Reliability Assessment in Event-B Development

Anton Tarasyuk, Elena Troubitsyna, and Linas Laibinis

˚Abo Akademi University, Finland
{anton.tarasyuk, elena.troubitsyna, linas.laibinis}@abo.fi

Abstract. Formal methods are indispensable for ensuring dependability of complex software-intensive systems. In particular, the B Method and its recent extension Event B have been successfully used in the development of several complex safety-critical systems. However, they are currently not supporting quantitative assessment of dependability attributes that is often required for certifying safety-critical systems. In this paper we demonstrate by example how to integrate reliability assessment into Event B development. This work shows how to conduct probabilistic assessment of system reliability at the development stage rather than at the implementation level.

Keywords: Event-based modeling, reliability assessment, formal verification, Markov processes

Introduction

To demonstrate dependability of a system it is often required to prove statistically that the probability of a catastrophic failure is acceptably low. Usually such an assessment is done after the development is finished, i.e., at the implementation level. However, in some cases the obtain design does not achieve the targeted level of reliability. Obviously, the cost of redesign might be very high.

In this paper we address the problem of reliability assessment at the development stage. We show how to integrate probabilistic assessment of system reliability with its formal modeling and verification in Event B. We demonstrate by example that in some cases we can obtain an algebraic solution connecting the overall system reliability with reliabilities of its components. In the other cases, we can employ probabilistic model checking to obtain a numeric solution.

Our ideas are exemplified via a small case study modeling and assessment of reliability of a data transmission. We believe that integrating probabilistic reasoning into the formal modeling significantly enhances the benefits of the later one. Indeed in this case our formal model would give us not only logical but also statistical evidences of system dependability.

1 Modelling and Refinement in Event B


Event B uses the Abstract Machine Notation for constructing and verifying models. An abstract machine encapsulates a state (the variables) of the model and provides operations on its state. A simple abstract machine has the following general form:
The machine is uniquely identified by its name $AM$. The state variables of the machine, $v$, are declared in the VARIABLES clause and initialised in INIT as defined in the INITIALISATION clause. The variables are strongly typed by constraining predicates of the machine invariant $I$ given in the INVARIANT clause. The invariant is usually defined as a conjunction of the constraining predicates and the predicates defining the properties of the system that should be preserved during system execution.

The dynamic behaviour of the system is defined by the set of atomic events specified in the EVENTS clause. An event is defined as follows:

$$E = \text{WHEN } g \text{ THEN } S \text{ END}$$

where the guard $g$ is conjunction of predicates over the state variables $v$, and the action $S$ is an assignment to the state variables.

The occurrence of events represents the observable behaviour of the system. The guard defines the conditions under which the action can be executed, i.e., when the event is enabled. The action can be either a deterministic assignment to the state variables or a non-deterministic assignment from a given set or an assignment according to a given postcondition. These assignments are denoted as $:=$, $\in$ and $|$ correspondingly. If several events are enabled then any of them can be chosen for execution non-deterministically. If none of the events is enabled then the system deadlocks.

The Event B models are formally defined using the weakest precondition semantics [4]. The weakest precondition semantics provides us with a foundation for establishing correctness of specifications and verifying refinements between them. For instance, we verify correctness of a specification by proving that its initialization and all events establish the invariant.

The basic idea underlying formal stepwise development by refinement is to design the system implementation gradually, by a number of correctness preserving steps, called refinements. The refinement process starts from creating an abstract, albeit unimplementable, specification and finishes with generating executable code. The intermediate stages yield the specifications containing a mixture of abstract mathematical constructs and executable programming artifacts.

Assume that the refinement machine $AM'$ is a result of refinement of the abstract machine $AM$: 
The machine $AM'$ might contain new variables as well as replace the abstract data structures of $AM$ with the concrete ones. The invariant of $AM' - I'$ defines not only the invariant properties of the refined model, but also the connection between the state spaces of $AM$ and $AM'$. For a refinement step to be valid, every possible execution of the refined machine must correspond (via $I'$) to some execution of the abstract machine. To demonstrate this, we should prove that $INIT'$ is a valid refinement of $INIT$, each event of $AM'$ is a valid refinement of its counterpart in $AM$ and that the refined specification does not introduce additional deadlocks, i.e.,

\[
wp(INIT', \neg wp(INIT, \neg I')) = true,
I \land I' \land g_i' \Rightarrow g_i \land wp(S', \neg wp(S, \neg InvC)), \quad \text{and}
I \land I' \land g_i \Rightarrow \bigvee_{i=1}^{N} g_i'.
\]

