DOI 10.1007/s00165-014-0305-z
BCS © 2014
Formal Aspects of Computing (2015) 27: 53–77

Formal Aspects of Computing

Integrating stochastic reasoning into Event-B development

Anton Tarasyuk*, Elena Troubitsyna and Linas Laibinis
Åbo Akademi University, Joukahaisenkatu 3-5, 20520 Turku, Finland

Abstract. Dependability is a property of a computer system to deliver services that can be justifiably trusted. Formal modelling and verification techniques are widely used for development of dependable computer-based systems to gain confidence in the correctness of system design. Such techniques include Event-B—a state-based formalism that enables development of systems correct-by-construction. While Event-B offers a scalable approach to ensuring functional correctness of a system, it leaves aside modelling of non-functional critical properties, e.g., reliability and responsiveness, that are essential for ensuring dependability of critical systems. Both reliability, i.e., the probability of the system to function correctly over a given period of time, and responsiveness, i.e., the probability of the system to complete execution of a requested service within a given time bound, are defined as quantitative stochastic measures. In this paper, we propose an extension of the Event-B semantics to enable stochastic reasoning about dependability-related non-functional properties of cyclic systems. We define the requirements that a cyclic system should satisfy and introduce the notions of reliability and responsiveness refinement. Such an extension integrates reasoning about functional correctness and stochastic modelling of non-functional characteristics into the formal system development. It allows the designer to ensure that a developed system does not only correctly implement its functional requirements but also satisfies given non-functional quantitative constraints.

Keywords: Event-B, Refinement, Probabilistic reasoning, Reliability, Responsiveness, Cyclic systems, Markov processes

1. Introduction

Formal methods—the mathematically-based approaches that provide the developers with rigorous ways to design and analyse systems—are extensively used in the design of dependable computer-based systems. Such methods include Event-B [Abr10, Abr96]—a formalism derived from the B Method [Abr05] to facilitate development of reactive and distributed systems. Event-B is a rigorous, state-based framework supporting the correct-by-construction system development. When developing a computer-based system in Event-B, we start from an abstract specification that defines the essential behaviour and properties of the system under construction. Via a number of correctness preserving model transformations—refinement steps, the abstract specification is transformed into a specification that is close to the desired implementation. In the development process, correctness of each refinement step is verified by proofs.

Correspondence and offprint requests to: A. Tarasyuk, E-mail: anton.tarasyuk@abo.fi

Formal development in Event-B allows us to ensure that a resulting detailed specification adheres to its abstract counterpart, i.e., it guarantees that the services provided by the system are functionally correct. However, to ensure dependability, it is essential to guarantee that the system is not only functionally correct but also satisfies a number of non-functional constraints. Usually such constraints are defined probabilistically. Currently, non-functional dependability-related system requirements are abstracted away in the process of system refinement. Yet, it would be desirable to re-use the created formal models for evaluating the impact of the chosen design decisions on the given non-functional requirements. Hence there is a clear need for integration between modelling of functional and non-functional system requirements.

In this paper, we extend the Event-B framework to enable stochastic modelling of reliability and responsiveness of cyclic systems. *Reliability* is the probability of the system functioning correctly over a given period of time under a given set of operating conditions [ALRL04, Vil95, O'C95], while *responsiveness* is the likelihood that the system successfully completes a service delivery within a certain time bound [TRF03, CS88]. These properties are dual in the sense that reliability can be defined as a probabilistic measure of the system staying operational during a certain time period, while responsiveness is a probabilistic measure of the system termination within a certain period of time. We rely on the notion of *iteration* as a discrete unit of time defining the unified time scale for cyclic systems, i.e., the systems that iteratively execute a predefined sequence of computational steps. We formally define the conditions that should be verified to ensure that the system under construction is indeed cyclic.

To enable explicit probabilistic reasoning about reliability and responsiveness, we propose a pragmatic approach to evaluating the impact of various possible refinement alternatives on system dependability. We advocate a dependability-explicit development process in which construction of a formal model of the functional system behaviour is intervened by stochastic evaluation of chosen design decisions on dependability. Such an approach aims at improving predictability in the system design by allowing the designers to assess the target system characteristics from the early development stages.

To facilitate stochastic assessment using Event-B models, we introduce a new language construct—the quantitative probabilistic choice—and define the semantics of the extended Event-B models. We show that, in the case of fully probabilistic systems, the underlying model of a probabilistically-enriched Event-B specification is a discrete-time Markov chain [KS60]. Moreover, in the case of the systems that contain both probabilistic and demonic behaviour, this model becomes a Markov decision process [Put05, Whi93].

To enable reliability- and responsiveness-explicit development in the probabilistically-augmented Event-B, we strengthen the notion of refinement by requiring that a refined model, in addition to being a proper functional refinement of its more abstract counterpart, also satisfies a number of quantitative constraints. These constraints ensure that the refined model improves (or at least preserves) the current system reliability and/or responsiveness. These additional constraints are derived from the fundamental properties of discrete-time Markov chains and Markov decision processes. We also compare the proposed definitions of probabilistic Event-B refinement with the traditional approach to probabilistic program refinement (see [MM05] for instance) and demonstrate by an example that the refinement conditions stipulated by the traditional approach are generally too strong for reasoning about such system properties as reliability and responsiveness. We believe that our work establishes sound mathematical foundations for integrating logical reasoning about functional correctness and probabilistic analysis of critical system properties.

The paper is structured as follows. In Sect. 2 we overview our formal framework—Event-B and discuss several advances in applying the framework to the development of dependable systems. In Sect. 3 we introduce the notion of cyclic systems, formulate the conditions required to verify their cyclic nature, and formally define the notion of a system iteration and its properties. In Sect. 4 we introduce the probabilistic choice operator and give an example of a probabilistic Event-B model. In Sects. 5 and 6 we present the strengthened notion of Event-B refinement for both fully probabilistic systems and systems with nondeterminism. In Sect. 7 we summarise the presented approach to stochastic reasoning in Event-B. Finally, in Sects. 8 and 9 we overview the related work in the field and give some concluding remarks.

2. Introduction to Event-B

Event-B [Abr10] is a formal framework derived from the B method [Abr05] to model parallel, distributed and reactive systems. The Rodin platform [Rod] provides tool support for modelling and formal verification by theorem proving in Event-B.

Event-B employs a top-down refinement-based approach to system development. The development starts from an abstract system specification that models the most essential behaviour and properties. Each refinement step introduces a representation of more detailed requirements into the system model. This results in elaborating on the data structures, dynamic behaviour and properties of the model. The logical consistency of system models and correctness of refinement steps are verified by mathematical proofs.

2.1. Event-B language

In Event-B, a system specification is defined using the notion of an *abstract state machine*. An abstract state machine encapsulates the model state, represented as a collection of model variables, and defines operations on this state via machine *events*. The occurrence of events together with the associated state changes represents the system behaviour.

Usually, an Event-B machine has an accompanying component called *context*. A context component can include user-defined carrier sets (types) as well as constants and their properties, which are given as a list of model axioms. In a most general form, an Event-B model can be defined as follows.

Definition 1 An Event-B model is a tuple $(\mathcal{C}, \mathcal{S}, \mathcal{A}, \upsilon, \Sigma, \mathcal{I}, \mathcal{E}, Init)$, where:

- $\mathcal{C}$ is a set of model constants;
- $\mathcal{S}$ is a set of model sets (types);
- $\mathcal{A}$ is a set of axioms over $\mathcal{C}$ and $\mathcal{S}$;
- υ is a set of model variables;
- Σ is the model state space, which is defined by all possible valuations of the model variables υ;
- $\mathcal{I}$ is the model invariant defined as a state predicate, i.e., $\mathcal{I} \in \Sigma \to Bool$;
- $\mathcal{E}$ is a non-empty set of model events, where each event e belonging to $\mathcal{E}$ is defined as a binary state relation, i.e., $e \in \Sigma \times \Sigma \to Bool$;
- $Init$ is a predicate defining an non-empty set of model initial states.

The model variables υ are strongly typed by the constraining predicates specified in the invariant $\mathcal{I}$ and initialised by the values satisfying the predicate $Init$. Furthermore, $\mathcal{I}$ may define other important properties that must be preserved by the system during its execution.

While specifying an event, we rely on the following syntax:

$e \mathrel{\widehat{=}}$ **any** a **where** G_e **then** R_e **end**,

where e is the event name, a is a list of local variables of the event, and G_e is the *event guard*—a model state predicate $G_e \in \Sigma \times \Gamma \to Bool$. Here Γ is the collective type of the event local variables. The *event action* $R_e \in \Gamma \times \Sigma \times \Sigma \to Bool$ is defined as a relation expressing the relationship between the values of local variables, the system states before, and the system states after event execution.

The event guard G_e defines the conditions under which such an execution can be performed, 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.

The event action R_e is usually specified as a parallel composition of state assignments. These assignments can be either deterministic or nondeterministic. A deterministic assignment $x := E(x, y)$, where $x, y \subseteq \upsilon$, 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 the initial values of variables x and y to some final value of x, denoted as x'. As a result of such non-deterministic assignments, the variables x can get any value either belonging to S or according to P.

If an event does not have local variables, it can be described simply as

$e \mathrel{\widehat{=}}$ **when** G_e **then** R_e **end**.

Here G_e is of the type $\Sigma \to Bool$, and R_e is of the type $\Sigma \times \Sigma \to Bool$.

2.2. Event-B semantics: model events

Essentially, an event e of the form **when** G_e **then** R_e **end** is a relation describing the corresponding state transformation from σ to σ', such that

$$e(\sigma, \sigma') = \mathcal{I}(\sigma) \wedge G_e(\sigma) \wedge R_e(\sigma, \sigma').$$

Here we treat the model invariant $\mathcal{I}$ as an implicit event guard. Note that, due to the possible presence of nondeterminism, the successor state σ' is not necessarily unique.

An event e of the form **any** a **where** G_e **then** R_e **end** is defined as follows:

$$e(\sigma, \sigma') = \exists a \cdot \mathcal{I}(\sigma) \wedge G_e(\sigma, a) \wedge R_e(a, \sigma, \sigma').$$

We can always rewrite the guard predicate $G_e(\sigma, a)$ into $G_{ev}(\sigma) \wedge G_{ea}(\sigma, a)$, where $G_{ev} \in \Sigma \rightarrow Bool$ is a condition on the global state, and $G_{ea} \in \Sigma \times \Gamma \rightarrow Bool$ is a predicate constraining the values of the local event variables. Then the above definition is equivalent to

$$e(\sigma, \sigma') = \mathcal{I}(\sigma) \wedge G_{ev}(\sigma) \wedge (\exists a \cdot G_{ea}(\sigma, a) \wedge R_e(a, \sigma, \sigma')).$$

It is easy to see that such an event is a special case of the simple event form, with the guard $G_{ev}(\sigma)$ and the action $(\lambda(\sigma, \sigma') \cdot \exists a \cdot G_{ea}(\sigma, a) \wedge R_e(a, \sigma, \sigma'))$. Without loss of generality, from now on we will consider only events of the simple form **when** G_e **then** R_e **end**.

In other words, the semantics of a single model event is given as a binary relation between pre- and post-states of the event. To clarify this relationship, we define two functions before and after of the type $\mathcal{E} \rightarrow \mathbf{2}^{\Sigma}$ in a way similar to [Ili11, TTL12]:

$$\mathsf{before}(e) = \{\sigma \in \Sigma \mid \mathcal{I}(\sigma) \wedge G_e(\sigma)\} \quad \text{and} \quad \mathsf{after}(e) = \{\sigma' \in \Sigma \mid \exists \sigma \in \Sigma \cdot \mathcal{I}(\sigma) \wedge G_e(\sigma) \wedge R_e(\sigma, \sigma')\}.$$

The latter definition can be also rewritten as follows:

$$\mathsf{after}(e) = \{\sigma' \in \Sigma \mid \exists \sigma \in \Sigma \cdot \sigma \in \mathsf{before}(e) \wedge R_e(\sigma, \sigma')\}.$$

One can see that, for a given event $e \in \mathcal{E}$ and any state $\sigma \in \Sigma$, e is enabled in σ if and only if $\sigma \in \mathsf{before}(e)$.

