Integrated Design and Process Technology, **IDPT-2002**
Printed in the United States of America, June, 2002
©2002 Society for Desing and Process Science

Computing Systems for Railways
A Rôle for Domain Engineering

Dines Bjørner[(a)], Chris George[(b)], and Søren Prehn[(c)]

(a) Dept. of Computer Science, Technical University of Denmark, DK–2800 Kgs. Lyngby, Denmark; db@imm.dtu.dk

(b) UNU/IIST, P.O. Box 3058, Macau SAR, via Hong Kong, China; cwg@iist.unu.edu

(c) Terma Electronics, Bregnerødvej 134, DK–3460 Birkerød, Denmark; snp@terma.com

ABSTRACT: Railways are usually considered to be very large scale, infrastructure supporting systems. They are distributed. They exhibit thousands of concurrent, but highly related (interwoven) activities. And, even without any computing or communication support, railway systems function as a result of critical interplays between railway staff, users and railway (and other, technological) equipment.

To develop computing systems support for railway activities requires, it is emphasised in this paper, first that the developers understand those parts of the railways which are to be supported, we call it *the domain* ($\mathcal{D}$); secondly that the developers understand what support is desired, we call it *the requirements* ($\mathcal{R}$); and, thirdly, that the developers correctly implement these requirements as some *computing system* ($\mathcal{S}$). Correctness is needed in order to deliver expected railway services, to avoid failures that may lead to injury to people or loss of property or human life. Correctness can be expressed as a relation: $\mathcal{D}, \mathcal{S} \models \mathcal{R}$: The computing system implementation ($\mathcal{S}$) can be shown to implement the requirements ($\mathcal{R}$) based on assumptions made about the domain ($\mathcal{D}$). To help ensure correctness, the domain, the requirements and the computing system are suitably formalised.

The formal models, $\mathcal{D}, \mathcal{R}$, and $\mathcal{S}$, are usually rather large documents. For developers and clients to trust that these formal specifications indeed do describe, what is intended, they need be structured such that their readers can grasp smaller parts ("*a page at a time*") and their composition. To develop a *domain description* we propose to decompose the overall development into the development of descriptions of domain intrinsics, domain technologies, domain management & organisation, domain rules & regulations, domain human behaviour, and possibly other *domain facets.*

The paper will briefly explain — and exemplify — some of these development concepts.

Invited paper for the IDPT'2002 Session on *Integration of Modelling Techniques for Train Control Systems* (Pasadena, Calif., 24–28 June 2002)

(a)'s Work on this paper was partially supported by the Commission of the European Communities (EU) under contract HPRN–CT–1999–00104 AMORE: Algorithmic Methods for Optimising Railways in Europe. (b–c)'s Work was partially supported by UNU/IIST.

I. Introduction

A. Domains, Requirements and Software Design

We assume the basic dogma: Before *software* can be *designed* it must be *requirements specified.* And before *requirements* can be expressed, an understanding of the world in which these *requirements* reside, the *domain,* must be formulated.

The *software design* describes how a computer (the hardware) is to proceed in order to achieve stated *requirements.* The *requirements* usually describe three things: (1) Which phenomena of the *domain* shall be supported by computing (the *domain requirements*); (2) which interface between the *machine* (*hardware + software*) and external phenomena: People, and other sensors and actuators, shall be provided (the *interface requirements*); and (3) what performance, dependability, maintenance, platform, and documentation measures are expected (the *machine requirements*).

Domain descriptions are indicative: Describe the "chosen world as it is", ie. the *domain* — without any reference to *requirements*, let alone *software design. Requirements prescriptions* are putative: Prescribe what there is to be, properties, not designs, of the *machine.*

Domain descriptions must describe the chosen *domain* with its imperfections, not try "paint a picture" of a "world as one would like it to be". In this paper we shall focus on such *domain descriptions.*

In this paper we only briefly touch upon the methodological principles, techniques and tools that allow the software developer, based on formal descriptions of the domain to rigorously *project, instantiate, extend* and *initialise* a *domain description* "into" a *domain requirements definition,* and, from *domain* and *interface requirements definitions,* to similarly rigorously develop *software architecture designs.* We cover such principles, techniques and tools in other papers, eg. [4], [6], [7], [8], and in our lecture notes[1].

B. The Problem to be Addressed

In this paper we shall survey some aspects of domain engineering only.

The overall problem that we are generally studying is that of methods for the development of large scale, typically infrastructure component software systems.

[1]URL: `http://www.imm.dtu.dk/~db/TheSEBook`

Excluded from our software development method concerns are therefore those related to the discovery, the invention of algorithms & data structures, for well--delineated problems such as sorting and searching, graph operations, fast Fourier transforms, parsing, *&c.* The borderline between infrastructure software systems and algorithms and & structures is indeed a fuzzy one — and one that we really do not wish to further investigate here. Suffice it to say that the infrastructure software systems we have in mind will indeed contain many examples of algorithms & data structures ! But as concerns the principles and techniques of methods — we only claim that we investigate some that are deemed applicable to infrastructure software systems development.

C. Some Typographical Conventions

The text alternates between paragraphs which either contain plain text, or `characterises`, or `defines` a concept, which are then usually followed by paragraphs which `discuss` the concept, and paragraphs which state a `principle`, a `technique`, or a `tool`. We use the ■ delimiter to show the end of the `specialised paragraphs`.

We make a distinction between characterisations and definitions: The former are (oftentimes necessarily) informal, the latter sometimes formalisable.

II. Domain Perspectives and Facets

We treat the subject of domain engineering in two parts. First we consider the plethora of stake-holders, that is: Individuals and institutions that are more-or-less interested in, or influenced by what goes on in the domain. Then we consider a concept of domain facets.

Modelling stake-holder perspectives and domain facets takes place, during development, "concurrently": One alternates "to-and-from" iteratively.

A. Stake–holders and Stake–holder Perspectives

Railway systems are characterised by a rather large set of diverse stake–holders. Failure of control systems are often due to misunderstandings of their diverse views of the domain and the sometimes thereby conflicting requirements.

A.1 Stake-holders

Characterisation: *Stake–holder.* By a stake–holder we mean a closely knit, tightly related group of either people or institutions, pressure groups — where the "fabric" that "relates" members of the group, "separates" these from other such stake-holder groups, and from non-stake-holder entities. ■

Examples: *Stake–holders* typically include enterprise staff: (i) owners, (ii) management (a) executive management (b) line management, and (c) "floor", ie. operations management, and (iii) workers of all kinds, (iv) families of the above, (v) the customers of the enterprise, (vi) competitors, and the external, "2nd tier" stake–holders: (vii) resource providers (a) IT resource providers[2], (b) non–IT/non–finance [3], and (c) financial service providers, (viii) regulatory agencies who oversee enterprise operations[4], (ix) local and state authorities, (x) politicians, and the (xi) "public at large". They all have a perspective on the enterprise. . ■

Discussion: It always makes good, commercial as well as technical, sense to incorporate the views of as many stake–holder groups as are relevant in the software development process. One need not refer to social, including so--called democratic, reasons for this inclusion. It is simply more fun to make sure that one has indeed understood as much of the domain (and, for that matter, as much of possible requirements) as is feasible, before embarking on subsequent, costly software development phases. ■

The Principle of Modelling the *Stake–holder* Concept expresses that the developer and the client, when setting out on a domain description, clearly defines which stake-holders must be recognised and duly involved in the development. ■

Technique of Modelling the *Stake–holder* Concept: Consider modelling each stake-holder group as a process, or a set of processes (ie. behaviour[s]), or define suitable stake-holder specific context and/or state components and associated (observer and generator) functions. ■

A.2 Stake-holder Perspectives

Characterisation: *Stake–holder Perspective.* By a stake–holder perspective we mean a partial description, a description which emphasises the designations, definitions and refutable assertions[5] that are particular to a given stake--holder group, or the interface between pairs, etc., of such. ■

Discussion: Each perspective usually gives rise to a distinct view of the domain. These views share properties. A good structuring of the "totality" of perspectives can be helped by suitable, usually algebraic specification language constructs, such as possibly the class, scheme and object constructs of the RAISE [23] Specification Language RSL [22]. We shall presently not illustrate this point. ■

The Principle of Modelling the *Stake–holder Perspective* Concept expresses that the developer and the client, when setting out on a domain description together, suitably as part of the contract, clearly defines which stake--holder perspectives must be recognised and duly included

[2] Viz.: Computer hardware and other IT equipment, software houses, facilities management, etc.

