TOWARDS A FORMAL MODEL OF **CyberRail**

Dines Bjørner^{1,2}

(1) *Department of Computer Science, School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543*

E-Mail: bjorner@comp.nus.edu.sg, URL: <http://www.comp.nus.edu.sg/cs/>

Peter Chiang², Morten S.T. Jacobsen², Jens Kielsgaard Hansen², and Michael P. Madsen²

(2) *Section of Computer Science and Engineering, Institute of Informatics and Computer Engineering, Technical University of Denmark, DK-2800 Kgs.Lyngby, Denmark*

E-Mail: {s001402,s001693,s001842,s001700}@student.dtu.dk

Martin Penicka^{1,3*}

(3) *Czech Technical University, Faculty of Transportation Sciences, Department of Applied Mathematics, Na Florenci 25, 110 00 Praha 1, Czech Republic*

E-Mail: penicka@fd.cvut.cz

Abstract Based on a number of reports and publications, primarily by Takahiko Ogino [14], [15], [16] (in these proceedings), and [17], on the emerging concept of **CyberRail**, we attempt to show what a formal domain model of **CyberRail** might look like, and what benefits one might derive from establishing and having such a formal model.

Keywords: **CyberRail**, Formalisation, Domain Model, RAISE, RSL, CSP, Non-determinism, Abstract Data Types

1. Background

The background for the work reported in this extended abstract is threefold: (i) Many years of actual formal specification as well as research into how to engineer such formal specifications, by the first author, of domains, including the railway domain [8] [7] [9] [10] [1] [6] [3] [2] [5] —using abstraction and

*MPs work was supported, in part, by the EU 5th IST FP Programme action: CoLogNET: The Computational Logic Network of Excellence.

modelling principles and techniques extensively covered in three forthcoming software engineering textbooks [4]. (ii) A term project with four MSc students. And (iii) Some fascination as whether one could formalise an essence of the novel ideas of **CyberRail**. We strongly believe that we can capture one crucial essence of **CyberRail** — such as this paper will show.

The formalisation of **CyberRail** is expressed in the RAISE [12] Specification Language, RSL [11]. RAISE stands for Rigorous Approach to Industrial Software Engineering. In the current abstract model we especially make use of RSL's parallel process modeling capability. It builds on, i.e., borrows from Tony Hoare's algebraic process concept of Communicating Sequential Processes, CSP [13].

2. A Rough Sketch Formal Model

2.1 An Overall **CyberRail** System

CyberRail consists of an index set of traveller behaviours and one cyber behaviour “running” in parallel. Each traveller behaviour is uniquely identified, $p:Tx$. Traveller behaviours communicate with the cyber behaviour. We abstract the communication medium as an indexed set of channels, $ct[p]$, from the cyber behaviour to each individual traveller behaviour, and $tc[p]$, from traveller behaviours to the cyber behaviour. Messages over channels are of respective types, CT and TC. The cyber behaviour starts in an initial state ω_i , and each traveller behaviour, p , starts in some initial state $m\sigma_i(p)$.

type

$Tx, \Sigma, \Omega, CT, TC$

$M\Sigma = Tx \multimap \Sigma$

channel

$\{ct[p]:CT, tc[p]:TC | p:Tx\}, cr:CR, rc:RC$

value

$m\sigma_i:M\Sigma, \omega_i:\Omega$

cyberrail_system: **Unit** $\rightarrow$ **Unit**

cyberrail_system() $\equiv \parallel \{ \text{traveller}(p)(m\sigma_i(p)) \mid p:Tx \} \parallel \text{cyber}(\omega)$

cyber: $\Omega \rightarrow \text{in } \{tc[p] | p:Tx\}, cr \text{ out } \{ct[p] | p:Tx\}, rc \text{ Unit}$

cyber(ω) $\equiv$

cyber_as_server(ω) $\parallel$ **cyber_as_proactive**(ω) $\parallel$ **cyber_as_co_director**(ω)

traveller: $p:Tx \rightarrow \Sigma \rightarrow \text{in } ct[p] \text{ out } tc[p] \text{ Unit}$

traveller(p)(σ) $\equiv \text{active_traveller}(p)(\sigma) \parallel \text{passive_traveller}(p)(\sigma)$

The cyber behaviour either acts as a server: Ready to engage in communication input from any traveller behaviour; or the cyber behaviour acts pro-actively: Ready to engage in performing output to one, or some traveller behaviours; or the cyber behaviour acts in consort with the “rest” of the transportation market (including rail infrastructure owners, train operators, etc.), in improving and changing services, and in otherwise responding to unforeseen circumstances of that market.