To reason about the event execution starting from a particular pre-state σ, we also introduce a single-state version of the function after:

$$\mathsf{after}_\sigma(e) = \{\sigma' \in \Sigma \mid \mathcal{I}(\sigma) \wedge G_e(\sigma) \wedge R_e(\sigma, \sigma')\}.$$

We can lift the above functions before and after for any set of the given events E, $E \subseteq \mathcal{E}$:

$$\mathsf{before}(E) = \bigcup_{e \in E} \mathsf{before}(e) \text{ and } \mathsf{after}(E) = \bigcup_{e \in E} \mathsf{after}(e).$$

In the special case when $E = \mathcal{E}$, the resulting set before($\mathcal{E}$) contains all the states when the modelled system is operational, i.e., when at least one event is enabled. Correspondingly, the complement of before($\mathcal{E}$) gives us those system states that, once reached, put the system into deadlock:

$$\mathsf{deadlocks}(\mathcal{E}) = \Sigma \setminus \mathsf{before}(\mathcal{E}).$$

The semantics of model events also allows us to define the notion of an execution trace of the modelled system.

Definition 2 (*Execution trace*) For any Event-B model we define its execution trace, *tr*, as a sequence of system states

$$< \sigma_0, \ldots, \sigma_i, \sigma_{i+1}, \ldots >$$

such that

(1) $\sigma_0 \in Init$;

(2) $\forall i \in \mathbb{N} \cdot \sigma_i, \sigma_{i+1} \in tr \Rightarrow \exists e \in \mathcal{E} \cdot \sigma_i \in \mathsf{before}(e) \wedge \sigma_{i+1} \in \mathsf{after}_{\sigma_i}(e)$.

Finally, we define the set $traces_M$ as a set containing all the execution traces of the Event-B model M.

2.3. Event-B semantics: initial model

The semantics of an entire Event-B model is completed by formulating a number of conditions—*proof obligations*, expressed in the form of logical sequents. In this paper we consider only the most important proof obligations that should be verified for the initial and refined models. The full list of proof obligations can be found in [Abr10].

In this paper we will heavily rely on the semantic functions before and after defined above. To keep our formalisation consistent and concise, we formulate all the presented proof obligations in terms of these functions.

The initial Event-B model should satisfy the event feasibility and invariant preservation properties. For each event e, its feasibility means that, whenever the event is enabled (in some particular state σ), its next-state relation is well-defined, i.e., there exists some reachable after-state:

$$\mathcal{A},\ \sigma \in \mathsf{before}(e)\ \vdash\ \exists \sigma' \cdot \sigma' \in \mathsf{after}_{\sigma}(e) \tag{FIS}$$

Each event e of an Event-B model should also preserve the model invariant:

$$\mathcal{A},\ \sigma' \in \mathsf{after}(e)\ \vdash\ \mathcal{I}(\sigma') \tag{INV}$$

Since the initialisation event has no initial state and guard, its invariant preservation proof obligation is simpler:

$$\mathcal{A},\ \mathit{Init}(\sigma')\ \vdash\ \mathcal{I}(\sigma') \tag{INIT}$$

2.4. Event-B semantics: refinement

Each Event-B refinement step typically introduces new variables and events into a more abstract model. The introduced new events correspond to stuttering steps that are not visible at the abstract level. The old, abstract model events may be also refined to reduce their nondeterminism and provide access to the new variables.

Moreover, Event-B formal development supports data refinement, allowing us to replace some abstract variables with their concrete counterparts. In that case, the invariant of a refining machine is extended (conjuncted) with a so called *gluing invariant* that formally defines the relationship between the abstract and concrete variables. Informally, it "glues" the state of the present machine to that of a (more) abstract machine being refined. A gluing invariant is used to prove simulation relationship between two machines, as formally defined in the refinement proof obligations presented below. For more details about using a gluing invariant in the Event-B refinement process, see [Abr10].

Let Σ_a, $\mathcal{I}_a$ and $\mathcal{E}_a$ be respectively the state space, invariant, and events of the abstract model. Similarly, let Σ_c, $\mathcal{I}_c$ and $\mathcal{E}_c$ be respectively the state space, invariant, and events of the (concrete) refined model. Finally, let $\mathcal{J}$ be a gluing invariant between Σ_a and Σ_c. To verify correctness of a refinement step, we need to prove a number of proof obligations for the refined model.

The first three proof obligations focus on the connection between the abstract events and their concrete refined versions. Let us assume that an abstract event $e_a \in \mathcal{E}_a$ is refined by a concrete event $e_c \in \mathcal{E}_c$.

The first proof obligation states that the refined event e_c should stay feasible:

$$\mathcal{A},\ \mathcal{I}_a(\sigma_a),\ \mathcal{J}(\sigma_a, \sigma_c),\ \sigma_c \in \mathsf{before}(e_c)\ \vdash\ \exists \sigma_c' \cdot\ \sigma_c' \in \mathsf{after}_{\sigma_c}(e_c) \tag{REF\_FIS}$$

The guard of the refined event e_c can be only strengthened in a refinement step:

$$\mathcal{A},\ \mathcal{I}_a(\sigma_a),\ \mathcal{J}(\sigma_a, \sigma_c),\ \sigma_c \in \mathsf{before}(e_c)\ \vdash\ \sigma_a \in \mathsf{before}(e_a) \tag{REF\_GRD}$$

The refined event e_c should preserve the concrete invariant $\mathcal{I}_c$. Moreover, its "execution" cannot be contradictory to that of the abstract event e_a:

$$\mathcal{A},\ \mathcal{I}_a(\sigma_a),\ \mathcal{J}(\sigma_a, \sigma_c),\ \sigma_c' \in \mathsf{after}_{\sigma_c}(e_c)\ \vdash \mathcal{I}_c(\sigma_c')\ \wedge\ \exists \sigma_a' \cdot\ \sigma_a' \in \mathsf{after}_{\sigma_a}(e_a)\ \wedge\ \mathcal{J}(\sigma_a', \sigma_c') \tag{REF\_INV}$$

To verify that all the concrete events (both old and new) do not introduce additional deadlocks into the model, we need to prove *relative deadlock freedom*:

$$\mathcal{A},\ \sigma_a \in \mathsf{before}(\mathcal{E}_a),\ \mathcal{J}(\sigma_a, \sigma_c),\ \mathcal{I}_c(\sigma_c)\ \vdash\ \sigma_c \in \mathsf{before}(\mathcal{E}_c) \tag{REF\_DLF}$$

Finally, we should demonstrate that the new events do not collectively diverge, i.e., they eventually return control to the old events. This is typically achieved by providing a natural number state expression (*variant*) and showing that each new event decreases it. Let $nvar \in \Sigma_c \rightarrow \mathbb{N}$ be the provided variant expression. Let also $e \in \widehat{\mathcal{E}}_c$ be a new concrete event, where $\widehat{\mathcal{E}}_c \subset \mathcal{E}_c$ is a set of new events of the refined model. Then the non-divergence proof obligation for the event e can be presented as follows:

$$\mathcal{A},\ \sigma_c' \in \mathsf{after}_{\sigma_c}(e)\ \vdash\ nvar(\sigma_c') < nvar(\sigma_c) \qquad \text{(REF\_VAR)}$$

The Event-B refinement process allows us to gradually introduce implementation details, while preserving functional correctness during stepwise model transformation. The model verification effort, in particular, automatic generation and demonstration of the required proof obligations, is significantly facilitated by the provided tool support—the Rodin platform.

Event-B facilitates correct-by-construction development of critical systems. However, to ensure system dependability, we should guarantee that the system is not only functionally correct but also meets desired non-functional requirements to system reliability, safety, responsiveness, etc. Since many of these properties depend on time, in the next section we will demonstrate how reliance on the notion of iteration allows us to implicitly introduce a model of time into Event-B specifications of cyclic systems. In its turn, it sets a scene for stochastic reasoning about critical system properties.

2.5. Development of dependable systems by refinement

Event-B facilitates the correct-by-construction development of dependable systems. In particular, it provides the system developers with a powerful mechanism for structuring complex system requirements and systematic reasoning about the system behaviour at different levels of abstraction. Moreover, various dependability-related aspects of the system behaviour can be modelled and verified in the Event-B refinement process. Over the recent years, a significant experience has been gained in refinement-based development of dependable systems from various domains. In particular, a significant contribution to establishing a formal dependability-explicit development process has been done within the EU projects RODIN [ROD04] and DEPLOY [DEP08].

Dependability is a multi-facet system characteristic that puts an emphasis on different system properties depending on the nature of an application. A number of modelling patterns, refinement strategies and integrated modelling techniques have been proposed to formally define and verify various aspects of dependability, and in particular fault tolerance.

Fault tolerance mechanisms are introduced into the system design to mask or mitigate the effect of component failures on the system-level services and, per se, maximise system reliability. Since system components are inherently unreliable, i.e., their behaviour might deviate from the nominal one during system functioning, coping with faults is important for ensuring dependability. In the formal dependability-explicit development, modelling the failure behaviour constitutes an intrinsic part of the specification at all levels of abstraction. It allows us to explicitly model non-deterministic failure occurrence and introduce the required fault tolerance mechanisms by refinement. Such an approach enables building large-scale fault tolerant systems with complex hierarchical fault tolerance mechanisms.

Essentially, while introducing fault tolerance mechanisms by refinement, we define the means for mitigating non-deterministic occurrence of faults. Hence, we resolve or reduce inherent nondeterminism of the specification. Usually there are several ways to achieve it, i.e., there are different alternative implementations of fault tolerance. Each of them is a valid refinement of the specification under consideration, i.e., each alternative is equivalent from the correctness point of view. Yet they might be different from the point of view of the non-functional requirements. Since these non-functional requirements often could be only evaluated probabilistically, it is desirable to enable evaluation of the refinement alternatives to optimise such design decisions throughout the development process. If there is a specific non-functional property of interest, we can consider such a formal development to be driven by this property.

Such an evaluation might not be required at each refinement step, since a particular refinement step could elaborate on the nominal system behaviour or refine the abstract data structures, i.e., it might not impact the non-functional system characteristics. On the other hand, when a refinement step introduces a representation of redundant component or computation re-execution, an assessment of the non-functional characteristics would give the designers an immediate feedback on appropriateness of the chosen design with respect to certain target characteristics.

In this paper, we aim at proposing a practical solution for integrating assessment of the non-functional characteristics of dependable systems, especially reliability and responsiveness, into the refinement process in Event-B. Since these properties depend on time, in the next section we will demonstrate how reliance on the notion of iteration allows us to implicitly introduce a model of time into Event-B specifications of cyclic systems. In its turn, it facilitates stochastic modelling of quantitative dependability properties.

3. Modelling of cyclic systems in Event-B

There is a large class of systems that exhibit a cyclic behaviour, i.e., the systems that iteratively execute a predefined sequence of steps. Typical representatives of cyclic systems are control and monitoring systems. For instance, one iteration of a control system usually includes reading the sensors that monitor the controlled physical processes, processing the obtained sensor values, and finally setting the actuators according to a predefined control algorithm. In principle, the system could operate in this way indefinitely. However, unforeseen conditions in the operating environment or component failures may affect the normal system functioning and lead to a shutdown.

3.1. The Event-B structure and control flow of a cyclic system

Let us start by describing the desired structure and control flow properties of cyclic systems we aim to model in Event-B. After completing computations performed at each iteration, the status of a cyclic system should be re-evaluated to decide whether it can continue its operation. Moreover, some actions may be needed to initiate a new iteration.

Therefore, it is convenient to split a formal model of a cyclic system into three groups of events. The first two groups, called IN and OUT, model the system behaviour at the beginning and the end of each system iteration respectively. These actions may include some necessary calculations as well as controlling actions to initialise and finalise a system iteration. The last group, called e_0, models the internal computations of a cyclic system that may be conducted within a single iteration.