[3] Viz.: Consumable goods, leasing agencies, etc.

[4] Viz.: Environment bureaus, financial industry authorities (eg.: The US Federal Reserve Board), food and drug administration (eg.: The US FDA), health authorities (eg.: The US HEW), etc., depending on the enterprise.

[5] Designations, definitions and refutable assertions are concepts defined in [36].

in the descriptions. ∎

Example: *Strategic, Tactical and Operations Resource Management.* We now present a rather lengthy example. It purports to illustrate the interface between a number of stake-holder perspectives. The stake-holders are here: An enterprise's top level, executive management (which plan, takes and follows up on strategic decisions), its line management (which plan, takes and follows up on tactical decisions), its operations management (which plan, takes and follows up on operational decisions), and the enterprise "workers" (who carry out decisions through tasks). Strategic management here has to do with upgrading or downsizing, ie. converting an enterprise's resources from one form to another — making sure that resources are available for tactical management. Tactical management here has to do with temporally scheduling and spatially allocating these resources, in preparation for operations management. Operations management here makes final scheduling and allocation, but now to tasks, in preparation for actual enterprise ("floor") operations.

Let R, Rn, L, T, E and A stand for resources, resource names, spatial locations, times, enterprises (with their estimates, service and/or production plans, orders on hand, etc.), respectively tasks (actions). SR, TR and OR stand for strategic, tactical and operational resource views, respectively.[6] SR expresses (temporal) schedules: Which sets of resources are either bound or free in which (pragmatically speaking: overall, ie. "larger") time intervals. TR expresses temporal and spatial allocations of sets of resources, in certain (pragmatically speaking: mode finer "grained", ie. "smaller") time intervals, and to certain locations. OR expresses that certain actions, A, are to be, or are being, applied to (parameter–named) resources in certain time intervals.

type R, Rn, L, T, E
  RS = R-**set**
  SR = $(T \times T) \xrightarrow[m]{} RS$,   SRS = SR-**infset**
  TR = $(T \times T) \xrightarrow[m]{} RS \xrightarrow[m]{} L$,   TRS = TR-**set**
  OR = $(T \times T) \xrightarrow[m]{} RS \xrightarrow[m]{} A$
  A = $(Rn \xrightarrow[m]{} RS) \xrightarrow{\sim} (Rn \xrightarrow[m]{} RS)$
value
  obs_Rn: $R \rightarrow Rn$
  srm: $RS \rightarrow E \times E \xrightarrow{\sim} E \times (SRS \times SR)$
  trm: $SR \rightarrow E \times E \xrightarrow{\sim} E \times (TRS \times TR)$
  orm: $TR \rightarrow E \times E \xrightarrow{\sim} E \times OR$
  p: $RS \times E \rightarrow$ **Bool**
  ope: $OR \rightarrow TR \rightarrow SR \rightarrow (E \times E \times E \times E) \rightarrow E \times RS$

The partial, including loosely specified, and in cases the non-deterministic functions: srm, trm and orm stand for strategic, tactical, respectively operations resource management. p is a predicate which determines whether the enterprise can continue to operate (with its state and in its environment, e), or not. To keep our model "small", we have had to resort to a "trick": Putting all the facts knowable and needed in order for management to function adequately into E ! Besides the enterprise itself, E also models its environment: That part of the world which affects the enterprise.

There are, accordingly, the following management functions:

Strategic resource management, srm(rs)(e,e''''), let us call the result (e',(srs,sr)) [see "definition" of the enterprise "function" below], proceeds on the basis of the enterprise (e) and its current resources (rs), and "ideally estimates" all possible strategic resource acquisitions (upgrading) and/or down-sizings (divestmments) (srs), and selects one, desirable strategic resource schedule (sr). The "estimation" is heuristic. Too little is normally known to compute sr algorithmically. One can, however, based on careful analysis of srm's pre/post conditions, usually provide some form of computerised decision support for strategic management.

Tactical resource management, trm(sr)(e,e''''), let us call the result(e'',(trs,tr)), proceeds on the basis of the enterprise (e) and one chosen strategic resource view (sr) and "ideally calculates" all possible tactical resource possibilities (trs), and selects one, desirable tactical resource schedule & allocation (tr). Again trm can not be fully algorithmitised. But some combinations of partial answer computations and decision support can be provided.

Operations resource management, orm(tr)(e,e''''), let us call the result (e''',or), proceeds on the basis of the enterprise (e) and one chosen tactical resource view (tr) and effectively decides on one operations resource view (or). Typically orm can be algorithmitised — applying standard operations research techniques.

We refer to [5] for details on the above and below model.

Actual enterprise operation, ope, enables, but does not guarantee, some "common" view of the enterprise: ope depends on the views of the enterprise, its context, state and environment, e, as "passed down" by management; and ope applies, according to prescriptions kept in the enterprise context and state, actions, a, to named (rn:Rn) sets of resources.

The above account is, obviously, rather "idealised". But, hopefully, indicative of what is going on.

Relating the above schematic example to the railway domain we may suggest: Resources R include access to (not necessarily ownership of) the rail net, rights to rent passenger train carriages and locomotives, staff, monies, etc. Strategic resources is, for example, about needing additional or changed rail net access rights, needing further or different kinds of train sets, etc. Strategic resource management , srm, typically brings many operators together, negotiating with rail infrastructure owners about access rights,

[6]In the formalisation, take for example that of OR, ie.: OR = $(T \times T) \xrightarrow[m]{} RS \xrightarrow[m]{} A$ = defines OR to be the type of maps ($\xrightarrow[m]{}$) from time periods (intervals ($T \times T$)) into maps from sets of resources RS into actions (A) [to be performed on these resources during the stated time interval]. These actions are partial functions ($\xrightarrow{\sim}$) from argument (Rn) named sets of resources (RS) into similarly such named results.

and with train set leasing (and lease finance) companies for rental of train sets, etc. srs:SRS designates all possible outcomes of a company's own strategic planning, sr:SR designates a negotiated solution. Tactical resources is, for example, now about the rostering of train staff (crew allocation), allocation of train sets to maintenance locations, etc. Tactical resource management , trm, typically involves negotiation with trade unions, with maintenance units, etc. trs:TRS designates all possible outcomes of a company's own tactical planning (its negotiating options), tr:TR designates a negotiated solution. *&c.*

To give a further abstraction of the "life cycle" of the enterprise, we "idealise" it, as now shown:

```
value
  enterprise: RS $\stackrel{\sim}{\to}$ E $\stackrel{\sim}{\to}$ Unit
  enterprise(rs)(e) ≡
    if p(rs)(e) then
      let (e′,(srs,sr)) = srm(rs)(e,e′′′′),
         (e′′,(trs,tr)) = trm(sr)(e,e′′′′),
         (e′′′,or) = orm(tr)(e,e′′′′),
         (e′′′′,rs′) = ope(or)(tr)(sr)(e,e′,e′′,e′′′) in
      let e′′′′′:E • p′(e′′′′,e′′′′′) in
      enterprise(rs′)(e′′′′′) end end
    else stop end

  p′: E × E → Bool
```

The enterprise re-invocation argument, rs′, a result of operations, is intended to reflect the use of strategically, tactically and operationally acquired, spatially and task allocated and scheduled resources, including partial consumption, "wear & tear", loss, replacements, etc.

The **let** e′′′′′:E • p′(e′′′′,e′′′′′) **in** ... shall model a changing environment.

