Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 280 (2011) 69–80

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Bounded Retransmission in Event-B‖CSP: a Case Study

Steve Schneider[1] Helen Treharne[2]

Department of Computing, University of Surrey Guildford, Surrey, UK

Heike Wehrheim[3]

Institut für Informatik, Universität Paderborn Paderborn, Germany

Abstract

Event-B‖CSP is a combination of Event-B and CSP in which CSP controllers are used in conjunction with Event-B machines to allow a more explicit approach to control flow. Recent results have provided an approach to stepwise refinement of such combinations. This paper presents a simplified Bounded Retransmission Protocol case study, inspired by Abrial's treatment of this example, to illustrate several aspects new in the approach. The case study includes refinement steps to illustrate four different aspects of this approach to refinement: (1) splitting events; (2) introducing convergent looping behaviour; (3) the relationship between anticipated, convergent, and devolved events; and (4) converging anticipated events.

Keywords: Event-B, CSP, Bounded Retransmission Protocol, Stepwise Refinement

1 Introduction

This paper presents a case study illustrating a refinement chain in a combination of CSP [7] and Event-B [6,1]. The case study is inspired by Abrial's treatment of the Bounded Retransmission Protocol [1], which was based on [4]. The approach is founded on the Event-B approach to stepwise refinement, in which additional detail is introduced at each stage, in particular new aspects of the state. New events need careful introduction, to relate to previous events and to control when they can occur. Our approach uses CSP rather than control variables in Event-B to manage the control flow of events in an explicit and visible way.

[1] Email: s.schneider@surrey.ac.uk

[2] Email: h.treharne@surrey.ac.uk

[3] Email: wehrheim@mail.uni-paderborn.de

doi:10.1016/j.entcs.2011.11.019

In Event-B, there are proof obligations at each stage to establish the validity of the refinement. The introduction of CSP allows some of the burden of proof to be handled within the CSP framework. In particular, those obligations concerned with flow of control can be discharged more easily with refinement checks. Establishing properties such as trace refinement and divergence-freedom for a model now allow a degree of automation. The technical details of this approach are given in [9,8]. The intention of this paper is to illustrate the kind of refinement steps that are now supported, and to provide an example of the approach.

2 CSP Background

CSP is a process algebra, which describes systems in terms of communicating components with particular attention to the interactions between them. Components consist of *processes*, which perform patterns of *events*, and which communicate by synchronising on events.

CSP provides a language to describe processes. $STOP$ is the process that can perform no events. $RUN(A)$ can perform any sequence of events from the set of events A. The prefix process $a \rightarrow P$ is initially ready to perform event a, and its subsequent behaviour is that of process P. The external choice $P \Box Q$ is a choice between process P and process Q. The parallel composition $P \parallel Q$ is the parallel combination of P and Q: they synchronise on events that they have in common, and can perform other events independently. The interleaved composition $P \mid\mid\mid Q$ is a parallel combination of P and Q where they execute independently and do not synchronise on any events. The abstraction process $P \setminus A$ behaves as P except that events in A are hidden: they are executed internally, and are no longer in the interface of P. Finally, a mapping f from one set of events to another can be used to rename alphabets: $f(P)$ is an alphabet renaming of P whereby $f(P)$ can perform $f(a)$ whenever P can perform a; similarly, $f^{-1}(P)$ can perform a whenever P can perform $f(a)$.

CSP also provides a variety of semantic models. In this paper we are primarily concerned with the *traces* model, which associated each process with a set of traces (sequences of events) that they can perform during some execution. The set of all possible traces of a process P is denoted $traces(P)$. Process P is *trace refined* by process Q if any trace of process Q is also a trace of process P. This is written $P \sqsubseteq_T Q$.

In this paper we are also concerned with *divergence*. A process diverges if it can perform an infinite sequence of internal events at some point. We generally aim to establish that processes do not diverge, and will make use of results for establishing divergence-freedom.

There are model-checking tools for CSP, such as FDR [3] and ProB [5]. These allow automated checking of refinement claims $P \sqsubseteq_T Q$, and also divergence-freedom checking for CSP processes, as well as other checks. All of the CSP proof obligations in this paper can be checked using FDR. A fuller explanation of CSP and its semantics can be found in [7].

3 Refinement principles of Event-B$\|$CSP

Event-B [1] models systems in terms of machines with state, and with events which update the state. Refinement between machines involves data refinement of existing events, and can also introduce new events.