2 Example of refinement in Event B

To illustrate modelling and refinement in Event B let us consider a small case study – modelling a simple data channel. The structure of the system is shown in Figure 1. The system comprises two nodes: a sender ($A$) and a receiver ($B$), and at each step $A$ can send a request to $B$. After sending a message $A$ stops sending and listens to the channel to get confirmation from receiver. The communication channel ($C$) can fail at each step, the failure is repairable and both failure and repair rates are assumed to be known from some reliability report. If a request (confirm) is sent while the channel remains unavailable, the message got lost and the system is shutting down. If the message is confirmed, $A$ can transmit the next request.

![Diagram](image.png)

**Fig. 1.** Case study: overall system structure
In our initial abstract specification we merely model the possibility of successful or failed transmission. The variable \( res \) is introduced to represent this. The variable is non-deterministically assigned the values \( TRUE \) or \( FALSE \). When \( res \) obtains the value \( FALSE \) then the failure has occurred and the system deadlocks.

\[
\begin{align*}
\text{MACHINE} & \quad \text{Channel0} \\
\text{VARIABLES} & \quad \text{res} \\
\text{INVARIANTS} & \quad \text{inv1}: \text{res} \in BOOL \\
\text{EVENTS} & \\
\text{Initialisation} & \quad \text{begin} \\
& \quad \text{act1}: \text{res} := TRUE \\
& \quad \text{end} \\
\text{Event} & \quad \text{evt1} \triangleq \\
& \quad \text{when} \\
& \quad \text{grd1}: \text{res} = TRUE \\
& \quad \text{then} \\
& \quad \text{act1}: \text{res} \in BOOL \\
& \quad \text{end} \\
\text{END} & \\
\end{align*}
\]

Our initial specification is deliberately simple to facilitate probabilistic modelling that we will discuss in the next section. Next we will show how to refine this specification to capture the entire set of requirements given above.

We introduce the variable \( msg \) to model the status of message being transmitted, the variable \( ch \) to model the status of the transmitting channel and the variable \( flag \) to model the desired sequence of events. This is a rather simple refinement step. The specification of it is given below.
MACHINE Channel2
REFINES Channel1
SEES Context0
VARIABLES
res
msg
flag
ch

INVARINTS
inv1: flag ∈ 0..1
inv2: ch ∈ BOOL

EVENTS
Initialisation
extended
begin
act1: res := TRUE
act2: msg := A
act3: flag := 0
act4: ch := TRUE
end
Event evt1 ≡
refines evt1
when
grd1: res = TRUE
grd2: msg = ok
grd3: flag = 1
then
act1: msg := A
act2: flag := 0
end
Event evt2 ≡
refines evt2
when
grd1: res = TRUE
grd2: msg = lost
grd3: flag = 1
then
act1: res := FALSE
end
Event evt3_1 ≡
refines evt3
when
grd1: res = TRUE
grd2: msg = A
grd3: flag = 1
grd4: ch = TRUE
then
act1: msg ∈ {A, B}
act2: flag := 0
end
Event evt3_2 ≡
refines evt3
when
grd1: res = TRUE
grd2: msg = A
grd3: flag = 1
grd4: ch = FALSE
then
act1: msg ∈ {A, lost}
act2: flag := 0
end
Event evt4_1 ≡
refines evt4
when
grd1: res = TRUE
grd2: msg = B
grd3: flag = 1
grd4: ch = TRUE
then
act1: msg ∈ {B, ok}
act2: flag := 0
end
Event evt4_2 ≡
refines evt4
when
grd1: res = TRUE
grd2: msg = B
grd3: flag = 1
grd4: ch = FALSE
then
act1: msg ∈ {B, lost}
act2: flag := 0
end
Event evt5 ≡
when
grd1: flag = 0
then
act1: ch ∈ BOOL
end
END
Next we will show how to integrate probabilistic reasoning into Event B modelling.