Thus there were two forms of recursion at play here: The simple tail-recursion, and the recursive "build-up" of the enterprise state e′′′′. The latter is the interesting one. Solution, by iteration towards some acceptable, not necessarily minimal fix-point, "mimics" the way the three levels of management and the "floor" operations change that state and "pass it around, up-&-down" the management "hierarchy". The operate function "unifies" the views that different management levels have of the enterprise, and influences their decision making. Dependence on E also models potential interaction between enterprise management and, conceivably, all other stake-holders. ∎

Discussion: We remind the reader that — in the previous example — we are "only" modelling the domain ! That model is, obviously, sketchy. But we believe it portrays important facets of domain modelling and stake-holder perspectives. The stake–holders were, to repeat: Strategy ("executive") management (srm, p), tactical ("line") management (trm), operations ("floor") management (orm), and the workers (ope). The perspective being modelled focused on two aspects: Their individual jobs, as "modelled" by the "functions" (srm, p, trm, orm, ope), and their interactions, as "modelled" by the passing around of arguments (e, e′, e′′, e′′′, e′′′′) The **let** e′′′′′:E • p′(e′′′′,e′′′′′) **in** ... which "models" the changing environment is thus summarising the perspectives of "all other" stake–holders !

We are modelling a domain with all its imperfections: We are not specifying anything algorithmically; all functions are rather loosely, hence partially defined, in fact only their signature is given. This means that we model well-managed as well as badly, sloppily, or disastrously managed enterprises. We can, of course, define a great number of predicates on the enterprise state and its environment (e:E), and we can partially characterise intrinsics — facts that must always be true of an enterprise, no matter how.

If we "programme-specified" the enterprise then we would not be modelling the domain of enterprises, but a specifically "business process engineered" enterprise. Or we would be into requirements engineering — we claim. ∎

Technique of Modelling the *Stake–holder Perspective* Concept: Emphasise how the distinct stake-holders interact, which phenomena in the domain they generate, share, or consume. This 'technique' follows up on the 'Stake–holder' modelling technique. ∎

A.3 Discussion

The stake-holder example given above is "sketchy". It identifies, we believe, the most important entities and operations that are relevant to a small number of interacting stake-holders. We believe that "rough sketches" like the above are necessary in the iterative development of domains.

B. Domain Facets

We shall outline the following facets:

Domain intrinsics: That which is common to all facets.

Domain support technologies: That in terms of which several other facets (intrinsics, management & organisation, and rules & regulations) are implemented.

Domain management & organisation: That which primarily determines and constrains communication between enterprise stake-holders.

Domain rules & regulations: That which guides the work of enterprise stake-holders, their interaction, and the interaction with non-enterprise stake-holders.

Domain human behaviour: The way in which domain stake-holders despatch their actions and interactions wrt. enterprise: dutifully, forgetfully, sloppily, yes even criminally.

We shall briefly characterise each of these facets. We venture to express "specification patterns" that "most closely capture" essences of the facet.

Separating the treatment of each of these (and possibly other) facets reflect a principle:

The Development Principle of *Separation of Concerns* expresses that when possible one should separate distinguishable concerns and treat them separately. ■

Discussion: We believe that the facets we shall present can be treated separately in most developments — but not necessarily always. Separation or not is a matter also of development as well as of presentation style.

The separation, in more generality, of computing systems development into the triptych of domain engineering, requirements engineering and machine (hardware + software) design, is also a result of separation of concerns — as are the separations of domain requirements, interface requirements and machine requirements (within requirements engineering), as well as the separations of software architecture and program organisation design [6]. ■

B.1 Intrinsics

Railways, although they have many "players and actors" revolve around some core notions: The rail net and trains on these.

• The Concept:

Characterisation: *Intrinsics:* That which is common to all facets. ■

• An Example:

Example: *Rail nets and switches.* We first give a summary view of a domain model for railway nets, first informally, then formally, leaving out axioms: A railway net consists of two or more stations and one or more lines. Nets, lines and stations consists of rail units. A rail unit is either a linear unit, or a switch unit, or a crossover unit, etc. Units have connectors. A linear unit has two connectors, a switch unit has three, a crossover unit has four, etc. A line is a linear sequence of connected linear units. A station usually has all kinds of units. A line connects exactly two distinct stations. A station contains one or more tracks (say, pragmatically, for passenger platforms or for cargo sidings). A path is a pair of connectors of a unit, and pragmatically defines a way for a train to traverse that unit. A unit is at any one time in a state (σ), which we may consider a set of paths. Over a lifetime a unit may attain one or another state in that unit's state space (ω).

```
type
  N, L, S, Tr, U, C
value
  obs_Ls: N → L-set, obs_Ss: N → S-set,
  obs_Us: (N|L|S) → U-set, obs_Cs: U → C-set
  obs_Trs: S → Tr-set
type
  P′ = U × (C×C), Σ = P-set, Ω = Σ-set
  P = {| p:P′ • let (u,(c,c′))=p in (c,c′)∈ obs_Σ(u) end |}
value
  obs_Σ: U → Σ, obs_Ω: U → Ω
```

[7] elaborates further on the model just presented: Formally defines the axioms that suitably constrain the types.

From the perspective of a train passenger or a cargo customer it is not part of the intrinsics that nets have units and units have connectors. Therefore also paths, states and state-spaces are not part of the intrinsics of a net as seen from such stake–holders.

From the perspective of the train driver and of those who provide the setting of switches and signalling in general, units, paths, and states are indeed part of the intrinsics: The intrinsics of a rail switch is that it can take on a number of states. A simple switch ($^{c_|}\overset{c_/}{\underset{c}{Y}}$) has three connectors: $\{c, c_|, c_/\}$. c is the connector of the common rail from which one can either "go straight" $c_|$, or "fork" $c_/$. So we have that ω_{g_s} :

$$\begin{aligned}&\{\{\},\\ &\{(c,c_|)\},\{(c,c_|),(c_|,c)\},\{(c_|,c)\},\\ &\{(c,c_/)\},\{(c,c_/),(c_/,c)\},\{(c_/,c)\},\\ &\{(c,c_/),(c_|,c)\},\{(c,c_/),(c_/,c),(c_|,c)\},\{(c_/,c),(c_|,c)\}\}\end{aligned}$$

ideally models a general switch. Any particular switch ω_{p_s} may have $\omega_{p_s} \subset \omega_{g_s}$. Nothing is said about how a state is determined: Who sets and resets it, whether determined solely by the physical position of the switch gear, or also by visible or virtual (ie. invisible, intangible) signals up or down the rail away from the switch. ■

• Methodological Consequences:

The Principle of Modelling the *Intrinsics* Domain Facet expresses that in any modelling one first form and describe the intrinsic concepts. ■

Technique of Modelling the *Intrinsics* Domain Facet: The intrinsics model of a domain is a partial specification. As such it involves the use of well–nigh all description principles. Typically we resort to property oriented models, ie. sorts and axioms. ■

• Discussion:

Thus the intrinsics become part of every one of the next facets. From an algebraic semantics point of view these latter are extension of the above.

B.2 Support Technologies

Railway systems are dominated by a rather large, if not "huge" varieties of technology legacy: Old, newer and latest mechanics, electro–mechanics, electronics and mecha–tronics support for large diversities of functions.

• The Concept:

Characterisation: *Support Technology* — that in terms of which several other facets (intrinsics, management & organisation, and rules & regulations) are implemented. ■

• An Example:

Example: *Railway switches.* An example of different technology *stimuli:* A railway switch, "in ye olde days" of the "childhood" of railways, was *manually "thrown";*

later it could be[7] mechanically controlled from a distance by *wires and momentum "amplification";* again later it could be electro-mechanically controlled from a further distance by *electric signals that then activated mechanical controls;* and today switches are usually *controlled in groups that are electronically interlocked.*

An aspect of supporting technology includes the recording of state-behaviour in response to external stimuli. Figure 1 indicates a way of formalising this aspect of a supporting technology.

