Towards Probabilistic Modelling in Event-B

Anton Tarasyuk, Elena Troubitsyna, Linas Laibinis

Towards Probabilistic Modelling in Event-B

Anton Tarasyuk[1,2], Elena Troubitsyna[2], and Linas Laibinis[2]

[1] Turku Centre for Computer Science
[2] Åbo Akademi University
Joukahaisenkatu 3-5 A, 20520 Turku, Finland
{anton.tarasyuk, elena.troubitsyna, linas.laibinis}@abo.fi

Abstract. Event-B provides us with a powerful framework for correct-by-construction system development. However, while developing dependable systems we should not only guarantee their functional correctness but also quantitatively assess their dependability attributes. In this paper we investigate how to conduct probabilistic assessment of reliability of control systems modeled in Event-B. We show how to transform an Event-B model into a Markov model amendable for probabilistic reliability analysis. Our approach enables integration of reasoning about correctness with quantitative analysis of reliability.

Keywords: Event-B, cyclic system, refinement, probability, reliability

1 Introduction

System development by refinement is a formalised model-driven approach to developing complex systems. Refinement enables correct-by-construction development of systems. Its top-down development paradigm allows us to cope with system complexity via abstraction, gradual model transformation and proofs. Currently the use of refinement is mainly limited to reasoning about functional correctness. Meanwhile, in the area of dependable system development – the area where the formal modelling is mostly demanded – besides functional correctness it is equally important to demonstrate that the system adheres to certain quantitatively expressed dependability level. Hence, there is a clear need for enhancing formal modelling with a capability of stochastic reasoning about dependability.

In this paper we propose an approach to introducing probabilities into Event-B modelling [1]. Our aim is to enable quantitative assessment of dependability attributes, in particular, reliability of systems modelled in Event-B. We consider cyclic systems and show that their behaviour can be represented via a common Event-B modelling pattern. We show then how to augment such models with probabilities (using a proposed probabilistic choice operator) that in turn would allow us to assess their reliability.

Reliability is a probability of system to function correctly over a given period of time under a given set of operating conditions [23, 24, 17]. It is often assessed using the classical Markov modelling techniques [9]. We demonstrate that Event-B models augmented with probabilities can be given the semantic of a Markov

process (or, in special cases, a Markov chain). Then refinement of augmented Event-B models essentially becomes reliability-parameterised development, i.e., the development that not only guarantees functional correctness but also ensures that reliability of refined model is preserved or improved. The proposed approach allows us to smoothly integrate quantitative dependability assessment into the formal system development.

The paper is structured as follows. In Section 2 we overview our formal framework – Event-B. In Section 3 we introduce a general pattern for specifying cyclic systems. In Section 4 we demonstrate how to augment Event-B models with probabilities to enable formal modelling and refinement of fully probabilistic systems. In Section 5 we generalise our proposal to the cyclic systems that also contain non-determinism. Finally, in Section 6 we overview the related work and give concluding remarks.

2 Introduction to Event-B

The B Method [2] is an approach for the industrial development of highly dependable software. The method has been successfully used in the development of several complex real-life applications [19, 5]. Event-B is a formal framework derived from the B Method to model parallel, distributed and reactive systems. The Rodin platform [21] provides automated tool support for modelling and verification (by theorem proving) in Event-B. Currently Event-B is used in the EU project Deploy [6] to model several industrial systems from automotive, railway, space and business domains.

In Event-B a system specification is defined using the notion of an abstract state machine [20]. An abstract machine encapsulates the state (the variables) of a model and defines operations on its state. A general form of Event-B models is given in Fig.1.

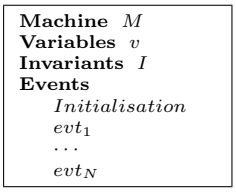

Fig. 1. An Event-B machine

The machine is uniquely identified by its name M. The state variables, v, are declared in the **Variables** clause and initialised in the *init* event. The variables are strongly typed by the constraining predicates I given in the **Invariants** clause. The invariant clause might also contain other predicates defining properties that should be preserved during system execution.

The dynamic behaviour of the system is defined by the set of atomic events specified in the **Events** clause. Generally, an event can be defined as follows:

$$evt \mathrel{\widehat{=}} \textbf{when } g \textbf{ then } S \textbf{ end},$$

where the guard g is a conjunction of predicates over the state variables v and the action S is an assignment to the state variables. If the guard g is $true$, an event can be described simply as

$$evt \mathrel{\widehat{=}} \textbf{begin } S \textbf{ end},$$

In its general form, an event can also have local variables as well as parameters. However, in this paper we use only the simple forms given above.

The occurrence of events represents the observable behaviour of the system. The guard defines the conditions under which the action can be executed, i.e., when the event is *enabled*. If several events are enabled at the same time, any of them can be chosen for execution nondeterministically. If none of the events is enabled then the system deadlocks.