Similarly any traveller behaviour acts as a client: Ready to engage in performing output to the cyber behaviour; or its acts passively: Ready to accept input from the cyber behaviour.

2.2 Travellers

2.2.1 Active Travellers. Active traveller behaviours alternate internally non-deterministically, ie., at their own choice, between *start (travel) planning* *st_pl*, *select (among suggested) travel plan(s)* *se_pl*, *change (travel) planning* *ch_pl*, *begin travel* *be_tr*, *board train* *bo_tr*, *leave train* *lv_tr*, *ignore train* *ig_tr*, *cancel travel* *ca_tr*, *seeking guidance* *se_gu*, *notifying cyber* *no_cy*, *entertainment* *ent*, *deposit resource* *de_re* (park car, ...), *claim resource* *cl_re* (retrieve car, ...), *get resource* *ge_re* (rent a car, ...), *return resource* *re_re* (return rent-car, ...), *going to restaurant* *rest* (or other), *change travel* *ch_tr*, *interrupt travel* *in_tr*, *resume travel* *re_tr*, *leave train* *le_tr*, *end travel* *en_tr*, and many other choices. Each of these normally entail an output communication to the cyber behaviour, and for those we can assume immediate response from the cyber behaviour, where applicable.

value

active_traveller: $p:Tx \rightarrow \Sigma \rightarrow \text{out } tc[p] \text{ in } ct[p] \text{ Unit}$

active_traveller(p)(σ) $\equiv$

let choice = *st_pl* $\sqcap$ *ac_pl* $\sqcap$ *ch_pl* $\sqcap$ *en_tr* $\sqcap$... $\sqcap$ *le_tr* $\sqcap$ *te_tr* **in**

let $\sigma' = \text{case choice of}$

st_pl $\rightarrow$ *start_planning*(p)(σ),
se_pl $\rightarrow$ *select_travel_plan*(p)(σ),
ch_pl $\rightarrow$ *change_travel_plan*(p)(σ),
be_tr $\rightarrow$ *begin_travel*(p)(σ),
bo_tr $\rightarrow$ *board_train*(p)(σ),
... $\rightarrow$..,
le_tr $\rightarrow$ *leave_train*(p)(σ),
en_tr $\rightarrow$ *end_travel*(p)(σ),
... $\rightarrow$..

end in

traveller(p)(σ') **end end**

```

start_planning:  $p:Tx \rightarrow \Sigma \rightarrow \text{out } tc[p] \text{ in } ct[p] \Sigma$ 
start_planning( $p$ )( $\sigma$ )  $\equiv$ 
  let ( $\sigma'$ , $plan$ ) = magic_plan( $\sigma$ ) in
     $tc[p]!plan$ ;
    let  $sps = ct[p]?$  in update $\Sigma((plan,sps))(\sigma')$  end end
...
update $\Sigma$ :  $Update \rightarrow \Sigma \rightarrow \Sigma$ 
type
   $Update == mkInPlRes(ip:InitialPlan,ps:Plan-set) \mid \dots$ 

```

2.2.2 Passive Travellers. When not engaging actively with the cyber behaviour, traveller behaviours are ready to accept any cyber initiated action. The traveller behaviour basically “assimilates” messages received from cyber — and may make use of these in future.

```

value
  passive_traveller:  $p:Tx \rightarrow \Sigma \rightarrow \text{in } ct[p] \text{ out } tc[p] \text{ Unit}$ 
  passive_traveller( $p$ )( $\sigma$ )  $\equiv$  let  $res = ct[p]?$  in update $\Sigma(res)(\sigma)$  end

