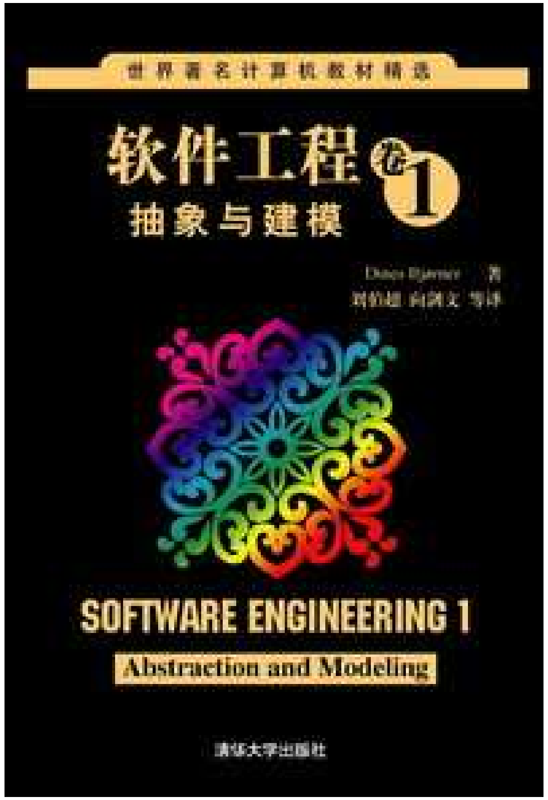

DRAWING TO A CLOSE

Begin of Lecture 8: First Session — Requirements Engineering

Domain and Interface Requirements

FM 2012 Tutorial, Dines Bjørner, Paris, 28 August 2012

Tutorial Schedule

12. Requirements Engineering

- We shall present a terse overview of
  - how one can "derive" essential fragments of **requirements prescriptions**
  - from a **domain description**.
- First we give,
  - in the next section,
  - a summary of the net domain, N,
  - as developed in earlier sections.

12.1. The Transport Domain — a Resumé

12.1.1. Nets, Hubs and Links

130. From a transport net one can observe sets of hubs and links.

type
130. N, HS, Hs = H-**set**, H, LS, Ls = L-**set**, L
131. HI, LI
15. LΣ = HI-**set**, HΣ = (LI×LI)-**set**
16. LΩ = LΣ-**set**, HΩ = HΣ-**set**

value
130. obs_HS: N → HS, obs_LS: N → LS
130. obs_Hs: N → H-**set**, obs_Ls: N → L-**set**
15. attr_LΣ: L → LΣ, attr_HΣ: H → HΣ
16. attr_LΩ: L → LΩ, attr_HΩ: H → HΩ

12.1.2. Mereology

131. From hubs and links one can observe their unique hub, respectively link identifiers and their respective mereologies.
132. The mereology of a link identifies exactly two distinct hubs.
133. The mereologies of hubs and links must identify actual links and hubs of the net.

value

131. uid_H: H → HI, uid_L: L → LI

131. mereo_H: H → LI-**set**, mereo_L: L → HI-**set**

axiom

132. ∀ l:L•**card** mereo_L(l)=2

133. ∀ n:N,l:L•l ∈ obs_Ls(n) ⇒

133. ∧ ∀ hi:HI•hi ∈ mereo_L(l)

133. ⇒ ∃ h:h•h ∈ obs_Hs(n)∧uid_H(h)=hi

133. ∧ ∀ h:H•h ∈ obs_Hs(n) ⇒

133. ∀ li:LI•li ∈ mereo_H(h)

133. ⇒ ∃ l:L•l ∈ obs_Ls(n)∧uid_L(l)=li

12.2. A Requirements “Derivation”

12.2.1. Definition of Requirements

IEEE Definition of ‘Requirements’

- By a requirements we understand (cf. IEEE Standard 610.12 [ieee-610.12]):
  - *“A condition or capability needed by a user to solve a problem or achieve an objective”.*

12.2.2. The Machine = Hardware + Software

- By ‘the **machine**’ we shall understand the
  - **software** to be developed and
  - **hardware** (equipment + base software) to be configured

  for the domain application.

12.2.3. Requirements Prescription

- The core part of the requirements engineering of a computing application is the **requirements prescription**.
  - A requirements prescription tells us which parts of the domain are to be supported by 'the machine'.
  - A requirements is to satisfy some **goal**s.
  - Usually the **goal**s cannot be prescribed in such a manner that they can serve directly as a basis for software design.
  - Instead we derive the requirements from the domain descriptions and then argue
    (incl. prove) that the **goal**s satisfy the requirements.
  - In this colloquium we shall not show the latter
    but shall show the former.

