

Formal Software Techniques**in****Railway Systems****Dines Bjørner****Dept. of CS&T, Techn.Univ. of Denmark****IFAC, Braunschweig, 13 June 2000****A Motivation**

- We see the rôle of formal techniques (of software development in the context of transportation systems)
- from two composite viewpoints:
 - The sociological and socio-economic,
 - and the computing science and software engineering viewpoints.

Sociological Motivation

- The domain of railways was, for perhaps a hundred years, characterised by stable staff. Once hired into a railway company, the employee remained there for life.
- And by not so rapidly changing technology.
- Manually thrown switches were in primary use for more than a hundred years. Electro-mechanically operated switches for at least fifty years.
- Now staff “drifts” in and out of the domain; old loyalties have gone; new staff has to be continually retrained; and staff which is not kept vigilantly to the same operating routines, day after day, easily forget them.
- Rail and train technology changes more rapidly than the now much shorter employment-time of staff.
- And computers and communication have come to stay.

AbstractFail #1

- We discuss a rôle for the use of formal techniques in
 - understanding the **domain** of railways,
 - in expressing **requirements** to,
 - and in the **design** of trustworthy software for the backbone and program packages for railways.
- We exemplify such techniques as applied to the recording of our **domain** understanding and hints at their use in **requirements** and **design**.
- We motivate the use of formal techniques, and survey, ever so briefly, the state-of-affairs of such use.
- We finally put forward a proposal for and invite you to join such a proposed European Railway Software Technology Research & Development project:

TRaIn: The Railway Infrastructure**(Sociological Motivation — Contd.)**

- Do they, computers and communication, offer a relief from the problems of migrant staff and itinerant technologies ?
- Yes, but !
 - The knowledge that loyal, life-time staff learned, through training, was predicated on that staff getting that knowledge re-enforced, again and again, year in and year out, by “drill-work”.
 - That knowledge was usually presented in master class:
 - The experienced staff verbally taught the new staff.
- Computing has to take heed and embody that knowledge.
 - We need to codify that knowledge.
 - To write it down.
 - In precise ways.
 - In both humanly understandable form,
 - and in forms that can be manipulated by machine.
- The technology of former days was based on natural sciences:
 - Switches were mechanical, then electro-mechanical, and are now seemingly electronic
 - — or are they ?
 - The technology of today — computing —
 - is not based on natural sciences.
 - It is based on mathematics and on the “man-mad[e]ness” of infrastructure components: transportation, etc.

(Sociological Motivation — Contd.)

- It takes time for this view to prevail.
 - In this talk we wish to indicate a way in which the knowledge of for example railways can be “codified”:
 - Written down in a form that is guaranteed to last beyond the current fashions of object-oriented, UML-like models. Namely in good, reliable, old-fashioned mathematics.
 - Not the mathematics, however, of analysis (differential equations, integrals, statistics, etc.),
 - but a simpler, more readily accessible mathematics:
 - that of discrete mathematics, algebras and, notably, logic.
- In this talk we wish, briefly, to illuminate the changes that are inherent in computing.
- These are changes that will profoundly affect age old institutions like railways — and, for that matter, any transportation domain.

(Method & Methodology — Contd.)

- By a *method* we understand a number of
 - *principles*, deployed [here] by engineers,
 - to *select and apply*, a number of
 - *techniques*
 - and *tools*
 - in order
 - efficiently to *analyse* and
 - efficiently to *synthesize*
 - efficient [as here] software.
- No one method suffice for a full application.
- Some techniques of some methods can be formally based and for some of these formally based tools can be made available.
- The term ‘formal method’ should really only be understood as ‘formal technique cum tool’.
- Some methods span phases, stages and steps of development (viz. VDM and Raise).
- Other methods cover just a single aspect of a single step (viz. STeP),
- most uses of Petri Nets and Statecharts.

Engineering Motivation**Some Delineations****The Sciences and Engineering**

- *Computer science* is the study and knowledge of the phenomena that can exist inside computers: Data and processes.
- *Computing science* is the study and knowledge of how to construct those devices: Data and processes.
- *Software engineering* is, to us, the triptych of
 - engineering understandings of the **domain** of applications — void of any reference to computing (*ℰc.*),
 - of engineering **requirements** to software for computing applications in the support of operations of the domain,
 - and of engineering the **design** of that software.
- An engineer “walks” the bridge between science and technology:
 - Designs technology based on scientific insight; and
 - analyses technology to ascertain its science content.

