Dines Bjørner's MAP-i Lecture # 9

Domain Requirements: Extension and Fitting

Thursday, 28 May 2015: 10:00–11:15

7.2.4. Domain Extension

Definition 30 . Extension: *By* **domain extension** *we understand the*

- *introduction of endurants and perdurants that were not feasible in the original domain,*
- *but for which, with computing and communication,*
- *and with new, emerging technologies,*
- *for example, sensors, actuators and satellites,*
- *there is the possibility of feasible implementations,*
- *hence requirement,*
- *that what is introduced becomes[27] part of the unfolding requirements prescription* ■

[27] become or becomes ?

7.2.4.1. The Core Requirements Example: Domain Extension

Example 82 . Domain Requirements. Extension Vehicles: Parts, Properties and Channels:

184 There is a domain, $\delta_{\mathcal{E}}$:$\Delta_{\mathcal{E}}$, which contains

185 a fleet, $f_{\mathcal{E}}$:$F_{\mathcal{E}}$,

186 of a set, $vs_{\mathcal{E}}$:$VS_{\mathcal{E}}$, of

187 extended vehicles, $v_{\mathcal{E}}$:$V_{\mathcal{E}}$ — their extension amounting to

a. a dynamic, active and biddable attribute[28], whose value, ti-gpos:TiGpos, at any time, reflects that vehicle's *time-stamped global positions*

b. The vehicle's GNSS receiver calculates its local position, lpos:LPOS, based on these signals.

c. Vehicles access these external attributes via the external attribute channel, attr_TiGPos_ch, cf. Item 100 on Slide 273.

d. The vehicle can, on its own volition, offer the timed local position, ti-lpos:TiLPos to the price calculator, $c_{\mathcal{E}}$:$C_{\mathcal{E}}$ along a vehicles-to-calculator channel, v_c_ch.

[28] See Sect. Slide 187.

type
184. $\Delta_{\mathcal{E}}$
185. $\mathsf{F}_{\mathcal{E}}$
186. $\mathsf{VS}_{\mathcal{E}} = \mathsf{V}_{\mathcal{E}}$**-set**
187. $\mathsf{V}_{\mathcal{E}}$
187a.. TiGPos = $\mathbb{T}$ × GPOS
187a.. TiLPos = $\mathbb{T}$ × LPOS
187b.. GPOS, LPOS
value
185. **obs_part**_$\mathsf{F}_{\mathcal{E}}$: $\Delta_{\mathcal{E}} \to \mathsf{F}_{\mathcal{E}}$
186. **obs_part**_$\mathsf{VS}_{\mathcal{E}}$: $\mathsf{F}_{\mathcal{E}} \to \mathsf{VS}_{\mathcal{E}}$
186. vs:**obs_part**_$\mathsf{VS}_{\mathcal{E}}(\mathsf{F}_{\mathcal{E}})$
channel
187c.. {attr_TiGPos_ch[vi]|viLVI•vi ∈ xtr_VIs(vs)}: TiGPos
187d.. {v_c_ch[vi,ci]
187d.. | vi:VI,ci:CI•vi∈vis∧ci=**uid**_C(c)}:(VI×TiLPos)
value
187a.. attr_TiGPos_ch[vi]?
187b.. loc_pos: GPOS → LPOS

- where vis:VI**-set** is the set unique vehicle identifiers of all vehicles of the requirements domain fleet, f:$\mathsf{F}_{\mathcal{R}_{\mathcal{E}}}$.

We define two auxiliary functions,

188 **xtr_vs**, which given a domain, or a fleet, extracts its set of vehicles, and

189 **xtr_vis** which given a set of vehicles generates their unique identifiers.

value

188. xtr_vs: $(\Delta_{\mathcal{E}}|F_{\mathcal{E}}|VS_{\mathcal{E}}) \rightarrow V_{\mathcal{E}}$**-set**
188. xtr_vs(arg) $\equiv$
188. **is**_$\Delta_{\mathcal{E}}$(arg) $\rightarrow$ **obs_part**_$VS_{\mathcal{E}}$(**obs_part**_$F_{\mathcal{E}}$(arg)),
188. **is**_$F_{\mathcal{E}}$(arg) $\rightarrow$ **obs_part**_$VS_{\mathcal{E}}$(arg),
188. **is**_$VS_{\mathcal{E}}$(arg) $\rightarrow$ arg
189. xtr_vis: $(\Delta_{\mathcal{E}}|F_{\mathcal{E}}|VS_{\mathcal{E}}) \rightarrow$ VI**-set**
189. xtr_vis(arg) $\equiv$ {**uid**_VI(v)|v $\in$ xtr_vs(arg)}

Example 83 . Domain Requirements. Extension Toll-road Net: Parts, Properties and Channels:

- We extend the domain with toll-gates for vehicles entering and exiting the toll-road entry and exit links.
- Figure 8 illustrates the idea of gates.