12.2.4. Some Requirements Principles

The "Golden Rule" of Requirements Engineering

- Prescribe only such requirements
  - that can be objectively shown to hold
  - for the designed software.

An "Ideal Rule" of Requirements Engineering

- When prescribing (including formalising) requirements,
  - also formulate tests (theorems, properties for model checking)
  - whose actualisation should show adherence to the requirements.

- We shall not show adherence to the above rules.

12.2.5. A Decomposition of Requirements Prescription

- We consider three forms of requirements prescription:
  - the **domain requirements**,
  - the **interface requirements** and
  - the **machine requirements**.
- Recall that the machine is the hardware and software (to be required).
  - **Domain requirements** are those whose technical terms are from the domain only.
  - **Machine requirements** are those whose technical terms are from the machine only.
  - **Interface requirements** are those whose technical terms are from both.

12.2.6. An Aside on Our Example

- We shall continue our "ongoing" example.
- Our requirements is for a tollway system.
- By a **requirements goal** we mean
  - *an objective*
  - *the system under consideration*
  - *should achieve* [LamsweerdeIEEE2001].
- The **goal**s of having a tollway system are:
  - to decrease transport times
    between selected hubs of a general net; and
  - to decrease traffic accidents and fatalities
    while moving on the tollway net
    as compared to comparable movements on the general net.

- The tollway net, however, must be paid for by its users.
  - Therefore tollway net entries and exits occur at tollway plazas
  - with these plazas containing entry and exit toll collectors
  - where tickets can be issued,
    respectively collected and
    travel paid for.
- We shall very briefly touch upon these toll collectors,
  in the Extension part (as from Slide 405) below.
- So all the other parts of the next section
  serve to build up to the Extension section.

12.3. Domain Requirements

- Domain requirements cover all those aspects of the domain —
  - parts and materials,
  - actions,
  - events and
  - behaviours —
- which are to be supported by ‘the machine’.

- Thus domain requirements are developed by systematically "revising" cum "editing" the domain description:
  - which parts are to be **projected:** left in or out;
  - which general descriptions are to be **instantiated** into more specific ones;
  - which non-deterministic properties are to be made more **determinate**; and
  - which parts are to be **extended** with such computable domain description parts which are not feasible without IT.

- Thus
  - projection,
  - instantiation,
  - determination and
  - extension

  are the basic engineering tasks of domain requirements engineering.

- An example may best illustrate what is at stake.
- The example is that of a tollway system —
  - in contrast to the general nets covered by description Items 130–133
  - (Slides 380–381).
  - See Fig. 4 on the next page.