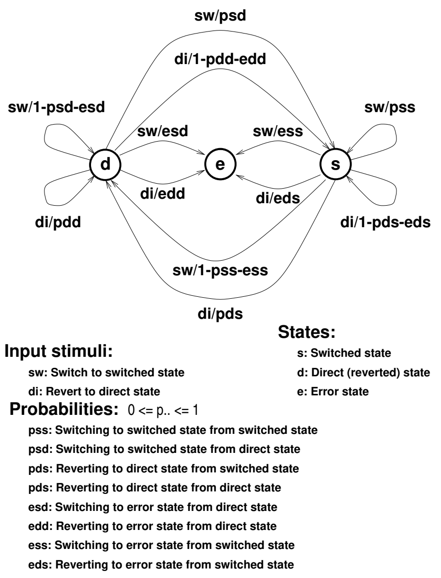
sw/psd
di/1-pdd-edd
sw/1-psd-esd
sw/pss
sw/esd
sw/ess
d
e
s
di/edd
di/eds
di/pdd
di/1-pds-eds
sw/1-pss-ess
di/pds
States:
Input stimuli:
s: Switched state
sw: Switch to switched state
d: Direct (reverted) state
di: Revert to direct state
e: Error state
Probabilities: 0 <= p.. <= 1
pss: Switching to switched state from switched state
psd: Switching to switched state from direct state
pds: Reverting to direct state from switched state
pds: Reverting to direct state from direct state
esd: Switching to error state from direct state
edd: Reverting to error state from direct state
ess: Switching to error state from switched state
eds: Reverting to error state from switched state

Fig. 1. Probabilistic State Switching

Figure 1 intends to model the probabilistic (erroneous and correct) behaviour of a switch when subjected to settings (to switched (s) state) and re–settings (to direct (d) state). A switch may go to the switched state from the direct state when subjected to a switch setting s with probability psd. Etcetera. ∎

• Another Example:

Another example shows another aspect of support technology.

Example: *Railway Optical Gates.* Train traffic (iTF), intrinsically, is a total function over some time interval, from time (T) to monotonically positioned (P) trains (TN).

Conventional optical gates "sample", at regular intervals, the intrinsic train traffic. Hence the collection of all optical gates is a partial function[8] from intrinsic to sampled train traffics (sTF).

type
  T, TN
  $P = U^*$
  $iTF = T \to (N \times (TN \xrightarrow[m]{} P))$
  $sTF = T \xrightarrow[m]{} (N \times (TN \xrightarrow[m]{} P))$
value
  [optical gates] og: $iTF \overset{\sim}{\to} sTF$
  [close] c: $N \times (P \times P) \overset{\sim}{\to}$ **Bool**
axiom
  $\forall$ itt:iTF • **let** stt = og(itt) **in** $\forall$ t:T • t $\in$ **dom** stt •
    t $\in$ **dom** itt $\wedge$ $\forall$ Tn:TN • tn $\in$ **dom** itt(t) $\Rightarrow$
      tn $\in$ **dom** stt(t) $\wedge$ c((itt(t))(tn),(stt(t))(tn)) **end**

The axioms express a property that one expects to hold for optical gates: That the optical gate–recorded train positions are close to those of the trains in the actual world

N was defined in Section II-B.1. Since units change state with time, N need be part of any model of traffic. ∎

• Methodological Consequences:

Technique of Modelling the *Support Technology* Domain Facet: The support technologies model of a domain is a partial specification — hence all the usual abstraction and modelling principles, techniques and tools apply. More specifically: Support technologies (st:ST) "implements" intrinsic contexts and states: $\gamma_i : \Gamma_i, \sigma_i : \Sigma_i$ in terms of "actual" contexts and states: $\gamma_a : \Gamma_a, \sigma_a : \Sigma_a$

type
  Syntax,
  Γ_i, VAL_i, Γ_a, VAL_a,
  ST = Γ_i $\overset{\sim}{\to}$ Γ_a
value
  sts:ST-**set**
axiom
  $\forall$ st:ST • st $\in$ sts $\Rightarrow$ ...

Support technology is not a refinement, but an extension. Support technology typically introduces considerations of technology accuracy, failure, etc. Axioms characterise members of the set of support technologies sts. An example axiom was given in the optical gate example . ∎

The Principle of Modelling the *Support Technology* Domain Facet is a principle that is relative to all other domain facets. It expresses that one must first describe essential intrinsics. Then it expresses that support technology is any means of implementing concrete instantiations of some intrinsics, of some management & organisation, and/or of some rules & regulations. Generally the principle states that one must always be on the look-out for and inspire new support technologies. The most abstract form of the principle is: *"What is a support technology one day becomes part of the domain intrinsics a future day"*. ∎

• Discussion:

[52] exemplify the use of the Duration Calculus [14], [18], [17], [13], [16] in describing supporting technologies

[7] 'It was, but can still be *&c.*'

[8] This example is due to my former MSc Thesis student Kristian M. Kalsing

that help achieve safe operation of a road level rail crossing.

The support technology facet descriptions "re–appear" in the requirements definitions: Projected, instantiated, extended and initialised [6]. In the domain description we "only" record our understanding of all aspects of support technology "failures". In the requirements definition we then follow up and make decisions as to which kinds of "breakdowns" the computing system, the machine, is to handle, and what is to be achieved by such "handlings".

B.3 Management and Organisation

Railway systems are characterised by usually highly structured management organisations, rules and regulations set up by upper echelons of management to be followed by lower levels and by "ground" staff and users.

Examples: *Train Monitoring* In China, as an example, re–scheduling of trains occur at stations and involves telephone negotiations with neighbouring stations ("up and down the lines"). Such re–scheduling negotiations by phone imply reasonably strict management & organisation (M&O). This kind of M&O reflects the geographical layout of the rail net. ▪

• The Concept:

Characterisation: *Management and Organisation:* That which primarily determines and constrains communication between enterprise stake-holders. ▪

• Conceptual Examples — I:

Discussion: People staff enterprises, the components of infrastructures with which we are concerned, for which we develop software. The larger these enterprises, these infrastructure components, are, the more need there is for management & organisation. The rôle of management is roughly, for our purposes, twofold: Firstly, to perform strategic, tactical and operational work, to make strategic, tactical and operational policies (cf. Section II-A.2) — including rules & regulations, cf. Section II-B.4 — and to see to it that they are followed. The rôle of management is, secondly, to react to adverse conditions: Unforeseen situations, and decide upon their handling, ie. conflict resolution.

Policy setting should help non-management staff operate normal situations — for which no management interference is thus needed. And management "back-stops" problems: Takes these problems off the shoulders of non-management staff.

To help management and staff know who's in charge wrt. policy setting and problem handling, a clear conception of the overall organisation is needed: Organisation defines lines of communication within management and staff and between these. Whenever management and staff has to turn to others for assistance they usually, in a reasonably well–functioning enterprise, follow the command line: The paths of organigrams — the usually hierarchical box and arrow/line diagrams. ▪

• Methodological Consequences — I:

Techniques of Modelling the *Management & Organisational* Domain Attributes Concepts: The management & organisation model of a domain is a partial specification — hence all the usual abstraction and modelling principles, techniques and tools apply. More specifically: Management is a set of predicates, observer and generator functions which either parameterise other, the operations functions, that is: Determine their behaviour, or yield results that become arguments to these other functions. We have indicated, earlier, some of the techniques. Organisation is a set of constraints on communication behaviours. "Hierarchical", rather than "linear", and "matrix" structured organisations can also be modelled as sets (of recursively invoked sets) of equations. ▪

• Conceptual Example — II:

Examples: *Management & Organisation* To relate "classical" organigrams to formal descriptions we first show such an organigram, see Figure 2, and then we show schematic processes which — for a rather simple case (ie. scenario) — model managers and managed !