In Event-B, when new events N_{i+1} are introduced in M_{i+1}, they can be assigned a status of 'convergent' or 'anticipated'. Furthermore, events which refine anticipated events of M_i can also be assigned a status of convergent or anticipated in M_{i+1}. Events which refine either convergent events or events without a status, are not assigned a status. Proof obligations arising from the status of events are that convergent events must decrease the variant of M_{i+1}; and anticipated events must not increase it. We will write $M_i \preccurlyeq M_{i+1}$ when the standard Event-B data refinement proof obligations hold between M_i and M_{i+1}, and so do the proof obligations on convergent and anticipated events.

In Event-B$\|$CSP, we deal with controlled components consisting of a non-divergent CSP process P_i, and an Event-B machine M_i, synchronising on their common events. The semantic foundation for this combination is given in [8].

A refinement step introduces a new non-divergent process P_{i+1} such that $P_i \sqsubseteq_T P_{i+1}$, and a new machine M_{i+1} such that $M_i \preccurlyeq M_{i+1}$. Events of M_{i+1} are present for one of two reasons:

(i) They may be refinements of events of M_i, with either the same name or a different name (this includes events which are exactly the same in each level). Refinement events give rise to a mapping f_{i+1} which maps events of M_{i+1} to events of M_i. The mapping is obtained from the **refines** clauses of event definitions, where $a \text{in} M_{i+1}$ **refines** $f_{i+1}(a) \text{in} M_i$. Note that Event-B allows one event in M_{i+1} to refine several in M_i in the most general case, but here we allow an event to refine at most one other.

(ii) They may be new events that do not refine any event in M_i. The new events for M_{i+1} will be denoted by N_{i+1}.

We extend Abrial's approach to the use of convergent and anticipated status by introducing an additional status: 'devolved'. Further, to support reasoning about divergence-freedom (i.e. that the system does not diverge when the new events are hidden), we will require all newly introduced events to be given a status, and all refinements of anticipated events to be given a status. A devolved event is treated similarly to an anticipated event, but in the context of Event-B$\|$CSP, responsibility for ensuring its convergence is devolved to the CSP controller P_{i+1} rather than delayed to some future refinement step as anticipated events are. Hence events that refine devolved events will not be assigned a status, in contrast to those refining anticipated events. Thus in M_{i+1} the only events with a status (convergent, anticipated, or devolved) are newly introduced events N_{i+1} and those that refine M_i's anticipated events. Figure 1 shows the events that we will use in our case study at the various refinement levels, with the mappings f_i also shown. Convergent, anticipated, and devolved events are labelled with (c), (a), and (d) respectively.

