Domain Engineering

book by Dines Bjørner, presentation by Tero Hasu

February 9, 2012

Contents

8 February 2012
INF329 course
Tero Hasu
<tero.hasu at ii.uib.no>

1 Dines Bjørner

- famous in the formal methods community
- current focus areas: (1) domain engineering, (2) requirements engineering, and (3) software design methods
- behind RAISE (Rigorous Approach to Industrial Software Engineering)
  - RAISE Specification Language (RSL) and tools
- homepage

2 a *domain*

A (somewhat cyclic) definition:

An application (or business) domain: a universe of discourse, an area of human and societal activity, …

3 some domains

Decreasing from "grand scale" (infrastructure components of society)

- financial services industry
- health care
- transportation
- …
- roads
- …
- buses
- …
- an automobile
- …
- wristwatch firmware ♠

4 "To understand is all."

- Should study man-made universes (domains) in-and-by-themselves, just like physicists study the universe.
- In isolation, without concern for requirements. (Bjørner's novelty)
- Regardless of whether the understanding can be "translated" into engineering tools and techniques.

5 motivation

- general domain understanding is not application specific
  - only domain specific
- clear and elegant understanding leads to better tools and better engineering
  - cf. e.g. λ-calculus and Scheme ♠

6 problems

- apparently not a popular research topic
  - "author urges younger scientists to get going"
- with current understanding, "to establish a trustworthy and believable theory of a [single] domain, it may take 10-15 years"
- not a single formalism will do

7 a *domain description*

- An informal narrative describing a domain, and a mathematical text formalising the description.
- Serves as axioms (assumed truths) on top of which can build theorems.

8 a *domain theory*

A domain description together with lemmas, propositions and theorems that can be proved about the description – and hence, can be claimed to hold in the domain.

9 a *domain model*

Something satisfying a domain description. Either:

- an actual, real domain "out there"; or
- a mathematical structure

10 domain modelling in engineering

- aerospace, chemical, civil etc. engineers
  - expected to model phenomena of domain in which artifacts placed
- software engineers
  - might model own artifacts (compilers, etc.)
  - seldom expected to model domain in which software operates

11 software engineering process

- domain engineering
- requirements engineering
- software design

12 domain engineering process

1. identification of and regular interaction with stakeholders
2. domain (knowledge) acquisition
3. domain analysis
4. domain modelling
5. domain verification
6. domain validation
7. domain theory formation

Stages 2 and 3 relate to domain description. Focus here on Stage 4.

13 central to domain engineering

Finding and expressing suitable abstractions.

- By observing phenomena.

  - at least when there is no existing knowledge: no implementation, no documentation, no domain experts ♠
    * e.g. Copernicus and the work that followed in modelling the *solar* system

- From repeated observations and identified patterns can form concepts. Possibly further generalise to more abstract concepts.
  - cf. category theory ♠
    * very abstract, but can help identify patterns *between* concepts

14 phenomena vs. concepts

- Phenomena are manifest.
  - Observed by senses or by measuring instruments.
- Concepts are defined.

15 domain abstractions

- entity
- function
  - over entities
- event
  - involving changes in entities
  - may be caused by function invocations
- behavior
  - structure of actions and events

16 higher-level abstractions

- state
  - an entity collection representing state
- action
  - application of a state-changing function

17 an *entity*

- Something we can point to;
- something that manifests; or
- something abstracted from the above.
- Either atomic or composite.
- Has attributes to describe it.

18 a *function*

Something which when applied to argument values yields entities (constituting the result value).

$$f : A \times B \to C \times D \tag{1}$$

19 an *event*

- An instantaneous change of state not directly brought about by explicitly willed action, but either by "external" forces or implicitly as a non-intended result of an explicitly willed action.
- e.g.:
  - bank account withdrawal with insufficient funds (internal event)
  - disruption caused by a bank robbery (external event)
- cf. exceptions ♠

20 a *behavior*

- A structure of actions and events.
- A sequence in the simplest case.
- A set of sequences or (sub)behaviors in more complex cases.
- With interleaved or "true" concurrency of sequences.
- Communication between behaviors by having shared events.

21 *mereology*

- A theory of part-hood relations.
- How entities are connected and composed.
- cf. entity-relationship model ♠
- cf. information model (FODA) ♠

22 a domain *facet*

- One among a finite set of "generic" ways of analysing a domain.
- E.g.: intrinsics, support technology, management and organisation, rules and regulations (and scripts), and human behavior.

23 facet: domain *intrinsics*

- Phenomena and concepts which are "basic" to any other facets.
- There may be several intrinsics, for different stakeholder perspectives.

24 facet: domain *support technology*

- Ways and means of implementation.
- E.g., or a rail unit switch for a railway.
- Support technologies typically reflect real-time embeddedness.

- Use techniques and languages similar to those for modelling event and process intensity, with the focus on temporal notions.

25 facet: domain *management* and *organisation*

- definition
  - management: people who set and enforce rules and strategies
  - organisation: structuring of staff levels
- Spans entity, function, event, and behavior "intensities".
- Typically requires full spectrum of modelling techniques and notations.