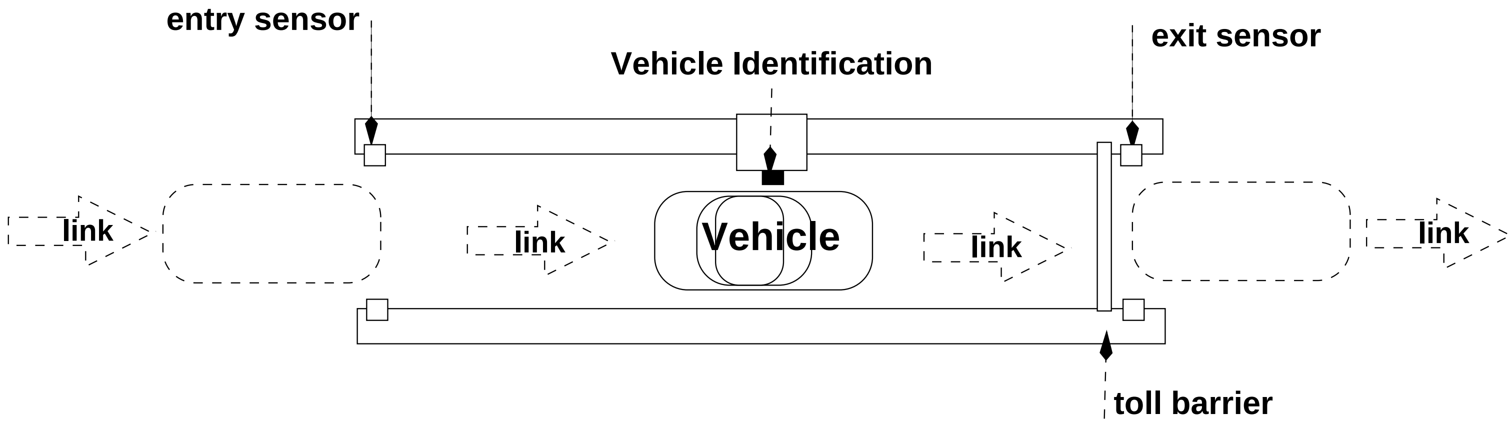

Figure 8: A toll plaza gate

- Figure 8 on the facing slide is intended to illustrate a vehicle entering (or exiting) a toll-road entry link.
  - The toll-gate is equipped with three sensors: an entry sensor, a vehicle identification sensor and an exit sensor.
  - The entry sensor serves to prepare the vehicle identification sensor.
  - The exit sensor serves to prepare the gate for closing when a vehicle has passed.
  - The vehicle identification sensor identifies the vehicle and "delivers" a pair: the current time and the vehicle identifier.
  - Once the vehicle identification sensor has identified a vehicle the gate opens.

190 There is the domain, δ:$\Delta_{\mathcal{E}}$,

191 which contains the extended net, n:$N_{\mathcal{E}}$, with the net extension amounting to the toll-road net, $TRN_{\mathcal{E}}$,

192 that is, the instantiated toll-road net, trn:$TRN_{\mathcal{I}}$, is extended, into trn:$TRN_{\mathcal{E}}$, with entry, eg:EG, and exit, xg:XG, toll-gates.

From entry- and exit-gates we can observe

a. their unique identifier and their mereology: being paired with the entry-, respectively exit link and the calculator (by their unique identifiers); further

b. a pair of gate enter and leave sensors modeled as external attribute channels, (ges:ES,gls:XS), and

c. a time-stamped vehicle identity sensor modeled as external attribute channels.

type

190 $\Delta_{\mathcal{E}}$

191 $\mathsf{N}_{\mathcal{E}}$

192 $\mathsf{TRN}_{\mathcal{E}} = (\mathsf{EG}\times\mathsf{XG})^* \times \mathsf{TRN}_{\mathcal{I}}$

192a. GI

value

190 **obs_part**_$\mathsf{N}_{\mathcal{E}}$: $\Delta_{\mathcal{E}} \rightarrow \mathsf{N}_{\mathcal{E}}$

191 **obs_part**_$\mathsf{TRN}_{\mathcal{E}}$: $\mathsf{N}_{\mathcal{E}} \rightarrow \mathsf{TRN}_{\mathcal{E}}$

192a. **uid**_G: (EG|XG) → GI

192a. **obs_mereo**_G: (EG|XG) → (LI×CI)

channel

192b. {attr_enter_ch[gi]|gi:GI•...} "enter"