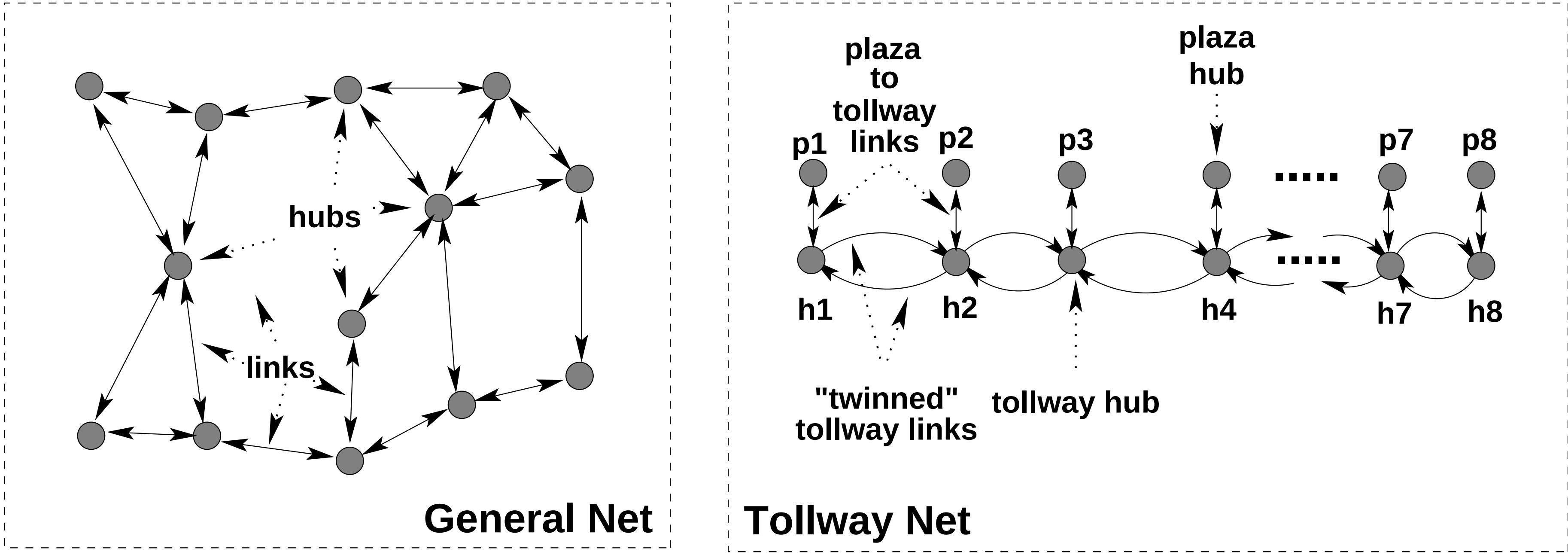

Figure 4: General and Tollway Nets

12.3.1. Projection

We keep what is needed to prescribe the tollway system and leave out the rest.

134. We keep the description, narrative and formalisation,
    (a) nets, hubs, links,
    (b) hub and link identifiers,
    (c) hub and link states,
135. as well as related observer functions.

type
134(a). N, H, L
134(b). HI, LI
134(c). HΣ, LΣ
value
135. obs_Hs,obs_Ls,obs_HI,obs_LI,
135. obs_HIs,obs_LIs,obs_HΣ,obs_L Σ

- We omit bringing the composite part concepts
- of HS, LS, Hs and Ls
- into the requirements.

12.3.2. Instantiation

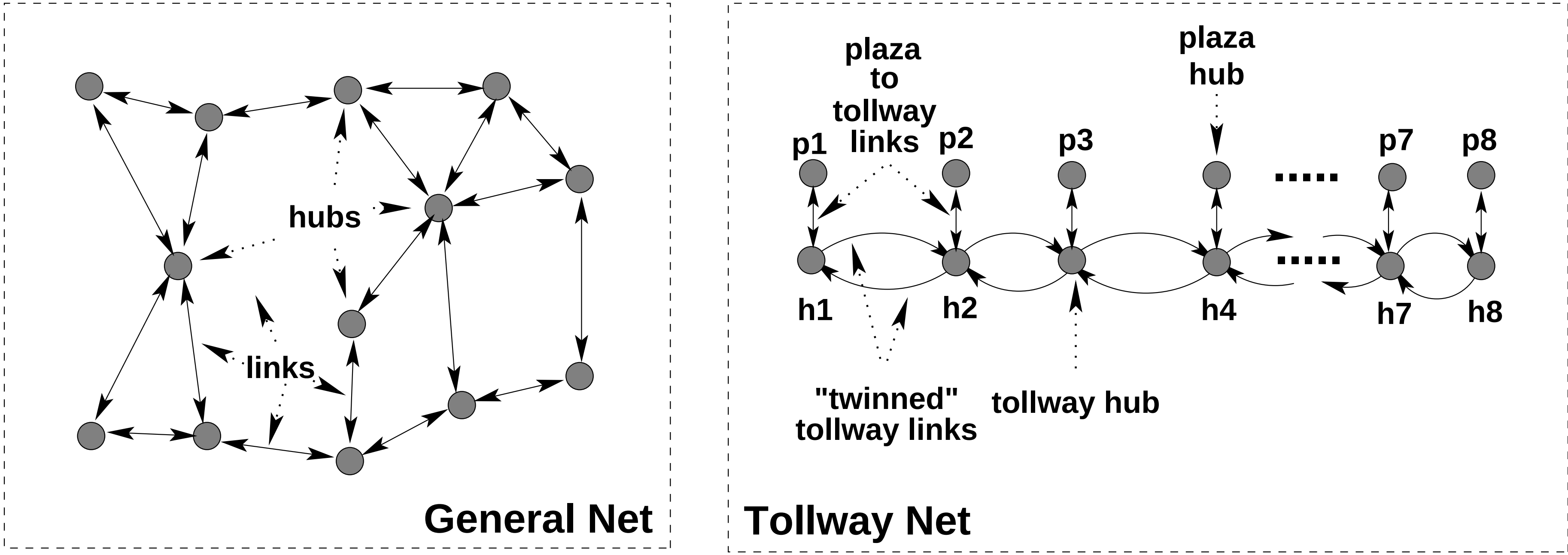

Figure 5: General and Tollway Nets

- From the general net model of earlier formalisations we instantiate, that is, make more concrete, the tollway net model now described.