```

2.2.3 Active Traveller Actions. The *active_traveller* behaviour performs either of the internally non-deterministically chosen actions: *start_planning*, *select_travel_plan*, *change_travel_plan*, *begin_travel*, *board_train*, ..., *leave_train*, or *end_travel*. They make use only of the “sum total state” (σ) that that traveller behaviour “is in”. Each such action basically communicates either of a number of plans (or parts thereof, here simplified into plans). Let us summarise:

```

type
  Plan
   $Request = Initial\_Plan \mid Selected\_Plan \mid Change\_Plan \mid Begin\_Travel$ 
     $\mid Board\_Train \mid \dots \mid Leave\_Train \mid End\_Travel \mid \dots$ 
   $Initial\_Plan == mkIniPl(pl:Plan)$ 
   $Selected\_Plan == mkSelPl(pl:Plan)$ 
   $Change\_Plan == mkChgPl(pl:Plan)$ 
   $Begin\_Travel == mkBTrav(pl:Plan)$ 
   $Board\_Train == mkBTrai(pl:Plan)$ 
  ...
   $Leave\_Train == mkLeTr(pl:Plan)$ 
   $End\_Travel == mkEnTr(pl:Plan)$ 
value
   $\forall f: p:Tx \rightarrow \Sigma \rightarrow \text{out } tc[p] \Sigma$ 

```

magic_f: $\Sigma \rightarrow \Sigma \times \text{Request}$

$f(p)(\sigma) \equiv \text{let } (\sigma', \text{req}) = \text{magic_f}(\sigma) \text{ in } \text{tc}[p]! \text{req}; \sigma' \text{ end}$

The `magic_functions` access and changes the state while otherwise yielding some `request`. They engage in no events with other than the traveller state. There are the possibility of literally “zillions” such functions, all fitted into the above sketched traveller behaviour.

2.3 cyber

2.3.1 cyber as Server. `cyber` is at any moment ready to engage in actions with any traveller behaviour. `cyber` is assumed here to respond immediately to “any and such”.

value

`cyber_rail_as_server:` $\Omega \rightarrow \text{in } \{\text{tc}[p] | p:\text{Tx}\} \text{ out } \{\text{ct}[p] | p:\text{Tx}\} \text{ Unit}$

`cyber_rail_as_server`(ω) $\equiv$

$\square \{ \text{let req} = \text{tc}[p]? \text{ in cyber}(\text{serve_traveller}(p, \text{req})(\omega)) \text{ end } | p:\text{Tx} \}$

`serve_traveller:` $p:\text{Tx} \times \text{Req} \rightarrow \Omega \rightarrow \text{in } \{\text{tc}[p] | p:\text{Tx}\} \text{ out } \{\text{ct}[p] | p:\text{Tx}\} \Omega$

`serve_traveller`(p, req)(ω) $\equiv$

case req of

`mkIniPl`(pl) $\rightarrow$

$\text{let } (\omega', \text{pls}) = \text{sugg_pls}(p, pl)(\omega) \text{ in } \text{ct}[p]! \text{pls}; \text{cyberrail}(\omega') \text{ end}$

`mkSelPl`(pl) $\rightarrow$

$\text{let } (\omega', \text{res}) = \text{res_pl}(p, pl)(\omega) \text{ in } \text{ct}[p]! \text{book}; \text{cyberrail}(\omega') \text{ end}$

`mkChgPl`(pl) $\rightarrow$

$\text{let } (\omega', pl') = \text{chg_pl}(p, pl)(\omega) \text{ in } \text{ct}[p]! pl'; \text{cyberrail}(\omega') \text{ end}$

`mkBTrav`(pl) $\rightarrow \dots$

`mkBTrai`(pl) $\rightarrow \dots$

$\dots$

`mkLeTr`(pl) $\rightarrow \dots$

`mkEnTr`(pl) $\rightarrow \dots$

end

2.3.2 cyber as Pro-Active. `cyber`, on its own volition, may, typically based on its accumulated knowledge of traveller behaviours, engage in sending messages of one kind or another to selected groups of travellers. Section 2.3.5 rough sketch-formalises one of these.