192b. {attr_leave_ch[gi]|gi:GI•...} "leave"

192c. {attr_passing_ch[gi]|gi:GI•...} TIVI

type

192c. TIVI = $\mathbb{T}$ × VI

We define some auxiliary functions over toll-road nets, trn:$TRN_{\mathcal{E}}$:

193 xtr_eGℓ extracts the ℓist of entry gates,

194 xtr_xGℓ extracts the ℓist of exit gates,

195 xtr_eGIds extracts the $s$et of entry gate identifiers,

196 xtr_xGIds extracts the $s$et of exit gate identifiers,

197 xtr_Gs extracts the $s$et of all gates, and

198 xtr_GIds extracts the $s$et of all gate identifiers.

value

193 xtr_eGℓ: TRN$_{\mathcal{E}}$ $\rightarrow$ EG*

193 xtr_eGℓ(pgl,_) $\equiv$

193 {eg|(eg,xg):(EG,XG)·(eg,xg)$\in$ **elems** pgl}

194 xtr_xGℓ: TRN$_{\mathcal{E}}$ $\rightarrow$ XG*

194 xtr_xGℓ(pgl,_) $\equiv$

194 {xg|(eg,xg):(EG,XG)·(eg,xg)$\in$ **elems** pgl}

195 xtr_eGIds: TRN$_{\mathcal{E}}$ $\rightarrow$ GI-**set**

195 xtr_eGIds(pgl,_) $\equiv$

195 {**uid**_GI(g)|g:EG·g $\in$ xtr_eGs(pgl,_)}

196 xtr_xGIds: TRN$_{\mathcal{E}}$ $\rightarrow$ GI-**set**

196 xtr_xGIds(pgl,_) $\equiv$

196 {**uid**_GI(g)|g:EG·g $\in$ xtr_xGs(pgl,_)}

197 xtr_Gs: TRN$_{\mathcal{E}}$ $\rightarrow$ G-**set**

197 xtr_Gs(pgl,_) $\equiv$

197 xtr_eGs(pgl,_) $\cup$ xtr_xGs(pgl,_)

198 xtr_GIds: TRN$_{\mathcal{E}}$ $\rightarrow$ GI-**set**

198 xtr_GIds(pgl,_) $\equiv$

198 xtr_eGIds(pgl,_) $\cup$ xtr_xGIds(pgl,_)

199 A well-formedness condition expresses

a. that there are as many entry end exit gate pairs as there are toll-plazas,

b. that all gates are uniquely identified, and

c. that each entry [exit] gate is paired with an entry [exit] link and has that link's unique identifier as one element of its mereology, the other elements being the calculator identifier and the vehicle identifiers.

The well-formedness relies on awareness of

200 the unique identifier, ci:CI, of the road pricing calculator, c:C, and

201 the unique identifiers, vis:VI-**set**, of the fleet vehicles.

```
value
200   ci:CI
201   vis:VI-set
axiom
199   ∀ n:N_R3, trn:TRN_R3 ·
199      let (exgl,(exl,hl,lll)) = obs_part_TRN_R3(n) in
199a.       len exgl = len exl = len hl = len lll + 1
199b.    ∧ card xtr_GIds(exgl) = 2 ∗ len exgl
199c.    ∧ ∀ i:Nat·i ∈ inds exgl·
199c.          let ((eg,xg),(el,xl)) = (exgl(i),exl(i)) in
199c.          obs_mereo_G(eg) = (uid_U(el),ci,vis)
199c.        ∧ obs_mereo_G(xg) = (uid_U(xl),ci,vis) end end
```

Example 84 . Domain Requirements. Extension Parts, Properties and Channels:

202 The road pricing calculator repeatedly receives

a. information, (vi,(τ,pos)):VITIPOS,

b. sent by vehicles as to their identify and time-stamped position

c. over a channel, v_c_ch indexed by the c:$C_{\mathcal{E}}$ and the vehicle identities.

203 The road pricing calculator has a number of attributes:

a. a traffic map, trm:TRM, which, for each vehicle inside the toll-road net, records a chronologically ordered list of each vehicle's timed position, (τ,vp), and

b. a (total) road location function, vplf:VPLF.

i The $v$ehicle $p$osition $l$ocation $f$unction, vplf:VPLF, is subject to another function, locate_VPos, which, given a local position, lpos:LPos, yields the vehicle position designated by the `GNSS`-provided position, or yields the response that the provided position is off the toll-road net.

ii This result is used by the road-pricing calculator to conditionally

A either update the traffic map, trm:TRM, recording also the relevant time,

B or reset that vehicle's traffic recording while send a bill for the just completed journey.

type

202a. $\mathsf{VITIPos} = \mathsf{VI} \times (\mathbb{T} \times \mathsf{LPos})$

value

202a. ... v_c_ch[ci,vi] ? ...