Domain Engineering

Page # 2

- Domain models initially express the very basics of their domain:
- This paper will illustrate such an
 - (i) *intrinsic* — void of any reference to how that domain may be instrumented:
 - How it is done so through:
 - * (ii) *supporting technologies*,
 - * (iii) *management & organisation*,
 - * (iv) *rules & regulations*,
 - * (v) *human behaviour*, etc.
 - Domain models (then) go on to then capture (ii-iii-iv-v-...).

Requirements Engineering

Publ. # 3

- At least two other papers of these proceedings illustrate, from the perspectives of both domain and requirements, issues of, amongst others, modelling the *support technology* of the group interlocked control of rail switches.
- Requirements usually reflect a triple of concerns:
 - To secure (a) support of domain operations (the *domain requirements*),
 - (b) the *interface* between humans (and/or other interfacing quantities) and the hardware/software to be built (the *interface requirements*), and
 - (c) performance, dependability, maintainability, platform and other “ility” issues (the *machine requirements*).
- We shall not cover other than the intrinsic facet in the current paper.

Software design

Publ. # 4

- Software design finally deals with what is normally considered the domain of software engineering — which in this paper is thus argued to encompass also the more novel domain (and the, since some time, well-established requirements) engineering aspects.
- *Software architecture* design implements the domain requirements and parts or all of the interface requirements, and
- *program organisation* implements machine requirements.
- Further steps of software design focus on specific platform issues — and it is here that modularisation, the use of standard, today typically object oriented, packages (CORBA, etc.) enters the design process.

Descriptions

Publ. # 5

Throughout the development process “reams and reams” of documents are being constructed:

- Informal briefs,
 - scope & span synopses,
 - rough sketches,
 - analyses,
 - narrative and
 - terminologies,
- as well as formal documents:
 - formalisations of the narratives
 - and verification of properties.
- Validation is the act of securing, with stake-holders of the domain that domain and requirements descriptions to a highest, if not fullest, extent, satisfy human, hence informal, expectations.
- And throughout the development process, one or another construction method is being applied.

Promises of Better Engineering

- Domain engineering, as a separate development phase, addresses, typically the “*separations of concern*” problem of engineering.
- Formal techniques, with their use of and facilities for abstraction, have made it possible to capture the domains of indeed very large scale applications.
- This modern software engineering —
 - now being applied, systematically and rigorously, to major components,
 - * viz. transportation systems,
 - * healthcare systems,
 - * financial service industry, etc.,
 - of societal infrastructures —
- enable us to embark upon and tackle the development of very large scale software systems
- whose thousands of packages cover, in the end, highly interwoven, highly interrelated, communicating and synchronising domain phenomena.

(Promises of Better Engineering — Contd.)

- So far only software can do that:
- The “one step” applications of
 - automatic control,
 - or of operations analysis techniques,
 cover only well defined “package” parts, where computing is able to link “it all” together.
- Automatic control is, classically based on classical mathematical models, say in the form of differential equations, being first used to model what can be modelled by such mathematical tools.
- Similar for operations analysis.
- The techniques and tools of automatic control and operations analysis are indeed very profound.
- Computing science has only begun to scratch the surface of its profoundness.

(Promises of Better Engineering — Contd.)

- But computing can do what the others can't:
 - Link the various “packages” together.
- In doing so, computing does not, as of yet, guarantee that there is a formal, mathematical understanding of what that “linking” means.
- The problem is, however, being under intense study.
- In general it could be labelled the problem of codesign, but currently that label is used in a much more narrow sense:
 - The co-design of hardware & software to solve an isolated, “small” problem.
- Our co-design is one of systems engineering.

A Railway Model

- The purpose of this section is to illustrate —
- by a “top-of-an-iceberg” example —
- that we can indeed capture all, of what need be captured,
- in precise informal as well as formal terms.
- We will explain, in simple words, and in simple mathematics —
- where the formula fits, “hand-in-glove” the simple, informal text —
- what is meant (in a technology independent abstraction) by the statics
 - of a railway net:
 - by rail lines and railway stations,
 - by the rail units they consists of,
 - the connectors that “glue” units together, etc.;
- and by the dynamics:
 - of the states of signals and switches,
 - by open and closed routes,
 - by train movement along routes (even off the net!),
 - by time tables,
 - schedules and
 - traffic in general.