Without losing generality, from now on we will treat all such groups of events as single events IN, e_0, and OUT (because they can always be merged into the corresponding single events, see, e.g., [Abr10]). Thus, each iteration of the modelled cyclic system can be represented by the following control flow on model events:

$$IN \longrightarrow e_0 \longrightarrow OUT.$$

During the Event-B refinement process, any of the events IN, OUT and e_0 can be refined. For simplicity, in this paper we assume that refinement will focus on elaborating the computation modelled by e_0. This reflects the refinement style that is frequently adopted in Event-B: the detailed representation of an algorithm is introduced in a number of new events $e_1, e_2, \ldots, e_n$ preceding e_0. In other words, these events model intermediate calculations on new data structures necessary to produce the system result in e_0. Then the control flow of a single iteration of a refined system looks as follows:

$$IN \longrightarrow e_1, \ldots, e_n \longrightarrow e_0 \longrightarrow OUT.$$

Note that we do not restrict in any way the execution order of the new events $e_1, e_2, \ldots, e_n$, i.e., any event branching or looping is possible. However, according to the Event-B semantics, $e_1, e_2, \ldots, e_n$ are not allowed to diverge (see the proof obligation rule REF_VAR), i.e., the new events will eventually return control to the event e_0.

3.2. Formal requirements for cyclic systems

Let us now formally define requirements imposed on the abstract and refined Event-B models of a cyclic system. Similarly to Sect. 2, we denote by $\mathcal{E}$ the set of all model events. Moreover, let $\widehat{\mathcal{E}}$ be all the new model events introduced during the refinement process. In other words,

$$\widehat{\mathcal{E}} = \{e_1, e_2, \ldots, e_n\} \quad \text{and} \quad \mathcal{E} = \{IN, e_0, OUT\} \cup \widehat{\mathcal{E}}.$$

Next we formulate a number of formal requirements that a model of a cyclic system has to satisfy. These properties can be generated and verified as additional proof obligations for a particular model under consideration.

Since the properties proved for an Event-B model are preserved for any of its refinements, it is often sufficient to verify these additional proof obligations for the newly introduced events only.

$$\{\sigma \in \Sigma \mid Init(\sigma)\} \subseteq \mathsf{before}(IN) \tag{1}$$

$$\mathsf{after}(IN) \subseteq \mathsf{before}(e_0) \cup \mathsf{before}(\widehat{\mathcal{E}}) \tag{2}$$

$$\mathsf{after}(\widehat{\mathcal{E}}) \subseteq \mathsf{before}(\widehat{\mathcal{E}}) \cup \mathsf{before}(e_0) \tag{3}$$

$$\mathsf{after}(e_0) \subseteq \mathsf{before}(OUT) \tag{4}$$

$$\mathsf{after}(OUT) \subseteq \mathsf{before}(IN) \cup \mathsf{deadlocks}(\mathcal{E}) \tag{5}$$

$$\forall e, f \in \{IN, OUT, e_0\} \cdot e \neq f \Rightarrow \mathsf{before}(e) \cap \mathsf{before}(f) = \varnothing \tag{6}$$

$$\forall e \in \{IN, OUT\} \cdot \mathsf{before}(e) \cap \mathsf{before}(\widehat{\mathcal{E}}) = \varnothing \tag{7}$$

The requirement (1) states that the system initialisation should enable the event IN. The requirements (2)–(5) stipulate the desired execution order of the involved model events, informally presented in the control flow given above. Specifically, the event IN must be followed by the event e_0 or any of the new model events [the requirement (2)]. The new events may loop or terminate by enabling e_0 [the requirement (3)]. The event e_0 is followed by the event OUT [the requirement (4)]. Finally, the event OUT may enable a start of a new iteration by the event IN or put the whole system into a deadlock [the requirement (5)].

The last two requirements (6) and (7) stipulate that the guards of the events IN, OUT and e_0 as well as IN, OUT and any new event from $\widehat{\mathcal{E}}$ should be disjoint, i.e., they cannot be enabled at the same time. This allows us to guarantee that the presented control flow is strictly followed.

The presented formulation of additional proof obligations ensuring a specific control flow of events is inspired by the approach given in [Ili11].

3.3. Example: abstract Event-B model of a cyclic system

In this section we present an example of modelling a simple cyclic system in Event-B. It can be easily shown that the modelled system satisfies the given cyclic system requirements (1)–(7).

Our example is a simple monitoring system that consists of a sensor, which produces certain measurements, and a controller, which periodically reads the sensor outputs. The controller analyses each reading to detect whether the sensor functions properly or it has failed. If no fault is detected, the system outputs the obtained sensor reading and continues to iterate in the same way. This behaviour constitutes the fully operational state of the system. However, if the controller detects a sensor failure then the system enters the degraded state, during which the sensor may recover. In this state the controller keeps periodically reading the sensor outputs to detect whether it has recovered. The system can stay in the degraded state for a limited period of time (we assume it cannot exceed 3 iterations). If sensor recovers from its failure within the allowed time limit, the system gets back to the fully operational state and its normal function is resumed. Otherwise, the system aborts.

Figure 1 shows an abstract Event-B model (machine `CS`) of such a cyclic system. For the sake of simplicity, we omit a representation of the output produced by the system, while the sensor behaviour is abstractly modelled by nondeterministically changing its status. The variable $phase$ models the phases that the system goes through within one iteration: first reading the sensor, then detecting a sensor failure, and finally either outputting some sensor reading or aborting. The variable st models the sensor status. When $st = 1$, the sensor is operational. Otherwise, i.e., when $st = 0$, it has failed. The value of the variable cnt corresponds to the number of successive iterations when the sensor remains faulty. The unrecoverable system failure occurs when the cnt value exceeds 3. In this case the variable $phase$ gets the value $abort$ and the system deadlocks.

We will use the model `CS` and its refinements as the running example of this paper.

3.4. Observable states and iterations

While reasoning about quantitative properties of a cyclic system, we are usually interested in the number of iterations that the system can perform before it terminates. This observation allows us to focus only on those

```
Machine CS
Variables $st, phase, cnt$
Invariants $st \in \{0,1\},\ phase \in \{in, det, out, abort\},\ cnt \in \mathbb{N}$
Events

  $Initialisation \mathrel{\widehat{=}}$
    begin
      $st, phase, cnt := 1, in, 0$
    end

  $IN \mathrel{\widehat{=}}$
    when
      $phase = in$
    then
      $st :\in \{0,1\}$
      $phase := det$
    end

  $e_0 \mathrel{\widehat{=}}$
    when
      $phase = det$
    then
      $cnt :| (st = 1 \Rightarrow cnt' = 0) \wedge (st = 0 \Rightarrow cnt' = cnt + 1)$
      $phase := out$
    end
  $OUT \mathrel{\widehat{=}}$
    when
      $phase = out$
    then
      $phase :| phase' \in \{in, abort\} \wedge (phase' = in \Leftrightarrow cnt \leq 3)$
    end
```

Fig. 1. Event-B model of a cyclic system

system states where a system iteration starts and finishes. We call such system states *observable*. We also distinguish an important subset of the observable states called *operational* states. Usually, essential properties of the system (such as dependability, performance and safety properties) can be guaranteed only while the system stays in the operational states.

Definition 3 (*Observable states*). For Event-B models satisfying the requirements (1)–(7), we define the observable system states as a set containing all the states where an iteration of a cyclic system may start or finish, i.e.,

$$\Sigma_{obs} = \mathsf{before}(IN) \cup \mathsf{after}(OUT).$$

From the requirement (5), we can also conclude that

$$\Sigma_{obs} \subseteq \mathsf{before}(IN) \cup \mathsf{deadlocks}(\mathcal{E}). \tag{8}$$

Since $\mathsf{before}(IN) \cap \mathsf{deadlocks}(\mathcal{E}) = \varnothing$, (8) also suggests that the set of observable states can be partitioned into two disjoint subsets of operational and non-operational (or terminating) states:

$$\Sigma_{obs} = \Sigma_{op} \cup \Sigma_{nop},$$

where $\Sigma_{op} = \mathsf{before}(IN)$ and $\Sigma_{nop} = \mathsf{after}(OUT) \setminus \mathsf{before}(IN) \subseteq \mathsf{deadlocks}(\mathcal{E})$.

States that are not in Σ_{obs} are called *unobservable*. Intuitively, introduction of the system observable states Σ_{obs} means that, for the external observer of a cyclic system, the core part of a cyclic system is a "black box" and only starting and ending points of iterations are visible. Moreover, since in this paper we aim at stochastic reachability analysis of cyclic systems, we assume that the set Σ is finite.

Before we formally define the notion of an iteration for the proposed generalised Event-B model of a cyclic system, let us to formulate one useful lemma.

Lemma 1 *If an Event-B model satisfies the requirements* (1)–(7)*, all the model events from* $\{e_0\} \cup \widehat{\mathcal{E}}$ *are defined on unobservable system states only, i.e.,*

$$\forall e \in \{e_0\} \cup \widehat{\mathcal{E}} \cdot (\mathsf{before}(e) \cup \mathsf{after}(e)) \cap \Sigma_{obs} = \varnothing.$$

Proof We only show the proof for the event e_0. The corresponding proof for any e such that $e \in \widehat{\mathcal{E}}$ is similar. From the requirement (6), we immediately have that

$$\mathsf{before}(e_0) \cap \mathsf{before}(IN) = \varnothing.$$

Moreover, by the definition of $\mathsf{deadlocks}(\mathcal{E})$, the following is true:

$$\mathsf{before}(e_0) \cap \mathsf{deadlocks}(\mathcal{E}) = \varnothing.$$

From these two propositions and the property (8), we have

$$\mathsf{before}(e_0) \cap \Sigma_{obs} = \varnothing. \tag{9}$$

Similarly, from the requirements (4) and (6), we can conclude that

$$\mathsf{after}(e_0) \cap \mathsf{before}(IN) = \varnothing.$$

From the requirement (4) and the definition of $\mathsf{deadlocks}(\mathcal{E})$, we also get

$$\mathsf{after}(e_0) \cap \mathsf{deadlocks}(\mathcal{E}) = \varnothing.$$

From the last two propositions and the property (8), we have

$$\mathsf{after}(e_0) \cap \Sigma_{obs} = \varnothing. \tag{10}$$

Finally, from (9) and (10), we conclude that

$$(\mathsf{before}(e_0) \cup \mathsf{after}(e_0)) \;\cap\; \Sigma_{obs} = \varnothing.$$

□

Each iteration of a cyclic system maps the current operational system state $\sigma \in \Sigma_{op}$ into a subset of Σ_{obs}. The resulting set of states represents all possible states that can be reached due to the system nondeterministic behaviour. Formally, we can define it as follows:

Definition 4 (*Iteration*) An iteration of a cyclic system M is a total function mapping the set of operational states to the powerset of observable system states

$$iter \in \Sigma_{op} \rightarrow \mathbf{2}^{\Sigma_{obs}},$$

such that, for any subsequent observable states σ and σ' in some system execution trace $tr \in traces_M$, the following holds

$$\sigma' \in iter(\sigma).$$

Theorem 1 *If an Event-B model satisfies the requirements* (1)–(7)*, the modelled system is cyclic and its iteration function* $iter$ *can be defined on all the system operational states.*

Proof Let us first consider the case when $\widehat{\mathcal{E}} = \varnothing$.

In that case, the requirements (1)–(7) guarantee that the system repeatedly executes the events IN, e_0, and OUT in the fixed order. Moreover, Lemma 1 states that the intermediate states, i.e., the pre- and post-states of the event e_0, are unobservable. This means that we can treat the events IN, e_0, and OUT as a single event

$$IN;\; e_0;\; OUT,$$

where ";" denotes the relational composition operator.

Then we can define the function $iter$ as

$$iter(\sigma) \;=\; \mathsf{after}_\sigma(IN;\; e_0;\; OUT),$$

for any $\sigma \in \mathsf{before}(IN)$.

In the case of $\widehat{\mathcal{E}} \neq \varnothing$, we rely on the proof obligation (REF_VAR), which states that the new events cannot diverge. This allows us to represent the overall execution of the new events as a single composite event $(\widehat{e})^*$, where $\widehat{e} = \bigcup_{e \in \widehat{\mathcal{E}}} e$ and * denotes the transitive relational closure operator.