In general, the action of an event is a parallel composition of assignments. The assignments can be either deterministic or non-deterministic. A deterministic assignment, $x := E(x, y)$, has the standard syntax and meaning. A nondeterministic assignment is denoted either as $x :\in S$, where S is a set of values, or $x :| P(x, y, x')$, where P is a predicate relating initial values of x, y to some final value of x'. As a result of such a non-deterministic assignment, x can get any value belonging to S or according to P.

The semantics of Event-B events is defined using so called before-after (BA) predicates [20]. A before-after predicate describes a relationship between the system states before and after execution of an event, as shown in Fig.2.

Action (S)	$BA(S)$
$x := E(x, y)$	$x' = E(x, y) \ \wedge \ y' = y$
$x :\in S$	$\exists t.\ (t \in Set \wedge x' = t) \ \wedge \ y' = y$
$x :\mid P(x, y, x')$	$\exists t.\ (P(x, t, y) \wedge x' = t) \ \wedge \ y' = y$

Fig. 2. Before-after predicates

where x and y are disjoint lists (partitions) of state variables, and x', y' represent their values in the after state. A before-after predicate for Event-B events is then constructed as follows:

$$BA(evt) \quad = \quad g \wedge BA(S).$$

The formal semantics provides us with a foundation for establishing correctness of Event-B specifications. In particular, to verify correctness of a specification, we need to prove that its initialisation and all events preserve the invariant.

Event-B employs a top-down refinement-based approach to system development. Development starts from an abstract system specification that models the most essential functional requirements. While capturing more detailed requirements, each refinement step typically introduces new events and variables into the abstract specification. These new events correspond to stuttering steps that are not visible at the abstract level. By verifying correctness of refinement, we ensure that all invariant properties of (more) abstract machines are preserved. A detailed description of the formal semantics of Event-B and foundations of the verification process can be found in [20].

3 Modelling of Cyclic Systems in Event-B

In this paper, we focus on modelling systems with cyclic behaviour, i.e. the systems that iteratively execute a predefined sequence of steps. Typical representatives of such cyclic systems are control and monitoring systems. An iteration of a control system includes reading the sensors that monitor the controlled physical processes, processing the obtained sensor values and setting actuators according to a predefined control algorithm. In principle, the system could operate in this way indefinitely long. However, different failures may affect the normal system functioning and lead to a shutdown. Hence, during each iteration the system status should be re-evaluated to decide whether it can continue its operation.

In general, *operational* states of a system, i.e., the states where system functions properly, are defined by some predicate $J(v)$ over the system variables. Usually, essential properties of the system (such as safety, fault tolerance, liveness properties) can be guaranteed only while system stays in the operational states. The predicate $J(v)$ partitions the system state space S into two disjoint classes of states – *operational* (S_{op}) and *non-operational* (S_{nop}) states, where $S_{op} \mathrel{\widehat{=}} \{s \in S \mid J.s\}$ and $S_{nop} \mathrel{\widehat{=}} S \setminus S_{op}$.

Abstractly, we can specify a cyclic system in Event-B as shown in Fig.3. In the machine CS, the variable st abstractly models the system state, which can be either operational ($J(st)$ is true) or failed ($J(st)$ is false). The event *iter* abstractly models one iteration of the system execution. As a result of this event, the system can stay operational or fail. In the first case, the system can execute its next iteration. In the latter case, the system deadlocks.

Machine CS
Variables st
Invariants
  $st \in STATE$
  ...
Events
  $Initialisation \mathrel{\widehat{=}}$
    begin
      $st :| J(st')$
    end
  $iter \mathrel{\widehat{=}}$
    when
      $J(st)$
    then
      $st :\in STATE$
    end

Fig. 3. A cyclic system

The **Invariants** clause (besides defining the variable types) can contain other essential properties of the system. Usually they are stated only over the operational states, i.e., they are of the form:

$$J(st) \quad \Rightarrow \quad ...$$

We can refine the abstract specification CS by introducing specific implementation details. For example, we may explicitly introduce new events modelling

the environment as well as reading the sensors or setting the actuators. The event *iter* can be also refined, e.g., into *detection* operation, which decides whether the system can continue its normal operation or has to shut down due to some unrecoverable failure. However, the Event-B refinement process will preserve the cyclic nature of the system described in the abstract specification CS.

The only other constraint we put on the refinement process is that all the new events introduced in refined models can be only enabled in operational system states, e.g., the event guards should contain the condition $J(v)$. To enforce this constraint, we propose a simple syntactic extension of the Event-B model structure. Specifically, we introduce a new clause **Operational guards** containing state predicates precisely defining the subset of operational system states. This is a shorthand notation implicitly adding the corresponding guard conditions to all events enabled in the operational states (except initialisation). We also assume that, like model invariants, operational guards are inherited in all refined models. By using this new clause, we can rewrite the system CS as follows.

