ISSN 2278-3091
Volume 8, No.1.4, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse7881.42019.pdf
https://doi.org/10.30534/ijatcse/2019/7881.42019

The Measurement of Safety Criteria in Safety Critical Systems

Muhammed Basheer Jasser[1], Jamilah Din[2] , Rodziah Atan, Yusmadi Yah Jusoh
Faculty of Computer Science and Information Technology/ University Putra Malaysia
43400 UPM Serdang, Selangor, Malaysia
[1]mbjasser@m.ieice.org; [1]mbjasser@gmail.com, [2]jamilahd@upm.edu.my

ABSTRACT

Safety critical systems use software to meet their functionalities. Failures in these software lead to a very high impact on the environment in which the safety critical systems are used. Several criteria are considered in safety critical systems, which are: reliability, availability, maintainability and safety. In this paper, we focus on measuring the safety criterion in safety critical systems. Safety has some factors that are: correctness, security, responsiveness and testability. Correctness is one of the most important factors of safety. Correctness also has sub-factors such as: completeness and consistency. In this paper, we investigate the safety criteria and safety measurement via two specific domains: transportation systems and business intelligent systems. Also, these criteria are employed in our developed prototype for verifying the safety criteria of safety critical systems and supporting the management of safety critical systems.

Key words : Safety critical systems, Safety, Correctness, Formal Methods, Formal Verification

1. INTRODUCTION

Safety critical systems [1], [2], [3] use specific software, appropriate processes, standards, specification languages and verification techniques to ensure a certain level of performance and assurance. This is because safety critical systems function in environments in which failures may lead to a loss of human lives and high cost assets.

Several criteria are considered in safety critical systems, which are: reliability, availability, maintainability and safety (RAMS). We concentrate in this paper on safety criterion. Safety has some factors [4] that are: correctness, security, responsiveness and testability. Correctness is one of the most important factors of safety. Correctness also has sub-factors such as: completeness and consistency.

There are several existing methods and techniques for measuring, specifying and verifying safety criterion and its factors. In [5], we studied quality criteria, attributes and metrics in the context of safety critical systems. In this paper, we put more emphasis on safety criteria and we investigate the safety measurement in two specific domains: transportation systems and business intelligent systems. We explore several case studies in these domains covering: the used specification language, the safety criterion to be measured, the verification technique used to verify the safety, the development level at which the system is verified, the automation level of verification and the tool used for the verification. These studied criteria are going to be employed in our developed prototype for verifying the safety criteria of safety critical systems and supporting the management of safety critical systems.

This paper is structured as follows. Section 2 presents a background and literature review on safety critical systems, safety criterion in general and formal methods. Section 3 presents some existing safety critical systems in specific domains and some details on measuring the safety of the systems in these domains. Section 4 resents an overview on how we integrate our developed prototype, which is for the management of safety critical systems, with a modelling platform for safety critical system. Section 5 presents the process model by which a user can specify safety critical systems and verify their safety. Section 6 concludes the work.

2. BACKGROUND AND LITERATURE REVIEW

In this section, we present a background and literature review on safety critical systems, safety criterion and formal methods. Safety critical system and safety criteria are discussed because they are the main topic of this article. Formal methods are discussed because they are the main and most important way to specify and verify safety critical systems.

2.1. Safety Critical Systems and Safety Criteria

Safety critical systems [1], [2] are those at which failures may lead to loss in human lives or high-cost assets. For that, specific standards and procedures should be followed when designing and developing safety critical systems. Also, specific criteria should be considered in safety critical systems, such as RAMS (Reliability, Availability, Maintainability, Safety).

In [6], Reliability is defined as "the probability of a component, or system, functioning correctly over a given period of time under a given set of operating conditions", Availability is defined as "the probability that the system will be functioning correctly at any given time", and Maintainability is defined as "the ability of a system to be maintained' and 'Maintenance is the action taken to retain a system in, or return a system to, its designed operating condition"

In [4], Safety is defined by some factors correctness, security, responsiveness and testability. Correctness is one of the most important factors of safety. Correctness also has sub-factors such as: completeness and consistency. Metrics are also defined for these factors. The consistency factor, for example, is defined as: the metric of the number of contradicting functions, interfaces, inputs, outputs, and data.