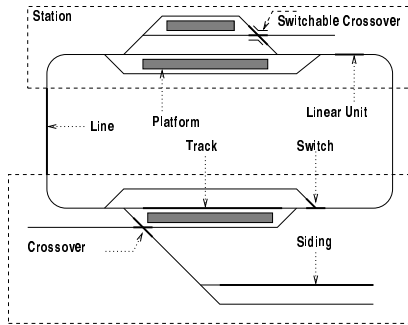
(A Railway Model — Contd.)

- The model given is claimed to describe the domain,
 - independent of any computing application.
- As shown, elsewhere we can then base requirements on domain models, like the one shown next,
- and from these requirements, in a trustworthy manner, develop dependable software.
- After this section we shall expand the view and postulate further applications.
- We divide the presentation into two parts:
 - a hierarchical (“topdown”) first part
 - followed by a compositional (“bottom-up”) second part.

(Hierarchical Presentation — Contd.)

- We focus on the railway net perspective of railway systems.
- We designate such components of the rail net which can be physically demonstrated,
 - but we abstract from a number of physical attributes —
 - they can always be simply “added” later on — and focus on the transport logics.
- Pls. inspect figure 1.

Figure 1: A “Model” Railway Net !

**(Hierarchical Presentation — Contd.)**

12. A unit is either linear, or a switch, or a simple crossover, or a switchable crossover, etc.
13. A unit has one or more connectors.²
14. For every connector there are at most two units which have that connector in common.
15. Every line of a net is connected to exactly two, distinct stations of the net.
16. A linear sequence of units is a non-cyclic sequence of linear units such that neighbouring units share connectors.

²A linear unit has two distinct connectors, a switch has three distinct connectors, crossovers have four distinct connectors.

(Hierarchical Presentation — Contd.)

Our natural, professional railway language description proceeds as follows:¹

1. A railway net consists of lines and two or more stations.
2. A railway net consists of units.
3. A line is a linear sequence of one or more linear units.
4. The units of a line must be units of the net of the line.
5. A station is a set of units.
6. The units of a station must be units of the net of the station.
7. No two distinct lines and/or stations of a net share units.
8. A station consists of one or more tracks.
9. A track is a linear sequence of one or more linear units.
10. No two distinct tracks share units.
11. The units of a track must be units of the station (of that track).

¹We enumerate the sentences for reference.

(Hierarchical Presentation — Contd.)

A corresponding, representationally abstract formal specification — please read it carefully, line-by-line, is:

type

N, L, S, Tr, U, C

value

1. obs_Ls: N → L-set
 1. obs_Ss: N → S-set
 2. obs_Us: N → U-set
 3. obs_Us: L → U-set
 5. obs_Us: S → U-set
 8. obs_TrS: S → Tr-set
 12. is_Linear: U → **Bool**
 12. is_Switch: U → **Bool**
 12. is_Simple_Crossover: U → **Bool**
 12. is_Switchable_Crossover: U → **Bool**
 13. obs-Cs: U → C-set
 16. lin_seq: U-set → **Bool**
- lin_seq(q) ≡
- let** us = obs_Us(us) **in**
- ∀ i:U · u ∈ us ⇒ is_Linear(u) ∧
- ∃ q:U* · len q = card us ∧ elems q = us ∧
- ∀ i:Nat · {i,i+1} ⊆ inds q ⇒ ∃ c:C ·
- obs-Cs(q(i)) ∩ obs-Cs(q(i+1)) = {c} ∧
- len q > 1 ⇒ obs-Cs(q(i)) ∩ obs-Cs(q(len q)) = {}
- end**

(Hierarchical Presentation — Contd.)**axiom**

1. $\forall n:N \cdot \mathbf{card} \text{ obs_Ss}(n) \geq 2,$
3. $\forall n:N, l:L \cdot l \in \text{obs_Ls}(n) \Rightarrow \text{lin_seq}(l)$
7. $\forall n:N, l,l':L \cdot$
 $\{l,l'\} \subseteq \text{obs_Ls}(n) \wedge l \neq l'$
 $\Rightarrow \text{obs_Us}(l) \cap \text{obs_Us}(l') = \{\}$
7. $\forall n:N, l:L, s:S \cdot$
 $l \in \text{obs_Ls}(n) \wedge s \in \text{obs_Ss}(n)$
 $\Rightarrow \text{obs_Us}(l) \cap \text{obs_Us}(s) = \{\}$
7. $\forall n:N, s,s':S \cdot$
 $\{s,s'\} \subseteq \text{obs_Ss}(n) \wedge s \neq s'$
 $\Rightarrow \text{obs_Us}(s) \cap \text{obs_Us}(s') = \{\}$
8. $\forall s:S \cdot \mathbf{card} \text{ obs_Trs}(s) \geq 1$
9. $\forall n:N, s:S, t:T \cdot$
 $s \in \text{obs_Ss}(n) \wedge t \in \text{obs_Trs}(s) \Rightarrow \text{lin_seq}(t)$

