(3. 3.8.)

Start of Lecture 4: REQUIREMENTS ENGINEERING

4. Requirements Engineering

- Whereas
  - a domain description presents a domain ***as it is***,
  - a requirements prescription presents a domain ***as it would be*** if some required machine was implemented (from these requirements).
- The **machine** is the **hardware** plus **software** to be designed from the requirements.
- That is, the *machine* is what the requirements are about.

- We distinguish between three kinds of requirements:
  - the ***domain requirements*** are those requirements which can be expressed solely using terms of the domain;
  - the ***machine requirements*** are those requirements which can be expressed solely using terms of the machine and
  - the ***interface requirements*** are those requirements which must use terms from both the domain and the machine in order to be expressed.

- We make a distinction between goals and requirements.
- Goals are what we expect satisfied by the software implemented from the requirements.
- But goals could also be of the system for which the software is required.
- First we exemply the latter, then the former.

Example 14 – Goals of a Toll Road System

- A goal for a toll road system may be
  - to decrease the travel time between certain hubs and
  - to lower the number of traffic accidents between certain hubs,

■ End of Example 14

Example 15 – Goals of Toll Road System Software

- The goal of the toll road system software is to help automate
  - the recording of vehicles entering, passing and leaving the toll road system
  - and collecting the fees for doing so.

■ End of Example 15

- Goals are usually expressed in terms of properties.
- Requirements can then be proved to satisfy the $\mathcal{G}$oals: $\mathcal{D},\mathcal{R} \models \mathcal{G}$.

Example 16 – Arguing Goal-satisfaction of a Toll Road System

- By endowing links and hubs with average traversal times for both ordinary road and for toll road links and hubs
  - one can calculate traversal times between hubs
  - and thus argue that the toll road system satisfies "quicker" traversal times.
- By endowing links and hubs with traffic accident statistics (real, respectively estimated)
  - for both ordinary road and for toll road links and hubs
  - one can calculate estimated traffic accident statistics between all hubs
  - and thus argue that the combined ordinary road plus toll road system satisfies lower traffic fatalities.

■ End of Example 16

Example 17 – Arguing Goal-satisfaction of Toll Road System Software

- By recording
  - tickets issued and collected at toll boths and
  - toll road hubs and links entered and left
  - as per the requirements specification brought in (forthcoming) Examples 19-23,
- we can eventually argue that
  - the requirements of (the forthcoming) Examples 19-23
  - help satisfy the goal of Example 15 on page 102.

■ End of Example 17

(4. Requirements Engineering)

- We shall assume that the (goal and) requirements engineer elicit both $\mathcal{G}$oals and $\mathcal{R}$equirements from requirements stakeholders.
- But we shall focus only on
  - domain and
  - interface

  requirements such as "derived" from domain descriptions.

(4. Requirements Engineering)

4.1. Business Process Re-engineering

- There are the business processes of the domain before installation of the required computing systems.
- The potential of installing computing systems invariably requires revision of established business processes.
- Business process re-engineering (BPR) is a development of new business processes
  - – whether or not complemented by computing and communication.
- BPR, such as we advocate it,
  - proceeds on the basis of an existing domain description and
  - outlines needed changes (additions, deletions, modifications) to entities, actions, events and behaviours
  - following the six domain facets.
- The goals help us formulate the BPR prescriptions.

(4. Requirements Engineering 4.1. Business Process Re-engineering)

Example 18 – Rough-sketching a Re-engineered Road Net

- Our sketch centers around a toll road net with toll booth plazas.
- The BPR focuses
  - first on entities, actions, events and behaviours,
  - then on the six domain facets.

(4. Requirements Engineering 4.1. Business Process Re-engineering)

64 **Re-engineered Entities:**

- We shall focus on a linear sequence of toll road intersections (i.e., hubs) connected by pairs of one-way (opposite direction) toll roads (i.e., links).
- Each toll road intersection is connected by a two way road to a toll plaza.
- Each toll plaza contains a pair of sets of entry and exit toll booths.
- (Example 20 brings more details.)