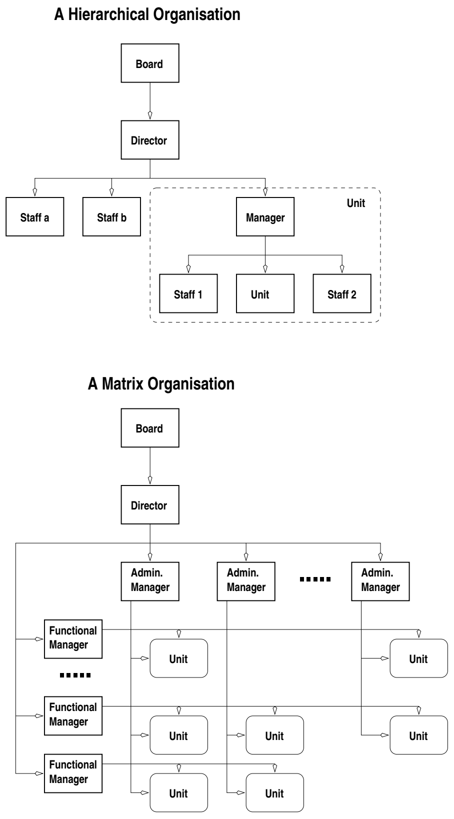

Fig. 2. Organisational Structures

```
type Msg, $\Psi$, $\Sigma$, Sx
channel { ms[i]:Msg | i:Sx }
value
  sys: Unit $\rightarrow$ Unit
  mgr: $\Psi \rightarrow$ in,out { ms[i] | i:Sx } Unit
  stf: i:Sx $\rightarrow \Sigma \rightarrow$ in,out ms[i] Unit
  sys() $\equiv$ || { stf(i)(i$\sigma$) | i:Sx } || mgr($\psi$)

  mgr($\psi$) $\equiv$
    let $\psi'$ = ... ;
      (|| {ms[i]!msg;f_m(msg)($\psi$)|i:Sx } )
      $\square$ ($\square$ {let msg$'$ = ms[i]? in
            g_m(msg$'$)($\psi$) end|i:Sx}) in
    mgr($\psi'$) end

  stf(i)($\sigma$) $\equiv$
    let $\sigma'$ = ... ;
      ((let msg = ms[i] ? in f_s(msg)($\sigma$) end)
       $\square$
       (ms[i] ! msg$'$ ; g_s(msg$'$)($\sigma$))) in
    stf(i)($\sigma'$) end

  f_m, g_m: Msg $\rightarrow \Psi \rightarrow \Psi$,
  f_s, g_s: Msg $\rightarrow \Sigma \rightarrow \Sigma$
```

Both manager and staff processes recurse (ie. iterates) over possibly changing states. Management process non--deterministically, external choice, "alternates" between "broadcast"–issuing orders to staff and receiving individual messages from staff. Staff processes likewise non--deterministically, external choice, "alternates" between receiving orders from management and issuing individual messages to management. The example also illustrates modelling stake-holder behaviours as interacting (here CSP–like, [29], [30], [48]) processes. ∎

• Methodological Consequences — II:

Discussion: The *strategic, tactical and operations resource management* example of Section II-A.2 illustrated another management & organisation description pattern. It is based on a set of, in this case, recursive equations. Any way of solving these equations, finding a suitable fix-point, or an approximation thereof, including just choosing and imposing an arbitrary "solution", reflects some management communication. The syntactic ordering of the equations — in this case: a "linear" passing of enterprise "results" from "upper" equations onto "lower" equations — reflects some organisation. ∎

The Principle of Modelling the *Management & Organisation* Domain Facets expresses that relations between resources, and decisions to acquire and dispose resources, to de–, re– and schedule and de–, re– and allocate resources, and to de–, re– and activate resources, are the prerogatives of well-functioning management, reflect a functioning organisation, and imply invocation of procedures that are modelled as actions that "set up" and "take-down" contexts and change states. As such these principles tell us which sub-problems of development to tackle. ∎

Techniques of Modelling the *Management & Organisation* Domain Facet: We have already, under techniques for modelling 'Stake–holder' and 'Stake–holder Perspectives', mentioned some of the techniques. Two "extremes" were shown: Earlier we modelled individual management groups by their respective functions (strm, trm, orm), and their interaction (ie. organisation) by "solutions" to a set of recursive equations ! Presently we modelled management & organisation, especially the latter, by communicating sequential behaviours. ∎

• Discussion:

The domain models of management and organisation, eventually find their way into requirements, and, hence, the software design — for the cases that the requirements are about computing support of management and its organisation.

Support to the solution of the recursive equations of the earlier stake–holder example may be offered in the form of constraint based logic solvers which may partially handle logic characterisations of the strategic and tactical management functions, and in the form of computerised support of message passing between the various management groups of the stake–holder example, as well as of the generic example of the present part.

B.4 Rules & Regulations

Railway systems are characterised by large varieties of rules for appropriate behaviour: Of trains, train despatch, monitoring and control, of supporting technology, and hence of humans at all levels.

• The Concept:

Characterisation: *Rule.* That which guides the work of enterprise stake-holders and their support technologies, as well as their interaction and the interaction with non-enterprise stake-holders. ∎

Characterisation: *Regulation.* That which stipulate what is to happen if a rule can be detected not to have been followed when such was deemed necessary. ∎

Rules & regulations are set by enterprises, equipment manufacturers, enterprise associations, [government] regulatory agencies, and by law.

• Two Examples:

We give two examples:

Example: *Train at Stations.*

Rule: *In China arrival and departure of trains at, respectively from railway stations are subject to the following rule: In any three minute interval at most one train may either arrive or depart.*

Regulation: *Disciplinary procedures*. ∎

and:

Example: *Train along Lines.*

Rule: *In many countries railway lines (between stations) are segmented into blocks or sectors. The purpose is to stipulate that if two or more trains are moving — obviously in the same direction — along the line, then there must be at least one free sector (ie. without a train) between any two such trains.*

Regulation: *Disciplinary procedures*. ∎

• Methodological Consequences:

Technique of Modelling the *Rules & Regulations* Domain Facets: There are usually three kinds of syntax involved wrt. (ie. when expressing) rules & regulations (resp. when invoking actions that are subject to rules & regulations): The syntaxes (Syntax_rul, Syntax_reg) describing rules, respectively regulations; and the syntax (Syntax_cmd) of [always current] domain external action stimuli. A rule, denotationally, is a predicate over domain stimuli, and current and next domain configurations (Γ). A regulation, denotationally, is a state changing function over domain stimuli, and current and next domain configurations (Γ). We omit treatment of [current] stimuli:

```
type
  Syntax_cmd, Syntax_rul, Syntax_reg, $\Gamma$
  Rules_and_Regulations = Syntax_rul $\times$ Syntax_reg
  RUL = $\Gamma \to \Gamma \to$ Bool,
  REG = $\Gamma \to \Gamma$
value
  interp: Syntax_rul $\to \Gamma \to$ RUL-set,
  interp: Syntax_reg $\to \Gamma \to$ REG

  valid: RUL-set $\to \Gamma \times \Gamma \to$ Bool
  valid(ruls)($\gamma$,$\gamma'$) $\equiv$
    $\forall$ rul:RUL • rul $\in$ ruls $\Rightarrow$ rul($\gamma$)($\gamma'$)
axiom
  $\forall$ (s_ruls,s_reg):Rules_and_Regulations,$\gamma$:$\Gamma$ •
    let (ruls,reg)=(interp(s_rul)($\gamma$),interp(s_reg)($\gamma$)) in
      $\exists$ $\gamma'$,$\gamma''$:$\Gamma$ • $\sim$valid(ruls)($\gamma$,$\gamma'$)
        $\Rightarrow$ reg($\gamma'$)=$\gamma''$ $\wedge$ valid(ruls)($\gamma$,$\gamma''$)
    end
```

Rules & regulations are therefore modelled by abstract or concrete syntaxes of syntactic rules etc., by abstract types of denotations, and by semantics definitions, usually in the form of axioms or denotation–ascribing functions. ∎

The Principle of Modelling the *Rules & Regulations* Domain Facet expresses that domains are governed by rules & regulations: By laws of nature or edicts by humans. Laws of nature can be part of intrinsics, or can be modelled as rules & regulations constraining the intrinsics. Edicts by humans usually change, but are usually considered part of an irregularly changing context, not a recurrently changing state. Modelling techniques follow these principles. ∎

• Rules & Regulation Scripts:

We discuss an issue, that arises with the above and which points to possible precautionary and/or remedial actions — as they would first be expressed in some requirements:

Discussion: Domain rules & regulations are usually formulated in "almost legalese", ie. in rather precise, albeit perhaps "stilted" subsets of the professional language of the domain in question. In cases such rules & regulation languages can be formalised, and we then call them script languages. A particular set of rules & regulations is thus a script. Such script languages can be mechanised: Making it "easy" for appropriate (rules & regulation issuing) stake-holders to script such scripts — and to have them inserted into their computing system: As predicates that detect rule violations, respectively suggest alternative actions (than causing a potentially violating action) or remedy an actual rule violation. ∎

The rules & regulations, that may be stipulated for a domain, can thus find their way into requirements that specify computerised support for their enforcement.

