to the roads within the toll road system, a car must pass through an entry booth and its entry gate. To leave the roads within the toll road system a car must pass through an exit booth and its exit gate.
- Cars collect tickets upon entry and return these tickets upon exit and pay a fee for having driven on the toll roads.
- The gates help ensure that cars have collected tickets and have paid their dues.

4.2.4.2. Descriptions

4.2.4.2.1. A RAISE/CSP Model

We use the CSP property [TheSEBook123, CARH:Electronic] of RSL.

- Toll Booth Plazas

With respect to toll road systems we focus on just their plazas: that is, where cars enter and leave the systems.

The below description is grossly simplified: instead of plazas having one or more entry and one or more exit booths (both with gates), we just assume one (pair: booth/gate) of each.

141. A toll plaza consists of a one pair of an entry booth and and entry gate and one pair of an exit booth and an exit gate.
142. Entry booths consist of an entry sensor, a ticket dispenser and an exit sensor.
143. Exit booths consist of an entry sensor, a ticket collector, a payment display and a payment component.

type

141. \( PZ = (EB \times G) \times (XB \times G) \)
142. \( EB = ... \)
143. \( XB = ... \)
4. An Ontology of Requirements Constructions

4.2. Domain Requirements

4.2.4. Extension

4.2.4.2. Descriptions

4.2.4.2.1. A RAISE/CSP Model

⊕ Cars ⊕

144. There are vehicles.

145. Vehicles have unique vehicle identifications.

type
144. V
145. VId

value
145. obs_VId: V \rightarrow VId

axiom
145. \forall v,v':V \cdot v \neq v' \Rightarrow \text{obs}_V\text{Id}(v) \neq \text{obs}_V\text{Id}(v')

⊕ Entry Booths ⊕

• The description now given is an idealisation.
• It assumes that everything works:
  – that the vehicles behave as expected and
  – that the electro-mechanics of booths and gates do likewise.

146. An entry_sensor registers whether a car is entering the entry booth or not,

(a) that is, for the duration of the car passing the entry_sensor that
sensor senses the car identification \text{cid}
(b) otherwise it senses “nothing”.

⊕ Gates ⊕

147. A ticket_dispenser

(a) either holds a ticket or does not hold a ticket, i.e., no_ticket;
(b) normally it does not hold a ticket;
(c) the ticket_dispenser holds a ticket soon after a car has passed the
entry_sensor;
(d) the passing car collects the ticket –
(e) after which the ticket_dispenser no longer holds a ticket.

148. An exit_sensor

(a) registers the identification of a car leaving the toll booth
(b) otherwise it senses “nothing”.

149. A gate

(a) is either closed or open;
(b) it is normally closed;
(c) if a car is entering it is secured set to close (as a security measure);
(d) once a car has collected a ticket it is set to open;
(e) and once a car has passed the exit_sensor it is again set to close.
4. An Ontology of Requirements Constructions

4.2. Domain Requirements

4.2.4. Extension

4.2.4.2. Descriptions

4.2.4.2.1 A RAISE/CSP Model

The Entry Plaza System

<table>
<thead>
<tr>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, CI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>obs_CI: (C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry_sensor: CI</td>
</tr>
<tr>
<td>ticket_dispenser: Ticket</td>
</tr>
<tr>
<td>exit_sensor: CI</td>
</tr>
<tr>
<td>gate_ch: G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>vs: V-set</td>
</tr>
<tr>
<td>eb: EB, xb: XB, eg, xg: G</td>
</tr>
</tbody>
</table>

system: G × EB × V-set × XB × G

system(eg, eb, vs, xb, xg) ≡
||{car(obs_CI(c), c | C ∈ cs} || entry_booth(eb) || entry_gate(eg) || ...