65 Re-engineered Actions:

- Cars enter and leave the toll road net through one of the toll plazas.
- Upon entering, car drivers receive, from the entry booth, a plastic/paper/electronic ticket which they place in a special holder in the front window.
- Cars arriving at intermediate toll road intersections choose, on their own, to turn either "up" the toll road or "down" the toll road — with that choice being registered by the electronic ticket.
- Cars arriving at a toll road intersection may choose to "circle" around that intersection one or more times — with that choice being registered by the electronic ticket.
- Upon leaving, car drivers "return" their electronic ticket to the exit booth and pay the amount "asked" for.

66 Re-engineered Events:

- A car entering the toll road net at a toll both plaza entry booth constitutes an event.
- A car leaving the toll road net at a toll both plaza entry booth constitutes an event.
- A car entering a toll road hub constitutes an event.
- A car entering a toll road link constitutes an event.

67 Re-engineered Behaviours:

- The journey of a car,
  - from entering the toll road net at a toll booth plaza,
  - via repeated visits to toll road intersections
  - interleaved with repeated visits to toll road links
  - to leaving the toll road net at a toll booth plaza,

  constitutes a behaviour — with
  - receipt of tickets,
  - return of tickets and
  - payment of fees

  being part of these behaviours.
- Notice that a toll road visitor is allowed to cruise "up" and "down" the linear toll road net – while (probably) paying for that pleasure (through the recordings of "repeated" hub and link entries).

68 Re-engineered Intrinsics:

- Toll plazas and abstracted booths are added to domain intrinsics.

69 Re-engineered Support Technologies:

- There is a definite need for domain-describing the failure-prone toll plaza entry and exit booths.

70 Re-engineered Rules and Regulations:

- Rules for entering and leaving toll booth entry and exit booths must be described as must related regulations.
- Rules and regulations for driving around the toll road net must be likewise be described.

71 **Re-engineered Scripts:**

- No need.

72 **Re-engineered Management and Organisation:**

- There is a definite need for domain describing
- the management and possibly distributed organisation
- of toll booth plazas.

73 **Re-engineered Human Behaviour:**

- Humans, in this case car drivers, may not change their behaviour in the spectrum from diligent and accurate via sloppy and delinquent to outright traffic-law breaking – so we see no need for any "re-engineering".

■ End of Example 18

4.2. Domain Requirements

- For the phase of domain requirements the requirements stakeholders "sit together" with the domain cum requirements engineers and read the domain description, line-by-line, in order to "derive" the domain requirements.
- They do so in five rounds (in which the BPR rough sketch is both regularly referred to and possibly, i.e., most likely regularly updated).
- Domain requirements are "derived" from the domain description.
- The goals then determine the derivations: which projections, instantiations, determinations, etcetera, to perform.

4.2.1. Projection

By *domain projection* we understand an operation

- that applies to a domain description
- and yields a domain requirements prescription.
- The latter represents a projection of the former
- in which only those parts of the domain are present
- that shall be of interest in the ongoing requirements development

Example 19 – Projection

- Our requirements is for a simple toll road:
  - a linear sequence of links and hubs outlined in Example 18:
    - ∗ see Items [1–11] of Example 1 on page 39
    - ∗ and Items [32–35] of Example 7 on page 68.

■ End of Example 19

4.2.2. Instantiation

- By *domain instantiation* we understand an operation
  - that applies to a (projected) domain description, i.e., a requirements prescription,
  - and yields a domain requirements prescription,
  - where the latter has been made more specific, usually by constraining a domain description.

Example 20 – Instantiation

- Here the toll road net topology as outlined in Example 18 on page 107 is introduced:
  - a straight sequence of toll road hubs
  - pairwise connected with pairs of one way links
  - and with each hub two way link connected to a toll road plaza.

type
$\mathsf{H, L, P = H}$
$\mathsf{N' = (H \times L) \times H \times ((L \times L) \times H \times (H \times L))^*}$
$\mathsf{N'' = \{|n:N' \cdot wf(n)|\}}$
value
$\mathsf{wf\_N'': N' \to \mathbf{Bool}}$
$\mathsf{wf\_N''((h,l),h',llhpl) \equiv \ldots\ 6\ \ lines\ \ldots\ !}$
$\mathsf{\alpha N: N'' \to N}$
$\mathsf{\alpha N((h,l),h',llhpl) \equiv \ldots\ 2\ lines\ \ldots\ !}$

- $\mathsf{wf\_N''}$ secures linearity;
- αN allows abstraction from more concrete N″ to more abstract N.