B.5 Human Behaviour

Railway systems are characterised by large physical distances between staff at all levels, staff whose behaviour cannot always be closely monitored.

• The Concept:

Discussion: Some people try their best to perform actions according to expectations set by their colleagues, customers, etc. And they usually succeed in doing so. They are therefore judged reliable and trustworthy, good, punctual professionals (*b_p*) of their domain. Some people set lower standards for their professional conduct: Are sometimes or often sloppy (*b_s*), make mistakes, unknowingly or even knowingly. And yet other people are outright delinquent (*b_d*) in the despatch of their work: Could'nt care less about living up to expectations of their colleagues and customers. Finally some people are explicitly criminal (*b_c*) in the conduct of what they do: Deliberately "do the opposite" of what is expected, circumvent rules & regulations, etc. And we must abstract and model, in any given situation where a human interferes in the "workings" of a domain action, any one of the above possible behaviours. ∎

Characterisation: *Human Behaviour*. The way in which domain stake-holders despatch their actions and interactions wrt. an enterprise: professionally, sloppily, delinquently, yes even criminally. ∎

• Methodological Consequences:

Techniques of Modelling the *Human Behaviour (I–II)* Domain Facet: We often model the "arbitrariness" of human behaviour by internal non-determinism:

```
... b_p $\sqcap$ b_s $\sqcap$ b_d $\sqcap$ b_c ...
```

The exact, possibly deterministic, meaning of each of the b's can be separately described.

In addition we can model human behaviour by the arbitrary selection of elements from sets and of subsets of sets:

```
type
  X
value
  hb_i: X-set ... $\to$ ... ,
  hb_i(xs,...) $\equiv$ let x:X $\bullet$ x $\in$ xs in ... end

  hb_j: X-set ... $\to$ ... ,
  hb_j(xs,...) $\equiv$ let xs$'$:X-set $\bullet$ xs$'$ $\subseteq$ xs in ... end
```

The above shows just fragments of formal descriptions of those parts which reflect human behaviour. Similar, loose, descriptions are used when describing faulty supporting technologies, or the "uncertainties" of the intrinsic world. ■

Technique of Modelling the *Human Behaviour (III)* Domain Facet: Commensurate with the above, humans interpret rules & regulations differently, and not always "consistently" in the sense of repeatedly applying the same interpretations. Our final specification pattern is therefore:

```
type
  RULS = RUL-set
  Action = $\Gamma \overset{\sim}{\to} \Gamma$-infset
value
  interpret: Syntax_rul $\to \Gamma \to$ RULS-infset

  human_behaviour: Action $\to$ Syntax_rr $\to \Gamma \overset{\sim}{\to} \Gamma$
  human_behaviour($\alpha$)(srr)($\gamma$) as $\gamma'$
    post
      let $\gamma$s = $\alpha(\gamma)$ in
      $\exists\ \gamma'$:$\Gamma \bullet \gamma' \in \gamma$s $\wedge$
        let rules:RULS $\bullet$ rules $\in$ interpret(srr)($\gamma$) in
          $\forall$ rule:RUL $\bullet$ rule $\in$ rules $\Rightarrow$ rule($\gamma$)($\gamma'$)
      end end
```

The above is, necessarily, sketchy: There is a possibly infinite variety of ways of interpreting some rule[s]. A human, in carrying out an action, interprets applicable rules and choose a set which that person believes suits some (professional, sloppy, delinquent or criminal) intent. "Suits" means that it satisfies the intent, ie. yields **true** on the pre/post state pair, when the action is performed — whether as intended by the ones who issued the rules & regulations or not. ■

Discussion: Please observe the difference between the version of interpret as indicated in the "Rules & Regulations" part of the paper and the present version: The former reflected the semantics as intended by the stake–holder who issued the rules & regulations. The latter reflects the professional, or the sloppy, or the delinquent, or the criminal semantics as intended by the similarly "qualified" staff which carries out the rule abiding or rule violating actions. Please also observe that we do not here exemplify any regulations. ■

The Principle of Modelling the *Human Behaviour* Domain Facet expresses what has now been mentioned several times, namely that some people are perfect: Follow rules & regulations as per intentions; other people are sloppy: Fail to follow the prescriptions; and yet other people are derelict or even criminal in the pursuit of their job: Deliberately flaunts rules & regulations. And the principle concludes that one must be prepared for the "worst". That is: Model it all. ■

III. Conclusion

Before software design for railway control applications — applications that are usually safety–critical — can be contemplated we must establish, firmly, the requirements. Before requirements for such applications can be defined, we must understand the domain. This is the main message of this paper.

A corollary message is that of providing a number of principles and techniques for developing domain descriptions, and in particular to illustrate such which are applicable to specific facets of (as here: Railway) domains.

A. Summary

We have covered a number of domain facets: *Intrinsics* ('the very basics'), *support technologies* (implementations of some parts of other facets), *management & organisation, rules & regulations*, and *human behaviour.* One can possibly think of other facets. With each domain facet the "full force" of all abstraction and modelling principles and techniques apply, and a careful "sequencing" ("fitting-in") of the treatment of 'that' facet wrt. other facets must be considered.

For each of the facets we have shown principles of and techniques for their modelling, and we have indicated that these facet models may eventually find their way into requirements models, and hence determine software designs.

B. Discussion

One will never be able, it is conjectured, to achieve a complete domain model. But one can do far better than is practice today — where no such models are even attempted. Most claims of domain models are biased towards contemplated software designs, embody requirements, or are covering only some of the domain (thus implicitly) being projected onto requirements, etc.

When indeed errors, ie. "holes" in the domain description, are still discovered, later, perhaps after final software delivery, then it is now easier, we claim, to pinpoint where these errors first occurred, and hence who were the perpetrators: The software, cum domain or requirements or design, developers, or the stake-holders, or both parties.

C. Relations to Requirements and Software Design

The results of informal as well as formal domain descriptions of supporting technologies, of management and organisations, of rules and regulations, and of human shortcomings find their way into those requirements which define computerised support for taking precautionary actions where technology failures and human errors can be detected.

In the validation interaction between the software developers — who are major "players" in the development of both domain descriptions and requirements definitions — and the domain stake–holders, in that validation process, we claim, many errors — that before could, and hence would, creep unconsciously into the software development — can now be avoided.

On one hand it is now easier to resolve legal issues, and, as well, to repair malfunctioning software. The latter because, in its development, from domains via requirements to designs, we adhere to an unstated principle: That of homomorphic development: *If two or more algebraically independent ("orthogonal") concepts are expressed in the domain and are to be "found", somehow, also in the software, then their implementation must be likewise distinguishable.*

IV. Bibliographical Notes

Space does not permit it, but below we refer to, and would have liked, more individually to comment on, a long list of additional reports, workshop papers and journal publications. Most are related to the issue of formalising requirements and design of mostly safety–critical railway software applications. Suffice it, for now, also for these additional references to serve as a starting point for studies: [34], [43], [44], [15], [37], [3], [26], [25], [19], [45], [12], [31], [38], [32], [20], [21], [46], [2], [24], [50], [11], [47], [27], [42], [33], [28], [41], [1], [51], [35], [60], [54], [40], [39], [53], [49], [9], [10]. As is seen, many of the papers are from five workshops organised by Formal Methods Europe: [55], [56], [57], [58], [59].