202b. ... v_c_ch[ci,vi] ! (vi,(τ,p)) ...

channel

202c. $\{$v_c_ch[ci,vi]|vi:VI•vi $\in$ vis$\}$:VITIPos

type

203a. $\mathsf{TRM} = \mathsf{VI} \underset{m}{\rightarrow} (\mathbb{T} \times \mathsf{VPos})^*$

203b. $\mathsf{VPLF} = \mathsf{LPos} \rightarrow \mathsf{VPos} \mid$ ″off_TRN″

value

203(b.)i locate_LH: LPos$\times$RLF $\rightarrow$ (VPos|″off_TRN″)

203(b.)iiA update_TRM: VI$\times$($\mathbb{T}\times$VPos)$\rightarrow$TRM$\rightarrow$TRM

203(b.)iiB reset_TRM: VI$\rightarrow$TRM$\rightarrow$TRM

Example 85 . Domain Requirements. Extension Main Sorts:

204 The main sorts of the road-pricing domain, $\Delta_{\mathcal{E}}$, are

a. the net, projected, instantiated (to include the specific toll-road net), made more determinate and now extended, $\mathsf{N}_{\mathcal{E}}$, with toll-gates;

b. the fleet, $\mathsf{F}_{\mathcal{E}}$,

c. of sets, VS, of extended vehicles, $\mathsf{V}_{\mathcal{E}}$;

d. the extended toll-road net, $\mathsf{TRN}_{\mathcal{E}}$, extending the instantiated toll-road net, $\mathsf{TRN}_{\mathcal{I}}$, with toll-gates; and

e. the road pricing calculator, $\mathsf{C}_{\mathcal{E}}$.

type

204. $\Delta_{\mathcal{E}}$

204a.. $\mathsf{N}_{\mathcal{E}}$

204b.. $\mathsf{F}_{\mathcal{E}}$

204c.. $\mathsf{VS}_{\mathcal{E}} = \mathsf{V}_{\mathcal{E}}$**-set**

204d.. $\mathsf{TRN}_{\mathcal{E}} = (\mathsf{EG}\times\mathsf{XG})^* \times \mathsf{TRN}_{\mathcal{I}}$

204e.. $\mathsf{C}_{\mathcal{E}}$

value

204a.. **obs_part**_$\mathsf{N}_{\mathcal{E}}$: $\Delta \rightarrow \mathsf{N}_{\mathcal{E}}$

204b.. **obs_part**_$\mathsf{F}_{\mathcal{E}}$: $\Delta \rightarrow \mathsf{F}_{\mathcal{E}}$

204c.. **obs_part**_$\mathsf{VS}_{\mathcal{E}}$: $\Delta \rightarrow \mathsf{VS}_{\mathcal{E}}$

204d.. **obs_part**_$\mathsf{TRN}_{\mathcal{E}}$: $\mathsf{N}_{\mathcal{E}} \rightarrow \mathsf{TRN}_{\mathcal{E}}$

204e.. **obs_part**_$\mathsf{C}_{\mathcal{E}}$: $\Delta \rightarrow \mathsf{C}_{\mathcal{E}}$

Example 86 . Domain Requirements. Extension Global Values:

- We exemplify a road-pricing system behaviour, in Example 87 on Slide 442,
- based on the following global values.

205 There is a given domain, $\delta_{\mathcal{E}}$:$\Delta_{\mathcal{E}}$;

206 there is the net, $n_{\mathcal{E}}$:$N_{\mathcal{E}}$, of that domain;

207 there is toll-road net, $trn_{\mathcal{E}}$:$TRN_{\mathcal{E}}$, of that net;

208 there is a set, $egs_{\mathcal{E}}$:$EG_{\mathcal{E}}$-**set**, of entry gates;

209 there is a set, $xgs_{\mathcal{E}}$:$XG_{\mathcal{E}}$-**set**, of exit gates;

210 there is a set, $gis_{\mathcal{E}}$:$GI_{\mathcal{E}}$-**set**, ofgate identifiers;

211 there is a set, $vs_{\mathcal{E}}$:$V_{\mathcal{E}}$-**set**, of vehicles;

212 there is a set, $vis_{\mathcal{E}}$:$VI_{\mathcal{E}}$-**set**, of vehicle identifiers;

213 there is the road-pricing calculator, $c_{\mathcal{E}}$:$C_{\mathcal{E}}$ and

214 there is its unique identifier, $ci_{\mathcal{E}}$:CI.