136. The net is now concretely modelled as a pair of sequences.
137. One sequence models the plaza hubs, their plaza-to-tollway link and the connected tollway hub.
138. The other sequence models the pairs of "twinned" tollway links.
139. From plaza hubs one can observe their hubs and the identifiers of these hubs.
140. The former sequence is of m such plaza "complexes" where $m \geq 2$; the latter sequence is of $m-1$ "twinned" links.
141. From a tollway net one can abstract a proper net.
142. One can show that the posited abstraction function yields well-formed nets, i.e., nets which satisfy previously stated axioms.

```
type
136. TWN = PC* × TL*
137. PC = PH × L × H
138. TL = L × L
value
137. obs_H: PH → H, obs_HI: PH → HI
axiom
140. ∀ (pcl,tll):TWN •
140.   2≤len pcl∧len pcl=len tll+1
value
141. abs_N: TWN → N
141. abs_N(pcl,tll) as n
141.   pre:  wf_TWN(pcl,tll)
141.   post:
141.     obs_Hs(n) =
141.       {h,h′|(h,_,h′):PC
141.       •(h,_,h′)∈ elems pcl}
141.   ∧ obs_Ls(n) =
141.       {l|(_,l,_):PC
141.         •(_,l,_)∈ elems pcl} ∪
141.       {l,l′|(l,l′):TL•(l,l′)∈ elems tll}

theorem:
142. ∀ twn:TWN • wf_TWN(twn)
142.       ⇒ wf_N(abs_N(twn))
```

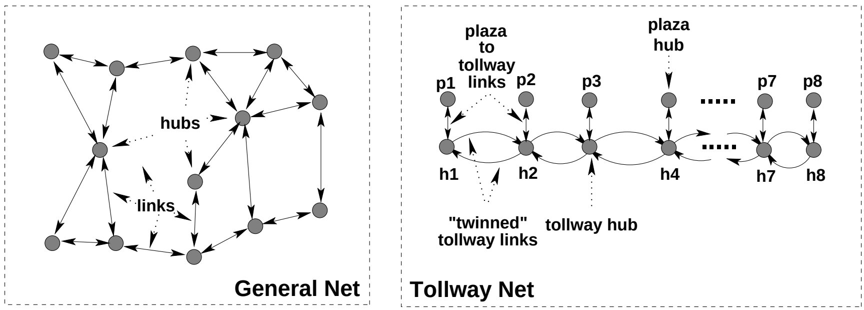

Figure 6: General and tollway Nets

12.3.2.1 Model Well-formedness wrt. Instantiation

- Instantiation restricts general nets to tollway nets.
- Well-formedness deals with proper mereology:
  that observed identifier references are proper.
- The well-formedness of instantiation of the tollway system model
  can be defined as follows:

143. The i'plaza complex, (p_i, l_i, h_i), is instantiation-well-formed if
    (a) link l_i identifies hubs p_i and h_i, and
    (b) hub p_i and hub h_i both identifies link l_i; and if
144. the i'th pair of twinned links, tl_i, tl'_i,
    (a) has these links identify the tollway hubs of the i'th and $i{+}1$'st plaza complexes ((p_i, l_i, h_i) respectively $(p_{i+1}, l_{i+1}, h_{i_1})$).

```
value
 Instantiation_wf_TWN: TWN → Bool
 Instantiation_wf_TWN(pcl,tll) ≡
143.    ∀ i:Nat · i ∈ inds pcl⇒
143.      let (pi,li,hi)=pcl(i) in
143(a).     obs_LIs(li)={obs_HI(pi),obs_HI(hi)}
143(b).    ∧ obs_LI(li)∈ obs_LIs(pi)∩ obs_LIs(hi)
144.     ∧ let (li′,li″) = tll(i) in
144.        i < len pcl ⇒
144.         let (pi′,li‴,hi′) = pcl(i+1) in
144(a).        obs_HIs(li) = obs_HIs(li′)
144(a).        = {obs_HI(hi),obs_HI(hi′)}
   end end end
```

12.3.3. Determination

- Determination, in this example, fixes states of hubs and links.
- The state sets contain only one set.
  - Twinned tollway links allow traffic only in opposite directions.
  - Plaza to tollway hubs allow traffic in both directions.
  - tollway hubs allow traffic to flow freely from
    - plaza to tollway links
    - and from incoming tollway links
    - to outgoing tollway links
    - and tollway to plaza links.
- The determination-well-formedness of the tollway system model can be defined as follows[29]:

[29] i ranges over the length of the sequences of twinned tollway links, that is, one less than the length of the sequences of plaza complexes. This "discrepancy" is reflected in out having to basically repeat formalisation of both Items 146(a) and 146(b).

12.3.3.1 Model Well-formedness wrt. Determination

- We need define well-formedness wrt. determination.
- Please study Fig. 7.

