Stijn Hoppenbrouwers

Requirements Engineering, Lecture 8:

Conversational Perspective on RE

Radboud University Nijmegen

Hooking Up with Lecture 4

- “Information Systems”
  - Computational
  - Socio-technical
- “System Development Systems”
  - Computational
  - Socio-technical

- What is a development system about?
- What does it essentially *do*?
- What doe its input and output consist of?
- What sort of interaction do users have with it?
- How (based on what principles) can you *steer* it?

More on Development Systems

- Input: all sorts of information “about the system”
  - Wishes, demands, standards
  - Functional, non-functional, technical
  - Many, many different aspects, stakes, and priorities
  - Architecture(s), requirements, various designs
  - But also the actual code, at various levels
- Output: “the system” (possibly, evolutionary)
  - Software? Implemented software? Deployed software?
  - Deployed & *managed* (human components!)
  - Is it the actual organization?
  - How about training people, or documentation?
- Can the development system and the operational system really be separated?

Development Systems & Communication

- How important is communication –in all its facets?
- How important is language (syntax-semantics-pragmatics)?

- “2nd order information system”?
  - Information system that brings forth information system(s)
  - Certainly, a limited, quite generic point of view
  - But fundamentally, perhaps, the most important one?
  - Any alternatives, please?

Syntax, Semantics, Pragmatics (1/3)

- Syntax = form/structure in language
- Most typically: “grammar”; composition rules of sentences and words (also called “morphosyntax”)
- However, alphabet and how words are spelled/pronounced essentially also belongs to syntax (sub-fields: phonology, phonetics,typology,morphology,lexicology)
- Also, syntax can go beyond sentences: the structure of conversations/texts.
- XML: way of defining and sharing syntax

Syntax, **Semantics**, Pragmatics (2/3)

- Semantics = “meaning”
- But only a part of meaning: *meaning without the interpretation-in-context part*
- Again, this can essentially be meaning of words, sentences, and conversations/texts
- There are different flavors of semantics:
  - Socio-cognitive semantics (“inside human brains”; meaning shared between humans)
  - Mathematical/formal semantics (can be boiled down to pure mathematical concepts)
  - Technical semantics (can be boiled down to machine states)
  - Mathematical and technical semantics can often be related (Turing machine etc.)
- Importantly, to express/talk about semantics, you need a language: syntax-semantics-pragmatics

Can semantics be captured/expressed 100%?

Syntax, Semantics, & **Pragmatics** (3/3)

- Pragmatics = language use in context
- Entails *personal* interpretation (very subjective)
- Not about form, but about what language *does*
- Also about link between language and “reality”
- “How to Do Things with Language”: Speech Acts
- Propositional content + intentionality
- Factual statements, questions, commands, subjective opinions

  “John will now close the door”

  “Could you please close that door, John?”

  “It’s freezing here, isn’t it John?”

Knowledge, Language, & Communication in IT and Organizations

- What can be observed, exchanged, stored all takes the shape of language
  - Natural language: syntax and semantics “open”
  - Semi-formal language: well described syntax
  - Formal languages (incl. programming languages): well-described syntax and semantics
- Interaction / communication between all actors in the development system can be captured in terms of:
  - Knowledge goals / strategies (“contents”: IT development fields)
  - Communication goals / strategies (pragmatics)
  - Language goals / strategies (syntax/semantics)

Why pragmatics in system development?

- Syntax and semantics are merely about *structures*
- To deal with processes, we need pragmatics (rules, principles, conventions)
- To “ground” language utterances in its social contex (knowledge *sharing, agreement, commitment*) we need pragmatics

- Statement = “good customers get a 10% discount”
- Is this a “true” statement, socially? Shared among everyone? Agreed to by everyone? Committed to by everyone?
- What is a “good customer”? Again: shared, agreed, committed? “Conversation about meaning”; DM!