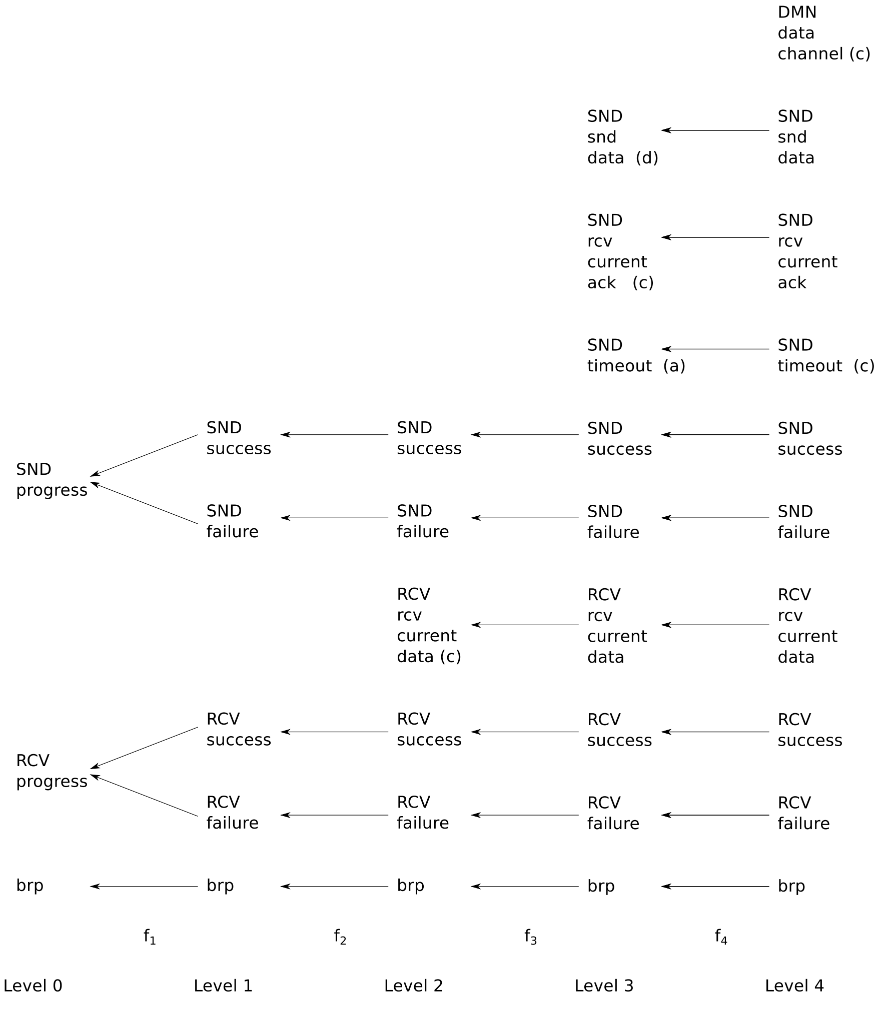

Fig. 1. Events introduced through the development

To establish the refinement relation, several proof obligations must be discharged:

(i) We require $M_i \preccurlyeq M_{i+1}$: the Event-B refinement relation holds between M_i and M_{i+1}.

(ii) We require $f_{i+1}^{-1}(P_i) \mathbin{|||} RUN(N_{i+1}) \sqsubseteq_T P_{i+1}$. If $N_{i+1} = \emptyset$ then this is equivalent to $f_{i+1}^{-1}(P_i) \sqsubseteq_T P_{i+1}$, also equivalent to $P_i \sqsubseteq_T f_{i+1}(P_{i+1})$.

It follows from Theorem 5.4 of [9] that a sequence of refinement steps from $P_0 \parallel M_0$ to $P_n \parallel M_n$, discharging these two obligations at each level, establishes the following relationship:

$$(P_0 \parallel M_0) \sqsubseteq_T f((P_n \parallel M_n) \setminus N)$$

SND_progress	RCV_progress	brp
when	**when**	**when**
$s\_st = working$	$r\_st = working$	$s\_st \neq working$
then	**then**	$r\_st \neq working$
$s\_st :\in \{success, fail\}$	$r\_st :\in \{success, fail\}$	**then**
end	**end**	$skip$
		end

$$
\begin{aligned}
P_0 &= S_0 \parallel R_0 \\
S_0 &= SND\_progress \rightarrow brp \rightarrow STOP \\
R_0 &= RCV\_progress \rightarrow brp \rightarrow STOP
\end{aligned}
$$

Fig. 2. Level 0: Machine M_0 events and control process P_0

where $f = f_n;\ \ldots;\ f_1$ is the composition of the event renamings, and $N = N_n \cup f_n^{-1}(N_{n-1}) \cup \ldots \cup (f_n;\ \ldots;\ f_2)^{-1}(N_1)$ is the set of all the new events introduced in the refinement steps, appropriately renamed.

To obtain divergence-freedom, we use a third proof obligation, to establish that the CSP controller does not allow devolved events to diverge:

(iii) Devolved events (like anticipated events) must not increase the variant. The additional proof obligation on devolved events (unlike anticipated events) is that if D_{i+1} is the set of all devolved events in M_{i+1}, then $P_{i+1} \setminus D_{i+1}$ must be divergence-free.

It follows from Corollary 5.18 of [9] that if M_n contains no anticipated events, then the combination $(P_n \parallel M_n) \setminus N$ is divergence-free.

4 Bounded Retransmission Protocol

This case study illustrates the transfer of a file by sending data packets over an unreliable medium. CSP is used to describe the repetitious behaviour in the sender (repeated transmission, and progress through the file) and the receiver (progressive receipt of the data packets), whereas the Event-B part of the model focuses on the state. For the purposes of this case study we focus only on the unreliability of the transmission medium, allowing reliable acknowledgements.

Level 0

In the initial level, given in Figure 2, we see the CSP controller split into a sender controller and a receiver controller. We begin with Abrial's model, with a single sender and a single receiver event. The event **brp** occurs after the protocol has completed.