Configurational Presentation

- We wish, now, to build up to the dynamics of the net:

- modelling, but abstractly,
- signals and their states,
- the state of switches
- as well as the intended use-state of rails:

17. A path, $p : P$, is a pair of connectors, (c, c') , of some unit.³
18. A state, $\sigma : \Sigma$, of a unit is the set of all open paths of that unit (at the time observed).⁴
19. A unit may, over its operational life, attain any of a (possibly small) number of different states ω, Ω .

³A path of a unit designate that a train may move across the unit in the direction from c to c' . We say that the unit is open in the direction of the path.

⁴The state may be empty: the unit is closed.

(Hierarchical Presentation — Contd.)

10. $\forall n:N, s:S, t,t':T \cdot$
 $s \in \text{obs_Ss}(n) \wedge \{t,t'\} \subseteq \text{obs_Trs}(s) \wedge t \neq t'$
 $\Rightarrow \text{obs_Us}(t) \cap \text{obs_Us}(t') = \{\}$
 14. $\forall n:N \cdot \forall c:C \cdot$
 $c \in \cup \{ \text{obs_Cs}(u) \mid u:U \cdot u \in \text{obs_Us}(n) \}$
 $\Rightarrow \mathbf{card} \{ u \mid u:U \cdot u \in \text{obs_Us}(n) \wedge c \in \text{obs_Cs}(u) \} \leq 2$
 15. $\forall n:N, l:L \cdot l \in \text{obs_Ls}(n) \Rightarrow$
 $\exists s,s':S \cdot \{s,s'\} \subseteq \text{obs_Ss}(n) \wedge s \neq s' \Rightarrow$
 $\mathbf{let} \text{ sus} = \text{obs_Us}(s), \text{ sus}' = \text{obs_Us}(s'), \text{ lus} = \text{obs_Us}(l) \mathbf{in}$
 $\exists u:U \cdot u \in \text{sus}, u':U \cdot u' \in \text{sus}', u'', u''':U \cdot \{u'', u'''\} \subseteq \text{lus} \cdot$
 $\mathbf{let} \text{ scs} = \text{obs_Cs}(u), \text{ scs}' = \text{obs_Cs}(u'),$
 $\text{ lcs} = \text{obs_Cs}(u''), \text{ lcs}' = \text{obs_Cs}(u''') \mathbf{in}$
 $\exists ! c, c':C \cdot c \neq c' \wedge \text{scs} \cap \text{lcs} = \{c\} \wedge \text{scs}' \cap \text{lcs}' = \{c'\}$
- end end**

(Configurational Presentation — Contd.)

20. A route is a sequence of pairs of units and paths —
21. such that the path of a unit/path pair is a possible path of some state of the unit, and such that “neighbouring” connectors are identical.
22. An open route is a route such that all its paths are open.
23. A train is modelled as a route.
24. Train movement is modelled as a discrete function (map) from time to routes such that for any two adjacent times the two corresponding routes differ by at most one of the following: a unit path pair has been deleted from (one or another end) of the open routes, or (similarly) added, or both, or no changes — a total of seven possibilities (i–vii).

type

- 17 $P = C \times C$
- 18 $\Sigma = \mathbf{P\text{-set}}$
- 19 $\Omega = \Sigma\text{-set}$
- 20 $R' = (U \times P)^*$
- 21 $R = \{ | r:R' \cdot \text{wf_R}(r) \}$
- 23 $\text{Trn} = R$
- 24 $\text{Mov} = T \xrightarrow{\text{mv}} \text{Trn}$

value

- 18 $\text{obs_}\Sigma: U \rightarrow \Sigma$
- 19 $\text{obs_}\Omega: U \rightarrow \Omega$

(Configurational Presentation — Contd.)