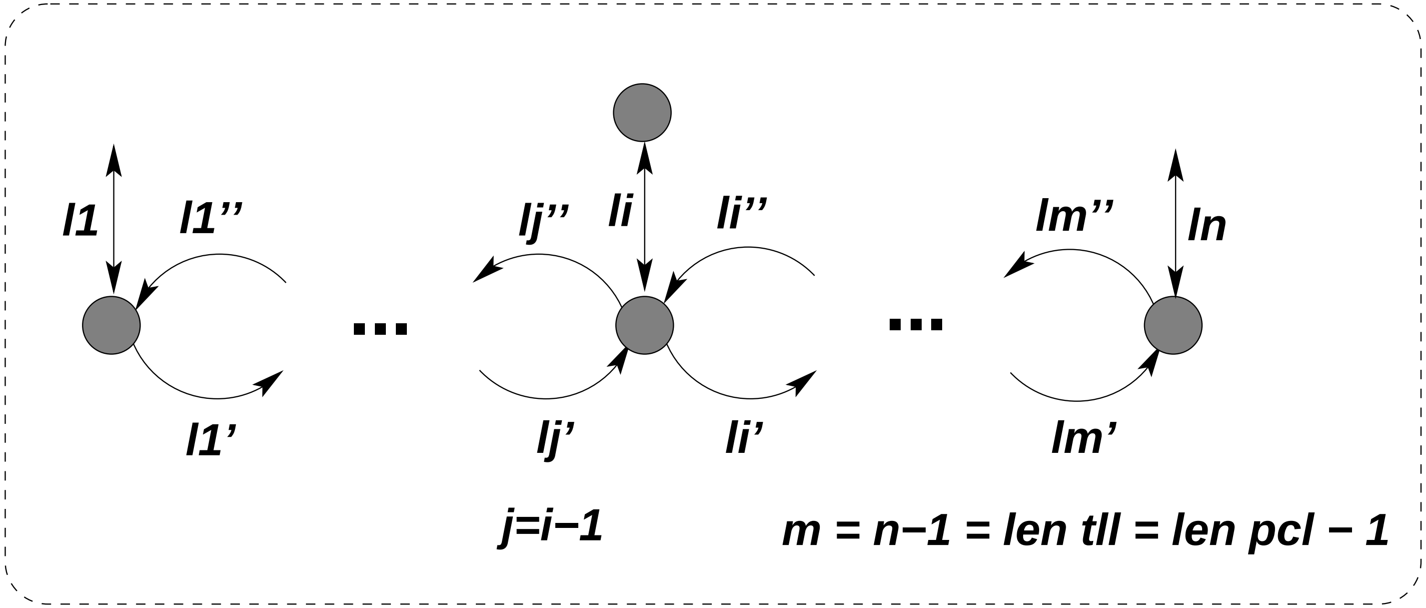

Figure 7: Hubs and Links

145. All hub and link state spaces contain just one hub, respectively link state.
146. The i'th plaza complex, pcl(i):(p_i, l_i, h_i) is determination-well-formed if
    (a) l_i is open for traffic in both directions and
    (b) p_i allows traffic from h_i to "revert"; and if
147. the i'th pair of twinned links (li', li'') (in the context of the i+1st plaza complex, pcl(i+1):$(p_{i+1}, l_{i+1}, h_{i+1})$) are determination-well-formed if
    (a) link l'_i is open only from h_i to h_{i+1} and
    (b) link l''_i is open only from h_{i+1} to h_i; and if
148. the jth tollway hub, h_j (for $1 \leq\ j \leq \mathbf{len}\,$pcl) is determination-well-formed if, depending on whether j is the first, or the last, or any "in-between" plaza complex positions,
    (a) [the first:] hub $i = 1$ allows traffic in from l_1 and l''_1, and onto l_1 and l'_1.
    (b) [the last:] hub $j = i + 1 = \mathbf{len}\,$pcl allows traffic in from $l_{\mathbf{len}\,\text{tll}}$ and $l''_{\mathbf{len}\,\text{tll}-1}$, and onto $l_{\mathbf{len}\,\text{tll}}$ and $l'_{\mathbf{len}\,\text{tll}-1}$.
    (c) [in-between:] hub $j = i$ allows traffic in from l_i, l''_i and l'_i and onto l_i, l'_{i-1} and l''_i.


```
value
146.     Determination_wf_TWN: TWN → Bool
146.     Determination_wf_TWN(pcl,tll) ≡
146.       ∀ i:Nat• i ∈ inds tll ⇒
146.         let (pi,li,hi) = pcl(i),
146.             (npi,nli,nhi) = pcl(i+1), in
146.             (li′,li″) = tll(i) in
145.           obs_HΩ(pi)={obs_HΣ(pi)}∧obs_HΩ(hi)={obs_HΣ(hi)}
145.        ∧ obs_LΩ(li)={obs_LΣ(li)}∧obs_LΩ(li′)={obs_LΣ(li′)}
145.        ∧ obs_LΩ(li″)={obs_LΣ(li″)}
146(a).     ∧ obs_LΣ(li)
146(a).          = {(obs_HI(pi),obs_HI(hi)),(obs_HI(hi),obs_HI(pi))}
146(a).     ∧ obs_LΣ(nli)
146(a).          = {(obs_HI(npi),obs_HI(nhi)),(obs_HI(nhi),obs_HI(npi))}
146(b).     ∧ {(obs_LI(li),obs_LI(li))}⊆obs_HΣ(pi)