26 facet: domain *rules* and *regulations*

- definition
  - rule: how expected to behave
  - regulation: prescription of remedial actions for rule breaking
- Usually expressed in terms of domain entities. Typically involving properties, axioms, state changes.
- May require various modelling techniques and notations, including constraint satisfaction notation and fuzzy logic.

27 sub-facet: domain *script*

- A rule or a regulation that has legally binding power.
- E.g., licenses of digital works. Whether can render, copy, edit, or sublicense a work.
  - (Bjørner's talk, video, Microsoft, 2008)
- Scripts are like programs.
- Techniques and notations for modelling programming languages apply.
  - E.g., *denotational semantics*, *operational semantics*.

28 facet: *human behavior*

- "Quality spectrum" for carrying out assigned work.
  - *diligent*, *sloppy*, *delinquent*, *criminal*
- Humans interpret rules and regulations differently and inconsistently.
- Specification languages allowing non-determinism and looseness preferable.

29 from descriptions to prescriptions

Domain descriptions serve as a basis for constructing *requirements prescriptions*. These specify *properties* (not implementations) of a machine (hardware and software) implementing them.

30 implementation relation

$$\mathcal{D}, \mathcal{M} \models \mathcal{R} \tag{2}$$

Machine $\mathcal{M}$ implements the requirements $\mathcal{R}$ in the context of the domain $\mathcal{D}$.

31 formal descriptions

- no single specification language suffices
  - "It seems highly unlikely and appears not to be desirable to obtain a single, "universal" specification language capable of "equally" elegantly, suitably abstractly modelling all aspects of a domain."

32 RAISE

- formal specification language (RSL)
- associated method for software development
  - stepwise refinement
  - invent and verify paradigm
- supporting tools

33 RSL

- supports different specification styles
  - algebraic or model-oriented
  - applicative or imperative
  - sequential or concurrent
- modular specifications
- types, values, variables, channels, axioms
- George and Haxthausen: The Logic of the RAISE Specification Language

34 RSL types and values

type Colour
value
  black, white : Colour
axiom
  black $\neq$ white

35 RSL: applicative functions

value
  reverse : **Int**$*$ $\to$ **Int**$*$
  reverse(l) $\equiv$
    if l $=\langle\rangle$ **then** $\langle\rangle$
    else reverse(**tl** l) $^\frown$ $\langle$**hd** l$\rangle$ **end**

36 RSL: imperative functions

variable v : **Int**
value
  add_to_v : **Int** → **write** v **Unit**
  add_to_v(x) ≡ v := v + x

37 model-oriented specification languages

- e.g., Z and VDM-SL
  - Among the most popular formal methods. Both are ISO Standards.
- Z notation (ISO/IEC 13568) semantics are based on logic and ZF set theory.
- VDM-SL (ISO/IEC 13817) is used to specify data types and operations on them.

38 property-oriented specification languages

e.g., CafeOBJ

- For specifying models and verifying their properties.
- Equational logic and theorem proving.
- Logical semantics based on institutions.

39 event-based languages

e.g., Petri nets

- For specifying distributed systems. States and transitions specified. Non-deterministic execution.

40 temporal languages

e.g., TLA+

- Temporal Logic of Actions
- by Leslie Lamport, the LaTeX creator
- for specifying concurrent and reactive systems
- PlusCal (an algorithm language) is based on it
  - an algorithm implemented in PlusCal can be automatically translated to a TLA+ specification for checking and reasoning (see Lamport, 2009)

41 process-based specification languages

e.g., CSP

- Communicating Sequential Processes
- a process algebra
- events, primitive processes, algebraic operators
- originally described in Hoare, 1978

42 industrial uses of RAISE

by Terma

42.1 an on-board demonstration application for ESA

- ESA (the European Space Agency)
- "RAISE was used to specify and develop part of a standard on-board instrument control unit, and the Ada translator was used to produce a prototype of the code."

42.2 Ørsted microsatellite

- "The spacecraft was assembled and integrated at Terma. Terma has used the RAISE method for developing its parts of the on-board software."

43 TRain (The Railway Domain)

www.railwaydomain.org

"Because we need a grand challenge project in order to gather enough momentum to make progress along the road to industrially scalable and useful, integrated formal techniques."

44 draft models of various railway domain aspects

Towards a TRain book

45 板寄せ方式 (いたよせほうしき)

- Itayose method is used for stock price formulation at TSE.
  - opening and closing prices, etc.
- Domain rules. Probably of interest for those developing trading applications.

46 a cautionary tale

- "An employee at Mizuho Securities, intending to sell one share at 610,000 yen, mistakenly typed an order to sell 610,000 shares at 1 yen."
  - Tetsuo Tamai: Social impact of information system failures
- Caused a "highly exceptional situation".
  - seven conditions holding at the same time
- Previously uncovered flaw in TSE Stock Order System meant order went through, and couldn't be cancelled. → 40,000,000,000¥ loss

47 Bjørner's formalisation of some of the system

Dines Bjørner: The TSE Trading Rules (2010)

48 further reading

- Bjørner's Software Engineering trilogy (Springer, 2006)
  - for more details
- Henry N. Pollack: Uncertain Science… Uncertain World (2003) ♠
  - on the difficulty of modelling the real world