Moreover, the requirements (1)–(7) guarantee that the system is now repeatedly executed as a composite event

$$IN;\; (\widehat{e})^*;\; e_0;\; OUT,$$

when Lemma 1 again enforces that all the intermediate states from the constructed set

$$\bigcup_{e \in \{e_0\} \cup \widehat{\mathcal{E}}} (\mathsf{before}(e) \cup \mathsf{after}(e))$$

are unobservable. □

Similarly as above, we can now define the function $iter$ as

$$iter(\sigma) \;=\; \mathsf{after}_\sigma(IN;\; (\widehat{e})^*;\; e_0;\; OUT),$$

for any $\sigma \in \mathsf{before}(IN)$. □

Essentially, Theorem 1 postulates that, no matter what refinement steps are taken, the Event-B refinement process will preserve the cyclic nature of a given system, provided that the formulated requirements (1)–(7) are verified. This means that we can use the notion of a system iteration as a *discrete unit of time* defining the unified time scale for any Event-B machine in the refinement chain.

The given requirements for modelling cyclic systems defined above narrow down the class of considered Event-B models. However, this makes such models amenable for integrating stochastic reasoning about the system behaviour. To achieve this goal, we first propose a semantic extension of the Event-B language.

4. Stochastic modelling in Event-B

Hallerstede and Hoang [HH07] have extended the Event-B framework with a new operator—*qualitative probabilistic choice*, denoted $\oplus|$. This operator assigns new values to state variables with some positive but generally unknown probability. The proposed extension aims 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 only 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 [MM05]. Hence such an extension is not interfering with the established refinement process.

In our previous work [TTL10b], we have proposed extending the Event-B modelling language with *quantitative probabilistic choice*, also denoted $\oplus|$. The introduced operator allows us to represent a 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 choice (assignment) has the following syntax

$$x \oplus| \; x_1 \; @ \; p_1; \; \ldots; \; x_n \; @ \; p_n, \qquad \text{where} \quad \forall i \in 1..n \cdot 0 < p_i \leq 1 \quad \text{and} \quad \sum_{i=1}^{n} p_i = 1.$$

It assigns to the variable x a new value x_i with the corresponding probability p_i. Similarly to Hallerstede and Hoang, we have restricted the use of the new probabilistic choice operator by introducing it only to replace the existing demonic one. Therefore, we can rely on the Event-B proof obligations to guarantee functional correctness of a refinement step. Moreover, the probabilistic information introduced in new quantitative probabilistic choices can be used to stochastically evaluate certain non-functional system properties as well as their preservation during the refinement process.

To illustrate the proposed extension, in Fig. 2 we present a probabilistic refinement of the abstract machine CS. In CS, a possible sensor failure and its recovery are modelled nondeterministically. In PCS, we refine the event IN by two events IN_1 and IN_2, separating the cases of detecting a sensor failure and its recovery. Moreover, we replace nondeterministic assignments to the sensor status by probabilistic ones. Here, the non-zero constants f and r represent the probabilities of sensor's failure and recovery correspondingly. According to the theory of probabilistic refinement [MM05], the machine PCS is a refinement of the machine CS.

The proposed probabilistic choice operator allows us to introduce a specific probabilistic information into Event-B models and, as a result, model (at least some subset of) probabilistic systems. Our goal, however, is to integrate stochastic reasoning into the entire Event-B development process. In the next section we will show how the notion of Event-B refinement can be strengthened to quantitatively demonstrate that the refined system is "better" (e.g., more reliable or responsive) than its abstract counterpart.

5. Modelling fully probabilistic cyclic systems

In this section we present a theoretical basis for formal verification of probabilistic cyclic systems in Event-B. We rely on the structure and properties for cyclic systems introduced in Sect. 3.

```
Machine PCS
Variables $st, phase, cnt$
Invariants $st \in \{0,1\},\ phase \in \{in, det, out, abort\},\ cnt \in \mathbb{N}$
Events
  $Initialisation \mathrel{\widehat{=}}$
    begin
      $st, phase, cnt := 1, in, 0$
    end
  $IN_1 \mathrel{\widehat{=}}$
    when
      $phase = in \land st = 1$
    then
      $st \oplus|\ 0 @ f;\ 1 @ (1-f)$
      $phase := det$
    end
  $IN_2 \mathrel{\widehat{=}}$
    when
      $phase = in \land st = 0$
    then
      $st \oplus|\ 1 @ r;\ 0 @ (1-r)$
      $phase := det$
    end
  $e_0 \mathrel{\widehat{=}}$
    when
      $phase = det$
    then
      $cnt :| (st = 1 \Rightarrow cnt' = 0) \land (st = 0 \Rightarrow cnt' = cnt + 1)$
      $phase := out$
    end
  $OUT \mathrel{\widehat{=}}$
    when
      $phase = out$
    then
      $phase :| phase' \in \{in, abort\} \land (phase' = in \Leftrightarrow cnt \leq 3)$
    end
```

Fig. 2. Cyclic system: introducing probabilities

5.1. Probability distribution

Since an Event-B model is essentially a state transition system, we can simulate its execution by producing a tree of reachable states. Each path in such a tree corresponds to one operational trace, while tree branching occurs due the presence of nondeterminism in an Event-B model. If we replace a particular nondeterministic choice by a probabilistic one, we essentially attach concrete weights (probabilities) to separate branches, reflecting how likely a particular branch will be chosen for execution.

Based on that, we can distinguish between two types of modelled systems—fully probabilistic systems, i.e., the systems containing only probabilistic branching, and the systems that behave both probabilistically and nondeterministically. The absence of nondeterminism in a fully probabilistic system additionally imposes a certain restriction on its initialisation event. Specifically, it can be either deterministic or probabilistic assignment.

Let us first consider fully probabilistic systems. The quantitative information present in a probabilistic Event-B model allows us to lift the notion of the system state to that of a probabilistic distribution over the system state:

Definition 5 (*Probability distribution*). For the system observable state space Σ_{obs}, the set of distributions over Σ_{obs} is

$$\overline{\Sigma}_{obs} \quad \widehat{=} \quad \left\{ \Delta \in \Sigma_{obs} \to [0,1] \mid \sum_{\sigma \in \Sigma_{obs}} \Delta(\sigma) = 1 \right\}.$$

Each iteration of a fully probabilistic cyclic system maps some initial operational state to a subset of Σ_{obs} according to some probabilistic distribution, i.e., we can define a single iteration of a probabilistic cyclic system as a total function

$$piter \in \Sigma_{op} \to \overline{\Sigma}_{obs}.$$

There is a simple connection between the iteration $iter$ of a cyclic system and its probabilistic counterpart $piter$—if some state σ' can be reached from a current state σ by $piter$ with a non-zero probability then it is also reachable by $iter$:

$$\forall \sigma \in \Sigma_{op}, \sigma' \in \Sigma_{obs} \cdot piter(\sigma)(\sigma') > 0 \Rightarrow \sigma' \in iter(\sigma).$$

For any state $\sigma \in \Sigma_{op}$, its distribution Δ_σ (where $\Delta_\sigma = piter(\sigma)$) is calculated from probabilistic choice statements present in a model. However, once the system terminates, it stays in a terminating state forever. This means that, for any state $\sigma \in \Sigma_{nop}$, its distribution Δ_σ is such that $\Delta_\sigma(\sigma) = 1$ and $\Delta_\sigma(\sigma') = 0$, if $\sigma' \neq \sigma$.

5.2. Definition of quantitative refinement

While developing a complex software system, the designer often should define critical non-functional constraints, such as required dependability or performance properties. These constraints explicitly describe the desired parameters of the system functioning and must be then taken into account during the development process. In this paper we focus on reasoning about two such constraints—system *reliability* and *responsiveness* (*response time*).

Often, it is not possible to formally guarantee that the system always satisfies a desired critical property. However, we can still assess the probability that the property is preserved by the system at a certain moment. Currently, Event-B does not explicitly support the notions of time and probability. In the previous sections we proposed a general approach for modelling cyclic systems in Event-B, where the progress of time is modelled by system iterations. Moreover, we proposed the semantic extension of the modelling language that allows us to augment Event-B models with probabilistic information about the system behaviour. Based on this information, we can strengthen the notion of Event-B refinement by additionally requiring that refined models meet reliability and responsiveness requirements with a higher probability.

Let us first consider system reliability. In engineering, reliability is generally measured by the probability that an entity $\mathcal{X}$ can perform a required function under given conditions for the time interval [0, t]:

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

Hence, for cyclic systems, reliability can be expressed as the probability that the system remains *operational* during a certain number of iterations. Let $X(t)$ be a function that returns the system state after t-th iteration, where $t \in \mathbb{N}$, and $X(0)$ is an initial system state such that $X(0) \in \Sigma_{op}$. Since any system iteration may start from only an operational system state, it holds that $\forall t > 1 \cdot X(t) \in \Sigma_{op} \Rightarrow X(t-1) \in \Sigma_{op}$, i.e., if the system is operational after t iterations then it has remained operational for all previous iterations as well. Then we can formally define the system reliability as follows:

$$R(t) = \mathbf{P}\{X(t) \in \Sigma_{op}\}.$$

It is straightforward to see that this property corresponds to the standard definition of reliability given above. Thus, while modelling a cyclic system, we can strengthen the notion of Event-B refinement from the reliability point of view in the following way:

Definition 6 (*Reliability refinement*) Let M_a and M_c be two probabilistic Event-B models of cyclic systems. Moreover, let Σ^a_{op} and Σ^c_{op} be the sets of operational states of M_a and M_c correspondingly. Then we say that M_c is a reliability refinement of M_a if and only if

1. M_c is an Event-B refinement of M_a ($M_a \sqsubseteq M_c$), and
2. $\forall\, t \in \mathbb{N}^+ \cdot \mathbf{P}\{X_a(t) \in \Sigma^a_{op}\} \leq \mathbf{P}\{X_c(t) \in \Sigma^c_{op}\}.$ (11)

The second condition essentially requires that the system reliability cannot decrease during the refinement process.

Dually, for a cyclic system that terminates by providing some particular service to the customers, our goal is to assess the probability that this service will be provided during a certain time interval t. This property can be expressed as the probability that there exist such value $i \in 1..t$ that $X(i) \in \Sigma_{nop}$. Clearly, for any $j > i$, it holds that $X(j) \in \Sigma_{nop}$. Hence the probability that the system will *terminate* during the time interval [0, t]:

$$Q(t) = \mathbf{P}\{X(t) \in \Sigma_{nop}\}.$$

Therefore, from the responsiveness point of view, we can strengthen the definition of Event-B refinement by also requiring that the refined system should be at least as responsive as the abstract one:

Definition 7 (*Responsiveness refinement*) Let M_a and M_c be two probabilistic Event-B models of cyclic systems. Moreover, let Σ^a_{nop} and Σ^c_{nop} be the sets of non-operational states of M_a and M_c correspondingly. Then we say that M_c is a responsiveness refinement of M_a if and only if

1. M_c is an Event-B refinement of M_a ($M_a \sqsubseteq M_c$), and
2. $\forall\, t \in \mathbb{N}^+ \cdot \mathbf{P}\{X_a(t) \in \Sigma^a_{nop}\} \leq \mathbf{P}\{X_c(t) \in \Sigma^c_{nop}\}.$ (12)

The second condition essentially requires that the system responsiveness cannot decrease during the refinement process.

Remark If the second, quantitative refinement condition of Definitions 6 and 7 holds not for all t, but for some interval $t \in 1..T,\ T \in \mathbb{N}^+$, we say that M_c is a partial reliability (responsiveness) refinement of M_a for $t \leq T$.

In practice, the refinement conditions (11) and (12) usually hold for specific values of probabilistic characteristics of system components, often only for a limited time interval. The length of the interval for which the quantitative refinement must be preserved is defined by the intended operation time of the system under development. For instance, such a standard fault-tolerance scheme as triple modular redundancy is implemented (and, consequently, is a valid refinement of a single-module system) only if, during the intended operation time, the reliability of the single-module system is greater than 0.5. Hence in practice we usually rely on the partial quantitative refinement (Remark 5.2). For more details about probabilistic modelling and reliability assessment of various fault-tolerance schemes in Event-B, see [TTL10a].