146(b).     ∧ {(obs_LI(nli),obs_LI(nli))}⊆obs_HΣ(npi)
147(a).     ∧ obs_LΣ(li′)={(obs_HI(hi),obs_HI(nhi))}
147(b).     ∧ obs_LΣ(li″)={(obs_HI(nhi),obs_HI(hi))}
148.        ∧ case i+1 of
148(a).          2 → obs_HΣ(h_1)=
148(a).                  {(obs_LΣ(l_1),obs_LΣ(l_1)), (obs_LΣ(l_1),obs_LΣ(l_1″)),
148(a).                  (obs_LΣ(l″_1),obs_LΣ(l_1)), (obs_LΣ(l″_1),obs_LΣ(l′_1))},
148(b).          len pcl → obs_HΣ(h_i+1)=
148(b).                 {(obs_LΣ(l_len pcl),obs_LΣ(l_len pcl)),
148(b).                  (obs_LΣ(l_len pcl),obs_LΣ(l′_len tll)),
148(b).                  (obs_LΣ(l″_len tll),obs_LΣ(l_len pcl)),
148(b).                  (obs_LΣ(l″_len tll),obs_LΣ(l′_len tll))},
148(c).          _ → obs_HΣ(h_i)=
148(c).                 {(obs_LΣ(l_i),obs_LΣ(l_i)), (obs_LΣ(l_i),obs_LΣ(l′_i)),
148(c).                  (obs_LΣ(l_i),obs_LΣ(l″_i−1)), (obs_LΣ(l″_i),obs_LΣ(l′_i)),
148(c).                  (obs_LΣ(l″_i),obs_LΣ(l′_i−1)), (obs_LΣ(l″_i),obs_LΣ(l′_i))}
146.         end end
```

12.3.4. Extension

- By **domain extension** we understand the
  - *introduction of domain entities, actions, events and behaviours that were not feasible in the original domain,*
  - *but for which, with computing and communication,*
  - *there is the possibility of feasible implementations,*
  - *and such that what is introduced become part of the emerging domain requirements prescription.*

12.3.4.1 Narrative

- The **domain extension** is that of the controlled access of vehicles to and departure from the tollway net:
  - the entry to (and departure from) tollgates from (respectively to) an `"an external"` net — which we do not describe;
  - the new entities of tollgates with all their machinery;
  - the user/machine functions:
    - upon entry:
      - driver pressing entry button,
      - tollgate delivering ticket;
    - upon exit:
      - driver presenting ticket,
      - tollgate requesting payment,
      - driver providing payment, etc.