Example of a grounded definition dialog I (1/4)

Participants: PM (product manager) BC (business consultant) A (analyst)

Goals: - initial definition of new business rule;
- share and agree;
- formalization level 0 (pre-formal)

```
PM: Let's make the amount of credit allowed variable,
    depending on customer status.
A:  And?
PM: Good customers get high credit limit, and bad
    customers get a lower credit limit, perhaps zero.
BC: That's a nice idea.
```

Example of a grounded definition dialog I (2/4)

```
A: OK, so:
    [Statement1,Share{PM,BC,A},Agree{PM,BC,A}]
    "Amount_of_credit_allowed is variable"]
    [Statement2,Share{PM,BC,A},Agree{PM,BC,A}
    "Amount_of_credit_allowed depends_on Customer_status"]
    [Statement3,Share{PM,BC,A},Agree{PM,BC,A}
    "Good_customer gets high_credit_rate"]
    [Statement4,Share{PM,BC,A},Agree{PM,BC,A}
    "Bad_customer gets low_credit_rate"]
 BC: Well, 2 implies 1, I suppose. Would 2 alone do?
 PM: yes, I don't see why not.
 A:  OK, 1 is thrown away:
     [Statement1,Share{PM,BC,A},Agree{},REJECTED
     "Amount of credit_allowed is variable‟,
     Argument{BC,"is implicit in statement2"}]
```

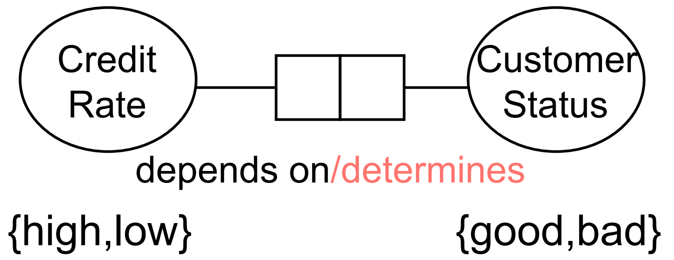
Example of a grounded definition dialog I (3/4)

```
PM: And where has the "or zero" gone?
A:  I thought that 0 is just a very low credit rate.
    Agreed?
PM: Yeah, I suppose so. OK.
BC: Yes, that makes sense.
A:  well then:
    [Statement4,Share{PM,BC,A},Agree{PM,BC,A}
    "Bad_customer gets low_credit_rate",
    Argument{A,"low includes 0"}]

BC: OK, but what defines a "good customer"?
```

... (and so on)

Example of a grounded definition dialog I (4/4)



red elements are still lacking in the example dialogue: could bring it to “form. Level 1” (initial formalization)

```
CustomerStatus(c,good) ⇒ CreditRate(c,high) ,
CustomerStatus(c,bad) ⇒ CreditRate(c,low) ,
CreditRate(c,low) ⇒ CreditAmount(c,x) ∧ x ≥ 0
```

- Rather incomplete and insufficient *formalization* so far
- The main point is made first, the rest is added progressively and insofar the participants are willing to/capable; otherwise, it is *explicitly* delegated (relates to *goals* of this and further conversations)
- The demands of the formalism and the domain are gradually reconciled and stepped up if required

System Development as an Interconnected, Goal-Driven Series of Conversations

- Goals are set (Stakeholder, SD goals, Knowledge goals, Communication Goals, Language Goals)
- RE goals: typically look like SD goals and Knowledge goals
- However, they have the other types as sub-goals
- Minimally, it is good to have *awareness* of these
- *All sorts of development goals will develop/emerge/change during the development process!!*

Development System Development Goals?

- RE of RE!
- Project Modeling as a system
- Communication/information modeling
- …

- (short discussion)

Knowledge Goals in System Development?