- 21 $\text{wf_R}: R' \rightarrow \mathbf{Bool}$
 $\text{wf_R}(r) \equiv$
 $\forall i: \mathbf{Nat} \cdot i \in \mathbf{inds} \ r \ \mathbf{let} \ (u, (c, c')) = r(i) \ \mathbf{in}$
 $(c, c') \in \cup \text{obs_}\Omega(u) \wedge i+1 \in \mathbf{inds} \ r \Rightarrow$
 $\mathbf{let} \ (_ , (c'', _)) = r(i+1) \ \mathbf{in} \ c' = c'' \ \mathbf{end} \ \mathbf{end}$
- 22 $\text{open_R}: R \rightarrow \mathbf{Bool}$
 $\text{open_R}(r) \equiv$
 $\forall (u, p): U \times P \cdot (u, p) \in \mathbf{elems} \ r \wedge p \in \text{obs_}\Sigma(u)$
- 24 $\text{wf_Mov}: \text{Mov} \rightarrow \mathbf{Bool}$
 $\text{wf_Mov}(m) \equiv \mathbf{card} \ \mathbf{dom} \ m \geq 2 \wedge$
 $\forall t, t': T \cdot t, t' \in \mathbf{dom} \ m \wedge t < t'$
 $\wedge \sim \exists t'': T \cdot t'' \in \mathbf{dom} \ m \wedge t < t'' < t' \Rightarrow$
 $\mathbf{let} \ (r, r') = (m(t), m(t')) \ \mathbf{in} \ \text{clauses (i) – (vii)} \ \mathbf{end}$

We leave it as an exercise to define clauses (i)–(vii) above.

(Other Aspects and Discussion)

- Models, informal and formal, such as shown above, can be developed for “the whole” of railway systems — indeed for any (other) transportation infrastructure component: Air traffic (with airports and airlines), shipping (for example emphasizing container logistics), and metropolitan transport (the interplay between taxis, buses, metro, regional trains, etc.).
- For railways we have, in this manner, developed, from the basis shown above, sub-models that include the following aspects:
- **Time tables:** T_n for train names, T stands (or could, eg. stand for modulo, say a week) times, and S_n for station names. TT stands for time tables: Every applicable train name associates to a journey, a list of station visits — with their arrival and departure times:

type T_n, T, S_n
 $TT = T_n \xrightarrow{m} (T \times S_n \times T)^*$

(Other Aspects and Discussion — Contd.)

- **Traffic:** Given that a train can be associated with not only its name but also its route position on the rails, TF stands for traffic: A continuous function from time, now absolute, with no modulo necessity, to usually stable nets and certainly monotonically changing train (TR) positions:

type TR
 $TF = T \rightarrow (N \times (TR \xrightarrow{m} R))$

value
 $\text{obs}T_n: TR \rightarrow T_n$
 $\text{obs}R_s: N \rightarrow R\text{-infset}$

axiom
 $\forall \text{tf}: TF, t: T \cdot$
 $t \in \mathbf{dom} \ \text{tf} \wedge \mathbf{let} \ (n, \text{tps}) = \text{tf}(t) \ \mathbf{in}$
 $\forall \text{tr}: TR \cdot \text{tr} \in \mathbf{dom} \ \text{tps} \Rightarrow$
 $\text{tps}(\text{tr}) \in \text{obs}R_s(n) \ \mathbf{end}$

- **Train Plans:** The train engine man is given, during briefing, before the train is despatched, a train plan. It gives a discrete, ie. not a continuous, description of where the train is expected to be at certain times, at which sequence, R of units. For all trains we get:

type
 $TP = T \xrightarrow{m} (N \times (T_n \xrightarrow{m} R))$

Train plans are discretisations of traffics — and, as is traffic, is modulo the net.

(Other Aspects and Discussion — Contd.)

- **Scheduling:** Given a net and a time table we can speak, as the meaning of the net and the time table, of all the traffics that satisfies the net and the time table:

value
 Meaning: $N \times TT \rightarrow TF\text{-infset}$

We can in particular devise a special planning function, **scheduling**, which decides upon a suitable set of train plans:

scheduling: $N \times TT \rightarrow TP$

Given a traffic and a trains plan we can now speak of monitoring whether trains in the traffic are on-schedule:

value
 ontime: $N \times TP \times TF \rightarrow \mathbf{Bool}$

Etcetera.