value

205. $\delta_{\mathcal{E}}:\Delta_{\mathcal{E}}$
206. $\mathsf{n}_{\mathcal{E}}:\mathsf{N}_{\mathcal{E}} = \textbf{obs\_part}\_\mathsf{N}_{\mathcal{E}}(\delta_{\mathcal{E}})$
207. $\mathsf{trn}_{\mathcal{E}}:\mathsf{TRN}_{\mathcal{E}} = \textbf{obs\_part}\_\mathsf{TRN}_{\mathcal{E}}(\mathsf{n}_{\mathcal{E}})$
208. $\mathsf{egs}_{\mathcal{E}}:\mathsf{EG}\textbf{-set} = \mathsf{xtr\_egs}(\mathsf{trn}_{\mathcal{E}})$
209. $\mathsf{xgs}_{\mathcal{E}}:\mathsf{XG}\textbf{-set} = \mathsf{xtr\_xgs}(\mathsf{trn}_{\mathcal{E}})$
210. $\mathsf{gis}_{\mathcal{E}}:\mathsf{XG}\textbf{-set} = \mathsf{xtr\_gis}(\mathsf{trn}_{\mathcal{E}})$
211. $\mathsf{vs}_{\mathcal{E}}:\mathsf{V}_{\mathcal{E}}\textbf{-set} = \textbf{obs\_part}\_\mathsf{VS}(\textbf{obs\_part}\_\mathsf{F}_{\mathcal{E}}(\delta_{\mathcal{E}}))$
212. $\mathsf{vis}_{\mathcal{E}}:\mathsf{VI}\textbf{-set} = \{\textbf{uid}\_\mathsf{VI}(\mathsf{v}_{\mathcal{E}})|\mathsf{v}_{\mathcal{E}}:\mathsf{V}_{\mathcal{E}}\cdot\mathsf{v}_{\mathcal{E}} \in \mathsf{vs}_{\mathcal{E}}\}$
213. $\mathsf{c}_{\mathcal{E}}:\mathsf{C}_{\mathcal{E}} = \textbf{obs\_part}\_\mathsf{C}_{\mathcal{E}}(\delta_{\mathcal{E}})$
214. $\mathsf{ci}_{\mathcal{E}}:\mathsf{CI}_{\mathcal{E}} = \textbf{uid}\_\mathsf{CI}(\mathsf{c}_{\mathcal{E}})$

Example 87 . Domain Requirements. Extension System Behaviour:

- We shall model the behaviour of the road-pricing system as follows:
  - we shall only model behaviours related to atomic parts;
  - we shall not model behaviours of hubs and links;
  - thus we shall model only
    - the set of behaviours of vehicles, veh,
    - the set of behaviours of toll-gates, gate, and
    - the behaviour of the road-pricing calculator, calc.

215 The road-pricing system behaviour, sys, is expressed as

a. the parallel, $\|$, (distributed) composition of the behaviours of all vehicles, with the parallel composition of
b. the parallel (likewise distributed) composition of the behaviours of all entry gates, with the parallel composition of
c. the parallel (likewise distributed) composition of the behaviours of all exit gates, with the parallel composition of
d. the behaviour of the road-pricing calculator,

value

```
215.   sys: Unit → Unit
215.   sys() ≡
215a..       ∥ {veh(uid_V(v),(ci,gis),𝕌TiGPos)|v:V•v ∈ vs_ℰ}
215b..    ∥ ∥ {gate("Entry")(uid_EG(eg),obs_mereo_G(eg),(𝕌enter,𝕌passing,𝕌leave))|eg:EG
215c..    ∥ ∥ {gate("Exit")(uid_EG(xg),obs_mereo_G(xg),(𝕌enter,𝕌passing,𝕌leave))|xg:XG•
215d..       ∥ calc(ci_ℰ,(vis_ℰ,gis_ℰ))(rlf)(trm)
```

Example 88 . Domain Requirements. Extension Vehicle Behaviour:

216 Instead of moving around by explicitly expressed internal non-determinism[29] vehicles move around by unstated internal non-determinism and instead receive their current position from the global positioning subsystem.

a. At each moment the vehicle receives its time-stamped local position, tilpos:TiLPos,

b. which it then proceeds to communicate, with its vehicle identification, (vi,tilpos), to the road pricing subsystem —

c. whereupon it resumes its vehicle behaviour.