5.3. Verification of quantitative refinement

To verify the first refinement condition of Definitions 6 and 7, we rely on the proof obligation rules that we discussed in Sect. 2. The tool support for Event-B—the Rodin platform—provides us with a means for generating and proving of all the required proof obligations, including the formalised requirements (1)–(7) for cyclic systems given in Sect. 3. However, it lacks the functionality needed to quantitatively verify refinement conditions (11) and (12). In this subsection we give a theoretical background that allows us to express the probabilistic reachability properties (11) and (12) in terms of the operational and non-operational states of cyclic systems.

Let us now consider in detail the behaviour of some fully probabilistic cyclic system M. We assume that the initial system state σ is determined by some probability distribution Δ_0 over the set of operational states Σ_{op} (which also covers the case of deterministic initialisation). After its first iteration, the system reaches some state $\sigma' \in \Sigma_{obs}$ with the probability $\Delta_\sigma(\sigma')$. At this point, if $\sigma' \in \Sigma_{nop}$, the system terminates. Otherwise, the system starts a new iteration and, as a result, reaches some state σ'' with the probability $\Delta_{\sigma'}(\sigma'')$, and so on. This process is completely defined by its state transition matrix P_M. More precisely, given that the state space Σ_{obs} is finite, we can enumerate it, i.e., assume that $\Sigma_{obs} = \{\sigma_1, \ldots, \sigma_n\}$, and define elements of the $n \times n$ transition matrix P_M as

$$\forall\, i, j \in 1..n \cdot P_M(\sigma_i, \sigma_j) \mathrel{\widehat{=}} \Delta_{\sigma_i}(\sigma_j) = \begin{cases} piter(\sigma_i)(\sigma_j) & \text{if } \sigma_i \in \Sigma_{op}, \\ 1 & \text{if } \sigma_i \in \Sigma_{nop} \text{ and } i = j, \\ 0 & \text{if } \sigma_i \in \Sigma_{nop} \text{ and } i \neq j. \end{cases}$$

In its turn, this matrix unambiguously defines the underlying Markov process—the absorbing discrete-time Markov chain [KS60], with the set of transient states Σ_{op} and the set of absorbing states Σ_{nop}. The state transition matrix of a Markov process together with its initial state allows us to calculate the probability that the defined Markov process, after a given number t of steps, will be in some particular state σ. Using the transition matrix P_M, we now can assess reliability and responsiveness of Event-B models as follows:

Proposition 1 *The reliability refinement condition* (11) *is equivalent to*

$$\forall\, t \in \mathbb{N}^+ \cdot \sum_{\sigma \in \Sigma^a_{op}} \left([\Delta^a_0] \cdot P^t_{M_a}\right)(\sigma) \leq \sum_{\sigma \in \Sigma^c_{op}} \left([\Delta^c_0] \cdot P^t_{M_c}\right)(\sigma),$$

where Σ^a_{op} and Σ^c_{op} are the sets of operational states of the systems M_a and M_c respectively, $[\Delta^a_0]$ and $[\Delta^c_0]$ are the initial state distribution row-vectors, and P^t is the matrix P raised to the power t.

Proof Let us first consider the DTMC M_a. The vector $[\Delta^a_0] \cdot P^t_{M_a}$ is a state probability vector of the chain M_a after t steps [KS60]. Hence by summing all the entries of this vector that correspond to the operational system states, we obtain the overall probability that the system is operational after t iterations. Then, clearly, $R_{M_a}(t) = \sum_{\sigma \in \Sigma^a_{op}} \left([\Delta^a_0] \cdot P^t_{M_a}\right)(\sigma)$. Similarly, $R_{M_c}(t) = \sum_{\sigma \in \Sigma^c_{op}} \left([\Delta^c_0] \cdot P^t_{M_c}\right)(\sigma)$, which proves Proposition 1. □

```
Machine RPCS
Variables $st1, st2, phase, cnt$
Invariants $st_1, st_2 \in \{0,1\},\ phase \in \{in, det, out, abort\},\ cnt \in \mathbb{N},\ \boxed{st = 0 \Leftrightarrow st_1 + st_2 = 0}$
Events
  $Initialisation \mathrel{\widehat{=}}$
    begin
      $st_1, st_2, phase, cnt := 1, 1, in, 0$
    end
  $IN_{12} \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st_1 = 1 \wedge st_2 = 1$
    then
      $st_1 \oplus|\ 0 @ f; 1 @ (1-f)$
      $st_2 \oplus|\ 0 @ f; 1 @ (1-f)$
      $phase := det$
    end
  $IN_{12} \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st_1 = 1 \wedge st_2 = 0$
    then
      $st_1 \oplus|\ 0 @ f;\ 1 @ (1-f)$
      $st_2 \oplus|\ 1 @ r;\ 0 @ (1-r)$
      $phase := det$
    end
  $IN_{13} \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st_1 = 0 \wedge st_2 = 1$
    then
      $st_1 \oplus|\ 1 @ r;\ 0 @ (1-r)$
      $st_2 \oplus|\ 0 @ f;\ 1 @ (1-f)$
      $phase := det$
    end
  $IN_2 \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st_1 = 0 \wedge st_2 = 0$
    then
      $st_1 \oplus|\ 1 @ r;\ 0 @ (1-r)$
      $st_2 \oplus|\ 1 @ r;\ 0 @ (1-r)$
      $phase := det$
    end
  $e_0 \mathrel{\widehat{=}}$
    when
      $phase = det$
    then
      $cnt :|\ (st_1 + st_2 > 0 \Rightarrow cnt' = 0) \wedge (st_1 + st_2 = 0 \Rightarrow cnt' = cnt + 1)$
      $phase := out$
    end
  $OUT \mathrel{\widehat{=}}$
    when
      $phase = out$
    then
      $phase :|\ phase' \in \{in, abort\} \wedge (phase' = in \Leftrightarrow cnt \leq 3)$
    end
```

Fig. 3. Cyclic system: probabilistic reliability refinement

Proposition 2 *The responsiveness refinement condition* (12) *is equivalent to*

$$\forall t \in \mathbb{N}^+ \cdot \sum_{\sigma \in \Sigma^a_{nop}} \left([\Delta^a_0] \cdot P^t_{M_a}\right)(\sigma) \;\leq\; \sum_{\sigma \in \Sigma^c_{nop}} \left([\Delta^c_0] \cdot P^t_{M_c}\right)(\sigma),$$

where Σ^a_{nop} and Σ^c_{nop} are the sets of terminating states of the systems M_a and M_c respectively, $[\Delta^a_0]$ and $[\Delta^c_0]$ are the initial state distribution row-vectors, and P^t is the matrix P raised to the power t.

Proof Similar to Proposition 1. □

To illustrate the use of our definitions of quantitative refinement in practice, let us revisit our simple example. In order to increase the reliability of our monitoring system, we implement a simple redundancy scheme by adding another (hot spare) sensor to the system. The machine RPCS (Fig. 3) is a result of refining the machine PCS by introducing such a hot spare sensor. In the refined specification, we replace the sensor st by two identical sensors st_1 and st_2. The probabilistic characteristics of these sensors are the same as those of st. The model gluing invariant $st = 0 \Leftrightarrow st_1 + st_2 = 0$ describes the refinement relationship between the corresponding abstract and concrete variables, i.e., the system would output the actual sensor readings only if no more than one sensor has failed.

Even for such simple models it is not trivial to find analytical representations of the reliability functions of PCS and RPCS. However, for some specific values of probabilistic parameters f and r, it is relatively easy to check whether the probabilistic Event-B model RPCS is a valid (partial) refinement of PCS (according to Definition 6 and Remark 5.2). For instance, Fig. 4 gives a comparison of the reliability functions $R_{\texttt{PCS}}$ and $R_{\texttt{RPCS}}$ for the specific case when $f = 10^{-3}$ and $r = 0.8$, and for the time interval $T = 5 \cdot 10^7$. The quantitative analysis was done using the PRISM probabilistic symbolic model checker [KNP11]. It is easy to see that the use of the hot spare redundancy scheme significantly increases the system reliability. In this particular case, the function $R_{\texttt{RPCS}}$ decreases very slowly and for $T = 5 \cdot 10^5$ its value is still very close to 1 (0.99995, to be more precise).

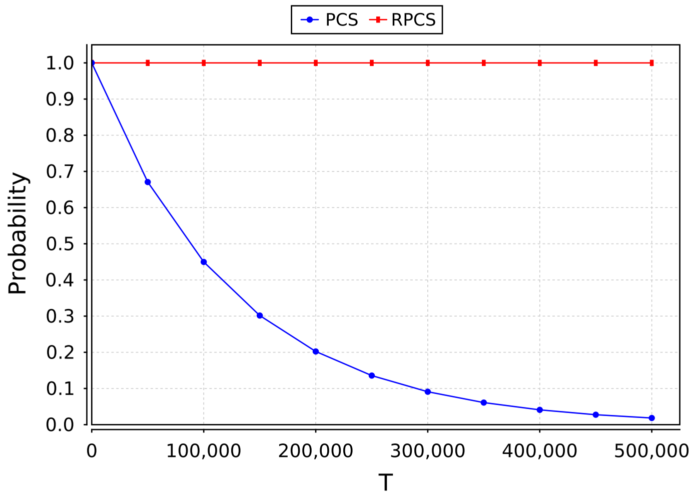

Fig. 4. Case study: reliability refinement for $f = 10^{-3}$ and $r = 0.8$

Let us now explain how probabilistic refinement from the reliability point of view differs from the traditional approach to probabilistic program refinement [MM05]. If we fix the number of system iterations t, the Event-B models PCS and RPCS can be considered as deterministic probabilistic programs according to [MM05]. More precisely, they can be viewed as two loops that can either terminate after t iterations in some state $\sigma \in \Sigma_{op}$ or deadlock after reaching some state $\sigma' \in \Sigma_{nop}$.

Let us define the set of operational states of PCS as $\Sigma_{op} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, where σ_0 is a fully operational state ($st = 1,\ cnt = 0$), while σ_k, for $k = 1, 2, 3$, are transient states ($st = 0,\ cnt = k$). Similarly, for RPCS, $\Sigma_{op} = \{\sigma_0^0, \sigma_0^1, \sigma_0^2, \sigma_1, \sigma_2, \sigma_3\}$, where σ_0^0, σ_0^1 and σ_0^2 are fully operational states ($st_1 = 1,\ st_2 = 1,\ cnt = 0$), ($st_1 = 1,\ st_2 = 0,\ cnt = 0$) and ($st_1 = 0,\ st_2 = 1,\ cnt = 0$) related to σ_0 by data refinement, while σ_k, for $k = 1, 2, 3$, are degraded states ($st_1 = 0,\ st_2 = 0,\ cnt = k$). The set of non-operational states Σ_{nop} consists of a single state σ_4, where σ_4 is ($st = 0,\ cnt > 3$) for PCS, or ($st_1 = 0,\ st_2 = 0,\ cnt > 3$) for RPCS. This set corresponds to a non-terminating state $\perp$ in terms of [MM05].

Let $P(\sigma, \sigma')$ be the probability that a program terminates in σ' after its execution starts from σ. Note that both programs PCS and RPCS have a single (feasible) initial state—σ_0 and σ_0^0 correspondingly, i.e., $\forall \sigma \in \Sigma_{op}, \sigma' \in \Sigma \cdot \sigma \neq \sigma_0 \, (\sigma \neq \sigma_0^0) \;\Rightarrow\; P(\sigma, \sigma') = 0$. For simplicity, let $P(\sigma_0, \sigma') = p(\sigma')$ and $P(\sigma_0^0, \sigma') = q(\sigma')$. Then, according to [MM05], the probabilistic program refinement $\mathtt{PCS} \sqsubseteq \mathtt{RPCS}$ is defined as follows:

$$\mathtt{PCS} \sqsubseteq \mathtt{RPCS} \;\;\Leftrightarrow\;\; p(\sigma_0) \leq \sum_{i=0}^{2} q(\sigma_0^i) \;\wedge\; (\forall\, k \in \{1, 2, 3\} \cdot\; p(\sigma_k) \leq q(\sigma_k)).$$

Let us now consider the case when $t = 1$. Then $p(\sigma_0) = 1 - f$, $p(\sigma_1) = f$, $\sum_{i=0}^{2} q(\sigma_0^i) = 1 - f^2$, $q(\sigma_1) = f^2$ and, finally, $p(\sigma_2) = p(\sigma_3) = q(\sigma_2) = q(\sigma_3) = 0$. Clearly, according to this definition, neither $\mathtt{PCS} \sqsubseteq \mathtt{RPCS}$ nor $\mathtt{RPCS} \sqsubseteq \mathtt{PCS}$ holds (as $p(\sigma_0) \leq \sum_{i=0}^{2} q(\sigma_0^i)$ and $p(\sigma_1) \geq q(\sigma_1)$).