(Other Aspects and Discussion — Contd.)

- **Shunting and Marshalling:** We can define what is meant by shunting and by marshalling yards, marshalling plans, and marshalling.
- **Passenger Service:** We can define what is meant by passenger services such as travel plan inquiry, ticket and seat reservation, ticketing, etc.
- **Logistics:** With nets, rolling stock control, passenger statistics and old time tables, we can speak of logistics planning: Constructing new time tables, etc.
- **Freight Services:** We can define what is meant by freight trains, by the logistics of planning and executing freight service: Reservation, loading, tracing and unloading freight, etc.
- **Rolling Stock Monitoring & Control:** And so on: We can model the rolling stock, its waiting, at sidings, to participate in passenger and/or freight train traffic; the gathering of such rolling stock into operable trains, etc.
- **Net Development:** You noticed that the net, N , was a parameter in traffic: To model that the net regularly undergoes maintenance and development. Old rails taken out of service, new lines and stations being put into service.

Challenges and Claims**An Infrastructure of Software**

The domain of railways is a micro-cosm:

- It represents a huge variety of possibilities for applications of software
- and all these applications “link together”:
- Are tightly related.
- Results of computations by means of some software packages has import as input data to other packages — yet little, if any effort, is made to vet that data:
 - To insure that an underlying model, not just of information (ie. data), but of their semantics, ie.
 - * of the actions to be taken on data and the interaction sequences, ie. the processes in which these data are shared with other processes —
 - that such underlying models indeed do exist and “bind” the information and its use, the data and its processing.
 - As it is today there is hardly any guarantee that statistical data gathered from one application, say ‘traffic and passenger monitoring’, is indeed correctly used by ‘timetabling’ software.

Other Formal Work

- There is, by now, a very rich literature on applying formal techniques to problems of railways.
- An EU sponsored network, **FME Rail** carried out by **FME: Formal Methods Europe** under the leadership of Dr. Peter Gorm Larsen, <http://www.ifad.dk/Projects/fmerail>, held five workshops on the topic across Europe in the period 1998–1999:
 - In The Netherlands,
 - England,
 - Austria,
 - Sweden
 - and France
 gathering a total of more than 300 railway IT people.
- A very extensive, partially annotated bibliography and a light assessment of international research and actual commercial use of formal techniques can be inspected:
 - <http://www.it.dtu.dk/~db/rail/bib.ps>
- So far the main emphasis — in the use of formal techniques — has been safety critical aspects of controlling the switches of railway stations, in modern technology known as *interlocking*.

(An Infrastructure of Software — Contd.)

- The realm of automatic control, the subject of this conference, interacts strongly with many facets of
 - statistics gathering,
 - time table planning,
 - train scheduling,
 - train despatch,
 - resource allocation and
 - scheduling, etc. —
 in a cycle back to the more-or-less automatic control of train movements.
- The field of **computing science** provides design tools, techniques, principles and methods that allow classical **automatic control** engineering designs to “link up” to more modern **operations analysis** (graph-theoretic and combinatorial resource optimisation).
- The computability based state, event and process modelling techniques of computing science provides means lacking in control theory and theories of optimisation.
- The understanding of this truth takes time to propagate.

A Socio-Economic Outlook

- This paper was invited with basically the given title.
- It therefore behooves us — based on many years of both scientific and engineering, theoretical as well as quite some substantial practical experience —
 - to make some statements as to the rôle of formal techniques in software development, and in particular in the development of software for the support of operations in the application domain of transportation and traffic.
- For sociological reasons it seems to take some time before formal techniques enter the domain of software for general software for common railway cum transportation support.
- In the area of software for safety critical applications there seems to be a strong trend towards using the formal techniques of this talk.
- In the area of general software there is hardly any “movement”.

(A Socio-Economic Outlook — Contd.)

- Meanwhile a few have to “fight it out”.
- It will be interesting to see whether the inevitable transition to trustworthy software conception, from domains via requirements to software design, will primarily be due to the candidates from universities wanting to have it no other way, or due to insurance companies demanding claimed “proofs of correctness” ?
- The fact is simple enough: There are enough universities around Europe — and Europe seems to be a leader in the research and use of formal techniques — which produce a reasonably quantity of very highly qualified candidates.
- There are enough problems that ought be “formally” tackled. But procurers either do not demand the “proven” quality, or the providers, the suppliers are not capable, or both !