[29] We refer to Items 157b., 157c. on Slide 343 and 158b., 158(c.)ii, 159 on Slide 345

value

```
216.    veh: vi:VI×(ci:CI×gis:GI-set)×𝕌TiGPos →
216.          out v_c_ch[ci,vi] Unit
216.    veh(vi,(ci,gis),attr_TiGPos_ch[vi]) ≡
216a..        let (τ,gpos) = attr_TiGPos_ch[vi]? in
216a..        let lpos = loc_pos(gpos) in
216b..        v_c_ch[ci,vi] ! (vi,(τ,lpos)) ;
216c..        veh(vi,(ci,gis),attr_TiGPos_ch[vi]) end end
216.        pre vi ∈ vis_ℰ ∧ ci = ci_ℰ ∧ gis = gis_ℰ
```

Example 89 . Domain Requirements. Extension Gate Behaviour:

- The entry and the exit gates have “vehicle enter”, “vehicle leave” and “vehicle time and identification” sensors.
  - The following assumption can now be made:
    - during the time interval between
    - a gate’s vehicle “enter” sensor having first sensed a vehicle entering that gate
    - and that gate’s “leave” sensor having last sensed that vehicle leaving that gate
    - that gate’s “vehicle time and identification” sensor registers the time when the vehicle is entering the gate and that vehicle’s unique identification.

- We sketch the toll-gate behaviour:

217 We parameterise the toll-gate behaviour as either an entry or an exit gate.

218 Toll-gates

a. inform the calculator of place (i.e., link) and time of entering and exiting of identified vehicles

b. over an appropriate array of channels.

219 Toll-gates operate autonomously and cyclically.

a. The **attr**_Enter event "triggers" the behaviour specified in formula line Item 219b.–219d..

b. The time-of-entry and the identity of the entering (or exiting) vehicle is sensed via external attribute channel inputs.

c. Then the road pricing calculator is informed of time-of-entry and of vehicle vi entering (or exiting) link li.

d. And finally, after that vehicle has left the entry or exit gate that toll-gate's behaviour is resumed.

- The toll-gate behaviour, gate:

type
217 EE = "Enter" | "Exit"
218a. GCM = EE $\times$ ($\mathbb{T}$ $\times$ VI $\times$ LI)
channel
218b. {g_c_ch[**uid**_GI(g),ci]|g:G,ci:CI•g $\in$ gates(trn)} GCM
value
219 gate: ee:EE$\times$gi:GI$\times$(ci:CI$\times$VI-**set**$\times$LI)$\times$($\mathbb{U}$enter$\times$$\mathbb{U}$passing$\times$$\mathbb{U}$leave) $\rightarrow$ **out** g_c_ch[gi
219 gate(ee,gi,(ci,vis,li),ea:(attr_enter_ch[gi],attr_passing_ch[gi],attr_leave_ch[gi])) $\equiv$
219a. attr_enter_ch[gi] ? ;
219b. **let** (τ,vi) = attr_passing_ch[gi] ? **in assert** vi $\in$ vis
219c. g_c_ch[gi,ci] ! (ee,(τ,(vi,li)));
219d. attr_leave_ch[gi] ?
219d. gate(ee)(gi,(ci,vis,li),ea)
219 **end**
219 **pre** ci = $ci_{\mathcal{E}}$ $\wedge$ vis = $vis_{\mathcal{E}}$ $\wedge$ li $\in$ $lis_{\mathcal{E}}$

Example 90 . Domain Requirements. Extension Calculator Behaviour:

220 The road-pricing calculator alternates between (offering to accept communication with)

a. either any vehicle

b. or any toll-gate.

```
220.   calc: ci:CI×(vis:VI-set×gis:GI-set)→RLF→TRM→
220a..        in {v_c_ch[ci,vi]|vi:VI•vi ∈ vis},
220b..           {g_c_ch[ci,gi]|gi:GI•gi ∈ gis}  Unit
220.   calc(ci,(vis,gis))(rlf)(trm) ≡
220a..      react_to_vehicles(ci,(vis,gis))(rlf)(trm)
220.   ⌈⌉⌊⌋
220b..      react_to_gates(ci,(vis,gis))(rlf)(trm)
220.       pre ci = ci_ℰ ∧ vis = vis_ℰ ∧ gis = gis_ℰ
```

221 If the communication is from a vehicle inside the toll-road net

a. then its toll-road net position, vp, is found from the road location function, rlf,

b. and the calculator resumes its work with the traffic map, trm, suitable updated,

c. otherwise the calculator resumes its work with no changes.