2.2. Formal Methods

Formal methods [7], [8] in the software engineering discipline allow the design, modelling, verification, and maintenance of hardware and software systems. Formal methods introduce preciseness, remove ambiguity in specifications, and support the verification of requirements and design properties. The specifications in formal methods could be viewed as mathematical models, which represent the intended behavior of the systems and they are used to model safety critical systems such as: railway control systems, nuclear power plant control systems, aircraft control systems, intelligent transport systems, and medical systems. There exist different kinds of formal specifications and each has its own advantages and limitations. Some formal specifications are considered at the system modeling like (B-Method [7], Event-B [8], Z-Method [9] and VDM [10], while another type is viewed as part of the system implementation level, in other words, the formal specification is added as supportive statements to the source code like Larch [11] and JML [12].

3. THE MEASUREMENT OF SAFETY CRITICAL SYSTEMS IN SPECIFIC DOMAINS

In this section, we provide an overview of some existing safety critical systems in some specific domain which are: transportation domain and business intelligent systems.

Also, we present details on measuring safety criteria in safety critical systems in these two specific domains. In addition, we explore several case studies in these domains covering: the safety criterion to measure, the verification technique used to verify the criterion, the development level at which the system is verified, the automation level of verification and the tool used for the verification.

3.1. Transportation Domain (COPPILOT: Platform Screen Door Controller)

The COPPILOT system studied in [13] is used in French railways in Paris, as in Figure 1. This is to prevent customers from entering or falling on train tracks. This project is to control the opening and closing of platform doors in relation to the train arrivals and departure and their doors opening and closing. The orders of opening and closing the platform doors should be verified to be safe and secure as this is related to the lives of passengers.

Figure 1:. COPPILOT: Platform Screen Door Controller

In this project, the safety is verified in terms of correctness. The functional constraints and safety properties are verified so that there is no possibility to establish forbidden connections between train and platform or between train and tracks. The formal method used for the specification and verification is the B-Method [7]. The verification technique employed for that is theorem proving in which the verification has been automated partially. This formal specification and verification has been performed using the tool Atelier-B at the modelling/ design and analysis development phase. Table1 summarises these details.

Table 1:. COPPILOT: Platform Screen Door Controller- Details

Language	B-Method
Criteria	Safety in terms of correctness: Verify in the overall system (PSD + controller) that functional constraints and safety properties are verified (no possibility to establish forbidden connections between train and platform or between train and tracks).
Verification Technique	Theorem Proving
Development Level	Modelling/Design and Analysis
Automation of Verification	Partial Automation as interactive proving exist for complex proofs that are not possible to be verified by theorem provers
Tool	Atelier-B

3.2. Business Intelligence Domain

An example of this domain is the electronic voting systems. Several electronic voting systems exist for facilitating the voting process that is considered a mission-critical task. Preserving the correctness and solidness of such systems should be tackled with utmost care. Some of the main entities of voting systems are the voters, the authorities that are responsible of registering the eligible voters and the tallying/counting authorities of the votes. Some of the main operations of the voting systems are: The registration of the eligible voters and candidates prior to the voting process, the authorization of the eligible voters and providing them with the right to vote based on their credentials, the voting operation by which the voters choose their preferred candidate, and finally the counting/tallying of the votes in a systematic manner.

An actual case study is the formal specification and verification of an electronic voting system used in USA named as ES&S [14]. This study is to model the safety, in terms of security, of the electronic voting system and reason about it. The ASTRAL formal language [15] is used to specify the voting processes of the machines of the voting system and their security properties. The safety criteria are recognized in this study in terms of correctness. In that regards, it should be verified that the system is secure under malicious attacks that try to change the voting process results. Theorem proving via the PVS interactive theorem prover is employed perform the proof obligations. The proof is partially automated because some complex proofs cannot be verified automatically by the prover. The specification and verification has been performed at the modeling/design&analysis development phase. Table 2 summarizes these details.

Table 2:. *ES&S. Electronic Voting System- Details*

Language	ASTRAL
Criteria	Safety in terms of security: Verify that the system is secure under malicious attacks
Verification Technique	Theorem Proving
Development Level	Modeling/Design and Analysis
Automation of Verification	Partial Automation as interactive proving exist for complex proofs that are not possible to be verified by theorem provers
Tool	PVS Interactive theorem prover

4. AUGMENTING SAFETY MEASUREMENT TO SAFETY CRITICAL SYSTEM MANAGEMENT

In this section, we present the steps we follow to allow measuring safety criteria of safety critical systems in our developed prototype, which is for the management of safety critical systems. The following steps are to be performed for this aim, as shown in Figure 2:

- Decision about the formal method: In this step, we decide a formal method to be used to specify and verify the safety criteria.
- Design the formal specification editor: In this step, we design a user interface in our prototype tool that includes an editor for specifying safety critical systems using the selected formal language.
- Integration of the editor with an external formal verification platform: In this step, we integrate the designed editor in the second step with an external formal verifier that may have a model checker or a theorem prover.