It is easy to show that the traditional definition of probabilistic program refinement always implies the probabilistic refinement proposed in this paper (Definition 6 and Definition 7), yet its conditions are generally too strong for reasoning about such system properties as reliability and responsiveness. The example presented above is a quite typical refinement step for formal development of dependable systems. The refined model introduces a fault tolerance mechanism that does not only increases the system probability to stay in fully operational states as well as decreases the probability of a system failure, but also reduces the probability that the system goes to or remains in some degraded (but still operational) state. In other words, in our refinement we consider the set of operational states Σ_{op} as the whole and only concern with the overall probability that the system stays within this set, no matter in which particular state. In contrast, in the traditional approach to probabilistic program refinement, all the operational states are "equally important", meaning that, for each operational state, the refinement relation can only increase the probability to terminate in this state by correspondingly decreasing the probability of system failure (non-termination).

The presented simple example demonstrates how we can incorporate formal reasoning about system reliability into the refinement process in Event-B. Since system responsiveness is the (mathematically) dual property, it can be modelled and verified in a similar way. In the next section we generalise our approach to the systems that combine both nondeterministic and probabilistic behaviour.

6. Modelling probabilistic cyclic systems with nondeterminism

It is not always possible to give precise probabilistic information for all nondeterministic choices in a specification of a cyclic system. As a result, system models often contain a mixture of nondeterministic and probabilistic choices, making reasoning about such systems more complicated. In this section we offer our solution to this problem, which is a generalisation of our approach presented in Sect. 5.

6.1. Generalised definition of quantitative refinement

For a cyclic system containing both probabilistic and nondeterministic choices, we define a single iteration as a total function

$$npiter \in \Sigma_{op} \rightarrow \mathbf{2}^{\overline{\Sigma}_{obs}},$$

i.e., this function maps a given observable operational state, $\sigma \in \Sigma_{op}$, into a *set of distributions* over the observable state space Σ_{obs}. The resulting set of distributions is built for all possible combinations of nondeterministic choices (i.e., execution traces) of a single system iteration.

Similarly as in the case of a fully probabilistic system, there is a simple connection between the iteration $iter$ of a cyclic system and its nondeterministic-probabilistic counterpart $npiter$. Specifically, if some state σ' can be reached from a current state σ with a non-zero probability according to some distribution from $npiter(\sigma)$ then it is also reachable by $iter$:

$$\forall \sigma \in \Sigma_{op}, \sigma' \in \Sigma_{obs} \cdot (\exists \Delta_\sigma \in npiter(\sigma) \cdot \Delta_\sigma(\sigma') > 0) \Rightarrow \sigma' \in iter(\sigma).$$

Now let us consider the behaviour of some nondeterministic-probabilistic cyclic system M in detail. We can assume that the initial system state σ belongs to the set of operational states Σ_{op}. After its first iteration, the system nondeterministically chooses some distribution Δ_σ from $npiter(\sigma)$ and then, according to this distribution, reaches some state σ' with the non-zero probability $\Delta_\sigma(\sigma')$. At this point, if $\sigma' \in \Sigma_{nop}$, the system terminates. Otherwise, the system starts a new iteration. It is easy to see that the behavioural semantics of a nondeterministic-probabilistic cyclic system in Event-B is defined by a Markov decision process with the absorbing set Σ_{nop} [Put05, Whi93].

Nondeterminism has the demonic nature in Event-B [BvW98], i.e., we do not have any control or information about which branch of execution will be chosen. Therefore, while reasoning about system reliability and responsiveness, we have to consider the worst-case scenario by always choosing the "worst" of available distributions. From the reliability and responsiveness perspective, it means that, while evaluating these properties of a probabilistic cyclic system with nondeterminism, we need to obtain the lowest bound of the performed evaluation. Therefore, we re-formulate the definitions of the reliability and responsiveness refinement for probabilistic cyclic systems as follows:

Definition 8 (*Reliability refinement*). Let M_a and M_c be two nondeterministic-probabilistic Event-B models of cyclic systems. Moreover, let Σ_{op}^a and Σ_{op}^c be the sets of operational states of M_a and M_c correspondingly. Then we say that M_c is a reliability refinement of M_a if and only if

1. M_c is an Event-B refinement of M_a ($M_a \sqsubseteq M_c$), and

2. $\forall t \in \mathbb{N}^+ \cdot \mathbf{P}_{min}\{X_a(t) \in \Sigma_{op}^a\} \leq \mathbf{P}_{min}\{X_c(t) \in \Sigma_{op}^c\}$, (13)

where $\mathbf{P}_{min}\{X(t) \in \Sigma_{op}\}$ is the minimum probability that the system remains operational during the first t iterations.

Definition 9 (*Responsiveness refinement*) Let M_a and M_c be two nondeterministic-probabilistic Event-B models of cyclic systems. Moreover, let Σ^a_{nop} and Σ^c_{nop} be the sets of non-operational states of M_a and M_c correspondingly. Then we say that M_c is a responsiveness refinement of M_a if and only if

1. M_c is an Event-B refinement of M_a ($M_a \sqsubseteq M_c$), and

2. $$\forall t \in \mathbb{N}^+ \cdot \mathbf{P}_{min}\{X_a(t) \in \Sigma^a_{nop}\} \le \mathbf{P}_{min}\{X_c(t) \in \Sigma^c_{nop}\}, \tag{14}$$

where $\mathbf{P}_{min}\{X(t) \in \Sigma_{nop}\}$ is the minimum probability that the system terminates during the first t iterations.

Remark If the second, quantitative refinement condition of Definitions 8 and 9 holds not for all t, but for some interval $t \in 1..T,\ T \in \mathbb{N}^+$, we say that M_c is a partial reliability (responsiveness) refinement of M_a for $t \le T$.

6.2. Verification of generalised quantitative refinement

To evaluate the worst-case reliability and responsiveness for probabilistic systems with nondeterminism, we have to calculate the minimum probabilities participating in (13) and (14). Let us assume that some cyclic system M is in an operational state σ, while *npiter* maps σ to the *finite* set of distributions $\overline{\Delta}_\sigma = \{\Delta^1_\sigma, \Delta^2_\sigma, \ldots\}$. If we want to evaluate the worst-case reliability of the system for this iteration, we just have to choose the distribution that maps σ to the set of operational states with the minimal probability, i.e., the probability $\min_{\Delta \in \overline{\Delta}_\sigma} \sum_{\sigma' \in \Sigma_{op}} \Delta(\sigma')$. Similarly, to evaluate the worst-case responsiveness of the system, we have to choose the distribution that maps σ to the set of terminating states with the minimal probability, i.e., the probability $\min_{\Delta \in \overline{\Delta}_\sigma} \sum_{\sigma' \in \Sigma_{nop}} \Delta(\sigma')$. However, when the goal is to evaluate the worst-case stochastic behaviour of the system within a time interval [0, t], where $t > 1$, the calculation process of the resulting minimal probability becomes more complex. Indeed, because of the presence of nondeterminism, we cannot simply rely on the calculated minimal probability for t iterations when calculating it for $t + 1$ iterations.

Let us first consider evaluation of system reliability. We define the worst-case reliability as a function $r(t, \sigma)$, the arguments of which are the number of system iterations t and some initial state σ. For now, we assume that the initial system state σ is deterministically defined. Later we consider more general cases when σ is given by some initial probability distribution or nondeterministically. The function $r(t, \sigma)$ returns the minimal value of the reliability function $R(t)$ over all possible state distributions.

The definition of $r(t, \sigma)$ is recursive. Two basic cases define the function values for the terminating (absorbing) states and the zero number of iterations respectively:

$$\sigma \in \Sigma_{nop} \Rightarrow \forall t \in \mathbb{N} \cdot r(t, \sigma) = 0 \quad \text{and} \quad \sigma \in \Sigma_{op} \Rightarrow r(0, \sigma) = 1.$$

Generally, for $\sigma \in \Sigma_{op}$, we can recursively define the function $r(t, \sigma)$ in the following way:

$$\forall t \in \mathbb{N}^+, \sigma \in \Sigma_{op} \cdot r(t, \sigma) = \min_{\Delta \in \overline{\Delta}_\sigma} \sum_{\sigma' \in \Sigma_{op}} \Delta(\sigma') \cdot r(t-1, \sigma').$$

Such an approach for defining an "absorbing" function is often used in the works based on Markov decision processes with absorbing sets (see [HW03] for instance). Note that the recursive function application essentially traverses all the possible operational state transitions and, based on that, operational state distributions, and then finds the minimal probability of the system staying operational.

Similarly, the stochastic evaluation of system responsiveness is based on the recursive function $q(t, \sigma)$. It returns the minimal probability of the system terminating within the time interval [0, t], when starting in the observable state σ. The basic cases are

$$\sigma \in \Sigma_{nop} \Rightarrow \forall t \in \mathbb{N} \cdot q(t, \sigma) = 1 \quad \text{and} \quad \sigma \in \Sigma_{op} \Rightarrow q(0, \sigma) = 0.$$

Generally, for $\sigma \in \Sigma_{op}$, we can recursively define the function $q(t, \sigma)$ in the following way:

$$\forall t \in \mathbb{N}^+, \sigma \in \Sigma_{op} \cdot q(t, \sigma)$$
$$= \min_{\Delta \in \overline{\Delta}_\sigma} \left[\sum_{\sigma' \in \Sigma_{op}} \Delta(\sigma') \cdot q(t-1, \sigma') + \sum_{\sigma' \in \Sigma_{nop}} \Delta(\sigma') \right] = \min_{\Delta \in \overline{\Delta}_\sigma} \sum_{\sigma' \in \Sigma_{obs}} \Delta(\sigma') \cdot q(t-1, \sigma').$$

Now we are ready to revisit our definitions of reliability and responsiveness refinement for probabilistic cyclic systems with nondeterminism. For such a cyclic system M, let us define column-vectors r_M^t and q_M^t with the elements $r_M^t(\sigma) = r(t, \sigma)$ and $q_M^t(\sigma) = q(t, \sigma)$ respectively. Now, assuming that the initial state of the system is not defined deterministically but given instead by some initial state distribution, we can formulate two propositions similar to Propositions 1 and 2 of Sect. 5:

Proposition 3 *Let us assume that the initial system state is defined according to some probability distribution. Then the reliability refinement condition* (13) *is equivalent to*

$$\forall t \in \mathbb{N}^+ \cdot [\Delta_0^a] \cdot r_{M_a}^t \ \le \ [\Delta_0^c] \cdot r_{M_c}^t,$$

where $[\Delta_0^a]$ *and* $[\Delta_0^c]$ *are the initial state distribution row-vectors for the systems* M_a *and* M_c *respectively.*

Proof Directly follows from our definition of r_M^t. □

Proposition 4 *Let us assume that the initial system state is defined according to some probability distribution. Then the responsiveness refinement condition* (14) *is equivalent to*

$$\forall t \in \mathbb{N}^+ \cdot [\Delta_0^a] \cdot q_{M_a}^t \ \le \ [\Delta_0^c] \cdot q_{M_c}^t,$$

where $[\Delta_0^a]$ *and* $[\Delta_0^c]$ *are the initial state distribution row-vectors for the systems* M_a *and* M_c *respectively.*

Proof Directly follows from our definition of q_M^t. □

In Sect. 5 we considered the machine initialisation is to be either deterministic or probabilistic. However, for the systems that contain both probabilistic and nondeterministic behaviour, we can also assume that we do not have precise information about the system initial state, i.e., the initialisation action is of the following form: $\sigma :\in S$, where $S \subseteq \Sigma_{op}$. In this case we can formulate the following two propositions:

Proposition 5 *Let us assume that the initial system state is defined nondeterministically. Then the reliability refinement condition* (13) *is equivalent to*

$$\forall t \in \mathbb{N}^+ \cdot \min_{\sigma \in S_a} r(t, \sigma) \ \le \ \min_{\sigma \in S_c} r(t, \sigma),$$

where S_a *and* S_c *are the sets of possible initial states for the systems* M_a *and* M_c *respectively.*

Proof Directly follows from our definition of $r(t, \sigma)$ and properties of the demonic nondeterminism. □

Proposition 6 *Let us assume that the initial system state is defined nondeterministically. Then the responsiveness refinement condition* (14) *is equivalent to*

$$\forall t \in \mathbb{N}^+ \cdot \min_{\sigma \in S_a} q(t, \sigma) \ \le \ \min_{\sigma \in S_c} q(t, \sigma),$$

where S_a *and* S_c *are the sets of possible initial states for the systems* M_a *and* M_c *respectively.*

Proof Directly follows from our definition of $q(t, \sigma)$ and properties of the demonic nondeterminism. □

We can also easily show that the proposed approach for modelling of probabilistic systems with nondeterminism is a generalisation of the approach presented in the previous section. Indeed, let us assume that we do not have any nondeterministic choices in our system. This means that, for any state σ, the set $\overline{\Delta}_\sigma$ is a singleton set, i.e., $\forall \sigma \in \Sigma_{op} \cdot \overline{\Delta}_s = \{\Delta_\sigma\}$. Then

$$r(t, \sigma) = \min_{\Delta \in \overline{\Delta}_\sigma} \sum_{\sigma' \in \Sigma_{op}} \Delta(\sigma') \cdot r(t-1, \sigma') = \sum_{\sigma' \in \Sigma_{op}} \Delta_\sigma(\sigma') \cdot r(t-1, \sigma') = R(t)$$

```
Machine NPCS
Variables $st, phase, cnt$
Invariants $st \in \{0,1\},\ phase \in \{in, det, out, abort\},\ cnt \in \mathbb{N}$
Events
  $Initialisation \mathrel{\widehat{=}}$
    begin
      $st, phase, cnt := 1, in, 0$
    end
  $IN_1 \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st = 1$
    then
      $st \oplus\!\mid\ 0 \ @\ f; 1 \ @\ (1-f)$
      $phase := det$
    end
  $IN_{21} \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st = 0$
    then
      $st \oplus\!\mid\ 1 \ @\ r_1; 0 \ @\ (1-r_1)$
      $phase := det$
    end
  $IN_{22} \mathrel{\widehat{=}}$
    when
      $phase = in \wedge st = 0$
    then
      $st \oplus\!\mid\ 1 \ @\ r_2; 0 \ @\ (1-r_2)$
      $phase := det$
    end
  $e_0 \mathrel{\widehat{=}}$
    when
      $phase = det$
    then
      $cnt :\mid (st = 1 \Rightarrow cnt' = 0) \wedge (st = 0 \Rightarrow cnt' = cnt + 1)$
      $phase := out$
    end
  $OUT \mathrel{\widehat{=}}$
    when
      $phase = out$
    then
      $phase :\mid phase' \in \{in, abort\} \wedge (phase' = in \Leftrightarrow cnt \leq 3)$
    end
```

Fig. 5. Cyclic system: combining probabilities with nondeterminism

and

$$q(t,\sigma) = \min_{\Delta \in \overline{\Delta}_\sigma} \sum_{\sigma' \in \Sigma_{obs}} \Delta(\sigma') \cdot q(t-1,\sigma') = \sum_{\sigma' \in \Sigma_{obs}} \Delta_\sigma(\sigma') \cdot q(t-1,\sigma') = Q(t).$$

Moreover, for fully probabilistic systems, we can easily prove (by induction) that

$$\forall t \in \mathbb{N}, \sigma \in \Sigma_{op} \cdot r(t,\sigma) + q(t,\sigma) = 1.$$

To demonstrate the use of our extended definitions of quantitative refinement, let us revisit our simple example. Figure 5 shows a model that refines the cyclic system PCS, yet combines both nondeterministic and probabilistic behaviour. Here we refine the event IN_2 of PCS by two events IN_{21} and IN_{22} that respectively model sensor recovery according to two different recovery procedures (i.e., two different repairmen). The choice between the active repairmen is modelled nondeterministically as IN_{21} and IN_{22} are always enabled simultaneously.

Once the sensor fails, the we have to assume that the "demon" follows the worst-case scenario strategy and, therefore, always chooses the worst repairman to repair the sensor, i.e., the event with the probability of recovery $\min\{r_1, r_2\}$, while the better repairman is totally ignored. This observation gives some intuition about the demonic nature of nondeterminism in Event-B. Furthermore, since the event IN_1, which models the sensor failure, is the same in both machines, to prove that NPCS is a valid refinement of PCS (according to Definition 8), it is enough to guarantee that in NPCS the sensor has the better recovery mechanism comparing to that of PCS. Clearly, this is true if and only if $r \leq \min\{r_1, r_2\}$.

7. Discussion

In this paper, we have proposed a pragmatic approach to integrating stochastic reasoning into the formal development of dependable systems in Event-B. This work builds mainly on the experience in the development of dependable systems gained within the EU projects RODIN [ROD04] and DEPLOY [DEP08]. These projects have made a significant contribution into establishing the formal dependability-explicit development process based on Event-B refinement.

Via abstraction, proof and incremental correctness-preserving refinement, Event-B allows us to build and verify models of large-scale systems. The confidence in the system design that is achieved by such a formal approach makes a significant contribution into achieving system dependability. However, ensuring dependability requires not only functional correctness but also satisfaction of the given non-functional characteristics. For instance, to certify a dependable system, it is often required to demonstrate that the probability of a dangerous failure is sufficiently low, the system reliability meets the required target, etc. Though such an evaluation could be conducted at the final stage of the development process, the cost of redevelopment can be significant if the desired characteristics are not achieved.

We argue that the approach proposed in this paper allows us to weave the probabilistic evaluation into the formal system development from the early design stages. It results in a more predictive development process, which enables continuous monitoring of the progress towards achieving the desired dependability attributes as well as allows the designers to optimise certain design decisions.

Since reliability and responsiveness are functions of time, in this paper we first had to address the problem of representing time in Event-B. Instead of modelling time explicitly, i.e., as an intrinsic part of an Event-B model, we have decided to reason about time implicitly by relying on the notion of a system iteration. This decision was motivated by several reasons. Firstly, the current approaches to modelling time in Event-B are not well-suited for bridging Event-B machines and discrete-time Markov processes. For instance, the approach by Iliasov et al. [ILT$^+$12] relies on building an additional model view (called the process view) in a dedicated modelling language and, hence, makes it hard to relate probabilistic Event-B and the process view models. Moreover, there is currently no automatic tool support for verification of timed systems in Event-B. The approach proposed by Butler et al. [BF02] significantly complicates the system model and hence raises the question of scalability. As a result, we have decided to narrow down the class of modelled systems to cyclic systems and rely on the notion of an iteration as a discrete unit of time defining the system time scale. We have also formally defined the verification conditions ensuring that the initial abstract model as well as all subsequent refined models preserve the cyclic system behaviour.

To enable stochastic reasoning in Event-B, we have introduced the quantitative probabilistic choice operator into the modelling language and also defined the quantitative refinement conditions that a refined model should satisfy. These conditions ensure that responsiveness and reliability are improved (or at least preserved) in the refinement process, as postulated in Definitions 6 – 9. The resulting integration of stochastic reasoning about the non-functional system properties with the modelling and verification of the system functionality facilitates the formal development process in Event-B. Indeed, it supports early assessment of the chosen design decisions and possible design alternatives with respect to the non-functional critical properties. Our approach to verification of quantitative model refinement is based on time-bounded reachability analysis of the underlying Markov processes.

In our work, we have aimed at finding a practical engineering solution to assessing dependability properties throughout the formal system development by refinement. The main goals that we have pursued were applicability of the proposed technique and possibility to re-use constructed Event-B models. To ensure applicability, we have had to address three main issues—$scalability$, $simplicity\ and\ availability\ of\ automatic\ toolsupport$.

In this paper, we focus on studying formal basis of integrating probabilistic reasoning into Event-B via bridging it with discrete-time Markov processes. We argue that, since our probabilistic extension is rather conservative, the scalability of the presented approach is characterised by the scalability of the Event-B method and the developer's choice of a suitable technique for analysis of Markov processes. The accumulated experience in industrial use of Event-B within the EU projects RODIN and DEPLOY has shown that the refinement approach offers a powerful tool for mastering system complexity. On the probabilistic side, to handle complexity posed by the size of the state space of large-scale systems, we can employ such techniques as lumping and probabilistic bisimulation (see, e.g., [KS60, LS91] for fully probabilistic systems and [Han95, SL95] for nondeterministic probabilistic systems).

The small running example of the paper was used merely to illustrate the proposed approach. We believe that large case studies that employ our continuous-time probabilistic extension of Event-B [TTL12] to some extent also demonstrate the scalability of the approach proposed in this paper. Thus, in [TPT$^+$12] we have undertaken a formal modelling of a satellite subsystem. At a certain stage of the refinement process, we introduced two alternative architectures for implementing fault tolerance—a duplicated architecture with a dynamic reconfiguration and a triplicated architecture. They both were fulfilling the same functional requirements but had different reliability and performance characteristics. The probabilistic analysis has allowed us to optimise the system design, i.e., to establish that even with the lower degree of redundancy the target reliability and performance parameters can be achieved. A similar study was performed in the context of fault tolerant multi-robotic systems [TPTL13]. We have assessed several alternative architectures to find the optimal balance between the likelihood of successful completion of the mission, the degree of redundancy, and system performance. In the future we plan to validate the applicability of the approach presented in this paper by performing complex case studies from different industrial domains.

To address the second issue, we proposed a new definition of reliability refinement. In contrast to the traditional definition of probabilistic refinement, we do not require from the designers to establish a refinement relation to prove each refinement step. Instead, one can reuse the probabilistically augmented Event-B specifications and employ the existing automated techniques for analysis of Markov models to compute the system reliability for specific values of the system's probabilistic characteristics and, therefore, to check the preservation of the quantitative refinement. Moreover, as we have already mentioned, in practice it is usually enough to check the partial refinement condition for a particular time bound specified by the intended operation time of the system.

To address the third issue, we can rely on the Rodin Platform [Rod]—an open extendable environment for modelling and verification in Event-B. Openness and extendability of the platform allow us to build a tool support—a dedicated plug-in that would facilitate the calculations presented in Sects. 5 and 6. The theoretical research described in this paper can be seen as a basis for creating such a tool. The approach proposed in this paper demonstrates how to conduct dependability assessment based on Event-B models. We have aimed at maximising the reuse of the created formal models and hence proposed a simple solution that requires augmenting the Event-B models with probabilistic information but does not require creating dedicated models for probabilistic reasoning. Indeed, the most time consuming and error-prone part of the probabilistic reliability assessment is the creation of an adequate system model. However, within our approach this part is taken care by the Event-B refinement process. Our experiments [TPT$^+$12, TTL12, TPTL13] with the PRISM probabilistic symbolic model checker [KNP11] show that such models can serve as an input for probabilistic model checking. In its turn, it offers the system developers a highly automated solution to the quantitative evaluation of dependability.

8. Related work

Hallerstede and Hoang [HH07] have proposed an extension of the Event-B framework to model the probabilistic system behaviour. Specifically, they introduce the qualitative probabilistic choice operator to reason about almost certain termination. This operator is used to bound demonic nondeterminism, and to facilitate proving convergence of the new events in Event-B models. In particular, they apply this technique to resolve the contention problem in Fireware protocol. In [YH10], Yilmaz and Hoang also successfully apply the qualitative probabilistic reasoning in Event-B to formalise the Rabin's choice coordination algorithm. The use of the qualitative probabilistic choice is currently supported by the Rodin tool [EBW]. However, the presented approach is not suitable for quantitative evaluation of system properties, since the introduced operator does not contain explicit probabilistic information.