car: CI × C → out entry_sensor, exit_sensor

in ticket_dispenser Unit

car(ci, c) ≡
entry_sensor ! ci ;
let ticket = ticket_dispenser ? assert: ticket ≠ no_ticket in
ticket_dispenser ! no_ticket ;
exit_sensor ! ci ;
car(add(ticket, c)) end

entry_booth: Unit → in entry_sensor, exit_sensor

out ticket_dispenser

out gate_ch Unit

entry_booth(b) ≡
gate_ch ! close ;
let ci = entry_sensor ? in
ticket_dispenser ! make_ticket(cid) ;
let res = ticket_dispenser ? assert: res = no_ticket ;
gate_ch ! open ;
let ci′ = exit_sensor ? assert: ci′ = ci ;
gate_ch ! close ;
entry_booth(add_TICKET(ticket, b)) end end end

entry_gate: G → in gate Unit

entry_gate(g) ≡
case gate_ch ? of
close → exit_gate(close) assert: g = open,
onopen → exit_gate(open) assert: g = close
end

add_Ticket: Ticket × C ~ C

pre add_Ticket(t, c): ~has_Ticket(c)
post: add_Ticket(t, c): has_Ticket(c)
4.2.4.2.2. A Duration Calculus Model

- We use the Duration Calculus [zcc+mrh2002,olderogdirks2008] extension to RSL.
- We abstract the channels of the RAISE/CSP model to now be Boolean-valued variables.

In the next section, “A Duration Calculus Model”, we shall start refining the descriptions given above.
- We do so in order to handle failures of vehicles to behave as expected and of the electro-mechanics of booths and gates.

The gate must be closed within no more than $\delta_{eg}$ time units after the entry_sensor has registered that a car is entering the toll booth.

A ticket must be in the ticket_dispenser within $\delta_{et}$ time units after the entry_sensor has registered that a car is entering the toll booth.

The ticket is in the ticket_dispenser at most $\delta_{tdc}$ time units.

The gate must be open within $\delta_{go}$ time units after a ticket has been collected.

The exit sensor is registering (i.e., is on) the identification of exiting cars and is not registering anything when no car is passing (i.e., is off).
4.2.4.2.3. A Timed Automata Model

- A timed automaton [Alur & Dill, 1994] for a configuration of an entry gate, its entry booth and a car is shown in Fig. 7 on the next page.
- Figure 8 on page 231 shows a car, an exit booth and its exit gate interactions.
- They are more-or-less “derived” from the example of Sect. 7.5 of [Alur & Dill, 1994] (Pages 42–45).
- The right half of the car timed automaton of Fig. 7 on the next page – is to be thought of as the same as the left half of the car timed automaton of Fig. 8 on page 231, – cf. the vertical dotted (...) line.

**Figure 7:** A timed automata model of gate, entry booth and car interactions

\[
\text{System: } G \times EV \times V\text{-set} \times XB \times G \rightarrow \text{Unit}
\]

\[
\text{System}(eg, eb, vs, xb, xg) \equiv \\
\quad \text{Entry\_Gate}(eg) \parallel \text{Entry\_Booth}(eb) \parallel \\
\quad \parallel \{\text{Car}(\text{obs\_Cl}(c), c)|c: C, v: Cc, v \in cs\} \parallel \\
\quad \text{Exit\_Booth}(xb) \parallel \text{Exit\_Gate}(xg)
\]
4. An Ontology of Requirements Constructions

4.2. Domain Requirements

4.2.4. Extension Descriptions

4.2.4.2. Timed Automata Model

Figure 8: A timed automata model of car, exit booth and gate interactions

Definition: Fitting.

- By domain requirements fitting we understand an operation
  - which takes \( n \) domain requirements prescriptions, \( d_i \) (\( i = \{1..n\} \)),
  - claimed to share \( m \) independent sets of tightly related sets of simple entities, actions, events and/or behaviours
  - and map these into \( n + m \) domain requirements prescriptions, \( \delta_{ij} \) (\( j = \{1..n+m\} \)),
  - where \( m \) of these, \( \delta_{n+k} \) (\( k = \{1..m\} \))
  - capture the \( m \) shared phenomena and concepts
  - and the other \( n \) prescriptions, \( \delta_{i} \) (\( i = \{1..n\} \)),
  - except that they now,
  - (instead of the “more-or-less” shared prescriptions, that are now consolidated in \( \delta_{n+k} \))
  - prescribe interfaces between \( \delta_{i} \) and \( \delta_{n+k} \) for \( i : \{1..n\} \).