References

[1] Patrick Behm. *METEOR : an industrial success in formal development.* In *FME Rail Workshop # 1* [55]. Matra (F).

[2] C. Bernardeschi, A. Fantechi, S. Gnesi, and G. Mongardi. *Proving safety properties for embedded control systems.* In A. Hlawiczka, J.G. Silva, and L.Simoncini, editors, *Dependable Computing - EDCC-2. Second European Dependable Computing Conference Proceedings; Taormina, Italy*, pages 321–32. Springer-Verlag; Berlin, Germany, 1996.

[3] Dines Bjørner. *A Architecture for Running Map Systems.* Technical Report db/arch/01, UNU/IIST, the UN University's International Institute for Software Technology, P.O.Box 3058, Macau; E-Mail: `library@iist.unu.edu`, February 1994.

[4] Dines Bjørner. *Domains as Prerequisites for Requirements and Software &c.* In M. Broy and B. Rumpe, editors, *RTSE'97: Requirements Targeted Software and Systems Engineering*, volume 1526 of *Lecture Notes in Computer Science*, pages 1–41. Springer-Verlag, Berlin Heidelberg, 1998.

[5] Dines Bjørner. *Domain Modelling: Strategic, Tactical & Operational Resource Management,* In Jim Davies, Bill Roscoe and Jim Woodcock, editors, *Festschrift to Tony Hoare: Millenium Perspective in Computer Science.* Oxford University and Microsoft, September 13–14, 1999; Palgrave Publs., pp 23–40, 2000.

[6] Dines Bjørner. *Where do Software Architectures come from ? Systematic Development from Domains and Requirements. A Re–assessment of Software Engineering ? South African Journal of Computer Science*, Number 22, March 1999, pp 3–13. Guest Editor: Chris Brink.

[7] Dines Bjørner. *Formal Software Techniques in Railway Systems.* In Eckehard Schnieder, editor, *9th IFAC Symposium on Control in Transportation Systems*, pages 1–12, Technical University, Braunschweig, Germany, 13–15 June 2000. *VDI/VDE-Gesellschaft Mess– und Automatisieringstechnik, VDI-Gesellschaft für Fahrzeug– und Verkehrstechnik.* Invited plenum lecture.

[8] Dines Bjørner. *Pinnacles of Software Engineering: 25 Years of Formal Methods. Annals of Software Engineering*, 2000. Eds. Dilip Patel and Wang Yi.

[9] Dines Bjørner, Jakob Braad, and Karin S. Mogensen (Eds.). *Models of Railway Systems: Domain.* In *FME Rail Workshop # 5* [59]; (60 pages) Dept. of IT, Techn. Univ. of Denmark.

[10] Dines Bjørner, Jakob Braad, and Karin S. Mogensen (Eds.). *Models of Railway Systems: Requirements.* In *FME Rail Workshop # 5* [59]; (62 pages) Dept. of IT, Techn. Univ. of Denmark.

[11] Dines Bjørner, C.W. George, and S. Prehn. *Scheduling and rescheduling of trains*, page 24 pages. Academic Press, 1999.

[12] Dines Bjørner, Dong YuLin, and S. Prehn. *Domain Analysis: A Case Study of Railway Station Management.* Technical Report db/03/01, UNU/IIST, UN University's International Institute for Software Technology, P.O.Box 3058, Macau; E-Mail: `library@iist.unu.edu`, November 12 1994. Presented at KICS'94: The Kunming (Yunnan, PRC) Intl. CASE Symposium, Nov. 1994.

[13] Zhou Chaochen. *Duration Calculi: An Overview.* Research Report 10, UNU/IIST, P.O.Box 3058, Macau, June 1993. Published in: *Formal Methods in Programming and Their Applications*, Conference Proceedings, June 28 – July 2, 1993, Novosibirsk, Russia; (Eds.: D. Bjørner, M. Broy and I. Pottosin) LNCS 736, Springer-Verlag, 1993, pp 36–59.

[14] Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. *A Calculus of Durations. Information Proc. Letters*, 40(5), 1992.

[15] Zhou Chaochen and Yu Huiqun. *A Duration Model for Railway Scheduling.* Technical Report 24b, UNU/IIST, P.O.Box 3058, Macau, May 1994.

[16] Zhou Chaochen, Zhang Jingzhong, Yang Lu, and Li Xiaoshan. *Linear Duration Invariants.* Research Report 11, UNU/IIST, P.O.Box 3058, Macau, July 1993. Published in: Formal Techniques in Real-Time and Fault-Tolerant systems, LNCS 863, 1994.

[17] Zhou Chaochen, Anders P. Ravn, and Michael R. Hansen. A*n Extended Duration Calculus for Real-time Systems.* Research Report 9, UNU/IIST, P.O.Box 3058, Macau, January 1993. Published in: *Hybrid Systems*, LNCS 736, 1993.

[18] Zhou Chaochen and Li Xiaoshan. *A Mean Value Duration Calculus.* Research Report 5, UNU/IIST, P.O.Box 3058, Macau, March 1993. Published as Chapter 25 in *A Classical Mind*, Festschrift for C.A.R. Hoare, Prentice-Hall International, 1994, pp 432–451.

[19] B. Dehbonei and L.-F. Mejia. *Formal methods in the railways signalling industry.* In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME '94: Industrial Benefit of Formal Methods. Second International Symposium of Formal Methods Europe. Proceedings; Barcelona, Spain*, pages 26–34. Springer-Verlag, Berlin, Germany; Lecture Notes in Computer Science LNCS, 1994.

[20] B. Dehbonei and L.-F. Mejia. *Formal development of safety-critical software systems in railway signaling.* In M. G. Hinchey and J. P. Bowen, editors, *Applications of Formal Methods*, Series in Computer Science, pages 227–252. Prentice Hall International, 1995.

[21] L.H. Eriksson. *Specifying railway interlocking requirements for practical use.* In Erwin Schoitsch, editor, *SAFECOMP'96: 15th International Conference on Computer Safety, Reliability and Security*, page 243, Vienna, Austria, 1996. Springer.

[22] Chris George, Peter Haff, Klaus Havelund, Anne Haxthausen, Robert Milne, Claus Bendix Nielsen, Søren Prehn, and Kim Ritter Wagner. *The RAISE Specification Language.* The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.

[23] Chris George, Anne Haxthausen, Steven Hughes, Robert Milne, Søren Prehn, and Jan Storbank Pedersen. *The RAISE Method.* The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.

[24] C.W. George. *A Theory of Distributed Train Rescheduling.* In Marie-Claude Gaudel and Jim Woodcock, editors, *FME'96: Industrial Benefit and Advances in Formal Methods*, pages 499–517. Springer-Verlag, March 1996.

[25] G. Guiho and L.-F. Mejia. *Operational safety critical software methods in railways.* In Anon, editor, *IFIP Transactions A (Computer Science and Technology),*, pages 262–9. IFIP World Congress, Hamburg, Germany, 1984.

[26] K.M. Hansen. *Validation of a railway interlocking model.* In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME '94: Industrial Benefit of Formal Methods. Second International Symposium of Formal Methods Europe. Proceedings; Barcelona, Spain*, pages 582–601. Springer-Verlag, Berlin, Germany; Lecture Notes in Computer Science LNCS 873, 1994.

[27] K.M. Hansen. *Formalising Railway Interlocking Systems.* In *FME Rail Workshop 2* [56], ScanRail Consult (now Atkins), Signalling Assessment, Pilestræde 58/6, DK–1112 Copenhagen K, Denmark, 1998.

[28] A.E. Haxthausen and J. Peleska. *Formal Development and Verification of a Distributed Railway Control System.* In *FME Rail Workshop # 1* [55]; Techn. Univ. of Denmark, resp. Univ. of Bremen, Germany.