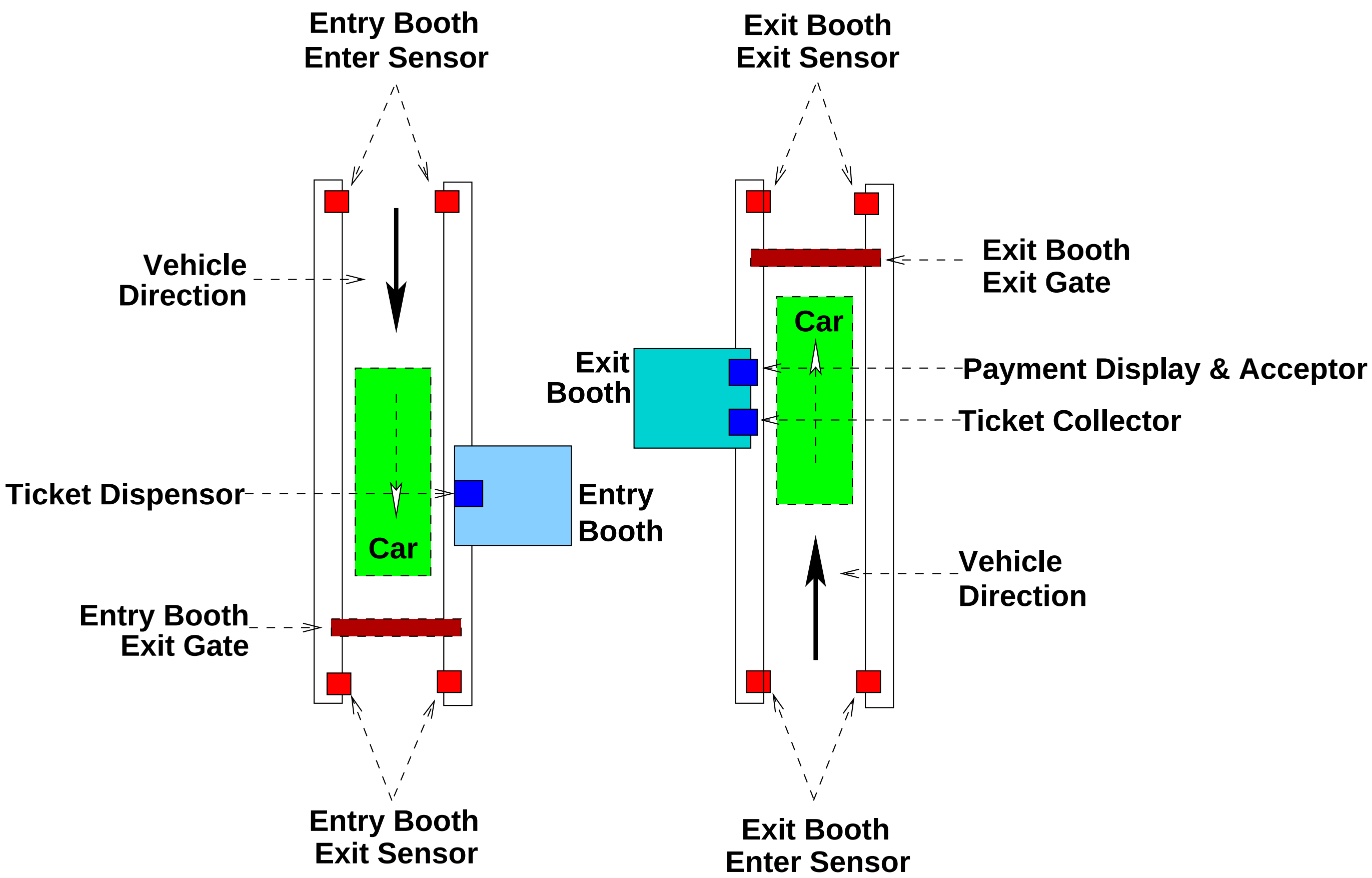

Figure 8: Entry and Exit Tollbooths

- One added (extended) domain requirements:
  - as vehicles are allowed to cruise the entire net
  - payment is a function of the totality of links traversed, possibly multiple times.
- This requires, in our case,
  - that tickets be made such as to be sensed somewhat remotely,
  - and that hubs be equipped with sensors which can record
  - and transmit information about vehicle hub crossings.
    - (When exiting, the tollgate machine can then access the exiting vehicles' sequence of hub crossings — based on which a payment fee calculation can be done.)
    - All this to be described in detail — including all the things that can go wrong (in the domain) and how drivers and tollgates are expected to react.

- We omit details of narration and formalisation.
  - In this case the extension description would entail a number of formalisations:
    - An initial one which relies significantly on the use of **RSL/CSP** [CARH:Electronic,TheSEBook1wo].
      It basically models tollbooth and vehicle behaviours.
    - A "derived" one which models temporal properties.
      It is expressed, for example, in the **Duration Calculus, DC** [zcc+mrh2002].
    - And finally a timed-automata [AluDil:94,olderogdirks2008] model which "implements" the **DC** model.

12.4. Interface Requirements Prescription

- A systematic reading of the domain requirements shall
  - result in an identification of all shared
    - parts and materials,
    - actions,
    - events and
    - behaviours.
- An entity is said to be a **shared entity** if it is mentioned in both
  - the **domain description** and
  - the **requirements prescription**.
- That is, if the entity
  - is present in the domain and
  - is to be present in the machine.

- Each such **shared phenomenon** shall then be individually dealt with:
  - **part** and **materials sharing** shall lead to interface requirements for **data initialisation and refreshment;**
  - **action sharing** shall lead to interface requirements for **interactive dialogues between the machine and its environment;**
  - **event sharing** shall lead to interface requirements for **how events are communicated between the environment of the machine and the machine.**
  - **behaviour sharing** shall lead to interface requirements for **action and event dialogues between the machine and its environment.**

12.4.1. Shared Parts

- The main **shared parts** of the main example of this section are
  - the net, hence the hubs and the links.
- As domain parts they repeatedly undergo changes with respect to the values of a great number of attributes and otherwise possess attributes — most of which have not been mentioned so far:
  - length, cadestral information, namings,
  - wear and tear (where-ever applicable),
  - last/next scheduled maintenance (where-ever applicable),
  - state and state space, and
  - many others.