3 Probabilistic modelling in Event B

Automatic tool support is essential for ensuring scalability of formal modelling. Hence availability of the tool support is essential when choosing the probabilistic basis underlying integration of probabilities into Event B. Maturity and good usability of probabilistic model checking using PRISM tool [5] encouraged us to choose discrete-time Markov chains (DTMCs) and Markov decision processes (MDPs) as the underlying semantics for our probabilistic enhancement of Event B modelling. In mathematics, a Markov chain is a stochastic process having the Markov property, it means that the description of the present state of the process fully captures all the information that could influence the future evolution of the process. Future states will be reached through a probabilistic process instead of a deterministic one, i.e. according to a certain probability distribution. Any Markov chain is completely defined by its matrix of transition probabilities and initial distribution. Markov decision processes are an extension of Markov chains, the difference is the addition of actions and rewards structures, in some cases MDP could be reduced to a Markov chain. More formal definitions of these of Markov processes can be found in many books on probability theory, e.g. [6–8].

Probabilistic model checking is an automatic formal verification technique for analysing quantitative properties of systems which exhibit stochastic behaviour. PRISM is a probabilistic model checker, a tool for formal modelling and analysis of systems which exhibit random or probabilistic behaviour. It supports three types of probabilistic models: discrete-time Markov chains, continuous-time Markov chains and Markov decision processes, plus extensions of these models with costs and rewards [5]. A system specification in PRISM is constructed as a parallel composition of modules, which can interact with each other. In general, a module in PRISM looks as follows:

\[
\text{module Module\_name} \\
\quad \text{var : Type} \text{ init . . . ;} \\
\quad [\text{grd1} \rightarrow p_1 : \text{action}_1 + \cdots + p_n : \text{action}_n; \text{grd2} \rightarrow q_1 : \text{action}'_1 + \cdots + q_m : \text{action}'_m; \cdots \\
\text{endmodule}
\]

As it is easy to see, Event B models can be easily augmented with the required probabilistic information and analysed using PRISM model checker. In the next section we will show how assess reliability of a system modeled in Event B using PRISM model checker as well as using mathematical theory of Markov processes.
4 Reliability Assessment in Event B Modelling

In engineering, reliability [9, 10] is generally measured by the probability that an entity $\mathcal{E}$ can perform a required function under given conditions for the time interval $[0, t]$: 

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

Let $T$ be the random variable measuring the uptime of the entity: 

$$R(t) = P[T > t] = 1 - P[T \leq t] = 1 - F(t).$$

There is a strong correlation between Event-B specifications and discrete time Markov processes. Let us consider some Event-B model where only local nondeterminism is possible, i.e. at every moment of time one and only one event might be enabled. In this case at every moment of time actions of the (single) enabled event define a number of possible future states of the system. As was mentioned before, only two types of actions are presented in Event-B: deterministic and non-deterministic assignments. Deterministic assignment defines the future system state unambiguously, in other words, it defines state transition with probability 1. Non-deterministic assignment defines a set of possible states, but the choice between them has daemonic nature. Replacing this non-deterministic choice with some probability distribution we obtain a set of possible probabilistic transitions similar to the ones in a discrete Markov chain. The replacement of such type is always valid because for any programs $\text{prog}$ and $\text{prog}'$: $\text{prog} \sqcap \text{prog}' \sqsubseteq \text{prog} \oplus \text{prog}'$, $\forall p \in [0, 1]$ [12]. If the global nondeterminism is also possible in an Event-B model then it can be represented by Markov decision process in a similar way.

Now let us return to our case study—modelling simple data channel. Let us assume that the transfer rate of $A$ and the service rate of $B$ are known.

The concrete specification consists of two groups of events, the first group models the channel’s availability and the second one models the message transferring process. These groups of events can be represented by two Markov chains, but while the transferring process depends on the channel’s availability, the corresponding Markov chain are not independent and reliability analysis of such system can be complex.

The probabilistic model checking PRISM can be used to assess reliability of our channel. The results of the analysis using PRISM are given below.

However, model checking does not give an analytical representation of reliability function. This might be disadvantageous, e.g., it does not give a guidance on how to choose components that allow to achieve the desired system reliability.