Machine CS
Variables st
Invariants
$st \in STATE$
...
Operational guards
$J(st)$
Events
$Initialisation \mathrel{\widehat{=}}$
begin
$st :| J(st')$
end
$iter \mathrel{\widehat{=}}$
begin
$st :\in STATE$
end

Fig. 4. A cyclic system

In general, the behaviour of some cyclic system M can be intuitively described by the sequential composition $(Initialisation; \mathbf{do}\, J \to E\, \mathbf{do})$, where $\mathbf{do}\, J \to E\, \mathbf{do}$ is a while-loop with the operational guard J and the body E that consists of all the machine events except initialisation. For example, the behaviour of CS can be described simply as $(Initialisation; \mathbf{do}\, J \to iter\, \mathbf{do})$.

Each iteration of the loop maps the current operational system state into a subset of S. The resulting set of states represents all possible states that can be reached due to system nondeterministic behaviour. Therefore, an iteration of a cyclic system M can be defined as a partial function $\mathscr{I}_M$ of the type $S_{op} \to \mathscr{P}(S)$.[3] The concrete definition of $\mathscr{I}_M$ can be derived from the composition of before-after predicates of the involved events. Moreover, we can also consider the behaviour of the overall system and observe that the final state of every iteration defines the initial state of the next iteration provided the system has not failed.

The specification pattern for modelling cyclic systems defined above restricts the shape of Event-B models. This restriction allow us to propose a scalable

[3] Equivalently, we can define an iteration as a relation between S_{op} and S.

approach to integrating probabilistic analysis of dependability into Event-B. This approach we present next.

4 Stochastic Modelling in Event-B

4.1 Introducing Probabilistic Choice

Hallerstede and Hoang [7] have extended the Event-B framework with a new operator – *qualitative probabilistic choice*, denoted $\oplus|$. This operator assigns new values to variables with some positive but generally unknown probability. The extension aimed at introducing into Event-B the concept of "almost-certain convergence"– probabilistically certain termination of new event operations introduced by model refinement. The new operator can replace a nondeterministic choice (assignment) statement in the event actions. It has been shown that any probabilistic choice statement always refines its demonic nondeterministic counterpart [13]. Hence such an extension is not interfering with traditional refinement process.

In this paper we aim at introducing *quantitative* probabilistic choice, i.e., the operator $\oplus|$ with precise probabilistic information about how likely a particular choice should be made. In other words, it behaves according to some known probabilistic distribution. The quantitative probabilistic assignment

$$x \oplus| \; x_1 \; @ \; p_1; \ldots; x_n \; @ \; p_n,$$

where $\sum_{i=1}^{n} p_i = 1$, assigns to the variable x a new value x_i with the corresponding non-zero probability p_i. Similarly to Hallerstede and Hoang, we can introduce probabilistic choice only to replace the existing demonic one.

To illustrate the proposed extension, in Fig.5 we present a small example of a probabilistic communication protocol implementing transmission over unreliable channel. Since the channel is unreliable, sent messages may be lost. In the model AM shown on the left-hand side, the occurrence of faults is modelled nondeterministically. Specifically, the variable msg_a is nondeterministically assigned *delivered* or *lost*. In the model AM', the nondeterministic choice is replaced by the probabilistic one, where the non-zero constant probabilities p and $1 - p$ express how likely a message is getting delivered or lost. According to the theory of probabilistic refinement [13], the machine AM' is a refinement of the machine AM. The model refinement relation is denoted $\sqsubseteq$.

Next we show how to define refinement between probabilistic systems modelled in (extended) Event-B. In particular, our notion of model refinement can be specialized to quantitatively demonstrate that the refined system is at least as reliable as its more abstract counterpart.

4.2 Fully Probabilistic Systems

Let us first consider fully probabilistic systems, i.e., systems containing only probabilistic nondeterminism. The quantitative information present in a proba-

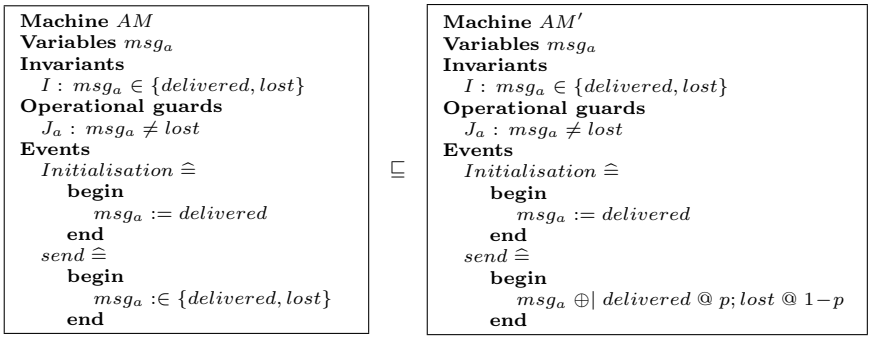

Fig. 5. A simple communication protocol: introducing probabilities

bilistic Event-B model requires lifting the notion of the system state to a probabilistic distribution over it:

Definition 1 (Probabilistic distribution). *For a system state space S, the set of distributions over S is*

$$\bar{S} \quad \widehat{=} \quad \{\Delta : S \to [0,1] \mid \sum_{s \in S} \Delta.s \ = \ 1\},$$

where $\Delta.s$ is the probability of reaching the state s.