SND_success	SND_failure
SND_success **refines** SND_progress **when** $s\_st = working$ $r\_st = success$ **then** $s\_st := success$ **end**	SND_failure **refines** SND_progress **when** $s\_st = working$ **then** $s\_st := failure$ **end**
RCV_success **refines** RCV_progress **when** $r\_st = working$ **then** $r\_st := success$ **end**	RCV_failure **refines** RCV_progress **when** $r\_st = working$ $s\_st = failure$ **then** $r\_st := failure$ **end**

invariant: $I_1 : s\_st = success \Rightarrow r\_st = success$

$$
\begin{aligned}
P_1 &= S_1 \parallel R_1 \\
S_1 &= (SND\_success \to brp \to STOP) \,\Box\, (SND\_failure \to brp \to STOP) \\
R_1 &= (RCV\_success \to brp \to STOP) \,\Box\, (RCV\_failure \to brp \to STOP)
\end{aligned}
$$

Fig. 3. Level 1: Machine M_1 events and control process P_1

Level 1

In the first refinement step the **progress** events are split into **success** and **failure** events, and an additional requirement on the relationship between the sender's and the receiver's final state is introduced. The resulting machine and controller are given in Figure 3. The associated renaming function is

$$
\begin{aligned}
f_1(SND\_success) = f_1(SND\_failure) &= SND\_progress \\
f_1(RCV\_success) = f_1(RCV\_failure) &= RCV\_progress \\
f_1(brp) &= brp
\end{aligned}
$$

There are no new events at this level.

Then $P_0 \sqsubseteq_T f_1(P_1)$. Also each event a of M_1 has that a **refines** $f_1(a)$. Hence

$$P_0 \parallel M_0 \sqsubseteq_T f_1(P_1 \parallel M_1)$$

RCV_rcv_current_data	RCV_success
status convergent **when** $r\_st = working$ $r+1 < n$ **then** $r := r+1$ $g := g \cup \{r+1 \mapsto p(r+1)\}$ **end**	**when** $r\_st = working$ $r+1 = n$ **then** $r\_st := success$ $r := r+1$ $g := g \cup \{r+1 \mapsto p(n)\}$ **end**

variant: $V_2 : n - r$

$$
\begin{aligned}
P_2 &= S_2 \parallel R_2 \\
S_2 &= S_1 \\
R_2 &= RCV\_rcv\_current\_data \to R_2 \\
&\quad \Box\ RCV\_success \to brp \to STOP \\
&\quad \Box\ RCV\_failure \to brp \to STOP
\end{aligned}
$$

Fig. 4. Level 2: Machine M_2 new and altered events, and control process P_2

Level 2

In the second refinement step, we introduce the data file $p : 1..n \to D$ to be transferred. Reception of data packets will be modelled with a new convergent event in the receiver part of the description, and an adjustment to **RCV_success**, with all other events remaining unchanged. A loop is introduced into the CSP controller. Observe that in this case it is the convergence of the B event that ensures that the new event cannot occur indefinitely.

N_2 is the set of events that have been newly introduced at this level. There is only one such event:

$$N_2 = \{RCV\_rcv\_current\_data\}$$

No event renaming has occurred, so f_2 will be the identity function and can be ignored. In fact this will be the case with all subsequent refinement levels.

The new event introduced for M_2, and the event strengthened from M_1 and M_2, are given in Figure 4, along with the control process P_2.

Then $P_1 \mid\mid\mid RUN(N_2) \sqsubseteq_T P_2$.

Hence $(P_1 \parallel M_1) \mid\mid\mid RUN(N_2) \sqsubseteq_T (P_2 \parallel M_2)$.

Level 3

In the third refinement step, we make use of the new status for events in controlled components: ‘devolved’. We introduce new events into the sender controller: a devolved event, a convergent event, and an anticipated event. We also refine two of

SND_snd_data
status
devolved
when
$s\_st = working$
then
$d := p(s+1)$
$db := TRUE$
end

SND_rcv_curr_ack
status
convergent
when
$s\_st = working$
$s + 1 < n$
$r = s + 1$
then
$s := s + 1$
$db := FALSE$
end

SND_timeout
status
anticipated
when
$TRUE$
then
skip
end

RCV_rcv_current_data
when
$r\_st = working$
$r + 1 < n$
$r = s$
$db = TRUE$
then
$r := r + 1$
$g := g \cup \{r + 1 \mapsto d\}$
end

RCV_success
when
$r\_st = working$
$r + 1 = n$
$r = s$
then
$r\_st := success$
$r := r + 1$
$g := g \cup \{r + 1 \mapsto d\}$
end

invariant: $J_3 : g = (1..r) \lhd p$
variant: $V_3 : (n - s)$

Fig. 5. Level 3: Machine M_3 new and changed events

the receiver events. These are given in Figure 5. All other events remain unchanged. We also introduce a data channel db which is set and reset by the sender when sending data.