4.2.5. Fitting

4.2.5.1. Examples

TO BE WRITTEN

4.3. Interface Requirements

Definition: Interface Requirements.

- Interface requirements are those requirements
  - which can on be expressed using professional terms
  - and from both the domain and the machine.

Thus, by interface requirements we understand

- the expression of expectations
  - as to which software-software, or software-hardware interface places (i.e., channels),
  - inputs and outputs (including the semiotics of these input/outputs)
  - there shall be in some contemplated computing system.
4. An Ontology of Requirements Constructions

4.3. Interface Requirements

4.3.1. But First: On Shared Phenomena and Concepts

Definition: Shared Phenomenon or Concept.

- A shared phenomenon (or concept) is a phenomenon (respectively a concept)
  - which is present in some domain (say in the form of facts, knowledge or information)
  - and which is also represented in the machine (say in the form of some entity, simple, action, event or behaviour).

- A phenomenon of a domain, when shared, becomes a concept of the machine.

- We shall give some examples – but they are just illustrative.

- Proper narration and formalisation is left to the reader!

4.3.2. Shared Simple Entities

Definition: Shared Simple Entity.

- By a shared simple entity we mean a simple entity
  - which both occurs
  - in the domain (as a phenomenon or a concept)
  - and in the machine.

- Simple entities that are shared between the domain and the machine must initially be input to the machine.

- Dynamically arising simple entities must likewise be input
  - and all such machine entities
  - must have their attributes updated, when need arise.

- Requirements for shared simple entities
  - thus entail requirements for their representation
  - and for their human/machine and/or machine/machine transfer dialogue.

- Main shared entities are those of hubs and links.

- Representations of hubs and links “within” the machine
  - necessarily abstracts many of the properties of hubs and links;
  - some (such) attributes may not be represented altogether.

- As for human input,
  - some man/machine dialogue
  - based around a set of visual display unit screens
  - with fields for the input of hub,
  - respectively link attributes
  - can then be devised.

- Etc.
4.3.3. Shared Actions

Definition: Shared Action.

• By a shared action we mean an action
  – that can only be partly computed by the machine.
  – That is, the machine,
    * in order to complete an action,
    * may have to inquire with the domain
      * (in order, say, to extract some measurable, time-varying
        simple entity attribute value)
    * in order to proceed in its computation.

4.3.4. Shared Events

Definition: Shared Event.

• By a shared event we mean
  – an event whose occurrence in the domain
  – need be communicated to the machine
    and, vice-versa,
  – an event whose occurrence in the machine
  – need be communicated to the domain.
4.3.5. Shared Behaviours

Definition: Shared Behaviour.

- By a shared behaviour we mean a behaviour
  - many of whose actions and events occur both
    - in the domain
    - and in the machine
  - (in some encoded form, and in the same sequence).

4.4. Machine Requirements

Definition: Machine Requirements.

- Machine requirements are those requirements which, in principle,
  - can be expressed without using professional domain terms
  - (for which these requirements are established).

- Thus, by machine requirements,
  - we understand requirements put specifically to,
    - i.e., expected specifically from, the machine.

- We normally analyse machine requirements into
  - performance requirements,  platform requirements and
  - dependability requirements,  documentation requirements.
  - maintenance requirements.

4.3.5.1. Example

- A typical toll road net use behaviour is as follows:
  - Entry at some toll plaza: receipt of electronic ticket,
  - placement of ticket in special ticket “pocket” in front window,
  - the raising of the entry booth toll gate;
  - drive up to [first] toll road hub (with electronic registration of time of occurrence),
  - drive down a selected link (with electronic registration of time of occurrence of entry to and exit from link),
  - then a repeated number of zero, one or more
    * toll road hub and
    * link visits –
    * some of which may be “repeats” –
  - ending with a drive down from a toll road hub to a toll plaza
  - with the return of the electronic ticket, etc.
End of Lecture 6: REQUIREMENTS – from Extension “out”