- We "split" our interface requirements development into two separate steps:
  - the development of $d_{r.net}$
    - (the common domain requirements for the shared hubs and links),
  - and the co-development of $d_{r.db:i/f}$
    - (the common domain requirements for the interface between $d_{r.net}$ and DB_{rel} —
- under the assumption of an available relational database system DB_{rel}

- When planning the common domain requirements for the net, i.e., the hubs and links,
  - we enlarge our scope of requirements concerns beyond the two so far treated ($d_{r.toll}$, $d_{r.maint.}$)
  - in order to make sure that the shared relational database of nets, their hubs and links, may be useful beyond those requirements.

- We then come up with something like
  - hubs and links are to be represented as tuples of relations;
  - each net will be represented by a pair of relations
    - a hubs relation and a links relation;
    - each hub and each link may or will be represented by several tuples;
  - etcetera.
- In this database modelling effort it must be secured that "standard" actions on nets, hubs and links can be supported by the chosen relational database system DB_{rel}.

12.4.1.1 Data Initialisation

- As part of $d_{r.net}$ one must prescribe **data initialisation**, that is provision for
  - an interactive user interface dialogue with a set of proper display screens,
    - one for establishing net, hub or link attributes (names) and their types and,
    - for example, two for the input of hub and link attribute values.
  - Interaction prompts may be prescribed:
    - next input,
    - on-line vetting and
    - display of evolving net, etc.
  - These and many other aspects may therefore need prescriptions.
- Essentially these prescriptions concretise the insert link action.

12.4.1.2 Data Refreshment

- As part of $d_{r.net}$ one must also prescribe **data refreshment**:
  - an interactive user interface dialogue with a set of proper display screens
    - one for updating net, hub or link attributes (names) and their types and,
    - for example, two for the update of hub and link attribute values.
  - Interaction prompts may be prescribed:
    - next update,
    - on-line vetting and
    - display of revised net, etc.
  - These and many other aspects may therefore need prescriptions.
- These prescriptions concretise remove and insert link actions.

12.4.2. Shared Actions

- The main **shared actions** are related to
  - the entry of a vehicle into the tollway system and
  - the exit of a vehicle from the tollway system.

12.4.2.1 Interactive Action Execution

- As part of $d_{r.toll}$ we must therefore prescribe
  - the varieties of successful and less successful sequences
  - of interactions between vehicles (or their drivers) and the toll gate machines.
- The prescription of the above necessitates determination of a number of external events, see below.
- (Again, this is an area of embedded, real-time safety-critical system prescription.)

12.4.3. Shared Events

- The main **shared external events** are related to
  - the entry of a vehicle into the tollway system,
  - the crossing of a vehicle through a tollway hub and
  - the exit of a vehicle from the tollway system.
- As part of $d_{r.toll}$ we must therefore prescribe
  - the varieties of these events,
  - the failure of all appropriate sensors and
  - the failure of related controllers:
    - gate opener and closer (with sensors and actuators),
    - ticket "emitter" and "reader" (with sensors and actuators),
    - etcetera.
- The prescription of the above necessitates extensive fault analysis.

12.4.4. Shared Behaviours

- The main **shared behaviours** are therefore related to
  - the journey of a vehicle through the tollway system and
  - the functioning of a toll gate machine during "its lifetime".
- Others can be thought of, but are omitted here.
- In consequence of considering, for example, the journey of a vehicle behaviour, we may "add" some further, extended requirements:
  - requirements for a vehicle statistics "package";
  - requirements for tracing supposedly "lost" vehicles;
  - requirements limiting tollway system access in case of traffic congestion; etcetera.

12.5. Machine Requirements

- The machine requirements
  - make hardly any concrete reference to the domain description;
  - so we omit its treatment altogether.

12.6. Discussion of Requirements "Derivation"

- We have indicated
  - how the **domain engineer**
  - and the **requirements engineer**
  - can work together
  - to "derive" significant fragments
  - of a **requirements prescription**.

- This puts **requirements engineering** in a new light.
  - Without a previously existing **domain descriptions**
  - the **requirements engineer** has to do double work:
    - both **domain engineering**
    - and **requirements engineering**
  - but without the **principles** of **domain description**,
    - as laid down in this tutorial
  - that job would not be so straightforward as we now suggest.

End of Lecture 8: First Session — Requirements Engineering

Domain and Interface Requirements

FM 2012 Tutorial, Dines Bjørner, Paris, 28 August 2012


Software
Engineering 1


Software
Engineering 2


Software
Engineering 3

SHORT BREAK