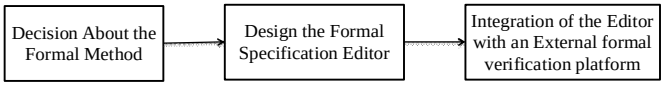

Figure 2: Steps for augmenting safety measurement to safety critical system management

In the next subsections, we explain every step individually in more details.

4.1. Decision about the Formal Method

A specific formal method is selected for the specification and verification of safety criteria of safety critical systems that is Event-B. Event-B [8] is a formal method that is based on action systems [16]. The mathematical notation used in Event-B is based on the set-theory [17]. Event-B differentiates the static and dynamic system parts. An Event-B context contains the types, axioms and constants, while an Event-B machine represents the changes of the state variables via events. Machines contain variables, events and invariants. Variables *v* define the machine state, constrained by the invariants *I(v)*. The events change the state of the machine. Proof obligations are to verify certain properties of a machine.

4.2. Design the Formal Specification Editor

We design a user interface in our prototype tool that includes an editor for specifying safety critical systems using the selected formal language Event-B. The Event-B specification is to be entered in the prototype in the editor for specifying the safety criteria.

4.3. Integration of the Editor with an External formal verification platform

The editor is integrated with the platform Rodin [18], which is a platform for the formal specification and verification of Event-B models. This integration is to: validate the entered specification in terms of its conformance to the Event-B formal language and to verify its safety (in terms of correctness) against the specified properties defined in the Event-B invariants.

5. THE PROCESS MODEL TO SPECIFY AND VERIFY SPECIFICATIONS OF SAFETY CRITICAL SYSTEMS

In this section, we explain the processes/steps followed to specify and verify a formal specification of a safety critical system. This is based on the steps that we have achieved in Section 4 for augmenting safety measurement into safety critical system management. The processes to be followed are, as in Figure 3:

- Specify the safety critical system in the editor that is for the formal specification.
- Validate the specification in terms of its conformance to the formal specification language.
- Verify the correctness of the specification against the correctness properties defined in the invariants.
- Show the output of the correctness verification.

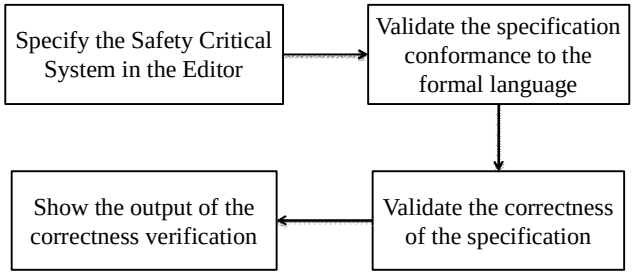

Figure 3:. The process model to specify and verify safety critical systems

In the next subsections, we explain every step of the process model individually in more details.

5.1. Specify the Safety Critical System in the Editor

The specification of a safety critical system may be entered in the editor designed in the tool. An E-voting system may be specified via Event-B. An Event-B machine may be specified by defining some set variables and events for the E-voting system. The specifications of the Event-B Machine, named *EVotingMachine*, and context, named *EVotingContext*, for the E-voting system are shown in Figure 4 and Figure 5, respectively.