- Share
- Agree
- Commit

- Explicitness of knowledge (see article “Understanding the Requirements on Modeling techniques”):
  - Formality
  - Quantifiability
  - Executability
  - Comprehensibility
  - Completeness

Communications goals and strategies in SD

- Who needs to communicate what to who, and why, and how?
- Execution plan
- Description languages
- Media
- Cognitive mode (analytic/experimental; knowledge handling)
- Social mode (expert-driven or participatory)
- Communication mode
  - Protocols (turn taking)
  - Participants
  - Patterns

Rational Conversations

- Cost / benefit balance
- Be rational about “means and ends”
- Optimally effective, efficient
- Measurement, reasoning, guidance
- Computational bookkeeping and AI required?
- Less *art*, more *science* of system development

- (Also see article “System Development as a Rational Communicative Process”)

Active goal-driven guidance of RE/SD conversations?

- How clever are you in steering/structuring RE conversations
- It’s about ALL conversations in a project, and how they relate
- LOTS of bookkeeping: not just contents, but also “conversation management”
- Strategies? Planning?
- If goals change, strategies change! *Evolutionary Development System* (complex, adaptive system)

A tool: conversation-based system development environment

- Part Computer Supported Cooperative Work System
- Part Knowledge Management System
- Part Decision Support System
- Part CASE-tool
- Part Dialogue system

Except:

- Think big, act small
- We’ve started at the (formal) basis: conceptual modeling (ORM) but now start putting process modeling central
- RE environment: not document-based, but conversation based; not product-oriented, but process-oriented; *goal driven*

Recent developments

- Method Engineering
  - Rule-based
  - Operational method = Information system
- Human-Computer Interaction approach
  - Not just languages
  - ALL aspects needed to make 2nd order ISs operational
  - HCI through modelling
- Method engineering as game design
  - Metaphor
  - But also link to concrete systems

Ddembe Williams: Applying System Dynamics to RE Projects

- Show paper

Approach to Method Modelling: Interaction System for Modelling

functionality	*realization*
Templates and views	View interface
Reference procedures	Workflow
Structured ToDo list	Generator
Speech acts	Deep interface
Rules and rule checking	Rule engine
State(s) of model and context	DBMS

Advantages of the approach

- Clear, goal-oriented, rule-based framework for methods
- Many possibilities for collaborative setup (multi-player)
- Advanced data gathering possible (interactions explicit)
- Usability / playability / HCI central (out-of-the-box approach)
- Operational process view on methods (SD link)
- Justifiably controlled working environment
- Effective guiding: score linked to quality system
- Emotive factor becomes concrete
- Clear link with virtual worlds / games (CASE tools)
- Possibilities for links to game theory (strategies)
- Many possibilities in education

Game Design Theory

- Järvinen 2006: “Games without Frontiers: Theories and Methods for Game Studies and Design”

- Games are systems (and may have sub-systems)
- Games are dynamic systems (structure, function, history)
- Games are/include *information systems* (which is why their computerization is so successful!)

- Rules, and Objects the rules Act on
- Communicative aspects of rules: communicative acts, “Game Rhetoric”

Game Design Theory: basic elements

- **Goals**

What players strive for

- **Components**

Concrete items that players care for (e.g. “pieces”)

- **End and Victory Conditions**

When the game is lost or won, or ends; introduce competition and control the game’s duration

- **Game mechanics**

The sorts of actions players can perform

Game Design Theory (2)

- **Environments**

Spatial constraints like a board or virtual space (not mandatory)

- **Themes**

Metaphors that add meaning to a game (not mandatory)

- **Interfaces**

Especially for video games, but picking up a piece on the board is also an interface

- And, of course, **rules**: gluing it all together
- Also, as part of the rules and the victory conditions: a **score system** (many alternatives)
- Games may involve a **jury** or **referee** or **game master**, so rules need not cover 100% of constraints, goals, and evaluation (scoring).

Method Engineering

- See thesis Roelofs

- Still lacking:
  - Score system
  - Limited games
  - Clearer goals
  - Implementation!

Latest developments and results

- Publications exploring and clarifying the principles
- Project at Everest B.V. to investigate “Gaming Aspects of the AQUIMA Tool Suite” (Wilmont)
- Development of actual Games:
  - Process Modeling Game: Schotten, Aarts
  - Supply Chain Construction Game (value modeling), with UvT
  - Game for testing Information Query Language & procedure (Claessens)