The topic of probabilistic formal modelling has been extensively explored by Morgan et al. in the context of probabilistic refinement calculus [MM05]—an extension of the standard refinement calculus. The introduced notion of probabilistic data refinement has been used, among other things, for assessment of system dependability (see [MMT98, MM05], for instance). Here, probabilistic programs are modelled using expectation transformers and probabilistic data refinement is verified via simulation between datatypes. In [Tro99], a similar approach is taken to enable reasoning about reliability in probabilistic action systems [ST96]—the extension of the action systems that combine both probabilistic and nondeterministic behaviour. However, proving simulation that implies data refinement between datatypes is an extremely difficult problem, which immediately raises the scalability issue. Moreover, the majority of non-functional system attributes, including those of dependability, explicitly depend on time. However, to the best of our knowledge, the notion of time is not defined in the probabilistic refinement calculus. The difference between these approaches to definition of probabilistic refinement and the one proposed in the current paper is addressed in Sect. 5.3.

Incorporation of probabilities into formal program semantics has also been investigated within the Unifying Theories of Programming (UTP) framework [HS06, BB12, SZ13]. In [BB12], the authors present an encoding of the semantics of the probabilistic guarded command language (pGCL) in UTP. The main contribution is a proposal to model pGCL programs as predicate transformers, where predicates are defined over probability distributions on before- and after-states. The proposed formulation also allows them to define a generic choice construct, covering conditional, nondeterministic and probabilistic choices. A long-term goal of this work is to develop a probabilistic extension of the Circus language, which is a fusion of Z, CSP, and refinement calculus. In [SZ13], the authors extend their previously developed theory of reversible computation by adding probabilistic choice. They also adapt and enhance the classical calculus of expectation transformers of McIver and Morgan [MM05] to enable reversible computations. Interactions of nondeterministic choice and probabilistic choice as well as feasibility and probabilistic choice are investigated as well. Similarly as in [MM05], probabilistic programs are identified with their convex closures, while refinement on such programs is defined as containment within these closures. In contrast to these approaches, we focus on formal modelling and refinement of probabilistic programs with respect to specific quantitative system characteristics (such as reliability and responsiveness), which are dependent on time. Our definition of probabilistic refinement allows redistribution of probabilities within the targeted operational states, provided that the overall probability of being in such states increases (decreases).

In [Rao95], Rao has proposed a generalisation of the UNITY formalism [CM88] that enables reasoning about probability and parallelism. Specifically, he generalises the weakest precondition semantics of UNITY to define a new predicate transformer—the weakest probabilistic precondition transformer. Relying on this extension, he also generalises certain relations of the UNITY to make them amenable for reasoning about probabilistic (parallel) programs. In particular, the new probabilistically leads-to relation allows for defining probabilistic progress properties. Similarly to the approach taken in [HH07], Rao does not aim at computing any kind of probabilistic measures but rather reasons about the progress properties that are attained with probability one. The proposed methodology has proved its worth in constructing and proving probabilistic algorithms [Rao95].

A connection between probabilistic reasoning and program refinement has been investigated by Meinicke and Solin [MS10]. The authors introduce a refinement algebra for reasoning about probabilistic program transformations. In particular, they investigate the data and atomicity refinement rules for probabilistic programs and explore the difference between probabilistic and non-probabilistic programs. They reason about the probabilistic program transformations without introducing a probabilistic choice operator or other explicit probabilistic attributes. Our approach is rather different from the one by Meinicke and Solin. We introduce the quantitative probabilistic choice operator, which explicitly defines concrete probabilistic values for different choices. The introduced probabilistic information is used to verify quantitative non-functional properties of the system and their preservation by refinement. Otherwise, we rely on the existing Event-B refinement framework to guarantee correctness of model transformations.

Ensuring a specific control flow within Event-B models have been investigated in [STW10]. The authors explore how process algebra descriptions can be defined alongside Event-B models and, in particular, how CSP can be used to provide explicit control flow for an Event-B model. Moreover, the methodology is proposed to construct models in a certain way to ensure the desired control flow and avoid possible deadlocks. The approach can be considered as complementary to ours. We are not interested in constructing formal models with the required control flow per se, but rather focus on verifying that given arbitrary models satisfy essential properties of cyclic systems.

9. Conclusions

In this paper we have proposed a pragmatic approach to integrating the stochastic reasoning about dependability (in particular reliability and responsiveness) of cyclic systems into the formal development by refinement in Event-B. We have made a number of technical contributions. Firstly, we have formally defined the conditions that should be verified to ensure that the system under construction has a cyclic behaviour. Secondly, we have proposed an extension of the Event-B language with the quantitative probabilistic choice construct and defined the proof semantics for the extended framework. Finally, we have demonstrated how to define reliability and responsiveness as the properties of extended Event-B models and integrate explicit stochastic reasoning about non-functional system properties into the Event-B refinement process.

The main novelty of our work is in establishing theoretical foundations for reasoning about probabilistic properties of augmented Event-B models. This result has been achieved by constraining the structure of considered Event-B models and consequently reducing the reasoning about time-dependent properties in general to the reasoning about these properties in terms of iterations. Since cyclic systems constitute a large class of critical systems, we believe that the imposed restrictions do not put significant limitations on the applicability of the proposed approach, yet at the same time allow us to represent the system models as discrete-time Markov chains or Markov decision processes. This, in its turn, enables the use of the well-established theory of Markov processes to verify time-bounded reachability properties.

References

[Abr96] Abrial J-R (1996) Extending B without changing it (for developing distributed systems). In: Habiras H (ed) First Conference on the B method, pp 169–190

[Abr05] Abrial J-R (2005) The B-Book: assigning programs to meanings. Cambridge University Press, Cambridge

[Abr10] Abrial J-R (2010) Modeling in Event-B. Cambridge University Press, Cambridge

[ALRL04] Avizienis A, Laprie J-C, Randell B, Landwehr CE (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur Comput 1(1):11–33

[BB12] Bresciani R, Butterfield A (2012) A UTP semantics of pGCL as a homogenous relation. In: IFM 2012. Springer, Belin, pp 191–205

[BF02] Butler M, Falampin J (2002) An approach to modelling and refining timing properties in B. In: Refinement of critical systems (RCS)

[BvW98] Back RJR, von Wright J (1998) Refinement calculus: a systematic introduction. Springer, Berlin

[CM88] Mani Chandy K, Misra J (1988) Parallel program design: a foundation. Addison-Wesley, USA

[CS88] Chu WW, Sit C-M (1988) Estimating task response time with contentions for real-time distributed systems. In: Real-Time Systems Symposium. IEEE Computer Society, pp 272–281

[DEP08] DEPLOY (2008) Industrial deployment of system engineering methods providing high dependability and productivity. IST FP7 IP Project. Online at http://www.deploy-project.eu/

[EBW] Event-B and Rodin Documentation Wiki. Qualitative probability plug-in. Online at http://wiki.event-b.org/index.php/Event-B_Qualitative_Probability_User_Guide.

[Han95] Hansson H (1995) Time and probability in formal design of distributed systems. Elsevier, London

[HH07] Hallerstede S, Hoang TS (2007) Qualitative probabilistic modelling in Event-B. In: Davies J, Gibbons J (eds) IFM 2007, Integrated Formal Methods. Springer, Berlin, pp 293–312

[HS06] He J, Sanders JW (2006) Unifying probability. In: Proceedings of First International Symposium on Unifying Theories of Programming, UTP 2006. Springer, New York, pp 173–199

[HW03] Hinderer K, Waldmann T-H (2003) The critical discount factor for finite Markovian decision processes with an absorbing set. In: Mathematical methods fo operations research. Springer, Berlin, pp 1–19

[Ili11] Iliasov A (2011) Use case scenarios as verification conditions: Event-B/Flow approach. In: SERENE 2011, Software Engineering for Resilient Systems. Springer, Berlin, pp 9–23

[ILT$^+$12] Iliasov A, Laibinis L, Troubitsyna E, Romanovsky A, Latvala T (2012) Augmenting Event-B modelling with real-time verification. In: FormSERA 2012, Formal Methods in Software Engineering: Rigorous and Agile Approaches. IEEE Press, New York, pp 51–57

[KNP11] Kwiatkowska M, Norman G, Parker D (2011) PRISM 4.0: Verification of probabilistic real-time systems. In: CAV'11, International Conference on Computer Aided Verification. Springer, Berlin, pp 585–591

[KS60] Kemeny JG, Snell JL (1960) Finite Markov chains. D. Van Nostrand Company, Princeton, NJ

[LS91] Larsen KG, Skou A (1991) Bisimulation through probabilistic testing. In: Information and Computation 94, pp 1–28

[MM05] McIver AK, Morgan CC (2005) Abstraction, refinement and proof for probabilistic systems. Springer, New York

[MMT98] McIver AK, Morgan CC, Troubitsyna E (1998) The probabilistic steam boiler: a case study in probabilistic data refinement. In: International Refinement Workshop, ANU, Canberra. Springer, Berlin

[MS10] Meinicke L, Solin K (2010) Refinement algebra for probabilistic programs. Form Asp Comput 22:3–31

[O'C95] O'Connor PDT (1995) Practical reliability engineering. 3rd edn. Wiley, Toronto

[Put05] Puterman M (2005) Markov decision processes. Discrete stochastic dynamic programming. Wiley, Toronto

[Rao95] Rao JR (1995) Extension of the UNITY methodology: compositionality, fairness and probability in parallelism. Springer, Berlin

[Rod] Rodin Platform. Integrated Development Environment for Event-B. Online at http://www.event-b.org/

[ROD04] RODIN: Rigorous Open Development Environment for Complex Systems. (2004) IST FP6 STREP project. Online at http://rodin.cs.ncl.ac.uk/

[SL95] Segala R, Lynch N (1995) Probabilistic simulations for probabilistic processes. Nord J Comput 2(2):250–273

[ST96] Sere K, Troubitsyna E (1996) Probabilities in action systems. In: Nordic Workshop on Programming Theory

[STW10] Schneider S, Treharne H, Wehrheim H (2010) A CSP approach to control in Event-B. In: IFM 2010. Springer, Berlin, pp 260–274

[SZ13] Stoddart B, Zeyda F (2013) A unification of probabilistic choice within a design-based model of reversible computation. Form Asp Comput 25:107–131

[TPT$^+$12] Tarasyuk A, Pereverzeva I, Troubitsyna E, Latvala T, Nummila L (2012) Formal development and assessment of a reconfigurable on-board satellite system. In: SAFECOMP 2012. Springer, Berlin, pp 210–222

[TPTL13] Tarasyuk A, Pereverzeva I, Troubitsyna E, Laibinis L (2013) Formal development and quantitative assessment of a resilient multi-robotic system. In: SERENE 2013. Springer, Berlin, pp 109–124

[TRF03] Trivedi K, Ramani S, Fricks R (2003) Recent advances in modeling response-time distributions in real-time systems. In: Proceedings of the IEEE 91(7):1023–1037

[Tro99] Troubitsyna E (1999) Reliability assessment through probabilistic refinement. Nord J Comput 6(3):320–342

[TTL10a] Tarasyuk A, Troubitsyna E, Laibinis L (2010) From formal specification in Event-B to probabilistic reliability assessment. In: DEPEND 2010. IEEE Computer Society, Los Alamitos, CA, pp 24–31

[TTL10b] Tarasyuk A, Troubitsyna E, Laibinis L (2010) Towards probabilistic modelling in Event-B. In: IFM 2010, Integrated Formal Methods. Springer, Berlin, pp 275–289

[TTL12] Tarasyuk A, Troubitsyna E, Laibinis L (2012) Formal modelling and verification of service-oriented systems in probabilistic Event-B. In: IFM 2012, Integrated Formal Methods. Springer, Berlin, pp 237–252
[Vil95] Villemeur A (1995) Reliability, availability, maintainability and safety assessment. Wiley, New York
[Whi93] White DJ (1993) Markov decision processes. Wiley, New York
[YH10] Yilmaz E, Hoang TS (2010) Development of Rabin's choice coordination algorithm in Event-B. ECEASST, 35

Received 24 April 2013
Revised 28 October 2013
Accepted 22 May 2014 by Michael J. Butler
Published online 11 July 2014