In *EVotingMachine*, the variables are: ***voters***: a set of eligible voters (*voters* ∈ ℙ(*VOTERS*)), ***candidates*** : a set of eligible candidates (*candidates* ∈ ℙ(*CANDIDATES*)), ***candidatevotes***: a total function (*candidatevotes* ∈ *candidates* → ℕ) that maps every candidate from *candidates* to a natural number ℕ that represents his own count of votes), ***VoterCandidate***: a partial function (*VoterCandidate* ∈ *voters* ⇸ *candidates*) that maps every voter with only one candidate at most). Indeed other variables may be defined as well depending on system requirements. *VOTERS*, *CANDIDATES* are defined types in the context *EVotingContext.*

```
MACHINE
  EVotingMachine
SEES
  EVotingContext
VARIABLES
  voters
  candidates
  VoterCandidate
  candidatevotes
INVARIANTS
  inv1   :   voters ∈ ℙ(VOTERS)
  inv2   :   candidates ∈ ℙ(CANDIDATES)
  inv3   :   VoterCandidate ∈ voters ⇸ candidates
  inv4   :   candidatevotes ∈ candidates → ℕ
EVENTS
  INITIALISATION   ≙
  STATUS
   ordinary
  BEGIN
   act1   :   voters ≔ ∅
   act2   :   candidates ≔ ∅
   act3   :   VoterCandidate ≔ ∅
   act4   :   candidatevotes ≔ ∅
  END
  registerVoter   ≙
  STATUS
   ordinary
  ANY
   v
  WHERE
   grd1   :   v ∈ VOTERS \ voters
  THEN
   act1   :   voters ≔ voters ∪ {v}
  END
  registerCandidate   ≙
  STATUS
   ordinary
  ANY
   c
  WHERE
   grd1   :   c ∈ CANDIDATES \ candidates
  THEN
   act1   :   candidates ≔ candidates ∪ {c}
   act2   :   candidatevotes≔candidatevotes ∪ {c↦0}
  END
  vote   ≙
  STATUS
   ordinary
  ANY
   v
   c
  WHERE
   grd1   :   v ∈ voters
   grd2   :   c ∈ candidates
   grd3   :   {v} ◁ VoterCandidate = ∅
  THEN
   act1   :   VoterCandidate ≔ VoterCandidate ∪ {v↦c}
   act2   :   candidatevotes(c)≔candidatevotes(c)+1
  END
END
```

Figure 4: Event-B machine specification for E-voting System

```
CONTEXT
  EVotingContext
SETS
  VOTERS
  CANDIDATES
END
```

Figure 5: Event-B context specification for E-voting System

In *EVotingMachine*, the events that may be specified are: ***registerVoter*** (that is to register a voter as an eligible voter for the election), ***registerCandidate*** (that is to register a candidate as an eligible candidate for the election), ***vote*** (to cast a voice for an eligible voter to an eligible candidate).

Some of the invariants that may be specified are that a voter should cast a vote for only one candidate. This is specified in the invariant *VoterCandidate* ∈ *voters* ⇸ *candidates* that means that a voter could only vote for one candidate at most.

Proof obligations are generated and proved in Rodin platform according to the Event-B specifications. Indeed, this is a formal specification of a very simple E-voting system and a more complex system may be specified based on more complex requirements.

5.2. Validate the Specification Conformance to the Formal Language

The formal specification, which has been entered in the editor, is then fed to the external platform Rodin to validate its conformance to the Event-B formal language. The validation output should be then returned to the editor to show it to the user.

If the specification conforms to the formal language of Event-B, then the editor should recognize the Event-B language clauses, otherwise it should show the syntax errors and their locations.

5.3. Verify the Correctness of the Specification

The verification of the specification correctness takes place when the specification conforms to the Event-B formal language. The specification is then fed to the platform Rodin to verify its correctness against the specified correctness properties defined in the Event-B invariants.

5.4. Show the output of the correctness verification

The verification result of the specification correctness should be returned to show if the specification is correct or not against the correctness criteria. Accordingly, safety critical system managers should use this result for taking decisions in managing safety critical systems.

6. CONCLUSION

Safety critical systems use specific software, appropriate processes, standards, specification languages and verification techniques to ensure a certain level of performance and assurance. Reliability, availability, maintainability and safety (RAMS) are some of the most important criteria that should be considered in safety critical systems.

We concentrate in this paper on the safety criterion, which has some factors that are: correctness, security, responsiveness and testability. Several existing methods and techniques have been introduced for measuring, specifying and verifying the safety criterion of safety critical systems. Among those methods and techniques are the formal methods. These allow the design, modelling, verification, and maintenance of hardware and software systems. Formal methods introduce preciseness, remove ambiguity in specifications, and support the verification of requirements and design properties. For that, formal methods are used to model and verify safety critical systems such as: intelligent transport systems and medical systems.

In this paper, we investigate the safety measurement in two specific domains, which are: transportation systems and business intelligent systems. We explore several case studies in these domains covering: the used specification language, the safety criterion to be measured, the verification technique used to verify the safety, the development level at which the system is verified, the automation level of verification and the tool used for the verification. These studied criteria are employed in our developed prototype for verifying the safety criteria of safety critical systems and supporting in their management.