■ End of Example 20

4.2.3. Determination

- By *domain determination* we understand an operation
  - that applies to a (projected and possibly instantiated) domain description, i.e., a requirements prescription,
  - and yields a domain requirements prescription,
  - where (attributes of) entities, actions, events and behaviours have been made less indeterminate.

Example 21 – Determination

- Pairs of links between toll way hubs are open in opposite directions;
- all hubs are open in all directions;
- links between toll way hubs and toll plazas are open in both directions.


```
type
  LΣ = (HI×HI)-set, LΩ = LΣ-set
  HΣ = (LI×LI)-set, HΩ = HΣ-set
  N′ = (H × L) × H × ((L × L) × H × (H × L))*
value
  ωLΣ: L → LΣ, ωLΩ: L → LΩ
  ωHΣ: H → HΣ, ωHΩ: H → HΩ
axiom
  ∀ ((h,l),h′,llhhl:⟨((l′,l″),h″,(h‴,l‴))⟩⌢llhhl′):N″ •
    ωLΣ(l)={(ωHI(h),ωHI(h′)),(ωHI(h′),ωHI(h))}∧
    ωLΣ(l‴)={(ωHI(h″),ωHI(h‴)),(ωHI(h‴),ωHI(h″))}∧
    ∀ i,i+1:Nat • {i,i+1}⊆inds llhhl ⇒
      let ((li,li′),hi,(hi″,li″))=llhhl(i), (_,hj,(hj″,lj″))=llhhl(i+1) in
      ωLΩ(li)= {{(ωHI(hi),ωHI(hj))}}∧ωLΩ(li′)={{(ωHI(hj),ωHI(hi))}}∧
      ωHΩ(hi)= { ... } ... 3 lines end
```

■ End of Example 21

4.2.4. Extension

- By *domain extension* we understand an operation
  - that applies to a (projected and possibly determined and instantiated) domain description, i.e., a (domain) requirements prescription,
  - and yields a (domain) requirements prescription.
  - The latter prescribes that a software system is to support, partially or fully, entities, operations, events and/or behaviours that were not feasible (or not computable in reasonable time or space) in a domain without computing support, but which are now are not only feasible but also computable in reasonable time and space.

Example 22 – Extension

- We extend the domain by introducing toll road entry and exit booths as well as electronic ticket hub sensors and actuators.
- There should now follow a careful narrative and formalisation of these three machines:
  - the car driver/machine "dialogues" upon entry and exit
  - as well as the sensor/car/actuator machine "dialogues" when cars enter hubs.
- The description
  - should first, we suggest, be ideal;
  - then it should take into account
    - ∗ failures of booth equipment,
    - ∗ electronic tickets,
    - ∗ car drivers,
    - ∗ and of sensors and actuators.

■ End of Example 22

4.2.5. Fitting

- By *domain requirements fitting* we understand an operation
  - which takes two or more (say n) domain requirements prescriptions, d_{r_i},
  - that are claimed to share entities, actions, events and/or behaviours and
  - map these into $n+1$ domain requirements prescriptions, δ_{r_i},
  - where one of these, $\delta_{r_{n+1}}$ capture the shared phenomena and concepts and the other n prescriptions, δ_{r_i},
  - are like the n "input" domain requirements prescriptions, d_{r_i},
  - except that they now, instead of the "more-or-less" shared prescriptions,
  - that are now consolidated in $\delta_{r_{n+1}}$, prescribe interfaces between δ_{r_i} and $\delta_{r_{n+1}}$ for $i : \{1..n\}$.

