Start of Lecture 11: CLOSING

5. Conclusion

- We discuss a number of issues.

5.1. What Have We Omitted

- Our coverage of domain and requirements engineering has focused on modelling techniques for domain and requirements facets.
- We have omitted the important software engineering tasks of
  - **stakeholder identification and liaison**,
  - **domain** and, to some extents also **requirements**, especially **goal acquisition and analysis**,
  - **terminologisation**, and
  - techniques for **domain and requirements and goal validation and [goal] verification** ($\mathcal{D},\mathcal{R} \models \mathcal{G}$).

5.2. Domain Descriptions Are Not Normative

- A description of, for example,
  - "the" domain of the *New York Stock Exchange* would describe
    - ∗ the set of rules and regulations governing the submission of sell offers and buy bids
    - ∗ as well as rules and regulations for clearing ('matching') sell offers and buy bids.
  - These rules and regulations appears to be quite different from those of the *Tokyo Stock Exchange*.
  - A normative description of stock exchanges would abstract these rules so as to be rather un-informative.
  - And, anyway, rules and regulations changes and business process re-engineering changes entities, actions, events and behaviours.
  - For any given software development one may thus have to rewrite parts of existing domain descriptions, or construct an entirely new such description.

5.3. "Requirements Always Change"

- This claim is often used as a hidden excuse for not doing a proper, professional job of requirements prescription, let alone "deriving" them, as we advocate, from domain descriptions.
- Instead we now make the following counterclaims
  - [1] "domains are far more stable than requirements" and
  - [2] "requirements changes arise more as a result of business process re-engineering than as a result of changing stakeholder ideas".

- Closer studies of a number of domain descriptions,
  - for example of a *financial service industry*,
  - reveals that the domain in terms of which an "ever expanding" variety of financial products are offered,
  - are, in effect, based on a small set of very basic domain functions which have been offered for well-nigh centuries !
- We thus claim that
  - thoroughly developed domain descriptions and
  - thoroughly "derived" requirements prescriptions
  - tend to stabilise the requirements re-design,
  - but never alleviate it.

5.4. What Can Be Described and Prescribed

- The issue of *"what can be described"* has been a constant challenge to philosophers.
  - Bertrand Russell covers, in a 1919 publication, *Theory of Descriptions*, and
  - in [Philosophy of Mathematics] a revision, as *The Philosophy of Logical Atomism*.
- The issue is not that straightforward.
- In two recent papers we try to broach the topic from the point of view of the kind of domain engineering presented in these lectures.
- Our approach is simple; perhaps too simple !
- We can describe what can be observed.

- We do so,
  - first by postulating types of observable phenomena and of derived concepts;
  - then by the introduction of *observer* functions and by axioms over these, that is, over values of postulated types and observers.
  - To this we add defined functions; usually described by pre/post-conditions.
    - ∗ The narratives refer to the "real" phenomena
    - ∗ whereas the formalisations refer to related phenomenological concepts.
- The narrative/formalisation problem is that one can 'describe' phenomena without always knowing how to formalise them.

5.5. What Have We Achieved – and What Not

- Earlier we made some claims.
- We think we have substantiated them all, albeit ever so briefly.
- Each of the domain facets
  - (intrinsics,
  - support technologies,
  - rules and regulations,
  - scripts [licenses and contracts],
  - management and organisation and
  - human behaviour)
- and each of the requirements facets
  - (projection,
  - instantiation,
  - determination,
  - extension and
  - fitting)
- provide rich grounds for both specification methodology studies and and for more theoretical studies.

5.6. Relation to Other Work

- The most obvious 'other' work is that of Michael jackson's [Problem Frames].
  - In that book Jackson, like is done here,
    * departs radically from conventional requirements engineering.
    * In his approach understandings of the domain, the requirements and possible software designs
    * are arrived at, not hierarchically, but in parallel, interacting streams of decomposition.

- Thus the 'Problem Frame' development approach iterates between concerns of
  - domains,
  - requirements and
  - software design.
- "Ideally" our approach pursues
  - domain engineering
  - prior to requirements engineering,
  - and, the latter, prior to software design.
- But see next.