type

`CR_act` $\equiv \text{gu_tr} \mid \text{no_tr} \mid \text{co_tr} \mid \text{wa_tr} \mid \dots$

value

```

cyber_as_proactive:  $\Omega \rightarrow \text{out } \{\text{ct}[p] \mid p:\text{Tx}\} \text{ Unit}$ 
cyber_as_proactive( $\omega$ )  $\equiv$ 
  let cho = gu_tr  $\sqcap$  no_tr  $\sqcap$  co_tr  $\sqcap$  wa_tr  $\sqcap$  ... in
  let  $\omega' = \text{case cho of gu\_tr} \rightarrow \text{guide\_traveller}(\omega),$ 
    no_tr  $\rightarrow \text{notify\_traveller}(\omega),$ 
    co_tr  $\rightarrow \text{commercial\_to\_travellers}(\omega),$ 
    wa_tr  $\rightarrow \text{warn\_travellers}(\omega),$ 
    ...  $\rightarrow \dots$  end in
  cyber( $\omega'$ ) end end

```

2.3.3 cyber as Co-Director. We do not specify this behaviour. It concerns the actions that cyber takes together with the “rest” of the transportation market. One could mention input from cyber_as_co_director to the train operators as to new traveller preferences, profiles, etc., and output from the rail (ie., net) infrastructure owners or train operators to cyber_as_co_director as to net repairs or train shortages, etc. The decomposition of CyberRail into cyber and the “rest”, may — to some — be artificial, namely in countries where there is no effective privatisation and split-up into infrastructure owners and train operators. But it is a decomposition which is relevant, structurally, in any case.

2.3.4 cyber Server Actions. We sketch:

value

```

sugg_plans:  $p:\text{Tx} \times \text{Plan} \rightarrow \Omega \rightarrow \Omega \times \text{Plan-set}$ 
res_pl:  $p:\text{Tx} \times \text{Plan} \rightarrow \Omega \rightarrow \Omega \times \text{Plan}$ 
chg_pl:  $p:\text{Tx} \times \text{Plan} \rightarrow \Omega \rightarrow \Omega \times \text{Plan}$ 
...

```

There are many other such traveller instigated cyber actions.

2.3.5 Pro-Active cyber Actions. We rough sketch just a single of the possible “dozens” of cyber initiated actions versus the travellers.

value

```

guide_traveller:  $\Omega \rightarrow \text{out } \{\text{ct}[p] \mid p:\text{Tx}\} \Omega$ 
guide_traveller( $\omega$ )  $\equiv$ 
  let ( $\omega', (\text{ps}, \text{guide})$ ) = any_guide( $\omega$ ) in broadcast(ps, guide) ;  $\omega'$  end

any_guide:  $\Omega \rightarrow \Omega \times (\text{Tx-set} \times \text{Guide})$ 

notify_traveller:  $\Omega \rightarrow \text{out } \{\text{ct}[p] \mid p:\text{Tx}\} \Omega$ 

```

```

commercial_to_travellers:  $\Omega \rightarrow \text{out } \{\text{ct}[p] \mid p:\text{Tx}\} \Omega$ 
warn_traveller:  $\Omega \rightarrow \text{out } \{\text{ct}[p] \mid p:\text{Tx}\} \Omega$ 
...

broadcast:  $\text{Tx-set} \times \text{CT} \rightarrow \text{Unit}$ 
broadcast(ps,msg)  $\equiv$ 
  case ps of {}  $\rightarrow$  skip, {p}  $\cup$  ps'  $\rightarrow$  ct[p]!msg; broadcast(ps',msg) end

```

type

```

CT = Guide | Notification | Commercial | Warning | ...
Guide == mkGui(...)
Notification == mkNot(...)
Commercial == mkCom(...)
Warning == mkWar(...)
...

```

3. Conclusion

A formalisation of a crucial aspect of **CyberRail** has been sketched. Namely the interplay between the rôles of travellers and the central **CyberRail** system.

Next we need analyse carefully all the action functions with respect to the way in which they use and update the respective states ($\sigma : \Sigma$) of traveller behaviours and the cyber behaviour ($\omega : \Omega$). At the end of such an analysis one can then come up with precise, formal descriptions, including axioms, of what the title of [16] refers to as the *Information Infrastructure*. We look forward to report on that in a near future.

The aim of this work is to provide a foundation, a domain theory, for **Cyber-Rail**. A set of models from which to “derive”, in a systematic way, proposals for computing systems, including software architectures.