220a.. react_to_vehicles(ci,(vis,gis))(rlf)(trm) $\equiv$
220a.. **let** (vi,(τ,lpos)) =
220a.. ⌈⌉⌈{v_c_ch[ci,vi]?|vi:VI•vi$\in$ vis} **in**
221. **if** vi $\in$ **dom** trm
221a.. **then let** vp = rlf(lpos) **in**
221b.. calc(ci,(vis,gis))(rlf)(trm†[vi↦trm^⟨(τ,vp)⟩]) **end**
221c.. **else** calc(ci,(vis,gis))(rlf)(trm) **end end**

222 If the communication is from a gate,

a. then that gate is either an entry gate or an exit gate;
b. if it is an entry gate
c. then the calculator resumes its work with the vehicle (that passed the entry gate) now recorded, afresh, in the traffic map, **trm**.
d. Else it is an exit gate and
e. the calculator concludes that the vehicle has ended its to-be-paid for journey inside the toll-road net, and hence to be billed;
f. then the calculator resumes its work with the vehicle (that passed the exit gate) now removed from the traffic map, **trm**.

```
220b..  react_to_gates(ci,(vis,gis))(rlf)(trm) ≡
220b..     let (ee,(τ,(vi,li))) =
220b..        ⌊⌋⌈⌉{g_c_ch[ ci,gi ]|gi:GI·gi∈ gis} in
222a..     case ee of
222b..        "Enter" →
222c..            calc(ci,(vis,gis))(rlf)(trm∪[ vi↦⟨(τ,(li,0))⟩ ]),
222d..        "Exit"  →
222e..            billing(vi,trm(vi)⌢⟨(τ,(li,1))⟩);
222f..           calc(ci,(vis,gis))(rlf)(trm\{vi}) end end
```

• • •

- We have made relevant external attributes explicit parameters of their (corresponding part) processes.
- We refer to Sect. 1.3.7.

7.2.4.2. A Domain Extension Operator

- Domain extension takes a (more-or-less) deterministic requirements description, $\mathcal{R}_{\mathcal{D}}$, and yields an **extended requirements prescription**, $\mathcal{R}_{\mathcal{E}}$, which extends the domain description, $\mathcal{D}$, and, "at the same time", "extends" the requirements prescription, $\mathcal{R}_{\mathcal{D}}$,
  - **type extension**: $\mathcal{R}_{\mathcal{D}} \rightarrow \mathcal{R}_{\mathcal{E}}$
- Semantically
  - $\mathcal{R}_{\mathcal{D}}$ denotes a possibly infinite set of meanings, say $\mathbb{R}_{\mathbb{D}}$, and
  - $\mathcal{R}_{\mathcal{E}}$ denotes a possibly infinite set of meanings, say $\mathbb{R}_{\mathbb{E}}$,
  - but now the relation $\mathbb{R}_{\mathcal{E}} \sqsubseteq \mathbb{R}_{\mathcal{D}}$ is not necessarily satisfied —
  - but instead some **conservative extension** relation $\mathbb{R}_{\mathbb{E}} \sqsupseteq \mathbb{D}_{\mathbb{D}}$ is satisfied.

7.2.5. Requirements Fitting

- Often a domain being described
- "fits" onto, is "adjacent" to, "interacts" in some areas with,
- another domain:
  - *transportation* with *logistics*,
  - *health-care* with *insurance*,
  - *banking* with *securities trading* and/or *insurance*,
  - and so on.

- The issue of requirements fitting arises
  - when two or more software development projects
  - are based on what appears to be the same domain.
- The problem then is
  - to harmonise the two or more software development projects
  - by harmonising, if not too late, their requirements developments.

7.2.5.1. Some Definitions

- We thus assume
  - that there are n domain requirements developments, d_{r_1}, d_{r_2}, ..., d_{r_n}, being considered, and
  - that these pertain to the same domain — and can hence be assumed covered by a same domain description.

Definition 31 . Requirements Fitting:

- *By* **requirements fitting** *we mean*
  - *a* **harmonisation** *of* $n > 1$ *domain requirements*
  - *that have overlapping (shared) not always consistent parts and*
  - *which results in*
    - n ***partial domain requirements***', $p_{d_{r_1}}$, $p_{d_{r_2}}$, $\ldots$, $p_{d_{r_n}}$, *and*
    - m ***shared domain requirements***, $s_{d_{r_1}}$, $s_{d_{r_2}}$, $\ldots$, $s_{d_{r_m}}$,
    - *that "fit into" two or more of the partial domain requirements* ■
- *The above definition pertains to the result of 'fitting'.*
- *The next definition pertains to the act, or process, of 'fitting'.*

Definition 32 . Requirements Harmonisation:

- *By* **requirements harmonisation** *we mean*
  - *a number of alternative and/or co-ordinated prescription actions,*
  - *one set for each of the domain requirements actions:*
    - *Projection,*
    - *Instantiation,*
    - *Determination and*
    - *Extension.*