Let us consider another approach based on mathematical theory of Markov chains. One of the possible approaches to work with two or more non-independent Markov chains is to try to define the correlation between them. For instance, the concrete specification of our case study can be represented by Markov chain (see Figure 2.) with transition probabilities depending on channel’s availability. The considered example is very simple and this is not a problem to derive an
analytical view for availability function \( \alpha \), namely
\[
\alpha(k) = \frac{P_R + P_F (1 - P_R - P_F)^k}{P_R + P_F}.
\]
But still, as can be seen, \( \alpha \) depends on time \( k \) and obtained Markov chain is time non-homogeneous, and analysis of such type of chains is not a good idea.

Another way to work with a number of non-independent Markov chains is to try to build their superposition [6]. This is a relatively easy task when all chains are regular, but in our case study the chain describing the message transferring process is absorbing. It is clear, that to describe two processes as a superposition we need to define a state space of an output process and find its transition matrix \( P \). For our example the superposition can be build by decomposing of two states \( A \) and \( B \) into four states \( A_1, A_2, B_1 \) and \( B_2 \), where \( A_1 \) represents a system state when the message is in \( A \) and the channel is available, \( A_2 \) – the message is in \( A \) and the channel is unavailable and so on. The corresponding state transition diagram is shown in Figure 3.

Now, when we know the transition matrix \( P \), we can calculate the fundamental matrix \( N = (I - Q)^{-1} \), where \( I \) is an identity matrix and \( Q \) represents the transitions between transient states. We can calculate also the matrix
\[
B = \{b_{ij}\} = N \cdot R,
\]
where \( R \) represents the transitions from transient to ergodic states and \( b_{ij} \) is the probability that the process starting in transient state \( s_i \) ends up in absorbing state \( s_j \) [7]. Let \( \beta \) be the probability of absorption in “Ok” state.
and from matrix $B$ we can find that

$$
\beta = \frac{(P_A + P_R - P_A \cdot P_F - P_A \cdot P_R) \cdot (P_B + P_R - P_B \cdot P_F - P_B \cdot P_R)}{(P_A + P_F + P_B - P_A \cdot P_F - P_A \cdot P_R) \cdot (P_B + P_F + P_B - P_B \cdot P_F - P_B \cdot P_R)}
$$

In our example the reliability function can be represented in terms of the total uptime of the system or in terms of a number of successful request-confirm transfers within given time interval. The last one is better in respect to reliability engineering since it gives more concrete information about the system functioning and omits the standby periods when failures are impossible. On the other hand, both representations are connected in a trivial way: as long as $\{A_1, A_2\}$ and $\{B_1, B_2\}$ comprise two non-communicable classes of transient states (process cannot goes back from $B$ to $A$), the mean number of times the process remains in non-absorbing states equals to the sum of the time periods the process spends in these two classes, namely $\frac{1}{P_A + P_B}$. Let $X$ be the random variable measuring the number of successful transfers, obviously it has a geometric distribution, and the mean of $X$ and the reliability function of the system are $E(X) = \frac{\beta}{1 - \beta}$ and $R(t) = \beta^t + 1$ respectively.

It is interesting to compare results obtained with probabilistic model checking to those obtained analytically. Let us consider the input vector of probabilistic characteristics $(P_A, P_B, P_F, P_R) = (0.5, 0.8, 0.01, 0.5)$, then $\beta = 0.9758$ and $E(X) = 40.3485$. The results of the analysis using PRISM are illustrated in Figure 4 and they are closely related to the analytical ones: the graph shows system unreliability in terms of the total uptime and the "Property details" window demonstrates the total expected number of sent requests, i.e. $E(X) + 1$.

Fig. 4. Formal verification with PRISM
Conclusion

In this paper we have shown an example of integrating formal development by refinement with probabilistic assessment of system reliability. We demonstrated that for rather small specifications we can obtain an algebraic solution expressing overall system reliability as a function of reliabilities of its components. However, we have also demonstrated that for complex systems we can obtain a numerical estimate of reliability using PRISM model checker. Our approach supports reliability assessment already at the development phase and can give guidance on optimizing the design from dependability point of view. Moreover, it can help us to early diagnose the problems of the chosen design, so that the desired level of dependability would be nevertheless achieved. The similar topic in the context of refinement calculus has been explored previously [11, 12]. However, we see a great benefit in integrating probabilistic reasoning into the framework that has a mature tool support [3]. As our future work it would be interesting to further explore the connection between Event B modeling and dependability assessment. In particular the topic of probabilistic data refinement seems to be promising.

References

5. PRISM probabilistic model checker http://www.prismmodelchecker.org/