We have introduced a way to allow measuring safety criteria of safety critical systems in our developed prototype. We have introduced a general process model to specify and verify a safety critical system using our tool.

We believe this would help safety critical system managers, modellers and developers in: correctly and precisely specify safety critical systems, measuring the safety criteria of safety critical systems and taking decisions in managing safety critical systems based on measurements.

ACKNOWLEDGEMENTS

Thanks to the Ministry of Higher Education (MOHE) and Research Management Center, Universiti Putra Malaysia (UPM) for the financial supports through FRGS Vote No: 08-01-15-1726FR

REFERENCES

[1] Rausand, M., **Reliability of safety-critical systems: theory and applications**. John Wiley & Sons, 2014. https://doi.org/10.1002/9781118776353

[2] Martins, L.E.G. and Gorschek, T., **Requirements engineering for safety-critical systems: A systematic literature review**. *Information and software technology*, 75, pp.71-89. 2016. https://doi.org/10.1016/j.infsof.2016.04.002

[3] Grant, E.S., Jackson, V.K. and Clachar, S.A., **Towards a Formal Approach to Validating and Verifying Functional Design for Complex Safety Critical Systems. GSTF** *Journal on Computing* (JoC), 2(1), 2018

[4] Singh, R.. **A systematic approach to software safety**. In Proceedings Sixth Asia Pacific Software Engineering Conference (ASPEC'99)(Cat. No. PR00509) (pp. 420-423). IEEE, 1999.

[5] Din, J., Atan, R., Jusoh, Y.Y. and Jasser, M. B., **Towards Employing Metrics in Measuring the Quality of Software Safety Critical Systems and Managing their Development**. *International Journal of Engineering & Technology* (pp. 135-139), 2018.

[6] Storey, N.R., **Safety critical computer systems**. Addison-Wesley Longman Publishing Co., Inc., 1996

[7] Abrial, J.R., **The B-book: assigning programs to meanings**. Cambridge University Press, 2005

[8] Abrial, J.R., **Modeling in Event-B: system and software engineering**. Cambridge University Press., 2010 https://doi.org/10.1017/CBO9781139195881

[9] Bowen, J.P., ***Formal specification and documentation using Z: A case study approach*** (Vol. 66). London: International Thomson Computer Press, 1996

[10] Bjørner, D., **The vienna development method (VDM)**. In *Mathematical Studies of Information Processing* (pp. 326-359). Springer, Berlin, Heidelberg, 1979. https://doi.org/10.1007/3-540-09541-1_33

[11] Garland, S.J., Guttag, J.V. and Horning, J.J**., An overview of Larch**. In *Functional Programming, Concurrency, Simulation and Automated Reasoning* (pp. 329-348). Springer, Berlin, Heidelberg, 1993. https://doi.org/10.1007/3-540-56883-2_15

[12] Leavens, G.T., Poll, E., Clifton, C., Cheon, Y., Ruby, C., Cok, D., Müller, P., Kiniry, J., Chalin, P., Zimmerman, D.M. and Dietl, W., **JML reference manual**, 2008

[13] Lecomte, T., Servat, T. and Pouzancre, G., **Formal methods in safety-critical railway systems**. In 10th Brasilian symposium on formal methods (pp. 29-31), 2007.

[14] Weldemariam, K., Kemmerer, R.A. and Villafiorita, A**. Formal analysis of an electronic voting system: An experience report**. *Journal of Systems and Software*, 84(10), pp.1618-1637, 2011 https://doi.org/10.1016/j.jss.2011.03.032

[15] Kolano, P.Z., Dang, Z. and Kemmerer, R.A., **The design and analysis of real-time systems using the ASTRAL software development environment**. *Annals of Software Engineering*, 7(1-4), pp.177-210, 1999. https://doi.org/10.1023/A:1018934104631

[16] Back, R.J. and Kurki-Suonio, R., **Decentralization of process nets with centralized control**. Distributed Computing, *3*(2), pp.73-87, 1989. https://doi.org/10.1007/BF01558665

[17] Abrial, J.R., **From Z to B and then Event-B: assigning proofs to meaningful programs**. In *International Conference on Integrated Formal Methods* (pp. 1-15). Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-38613-8_1

[18] Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F. and Voisin, L., **Rodin: an open toolset for modelling and reasoning in Event-B**. *International journal on software tools for technology transfer*, 12(6), pp.447-466, 2010 https://doi.org/10.1007/s10009-010-0145-y