- The recent book [Axel van Lamsweerde]
  - appears to represent the most definitive work on Requirements Engineering today.
  - Much of its requirements and goal acquisition and analysis techniques
  - carries over to main aspects of domain acquisition and analysis techniques
  - and the goal-related techniques of [Lamsweerde] apply to determining which
    * projections,
    * instantiation,
    * determination and
    * extension operations

    to perform on domain descriptions.

5.7. "Ideal" Versus Real Developments

- The term 'ideal' has been used in connection with 'ideal development' from domain to requirements.
- We now discuss that usage.
- Ideally software development could proceed
  - from developing domain descriptions
  - via "deriving" requirements prescriptions
  - to software design,

  each phase involving extensive
  - formal specifications,
  - verifications (formal testing, model checking and theorem proving) and validation.

- More realistically
  - less comprehensive domain description development (D)
  - may alternate with both requirements development (R) work
  - and with software design (S) –
  - in some
    - controlled,
    - contained
    - iterated and
    - "spiralling"

    manner
  - and such that it is at all times clear which development step is what: $\mathcal{D}$, $\mathcal{R}$ or $\mathcal{S}$!

5.8. Description Languages

- We have used the RSL specification language, for the formalisations of this report,
- but any of the model-oriented approaches and languages offered by
  - Alloy,
  - B, Event B,
  - RAISE,
  - VDM and
  - Z,

  should work as well.

- No single one of the above-mentioned formal specification languages, however, suffices.
- Often one has to carefully combine the above with elements of
  - Petri Nets,
  - CSP,
  - MSC,
  - Statecharts,

  and/or some temporal logic, for example
  - either DC or
  - TLA+.
- Research into how such diverse textual and diagrammatic languages can be combined is ongoing.

5.9. Entailments

- $\mathcal{D}, \mathcal{R} \models \mathcal{G}$
  - From the $\mathcal{D}$omain and the $\mathcal{R}$equirements we can reason that the $\mathcal{G}$oals are met.
- $\mathcal{D}, \mathcal{S} \models \mathcal{R}$
  - In a proof of correctness of $\mathcal{S}$oftware design with respect to $\mathcal{R}$equirements prescriptions one often has to refer to assumptions about the $\mathcal{D}$omain.
  - Formalising our understandings of the $\mathcal{D}$omain, the $\mathcal{R}$equirements and the $\mathcal{S}$oftware design enables proofs that the software is right and the formalisation of the "derivation" of $\mathcal{R}$equirements from $\mathcal{D}$omain specifications help ensure that it is the right software [Boehm81].

5.10. Domain Versus Ontology Engineering

- In the information science community an ontology is a
  - "formal, explicit specification of a shared conceptualisation".
- Most of the information science ontology work seems aimed primarily at axiomatisations of properties of entities.
- Apart from that there are many issues of "ontological engineering" that are similar to the triptych kind of domain engineering;
  - but then, we claim, that domain engineering goes well beyond ontological engineering and makes free use of whatever formal specification languages are needed.

6. Bibliographical Notes

6.1. Description Languages

- Besides using
  - as precise a subset of a national language, as here English, as possible, and in enumerated expressions and statements,
  - we have "paired" such narrative elements with corresponding enumerated clauses of a formal specification language.
- We have been using the RAISE Specification Language, RSL in our formal texts.
- But any of the model-oriented approaches and languages offered by
  - Alloy,
  - CafeOBJ [futatsugi2000a],
  - Event B,
  - VDM and
  - Z,

  should work as well.

- No single one of the above-mentioned formal specification languages, however, suffices.
- Often one has to carefully combine the above with elements of
  - Petri Nets,
  - CSP: Communicating Sequential Processes,
  - MSC: Message Sequence Charts,
  - Statecharts,
  - and some temporal logic, for example
    - DC: Duration Calculus
    - or TLA+.
  - And even then !

End of Lecture 11: CLOSING