Each iteration of a fully probabilistic system then maps some initial operational state to a subset of S according to some probabilistic distribution, i.e., we can define a single iteration $\mathscr{PI}_M$ of a probabilistic cyclic system M as a partial function of the type $S_{op} \to \bar{S}$.

There is a simple connection between iteration $\mathscr{I}_M$ of a cyclic system M and and its probabilistic counterpart $\mathscr{PI}_M$ – some state is reachable by $\mathscr{I}_M$ if and only it is reachable by $\mathscr{PI}_M$ with a non-zero probability:

$$\forall s, \, s' \cdot \ s \in \mathbf{dom}.\mathscr{I}_M \ \wedge \ s' \in \mathscr{I}_M.s \ \Leftrightarrow \ s \in \mathbf{dom}.\mathscr{PI}_M \ \wedge \ \mathscr{PI}_M.s.s' > 0,$$

where **dom** is the function domain operator.

For example, it is straightforward to see that for our model AM of the communication channel, the iteration function $\mathscr{I}_{AM}$ is

$$\mathscr{I}_{AM} = \{delivered \mapsto \{delivered, lost\}\},$$

while the probabilistic iteration function $\mathscr{PI}_{AM'}$ for the model AM' is

$$\mathscr{PI}_{AM'} = \{delivered \mapsto \{delivered \mapsto p, lost \mapsto (1-p)\}\}.$$

As it was mentioned before, all elements of system state are partitioned into two disjoint classes of operational and non-operational states. For any state $s \in S_{op}$, its distribution Δ is defined by probabilistic choice statements (assignments)

presented in an Event-B machine. However, once the system fails, it stays in the failed (non-operational) state. This means that, for any state $s \in S_{nop}$, its distribution Δ is such that $\Delta.s = 1$ and $\Delta.s' = 0$, if $s' \neq s$.

Once we know the probabilistic state distribution Δ, we can quantitatively assess the probability that the operational guard J is preserved by a single iteration. However, our goal is to evaluate system reliability. In engineering, reliability [24, 17] is generally measured by the probability that an entity $\mathscr{E}$ can perform a required function under given conditions for the time interval $[0, t]$:

$$R(t) = \mathbf{P}\{\mathscr{E} \textit{ not failed over time } [0, t]\}.$$

Hence reliability can be expressed as the probability that J remains *true* during a certain number of iterations, i.e., the probability of system staying operational for k iterations:

$$R(t) = \mathbf{P}\{\Box^{\leq k} J\}.$$

Here we use the modal operator $\Box$ borrowed from temporal logic (LTL or (P)CTL, for instance). The formula $(\Box^{\leq k} J)$ means that J holds *globally* for the first k iterations. It is straightforward to see that this property corresponds to the standard definition of reliability given above.

Let M and M' be probabilistic Event-B models of cyclic systems. We strengthen the notion of Event-B refinement by additionally requiring that the refined model will execute more iterations before shutdown with a higher probability:

Definition 2 (Refinement for probabilistic cyclic systems). *For two probabilistic Event-B models M and M' of cyclic systems such that $M \mathrel{\widehat{=}} (Initialisation; \mathbf{do}\, J \to E\, \mathbf{do})$ and $M' \mathrel{\widehat{=}} (Initialisation'; \mathbf{do}\, J' \to E'\, \mathbf{do})$, we say that M' is a refinement of M, if and only if*

1. *M' is an Event-B refinement of M ($M \sqsubseteq M'$), and*
2. *$\forall k \in \mathbb{N}_1 \cdot \mathbf{P}\{\Box^{\leq k} J\} \leq \mathbf{P}\{\Box^{\leq k} J'\}$.*

Remark 1. If the second condition of Definition 2 holds not for all k, but for some interval $k \in 1..K$, $K \in \mathbb{N}_1$, we say that M' is a *partial* refinement of M for $k \leq K$.

From the reliability point of view, a comparison of probabilistic distributions corresponds to a comparison of how likely the system would fail in its next iteration. This consideration allows us to define an order over the set $\bar{S}$ of system distributions:

Definition 3 (Ordering over distributions). *For two distributions $\Delta, \Delta' \in \bar{S}$ we define the ordering relation $\preceq$ as follows*

$$\Delta \preceq \Delta' \quad \Leftrightarrow \quad \sum_{s \in S_{op}} \Delta.s \leq \sum_{s \in S_{op}} \Delta'.s.$$

It is easy to see that the ordering relation $\preceq$ defined in this way is reflexive and transitive and hence is a total preorder on S. Let us note that the defined order is not based on pointwise comparison between the corresponding single state probabilities. Instead, we rely on the accumulated likelihood that the system stays operational.