References

- [1] Dines Bjørner. Formal Software Techniques in Railway Systems. In Eckehard Schnieder, editor, *9th IFAC Symposium on Control in Transportation Systems*, pages 1–12, Braunschweig, Germany, 13–15 June 2000. Invited talk.
- [2] Dines Bjørner. Dynamics of Railway Nets: On an Interface between Automatic Control and Software Engineering. In *CTS2003: 10th IFAC Symposium on Control in Transportation Systems*, Oxford, UK, August 4–6 2003. Elsevier Science Ltd.
- [3] Dines Bjørner. New Results and Trends in Formal Techniques for the Development of Software for Transportation Systems. In *FORMS2003: Symposium on Formal Methods for Railway Operation and Control Systems*. 2003. Budapest, Hungary. Editors: G. Tarnai and E. Schnieder, Germany.

- [4] Dines Bjørner. *Software Engineering*, volume Vol. 1: Abstraction and Modelling, Vol. 2: Advanced Specification Techniques, Vol. 3: From Domains via Requirements to Software. Springer-Verlag, 2004–2005.
- [5] Dines Bjørner, Chris George, Anne E. Haxthausen, Christian Krog Madsen, Steffen Holmslykke, and Martin **Ponioka**. “UML”-ising Formal Techniques. In *INT 2004: Third International Workshop on Integration of Specification Techniques for Applications in Engineering*. 28 March 2004, ETAPS, Barcelona, Spain. INT-2004 Proceedings, Springer-Verlag.
- [6] Dines Bjørner, Chris W. George, and Søren Prehn. Computing Systems for Railways — A Rôle for Domain Engineering. Relations to Requirements Engineering and Software for Control Applications. In *Integrated Design and Process Technology*. Editors: Bernd Kraemer and John C. Petterson, 24–28 June 2002. Society for Design and Process Science.
- [7] Dines Bjørner, C.W. George, and S. Prehn. *Scheduling and Rescheduling of Trains*, chapter 8, pages 157–184. *Industrial Strength Formal Methods in Practice*, Eds.: Michael G. Hinchey and Jonathan P. Bowen. FACIT, Springer-Verlag, London, England, 1999.
- [8] Dines Bjørner, Dong Yu Lin, and S. Prehn. Domain Analyses: A Case Study of Station Management. In *KICS'94: Kunming International CASE Symposium, Yunnan Province, P.R.ofChina*. Software Engineering Association of Japan, 16–20 November 1994.
- [9] Dines Bjørner, Søren Prehn, and Chris W. George. Formal Models of Railway Systems: Domains. FME Rail Workshop on Formal Methods in Railway Systems, FM'99 World Congress on Formal Methods, France.
- [10] Dines Bjørner, Søren Prehn, and Chris W. George. Formal Models of Railway Systems: Requirements. FMERail Workshop on Formal Methods in Railway Systems, FM'99 World Congress on Formal Methods, France.
- [11] 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 Hempstead, England, 1992.
- [12] Chris George, Anne Haxthausen, Steven Hughes, Robert Milne, Søren Prehn, and Jan Storbak Pedersen. *The RAISE Method*. The BCS Practitioner Series. Prentice-Hall, Hemel Hempstead, England, 1995.
- [13] C.A.R. Hoare. *Communicating Sequential Processes*. C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985.
- [14] Takahiko Ogino. Aiming for Passenger Interoperability. Technical report, Railway Technical Research Institute, Transport Information Technology Division, Railway Technical Research Institute, 2-8-38 Hikari-cho, Kokubunji-shi, Tokyo, 185-8540 Japan, 2003.
- [15] Takahiko Ogino. CyberRail: For Urban Mobility Tomorrow. Technical report, Railway Technical Research Institute, Transport Information Technology Division, Railway Technical Research Institute, 2-8-38 Hikari-cho, Kokubunji-shi, Tokyo, 185-8540 Japan, 2004.
- [16] Takahiko Ogino. CyberRail: Information Infrastructure for New Intermodal Transport Business Model. In *Topical Days @ IFIP World Computer Congress 2004*, IFIP Series. IFIP, Kluwer Academic Press, August 2004.
- [17] Takahiko Ogino, Koivhi Goto, Ryuji Tsuchiya, Kiyotaka Seki, and Akihiko Matsuoka. CyberRail and its significance in the coming ubiquitous society. In , 2004.