The CSP controller, shown in Figure 6, is used to manage the flow of events in the sender. In the pure Event-B version [1], an additional control variable is needed to manage the interaction between the sender events. Here, the relationship between their occurrence is given explicitly in S_3.

The requirement $M_2 \preccurlyeq M_3$ requires that **SND_rcv_curr_ack** decreases the variant V_3, that **SND_timeout** does not increase V_3, and that the strengthened receiver events are appropriate refinements. We must also show that the devolved event **SND_snd_data** does not increase V_3.

Then $P_2 \mid\mid\mid RUN(N_3) \sqsubseteq_T P_3$, where

$$N_3 = \{SND\_snd\_data\ ,\ SND\_rcv\_curr\_ack\ ,\ SND\_timeout\}$$

Observe also that $P_3 \setminus D_3$ is divergence-free, where $D_3 = \{SND\_snd\_data\}$.

Thus $(P_2 \parallel M_2) \mid\mid\mid RUN(N_3) \sqsubseteq_T (P_3 \parallel M_3)$.

$$
\begin{aligned}
P_3 &= S_3 \parallel R_3 \\
S_3 &= SND\_snd\_data \rightarrow \; SND\_rcv\_curr\_ack \rightarrow S_3 \\
&\qquad\qquad\qquad\qquad\quad \Box\; SND\_success \rightarrow brp \rightarrow STOP \\
&\qquad\qquad\qquad\qquad\quad \Box\; SND\_fail \rightarrow brp \rightarrow STOP \\
&\qquad\qquad\qquad\qquad\quad \Box\; SND\_timeout \rightarrow S_3 \\
R_3 &= R_2
\end{aligned}
$$

Fig. 6. Level 3: Control process P_3

Level 4

In the final refinement step, we refine the anticipated event **SND_timeout** by a convergent event. This is achieved by introducing a counter variable c which places a bound on the number of times the *SND_timeout* event can occur without receiving an acknowledgement.

We also model the unreliability of the data channel by introducing the new event **DMN_data_channel** corresponding to loss of data. The new event and the changed events are given in Figure 7.

At this level, the timeout is refined to a convergent event. Also, the new event **DMN_data_channel**, which resets the data channel db, is convergent. All events in M_3 are refined by their corresponding events in M_4. Hence $M_3 \preccurlyeq M_4$. Thus $(P_3 \parallel M_3) \mid\mid\mid RUN(N_4) \sqsubseteq_T (P_4 \parallel M_4)$.

Refinement chain

Finally, we consider the whole chain of refinements from $P_0 \parallel M_0$ to $P_4 \parallel M_4$.

The set of all new events introduced is given by $N = N_2 \cup N_3 \cup N_4$. The relationship between the initial and final levels is:

$$P_0 \parallel M_0 \sqsubseteq_T f_1((P_4 \parallel M_4) \setminus N)$$

Further, there are no anticipated events left in M_4. Hence $(P_4 \parallel M_4) \setminus N$ is divergence-free.

5 Discussion

This paper has shown the development of a simple bounded retransmission protocol in Event-B‖CSP through a chain of refinement steps. Each step illustrates a refinement rule underpinned by the Event-B‖CSP semantics. The result is a description of the protocol with a clear relationship to the original specification. Further, though not considered explicitly in this paper, the protocol transmitting the file is also deadlock-free prior to completing the file transfer. Establishing this requires rules concerned with failures refinement or deadlock-freedom beyond the scope of this paper, and will be addressed in a subsequent paper.

Our example has been chosen in part to enable comparison with the pure Event-B approach taken in [1]. We believe that inclusion of the CSP controllers alongside

SND_rcv_curr_ack	SND_timeout	DMN_data_channel
when $s\_st = working$, $s+1 < n$, $r = s+1$ **then** $s := s+1$, $db := FALSE$, $c := 0$ **end**	**status** convergent **when** $c < MAX$ **then** $c := c+1$ **end**	**status** convergent **when** $db = TRUE$ **then** $db := FALSE$ **end**

SND_rcv_curr_ack
  when
    $s\_st = working$
    $s + 1 < n$
    $r = s + 1$
  then
    $s := s + 1$
    $db := FALSE$
    $c := 0$
  end