McIver and Morgan [13] have considered deterministic probabilistic programs with possible nontermination. They have defined the set of (sub-)distributions for terminating programs, with the order over distributions introduced as $\Delta \preceq \Delta' \Leftrightarrow (\forall s \in S \cdot \Delta.s \leq \Delta'.s)$. Such a pointwise definition of an order is too strong for our purposes. We focus on quantitative evaluation of system reliability, treating all the operational states in system distributions as one class, i.e., we do not distinguish between single operational states or their groups. In our future work it would be interesting to consider a more fine-grained classification of operational states, e.g., taking into account different classes of degraded states of the system.

The order over final state distributions can be in turn used to define the order over the associated initial states:

Definition 4 (Ordering over states). *Let M be a probabilistic cyclic system. Then, for its iteration $\mathscr{PI}_M$, any initial states $s_i, s_j \in S_{op}$ and distributions $\Delta_i, \Delta_j \in \bar{S}$ such that $\Delta_i = \mathscr{PI}_M.s_i$ and $\Delta_j = \mathscr{PI}_M.s_j$, we define the ordering relation $\preceq_M$ as*

$$s_i \preceq_M s_j \quad \Leftrightarrow \quad \Delta_i \preceq \Delta_j$$

We can use this state ordering to represent the system state space S as an ordered set $\{s_1, \ldots, s_n\}$, where $n \in \mathbb{N}_{\geq 2}$ and $(\forall i \in 1..(n-1) \cdot \Delta_{i+1} \preceq \Delta_i)$.

Generally, all the non-operational states S_{nop} can be treated as a singleton set, since we do not usually care at which particular state the operational guard has been violated. Therefore, by assuming that $S = \{s_1, \ldots, s_n\}$ and $S_{nop} = \{s_n\}$, it can be easily shown that s_n is the least element (bottom) of S:

$$\Delta_n.s_n = 1 \Rightarrow \forall i \in 1..n \cdot s_n \preceq_M s_i$$

Now let us consider the behaviour of some cyclic system M in detail. We can assume that the initial system state s_1 belongs to the ordered set $\{s_1, \ldots, s_n\}$. This is a state where the system works "perfectly". After its first iteration, the system goes to some state s_i with the probability $\Delta_1.s_i$ and s_i becomes the current system state. At this point, if $i = n$, system shutdown is initiated. Otherwise, the system starts a new iteration and, as a result, goes to some state s_j with the probability $\Delta_i.s_j$ and so on. It is easy to see that this process is completely defined by the following state transition matrix

$$P_M = \begin{pmatrix} \Delta_1.s_1 & \Delta_1.s_2 & \ldots & \Delta_1.s_n \\ \Delta_2.s_1 & \Delta_2.s_2 & \ldots & \Delta_2.s_n \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_n.s_1 & \Delta_n.s_2 & \ldots & \Delta_n.s_n \end{pmatrix},$$

which in turn unambiguously defines the underlying Markov process (absorbing discrete time Markov chain, to be precise).

Let us note that the state transition matrix of a Markov chain and its initial state allow us to calculate the probability that the defined Markov process (after k steps) will be in a state s_i (see [9] for example). Let assume that the operational states of the system are ordered according to Definition 4 and initially a system is in the state s_1. Then we can rewrite the second condition of Definition 2 in the following way:

Proposition 1. *For two probabilistic Event-B models M and M' such that $M \mathrel{\widehat{=}} (Initialisation; \mathbf{do}\, J \to E\, \mathbf{do})$ and $M' \mathrel{\widehat{=}} (Initialisation'; \mathbf{do}\, J' \to E'\, \mathbf{do})$, the inequality*

$$\forall k \in \mathbb{N}_1 \cdot \mathbf{P}\{\square^{\leq k} J\} \leq \mathbf{P}\{\square^{\leq k} J'\}$$

is equivalent to

$$\forall k \in \mathbb{N}_1 \cdot ((P_{M'})^k)_{1n'} \leq ((P_M)^k)_{1n}, \tag{1}$$

where $S = \{s_1, \ldots, s_n\}$ and $S' = \{s_1, \ldots, s_{n'}\}$ are the ordered system state spaces of M and M' accordingly, and $(\ldots)_{1n}$ is a $(1n)$-th element of a matrix.

Proof Directly follows from our definition of the order on state distributions and fundamental theorems of the Markov chains theory. ■

In general, the initial system state is not necessarily the given state s_1 but can be defined by some initial state distribution Δ_0. In this case the inequality (1) should be replaced with

$$([\Delta_0'] \cdot P_{M'}^k)(n') \leq ([\Delta_0] \cdot P_M^k)(n),$$

where $[\Delta_0] = \begin{pmatrix} \Delta_0.s_1 \\ \vdots \\ \Delta_0.s_n \end{pmatrix}$, $[\Delta_0'] = \begin{pmatrix} \Delta_0'.s_1 \\ \vdots \\ \Delta_0'.s_{n'} \end{pmatrix}$ and $([\Delta_0] \cdot P_M^k)(n)$ is the n-th component of the column vector $([\Delta_0] \cdot P_M^k)$.