(A Socio-Economic Outlook — Contd.)

- The problem, there, seems to be that most managers of IT departments of the transportation industry: providers as well as users, are not software professionals.
- In other words: They really are neither computing scientifically cum software technologically mature nor engineeringly responsible.
- The current lot of programmers possess the enviable position of being considered indispensable — managed by non-professionals — where, as a matter of cool fact, they ought be laid off: The programmers and their management !
- A long period therefore confronts us, in which the industry ever so slowly turns around and becomes mature and responsible.

Conclusion**Formal Techniques and Tools**

- In this paper we have used the Raise Specification Language, RSL, and thus implied use of the Raise software development method.
- There are many complementary as well as “competing” approaches:
 - B,
 - VDM-SL,
 - Z,
 - and others.
- It seems, to this author, that B is currently being very successfully applied in “real-life”, commercial projects.

A Reinforcement

- In a rather personal style — but that is fortunately what invited talk from long experienced people are all about, ie. to be expected — I have related some 25 years of hard-won experience:
 - For the fascinating field of transportation, a highly structured and disciplined field, the combination of automatic control with operations analysis as linked together by computing science, that is:
 - The well-managed collaboration of control engineers and operations ‘researchers’ with software engineers, brings exciting promises.
- The former disciplines: Automatic control and operations analysis, has, since long, adhered to strong, formal foundations, linking, in the case of automatic control, the natural sciences with mathematical models, and, in the case of operations analysis, the human decision processes with computing.
- The new corner in the triangle: That of computing science, is now no longer a trivial means of “hacking” code. Computing science, with domain engineering “at the top”, is offering radically more fundamental, broad and deep, all-encompassing models of “all” of railways.
- That is the challenge: For owners, operators and procurers of the transportation domains, to understand and professionally exploit this.

TRaIn: A Pan-European — and beyond — R&D Project Proposal

- I think it is high time that we gather forces:
 - Classical automatic control,
 - operations research, and
 - modern computing science,
- to embark on
 - exploratory,
 - experimental, as well as
 - committed
- R&D wrt. extensive domain models for railway transport systems:
 - The net and its signalling;
 - trains, their despatch, monitoring & control;
 - rolling stock monitoring & control:
 - * Allocation & scheduling, and
 - * shunting and marshalling;
 - time-tabling, scheduling, and re-scheduling;
 - passenger handling: Reservations, ticketing, etc.;
 - freight handling: loading, unloading, tracing, etc.;
- etc.

(TRaIn: A Pan-European — and beyond — R&D Project Proposal — Contd.)

- Only through joint R&D can we achieve a much needed clarification and understanding of
- strategic, tactical, and operational planning and operations:
 - (A) Strategics — resource upgrading and downsizing:
 - * (i) New transport services: passenger, freight;
 - * (ii) new lines and stations;
 - * (iii) or the suspension of the services;
 - * (iv) financial management, *ℰc.*
 - (B) Tactics:
 - * (i) time-tabling,
 - * (ii) fee structures,
 - * (iii) spatial resource allocation & scheduling, *ℰc.*
 - (C) Operations — task allocation & scheduling:
 - * (i) Planning:
 - (a) Traffic scheduling and rescheduling,
 - (b) resource deployment and re-deployment, *ℰc.*
 - * (ii) Operations:
 - (a) Traffic monitoring & control,
 - (b) passenger and freight monitoring & control, *ℰc.*
 - * (iii) Automation:
 - (a) ATP: Automatic train protection,
 - (b) ATC: Automatic train control, *ℰc.*
- *ℰc.*: Please join this project proposal !
- <http://www.it.dtu.dk/db/train/train.html>

Acknowledgements

- The author is grateful to Prof. Schnieder, Univ. of Braunschweig, for promoting this paper.
- Over the years the author has had the great joy of working with Messrs. Søren Prehn, Chris W. George and others on the specific (railway) as well as general (methodology) subjects covered and exemplified by this paper.
- My thanks goes to them in no small measure.
- The origins of the model of this paper seems to derive from work done by Prehn and George.
- Other models are more concrete — and would not have given us the degree of freedom and the desired generality.
- Finally I wish to acknowledge, with great pleasure, the leadership of Dr. Peter Gorm Larsen of the EU sponsored FME Rail project.
- Contact me: db@it.dtu.dk, URL: www.it.dtu.dk/~db