SND_timeout
  status
    convergent
  when
    $c < MAX$
  then
    $c := c + 1$
  end

DMN_data_channel
  status
    convergent
  when
    $db = TRUE$
  then
    $db := FALSE$
  end

SND_success
  when
    $s\_st = working$
    $s + 1 = n$
  then
    $s\_st := success$
    $c := 0$
  end

SND_failure
  when
    $s\_st = working$
    $c = MAX$
  then
    $s\_st := failure$
    $c := c + 1$
  end

RCV_failure
  when
    $r\_st = working$
    $c = MAX + 1$
  then
    $r\_st := failure$
  end

variant: $V_4 : (MAX - c) + \#(\{FALSE\} - \{db\})$

$$P_4 = P_3 \mid\mid\mid RUN(N_4)$$
where
$$N_4 = \{DMN\_data\_channel\}$$

Fig. 7. Level 4: Machine M_4 new and changed events, and control process P_4

the Event-B description has allowed a clearer and more natural expression of the flow of control of events, particularly with respect to the timeout and repeated transmission of the data. It also allows for simpler event descriptions in the Event-B machine, since control variables in event guards and assignments can be removed where their effect is now taken care of by the CSP controller. For example, in the pure Event-B version, at Level 3 there are several control bits w, ab, db, which are set and reset by events, and are used within event guards, to manage the flow of control. In any state, no more than one of them can have the value 1. Thus the event **SND_snd_current_data** is given as follows:

SND_snd_current_data
  when
    $s = working$
    $w = 1$
    $p + 1 < n$
  then
    $d := a(p + 1)$
    $w := 0$
    $db := 1$
    $l := 0$
  end

Here we see that the control value $w = 1$ is used (alongside other conditions) to guard this event. After the event occurs w is set to 0, and a different control value db is set to 1. This results in different events being enabled and gives rise to a flow of control. The equivalent event in our example, **SND_snd_data**, appears explicitly in the CSP control process S_3 in Figure 6. Its place in the overall flow of control is more readily apparent: it either leads to success or failure, or else to an acknowledgement or timeout which reactivates it. In our view the overall behaviour of the system is easier to understand. The cost of this benefit is the need to reconcile two formalisms, and some overhead in ensuring consistency between them.

In terms of tool support available for the approach, one notable model-checking tool that checks combinations of CSP with Event-B (and also classical B) is ProB [5], which allows Event-B machines with CSP controllers to be explored for consistency. Results from this form of model-checking augment our approach, since it supports the verification of machine invariants under CSP controllers, even if the machine in isolation is not consistent. Our rules for establishing consistency do not yet cover this case, since they require consistency of the Event-B machine. ProB also supports refinement checking of combinations, though currently this is practicable only on small examples. Alongside ProB, support for the approach will also come from Event-B tools such as the RODIN platform [2], and from CSP tools such as FDR [3] which can be used to check the proof obligations on the CSP controllers.

Acknowledgement

We are grateful to the anonymous reviewers for their constructive comments.

References

[1] Abrial, J.-R., "Modeling in Event-B: System and Software Engineering," Cambridge University Press, 2010.

[2] Abrial, J.-R., M. J. Butler, S. Hallerstede, T. S. Hoang, F. Mehta and L. Voisin, *Rodin: an open toolset for modelling and reasoning in Event-B*, STTT **12** (2010), pp. 447–466.

[3] Formal Systems (Europe) Ltd., *The FDR model checker*, `http://www.fsel.com/` (accessed 8/3/11).

[4] Groote, J. F. and J. van de Pol, *A bounded retransmission protocol for large data packets*, in: *AMAST*, 1996, pp. 536–550.

[5] Leuschel, M. and M. J. Butler, *ProB: an automated analysis toolset for the B method*, STTT **10** (2008), pp. 185–203.

[6] Métayer, C., J.-R. Abrial and L. Voisin, *Event-B language* (2005), RODIN Project Deliverable 3.2, `http://rodin.cs.ncl.ac.uk/deliverables/D7.pdf`, accessed 25/5/10.

[7] Schneider, S., "Concurrent and Real-time Systems: The CSP approach," Wiley, 1999.

[8] Schneider, S., H. Treharne and H. Wehrheim, *A CSP approach to control in Event-B*, in: *IFM*, 2010, pp. 260–274.

[9] Schneider, S., H. Treharne and H. Wehrheim, *Stepwise refinement in Event-B‖CSP. Part 1: Safety*, Technical Report CS-11-03, University of Surrey (2011).