To illustrate our approach to refining fully probabilistic systems, let us revisit our transmission protocol example. To increase reliability of transmission, we refine the protocol to allow the sender to repeat message sending in case of delivery failure. The maximal number of such attempts is given as the predefined positive constant N. The resulting Event-B model CM is presented in Fig.6. Here the variable *att* represents the current sending attempt. Moreover, the event *send* is split to model the situations when the threshold N has been accordingly reached and not reached.

The Event-B machine CM can be proved to be a probabilistic refinement of its abstract probabilistic model (the machine AM' in Fig.5) according to Definition 2.

In this section we focused on fully probabilistic systems. In the next section we generalize our approach to the systems that also contain nondeterminism.

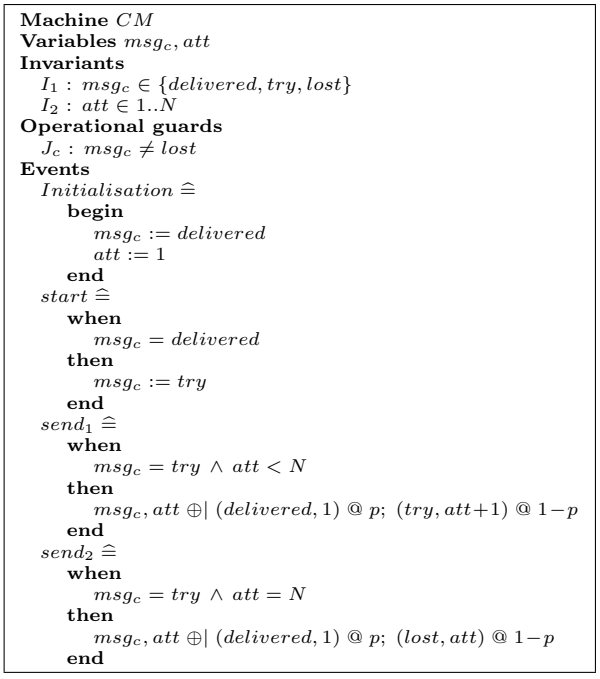

Fig. 6. A simple communication protocol: probabilistic refinement

4.3 Probabilistic Systems with Nondeterminism

For a cyclic system M containing both probabilistic and demonic nondeterminism we define a single iteration as the partial function $\mathscr{PI}_M$ of the type $S_{op} \to \mathscr{P}(\bar{S})$, i.e., as a mapping of the operational state into a *set* of distributions over S.

Nondeterminism has a demonic nature in Event-B. Hence such a model represent a worst case scenario, i.e., choosing the "worst" of operative sub-distributions – the distributions with a domain restriction on S_{op}. From reliability perspective, it means that while assessing reliability of such a system we obtain the lowest bound estimate of reliability. In this case the notions of probabilistic system refinement and the state ordering are defined as follows:

Definition 5 (Refinement for nondeterministic-probabilistic systems). *For two nondeterministic-probabilistic Event-B models M and M' of cyclic systems such that $M \mathrel{\widehat{=}} (Initialisation; \mathbf{do}\, J \to E\, \mathbf{do})$ and $M' \mathrel{\widehat{=}} (Initialisation'; \mathbf{do}\, J' \to E'\, \mathbf{do})$, we say that M' is a refinement of M, if and only if*

1. *M' is an Event-B refinement of M ($M \sqsubseteq M'$);*
2. *$\forall k \in \mathbb{N}_1 \cdot \mathbf{P}_{min}\{\Box^{\leq k}\, J\} \leq \mathbf{P}_{min}\{\Box^{\leq k}\, J'\}$,*

where $\mathbf{P}_{min}\{\Box^{\leq k}\, J\}$ *is the minimum probability that* J *remains true during the first* k *iterations.*

Remark 2. If the second refinement condition of the Definition 5 holds not for all k, but for some interval $k \in 1..K$, $K \in \mathbb{N}_1$, we say that M' is a *partial* refinement of M for $k \leq K$.

Definition 6 (Ordering over distributions).
For two sets of distributions $\{\Delta_{i_l} \mid l \in 1..L\}$ *and* $\{\Delta_{j_k} \mid k \in 1..K\} \in \mathscr{P}(\bar{S})$*, we define the ordering relation* $\preceq$ *as*

$$\{\Delta_{i_l} \mid l \in 1..L\} \preceq \{\Delta_{j_k} \mid k \in 1..K\} \Leftrightarrow \min_l(\sum_{s \in S} \Delta_{i_l}.s) \leq \min_k(\sum_{s \in S} \Delta_{j_k}.s).$$

As in the previous section, the order over final state distributions can be in turn used to define the order over the associated initial states:

Definition 7 (Ordering over states). *Let* M *be a nondeterministic-probabilistic system. Then, for its iteration* $\mathscr{PI}_M$*, any initial states* $s_i, s_j \in S_{op}$ *and sets of distributions* $\{\Delta_{i_l} \mid l \in 1..L\}, \{\Delta_{j_k} \mid k \in 1..K\} \in \mathscr{P}(\bar{S})$ *such that* $\{\Delta_{i_l} \mid l \in 1..L\} = \mathscr{PI}_M.s_i$ *and* $\{\Delta_{j_k} \mid k \in 1..K\} = \mathscr{PI}_M.s_j$*, we define the ordering relation* $\preceq_M$ *as*

$$s_i \preceq_M s_j \quad \Leftrightarrow \quad \{\Delta_{i_l} \mid l \in 1..L\} \preceq \{\Delta_{j_k} \mid k \in 1..K\}$$

The underlying Markov process representing the behaviour of a nondeterministic-probabilistic cyclic system is a simple form of a Markov decision process. For every $i \in 1, \ldots, (n-1)$, let us define $\underline{\Delta}_i = \min_l(\sum_{s \in S} \Delta_{i_l}.s)$ and $\underline{\Delta}_n = \Delta_n$. Then, the state transition matrix that represents the worst-case scenario system behaviour is defined in the following way:

$$\underline{P}_M = \begin{pmatrix} \underline{\Delta}_1.s_1 & \underline{\Delta}_1.s_2 & \ldots & \underline{\Delta}_1.s_n \\ \underline{\Delta}_2.s_1 & \underline{\Delta}_2.s_2 & \ldots & \underline{\Delta}_2.s_n \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\Delta}_n.s_1 & \underline{\Delta}_n.s_2 & \ldots & \underline{\Delta}_n.s_n \end{pmatrix},$$

and the second refinement condition of Definition 5 can be rewritten as follows:

Proposition 2. *For two nondeterministic-probabilistic Event-B models* M *and* M' *such that* $M \mathrel{\widehat{=}} (Initialisation; \mathbf{do}\, J \rightarrow E\, \mathbf{do})$ *and* $M' \mathrel{\widehat{=}} (Initialisation'; \mathbf{do}\, J' \rightarrow E'\, \mathbf{do})$*, the inequality*

$$\forall k \in \mathbb{N} \cdot \mathbf{P}\{\Box^{\leq k}\, J\} \leq \mathbf{P}\{\Box^{\leq k}\, J'\}$$

is equivalent to

$$\forall k \in \mathbb{N}_1 \cdot ((\underline{P}_{M'})^k)_{1n'} \leq ((\underline{P}_M)^k)_{1n}, \tag{2}$$

where $S = \{s_1, \ldots, s_n\}$ *and* $S' = \{s_1, \ldots, s_{n'}\}$ *are the ordered system state spaces of* M *and* M' *accordingly.*

Proof This proof is the same as the proof for Proposition 1. ■

Similarly as for fully-probabilistic systems, if the initial system state is not a single state s_1, but instead it is defined by some initial state distribution Δ_0, then the inequality (2) is replaced by

$$([\Delta_0'] \cdot \underline{P}^k_{M'})(n') \leq ([\Delta_0] \cdot \underline{P}^k_{M})(n).$$

4.4 Discussion

For fully probabilistic systems, we can often reduce the state space size using the lumping technique [9] or equally probabilistic bisimulation [12]. For nondeterministic probabilistic systems, a number of bisimulation techniques [8, 22] have been also developed.

For simple system models, deriving the set of state distributions $\bar{S}$ and calculating reliability probabilities P^k_M for each refinement step can be done manually. However, for complex real-size systems this process can be extremely time and effort consuming. Therefore, it is beneficial to have an automatic tool support for routine calculations. Development and verification of Event-B models is supported by the Rodin Platform [19] – integrated extensible development environment for Event-B. However, at the moment the support for quantitative verification is sorely missing. To prove probabilistic refinement of Event-B models according to Definition 2 and Definition 5, we need to extend the Rodin platform with a dedicated plug-in or integrate some external tool.

One of the available automated techniques widely used for analysing systems that exhibit probabilistic behaviour is probabilistic model checking [4, 10]. In particular, the probabilistic model checking frameworks like PRISM or MRMC [18, 16] provide good tool support for formal modelling and verification of discrete- and continuous-time Markov processes. To enable the quantitative reliability analysis of Event-B models, it would be advantageous to develop a Rodin plug-in enabling automatic translation of Event-B models to existing probabilistic model checking frameworks.

5 Related Work and Conclusions

5.1 Related Work

The Event-B framework has been extended by Hallerstede and Hoang [7] to take into account model probabilistic behaviour. They introduce qualitative probabilistic choice operator to reason about almost certain termination. This operator attempts to bound demonic nondeterminism that, for instance, allows us to demonstrate convergence of certain protocols. However, this approach is not suitable for reliability assessment since explicit quantitative representation of reliability is not supported.

Several researches have already used quantitative model checking for dependability evaluation. For instance, Kwiatkowska et al. [11] have proposed an

approach to assessing dependability of control systems using continuous time Markov chains. The general idea is similar to ours – to formulate reliability as a system property to be verified. This approach differs from ours because it is aims at assessing reliability of already developed systems. However, dependability evaluation late at the development cycle can be perilous and, in case of poor results, may lead to major system redevelopment causing significant financial and time losses. In our approach reliability assessment proceeds hand-in-hand with the system development by refinement. It allows us to assess dependability of designed system on the early stages of development, for instance, every time when we need to estimate impact of unreliable component on the system reliability level. This allows a developer to make an informed decision about how to guarantee a desired system reliability.