Example 23 – Fitting

- We assume three ongoing requirements development projects, all focused around road transport net software systems:
  - (i) road maintenance,
  - (ii) toll road car monitoring and
  - (iii) bus services on ordinary plus toll road nets.
- The main shared phenomenon is the road net, i.e., the links and the hubs.
- The consolidated, shared road net domain requirements prescription, $\delta_{r_{n+1}}$, is to become a prescription for the domain requirements for shared hubs and links.
- Tuples of these relations then prescribe representation of all hub, respectively all link attributes – common to the three applications.
- Functions (including actions) on hubs and links become database queries and updates. Etc.

■ End of Example 23

4.2.6. Discussion:

- This section has very briefly surveyed and illustrated domain requirements.
- The reader should take cognizance of the fact that these are indeed "derived" from the domain description.
- They are not domain descriptions, but, once the business process re-engineering has been adopted
- and the required software has been installed,
- then the domain requirements become part of a revised domain description !

4.3. Interface Requirements

- By interface requirements we understand such requirements which are concerned with the phenomena and concepts *shared* between the domain and the machine.
- Thus such requirements can only be expressed using terms from both the domain and the machine.
- We tackle the problem of "deriving", i.e., constructing interface requirements by tackling four "smaller" problems:
  - those of "deriving" interface requirements for
    - ∗ entities,
    - ∗ actions,
    - ∗ events and
    - ∗ behaviours

    respectively.
  - Again goals help state which phenomena and concepts are to be shared.

4.3.1. Entity Interfaces

- Entities that are shared between the domain and the machine must initially be input to the machine.
- Dynamically arising or attribute value changing entities must likewise be input and all such machine entities must have their attributes updated, when need arise.
- Requirements for shared entities thus entail
  - requirements for their representation
  - and for their human/machine and/or machine/machine transfer-dialogues.

Example 24 – Shared Entities

- Main shared entities are those of hubs and links.
- We suggest that eventually a relational database be used for representing hubs links in relations.
- 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.

■ End of Example 24

4.3.2. Action Interfaces

- 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
  - (some measurable, time-varying entity attribute value, or some domain stakeholder)
  - in order to proceed in its computation.

Example 25 – Shared Actions

- In order for a car driver to leave an exit toll both the following component actions must take place:
  - the driver inserts the electronic pass in the exit toll booth machine;
  - the machine scans and accepts the ticket and calculates the fee for the car journey from entry booth via the toll road net to the exit booth;
  - the driver is alerted to the cost and is requested to pay this amount;
  - once paid the exit booth toll gate is raised.
- *Notice that a number of details of the new support technology is left out.*
- *It could either be elaborated upon here, or be part of the system design.*

■ End of Example 25

4.3.3. Event Interfaces

- 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.

Example 26 – Shared Events

- The arrival of a car at a toll plaza entry booth is an event that must be communicated to the machine so that the entry booth may issue a proper pass (ticket).
- Similarly for the arrival at a toll plaza exit booth so that the machine may request the return of the pass and compute the fee.
- The end of that computation is an event that is communicated to the driver (in the domain) requesting that person to pay a certain fee after which the exit gate is opened.

■ End of Example 26

4.3.4. Behaviour Interfaces

- By a shared behaviour we understand
  - a sequence of zero, one or more
    - shared actions and
    - shared events.

Example 27 – Shared Behaviour

- 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 Example 27

4.3.5. Discussion

- Once the machine has been installed
- it, the machine, is part of the new domain !

4.4. Machine Requirements

- We shall not cover this stage of requirements development other than saying that it consists of the following concerns:
  - performance requirements (storage, speed, other resources),
  - dependability requirements (availability, accessibility, integrity, reliability, safety, security),
  - maintainability requirements (adaptive, extensional, corrective, perfective, preventive),
  - portability requirements (development platform, execution platform, maintenance platform, demo platform) and
  - documentation requirements.

- Only dependability seems to be subjectable to rigorous, formal treatment.
- The **discussions** of earlier carry over to this paragraph.
- That is, once the machine has been installed it, the machine, is part of the new domain !

End of Lecture 4: REQUIREMENTS ENGINEERING