[29] C.A.R. Hoare. *Communicating Sequential Processes. Communications of the ACM*, 21(8), Aug. 1978.

[30] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[31] M. Ingleby. *Safety properties of a control network: Local and global reasoning in machine proof.* In *Proceedings of Real Time Systems*. Paris, January 1994, Huddersfield HD1 3DH, UK, 1998. School of Computing and Mathematics, University of Huddersfield.

[32] M. Ingleby. *A Galois theory of local reasoning in control systems with compositionality.* In *Proceedings of Mathematics of Dependable Systems*. Oxford UP (UK), 1995, Huddersfield HD1 3DH, UK, 1998. School of Computing and Mathematics, University of Huddersfield.

[33] M. Ingleby. *A predicate logic for harmonised interlocking functions.* In *FME Rail Workshop 1* [55], Huddersfield HD1 3DH, UK, 1998. School of Computing and Mathematics, University of Huddersfield.

[34] M. Ingleby and I. Mitchell. *Proving safety of a railway signalling system incorporating geographic data. SAFECOMP 1992: Safety of Computer Control Systems 1992*, pages 129–134, 1992.

[35] David Jackson. *Mechanical Verification of British Rail Interlocking.* In *FME Rail Workshop # 2* [56]. Praxis Critical Systems (UK).

[36] Michael A. Jackson. *Software Requirements & Specifications: a lexicon of practice, principles and prejudices.* ACM Press. Addison-Wesley Publishing Company, Wokingham, nr. Reading, England; E-mail: ipc@awpub.add-wes.co.uk, 1995. ISBN 0-201-87712-0; xiv + 228 pages.

[37] T. King. *Formalising British Rail's signalling rules. Lecture Notes in Computer Science*, 873, 1994.

[38] T. King. Formalising British Rail's signalling rules. In M. Bertran M. Naftalin, T. Denvir, editor, *FME'94: Industrial Benefit of Formal Methods*, pages 45–54. Springer-Verlag, October 1994.

[39] Peter Gorm Larsen. *A VDM-SL Specification of the Dwarf Signal Controller.* In *FME Rail Workshop # 3* [57], IFAD (DK).

[40] Peter Gorm Larsen. The KLV System in VDM–SL. In *FME Rail Workshop # 3* [57], IFAD (DK).

[41] Fernando Mejia. *Reverse engineering of a safety critical software with B.* In *FME Rail Workshop # 1* [55], GEC Alsthom (F).

[42] L.-F. Mejia. *Formalising existing safety-critical software.* In *FME Rail Workshop 2* [56]; 33, rue des Bateliers, F–93400 Saint–Ouen, France, 1998. Alstom Transport.

[43] Markus Montigel. *Formal representation of track topologies by double vertex graphs.* In *Proceedings of Railcomp 92 held in Washington DC, Computers in Railways 3*, volume 2: Technology. Computational Mechanics Publications, 1992.

[44] Markus Montigel. *Elemente eines Computergestützten Werkzeugs zur Entwicklung von Eisenbahnsicherungsanlagen mit Petri-Netzen.* Technical Report Schriftenreihe des IVT Nr. 92, IVT: Institut für Verkerhrsplanung, Transporttechnik, Strassen- und Eisenbahnbau, ETH, Zürich, Dezember 1992. In German.

[45] Markus Montigel. *Modellierung und Gewährleistung von Abhängigkeiten in Eisenbahnsicherungsanlagen.* PhD thesis, ETH: Swiss Federal Institute of Technology, ETH Honggerberg, CH-8093 Zürich, Swtizerland, June 1994.

[46] J. Peleska and M. Siegel. *From testing theory to test driver implementation.* In M.-C. Gaudel and J. Woodcock, editors, *FME '96: Industrial Benefit and Advances in Formal Methods. Third International Symposium of Formal Methods Europe. Proceedings; Oxford, UK*, pages 538–56. Springer-Verlag; Berlin, Germany, 1996.

[47] Jakob Lyng Petersen. *Mathematical Methods for validating Railway Interlocking Systems.* PhD thesis, Dept. of IT, Techn. Univ. of Denmark, Bldg. 344, DK–2800 Lyngby, February, November 1998.

[48] A.W. Roscoe. *Theory and Practice of Concurrency.* Prentice–Hall, 1997.

[49] Denis Sabatier. *The B Method in Railways.* In *FME Rail Workshop # 4* [58], Steria, France.

[50] A.C. Simpson, J.C.P. Woodcock, and J.W. Davies. *The Mechanical Verification of Solid State Interlocking Geographic Data.* In L. Groves and S. Reeves, editors, *Proceedings of Formal Methods Pacific*, pages 223–242, Wellington, New Zealand, 9–11 July 1997. Springer–Verlag.

[51] Andy Simpson. *CSP applied in Model-Checking for Interlocking Safety.* In *FME Rail Workshop # 2* [56]. Oxford University (UK).

[52] Jens U. Skakkebæk, Anders P. Ravn, Hans Rischel, and Zhou ChaoChen. *Specification of Embedded, Real-time Systems.* EuroMicro Workshop on Formal Methods for Real-time Systems, 1992 December 1991. The example: A railway road/rail crossing.

[53] J. Woodcock. *A CSP Model of the Alcatel Dwarf Case Study.* In Thierry Lecomte and Peter Gorm Larsen, editors, *FME Rail Workshop # 5* [59], Oxford Univ, Programming Research Group.

[54] Michael Meyer zu Hörste. *Modelling and Simulation of Train Control Systems with Petri Nets.* In *FME Rail Workshop # 4* [58]; Technische Universität Braunschweig (D).

FME Rail Workshop Proceedings:

[55] P.G. Larsen, editor. *FME Rail Workshop # 1*, volume 1 of *FME Rail Seminars*, Forskerparken, DK–6000 Odense, Denmark, 8–9 June 1998. FME: Formal Methods Europe, IFAD. ESSI Project 26538. Workshop venue: Breukelen, The Netherlands. Organised by Origin Nederland, a member of the Philips group of companies, P.O.Box 1444, NL–3430 BK Nieuwegein, The Netherlands.

[56] J.C.P. Woodcock, editor. *FME Rail Workshop # 2*, volume 2 of *FME Rail Seminars*, Keble Court, 26 Temple Street, Oxford OX4 1JS, UK, Telephone: +44 1865 728460, Telefax: +44 1865 201114, October 1998. ESSI Project 26538. Workshop venue: Canary Wharf, London Docklands, England. Organised by Formal Systems Ltd., Oxford. Hosted by London Underground.

[57] Maria Fahlén, editor. *FME Rail Workshop # 3*, volume 3 of *FME Rail Seminars*, Falun, Sweden, May 12–14 1998. FME: Formal Methods Europe, Banverket. ESSI Project 26538. Workshop venue: Stockholm, Sweden. Organised by Banverket (Swedish Rail's Infrastructure Division), Falun, Sweden.

[58] Markus Montigel, editor. *FME Rail Workshop # 4*, volume 4 of *FME Rail Seminars*, Herzogenburgerstr. 68, A-3100 St. Pölten, Austria, February 17–19 1999. FME: Formal Methods Europe, Fachhochschulstudiengang St. Pölten. ESSI Project 26538. Workshop venue: St. Pölten, Austria. Organised by Fachhochschulstudiengang St. Pölten and Alcatel, Austria.

[59] Thierry Lecomte and Peter Gorm Larsen, editors. *FME Rail Workshop # 5*, volume 5 of *FME Rail Seminars*. FME: Formal Methods Europe, Springer Verlag, September 20–24 1999. ESSI Project 26538. Workshop venue: Toulouse, France. Organised as part of FM'99: World Congress of Formal Methods.

TRAIN, A Project Proposal:

[60] Dines Bjørner. Train: The Railway Infrastructure — an R&D project proposal, 1998. This is a 125 page draft proposal for a thorough study and support tool (demo etc.) development of railway domains. We refer to `http://www.imm.dtu.dk/~db/train/train.html` and `http://www.imm.dtu.dk/~db/train/train.ps`. Pls. state your interest.