A similar topic in the context of refinement calculus has been explored by Morgan et al. [14, 13]. In this approach the probabilistic refinement has been used to assess system dependability. Such an approach is much stronger than the approach described in this paper. Probabilistic refinement allows the developers to obtain algebraic solutions even without pruning the system state space. Meanwhile, probabilistic verification gives us only numeric solutions for restricted system models. In a certain sense, our approach can be seen as a property-wise refinement evaluation. Indeed, while evaluating dependability, we essentially check that, for the same samples of system parameters, the probability of system to hold a certain property is not decreased by refinement.

5.2 Conclusions

In this paper we proposed an approach to integrating probabilistic assessment of reliability into Event-B modelling. We defined reliability of a cyclic system as the probability of the system to stay in its operational state for a given number of iterations. Our approach to augmenting Event B models with probabilities allows us to give the semantic of a Markov process (or, in special cases, a Markov chain) to augmented models. In turn, this allow us to algebraically compute reliability by using any of numerous automated tools for reliability estimation.

In general, continuous-time Markov processes are more often used for dependability evaluation. However, the theory of refinement of systems with continuous behaviour has not reached maturity yet [3, 15]. In this paper we showed that, by restricting the shape of Event-B models and augmenting them with probabilities, we can make a smooth transition to representing a cyclic system as a Markov process. This allow us to rely on standard techniques for assessing reliability.

In our future work it would be interesting to explore continuous-time reasoning as well as generalise the notion of refinement to take into account several dependability attributes.

Acknowledgments

This work is supported by IST FP7 DEPLOY Project. We also wish to thank the anonymous reviewers for their helpful comments.

References

1. Abrial, J.R.: Extending B without Changing it (for Developing Distributed Systems). In: Habiras, H. (ed.) First Conference on the B method, pp. 169–190. IRIN Institut de recherche en informatique de Nantes (1996)
2. Abrial, J.R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press (2005)
3. Back, R.J.R., Petre, L., Porres, I.: Generalizing Action Systems to Hybrid Systems. In: FTRTFT 2000, LNCS 1926. pp. 202–213. Springer (2000)
4. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT press (2008)
5. Craigen, D., Gerhart, S., T.Ralson: Case study: Paris metro signaling system. In: IEEE Software. pp. 32–35 (1994)
6. EU-project DEPLOY: Online at http://www.deploy-project.eu/
7. Hallerstede, S., Hoang, T.S.: Qualitative probabilistic modelling in Event-B. In: Davies, J., Gibbons, J. (eds.) IFM 2007, LNCS 4591. pp. 293–312 (2007)
8. Hansson, H.: Time and Probability in Formal Design of Distributed Systems. Elsevier (1995)
9. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. D. Van Nostrand Company (1960)
10. Kwiatkowska, M.: Quantitative verification: models techniques and tools. In: ESEC/FSE'07. pp. 449 – 458. ACM (2007)
11. Kwiatkowska, M., Norman, G., Parker, D.: Controller dependability analysis by probabilistic model checking. In: Control Engineering Practice. pp. 1427–1434 (2007)
12. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. In: Information and Computation 94. pp. 1–28 (1991)
13. McIver, A.K., Morgan, C.C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer (2005)
14. McIver, A.K., Morgan, C.C., Troubitsyna, E.: The probabilistic steam boiler: a case study in probabilistic data refinement. In: Proc. International Refinement Workshop, ANU, Canberra. Springer (1998)
15. Meinicke, L., Smith, G.: A Stepwise Development Process for Reasoning about the Reliability of Real-Time Systems. In: Integrated Formal Methods IFM2007, LNCS 4591. pp. 439–456. Springer (2007)
16. MRMC – Markov Reward Model Checker: Online at http://www.mrmc-tool.org/
17. O'Connor, P.D.T.: Practical Reliability Engineering, 3rd ed. John Wiley & Sons (1995)
18. PRISM – Probabilistic Symbolic Model Checker: Online at http://www.prismmodelchecker.org/
19. Rigorous Open Development Environment for Complex Systems (RODIN): IST FP6 STREP project, online at http://rodin.cs.ncl.ac.uk/
20. Rigorous Open Development Environment for Complex Systems (RODIN): Deliverable D7, Event-B Language, online at http://rodin.cs.ncl.ac.uk/
21. RODIN. Event-B Platform: Online at http://www.event-b.org/
22. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. In: Nordic Journal of Computing, 2(2). pp. 250–273 (1995)
23. Storey, N.: Safety-Critical Computer Systems. Addison-Wesley (1996)
24. Villemeur, A.: Reliability, Availability, Maintainability and Safety Assessment. John Wiley & Sons (1995)