- *They are – we assume n separate software product requirements:*
  - Projection:
    - *If the n product requirements do not have the same projections,*
    - *then identify a common projection which they all share,*
    - *and refer to it is the* **common projection**.
    - *Then develop, for each of the n product requirements,*
    - *if required,*
    - *a* **specific projection** *of the common one.*
    - *Let there be m such specific projections, $m \leq n$.*

- *Instantiation:*
  - *First instantiate the common projection, if any instantiation is needed.*
  - *Then for each of the m specific projections*
  - *instantiate these, if required.*
- *Determination:*
  - *Likewise, if required, "perform" "determination" of the possibly instantiated common projection,*
  - *and, similarly, if required,*
  - *"perform" "determination" of the up to m possibly instantiated projections.*

- *Extension:*
  - *Finally "perform extension" likewise:*
  - *First, if required, of the common projection (etc.),*
  - *then, if required, on the up m specific projections (etc.).*
- *These harmonization developments may possibly interact and may need to be iterated* ■

- By a **partial domain requirements** we mean a domain requirements which is short of (that is, is missing) some prescription parts: text and formula ■
- By a **shared domain requirements** we mean a domain requirements ■

- By **requirements fitting** m *s*hared *d*omain *r*equirements texts, $sdrs$, into n partial domain requirements we mean that
  - there is for each *p*artial *d*omain *r*equirements, pdr_i,
  - an identified *s*ubset of $sdrs$ (could be all of $sdrs$), $ssdrs_i$,
  - such that textually conjoining $ssdrs_i$ to pdr_i,
  - i.e., $ssdrs_i \oplus pdr_i$
  - can be claimed to yield the "original" d_{r_i},
  - that is, $\mathcal{M}(ssdrs_i \oplus pdr_i) \subseteq \mathcal{M}(d_{r_i})$,
  - where $\mathcal{M}$ is a suitable meaning function over prescriptions ∎

7.2.5.2. Requirements Fitting Procedure — A Sketch

- Requirements fitting consists primarily of a pragmatically determined sequence of analytic and synthetic ('fitting') steps.
  - It is first decided which n domain requirements documents to fit.
  - Then a 'manual' analysis is made of the selected, n domain requirements.
  - During this analysis tentative shared domain requirements are identified.
  - It is then decided which m shared domain requirements to single out.
  - This decision results in a tentative construction of n partial domain requirements.
  - An analysis is made of the tentative partial and shared domain requirements.
  - A decision is then made
    - whether to accept the resulting documents
    - or to iterate the steps above.

7.2.5.3. Requirements Fitting – An Example

Example 91 . Domain Requirements. Fitting A Sketch:

- We postulate two domain requirements:
  - We have outlined a domain requirements development for software support for a road-pricing system.
  - We have earlier hinted at domain operations related to insertion of new and removal of existing links and hubs.
- We can therefore postulate that there are two domain requirements developments, both based on the transport domain:
- one, $d_{r_{\mathsf{toll}}}$, for a road-pricing system, and,
- another, $d_{r_{\mathsf{maint.}}}$, for a toll-road link and hub building and maintenance system monitoring and controlling link and hub quality and for development.

- The fitting procedure now identifies the shared awareness by both dr_{toll} and $dr_{\text{maint.}}$ of nets (N), hubs (H) and links (L).
  - We conclude from this that we can single out a common requirements for software that manages net, hubs and links.
  - Such software requirements basically amounts to requirements for a database system.
  - A suitable such system, say a relational database management system, DB_{rel}, may already be available with the customer.

- In any case, where there before were two requirements $(d_{r_{\text{toll}}}, d_{r_{\text{maint.}}})$ there are now four:
  - $d'_{r_{\text{toll}}}$, a modification of $d_{r_{\text{toll}}}$ which omits the description sections pertaining to the net;
  - $d'_{r_{\text{maint.}}}$, a modification of $d_{r_{\text{maint.}}}$ which likewise omits the description sections pertaining to the net;
  - $d_{r_{\text{net}}}$, which contains what was basically omitted in $d'_{r_{\text{toll}}}$ and $d'_{r_{\text{maint.}}}$; and
  - $d_{r_{\text{db:i/f}}}$ (db:i/f for database interface) which prescribes a mapping between type names of $d_{r_{\text{net}}}$ and relation and attribute names of DB_{rel} ■

- Much more can and should be said, but this suffices as an example in a software engineering methodology paper.

7.2.6. Domain Requirements Consolidation

- After projection, instantiation, determination, extension and fitting,
  - it is time to review, consolidate and possibly restructure (including re-specify)
  - the domain requirements prescription
  - before the next stage of requirements development.

Dines Bjørner's MAP-i Lecture # 9

End of MAP-i Lecture # 9: Domain Requirements: Extension and Fitting

Thursday, 28 May 2015: 10:00–11:15