Analysis of Error Propagation in Safety Critical Software Systems: An Approach Based on UGF

R. Selvarani and R. Bharathi

Abstract Designing fault free software systems becomes an essential practice towards Safety Critical Software System (SCSS) manufacturing. The error free scenario of SCSS will support the systems perfect functioning. The proposed approach is based on universal generating function to compute the error inclusion in the output of the selected safety critical system. This paper presents an Error Propagation Metric called Safety Metric SM_{EP}, which can be characterized depending on the performance rate of the software module. Through this, the performance distribution of system modules and the system with respect to safety metric SM_{EP} is quantified.

Keywords Error • Safety metric • Performance • Modules • Subsystem • System

1 Introduction

SCSS may contain faults, although the system has been well tested, used and documented. If one part of a system fails, this can affect other parts and in worst-case results in partial or even total system failure. To avoid such incidents research on failure analysis is of high importance. A failure of a safety critical system can be defined as "the non performance or incapability of the system or a component of the system to perform or meet the expectation for a specific time under stated environmental conditions". Error propagation between software modules is a qualitative factor that reflects on the reliability of a safety critical software product.

R. Selvarani
Computer Science and Engineering, Alliance University, Bangalore, India
e-mail: selvarani.riic@gmail.com

R. Bharathi (✉)
Information Science and Engineering, PESIT-BSC, Bangalore, India
e-mail: rbharathi@pes.edu; sbharathi235@gmail.com

A.K. Sangaiah et al. (eds.), *Intelligent Decision Support Systems for Sustainable Computing*, Studies in Computational Intelligence 705,
DOI 10.1007/978-3-319-53153-3_13

This work focuses on modeling the failure analysis of SCSS using Universal Generating Function (UGF) [1], through the concept of Error Propagation (EP). The error propagation probability is a condition that once an error occurs in a system module, it might propagate to other modules and thereby cascades the error to the system output [2]. The error propagation analysis is a vital activity for the efficient and robust designing of safety critical software system. Generally, the functioning of SCSS is considered between two major states, perfect functioning and failure state. Here we are considering several intermittent states between the two major states for the failure analysis. Hence these systems can be termed as Multistate Systems (MS) in our research. The reliability of a MS can be defined as a measure of the capability of the system to execute required performance level [1].

The error in a software module might trigger an error across the other system modules that are interconnected [3]. Propagation analysis may be used to identify the critical modules in a system, and to explore how other modules are affected in the presence of errors. This concept will aid in system testing and debugging through generating required test cases, that will stimulate fault activation in the identified critical modules and facilitate error detection [4].

The kind of errors under consideration might be due to faulty design, which could result in errors and data errors due to wrong data, late data or early data. In this work, we assess the impact of EP across modules, by analyzing the error propagation process through probabilistic approach and arrive at a general expression to estimate the performance distribution of each module, subsystem and system. The reliability and performance of a multistate safety critical system can be computed by using Universal Generating Function (UGF) technique [5]. The UGF technique applied for analyzing the EP in safety critical systems in this paper is adapted by following the procedure demonstrated by Levitin et al. [5, 6].

The chapter is structured as follows: Related Terminologies are given in Sect. 2. Section 3 discusses on related works on error propagation. The influence of EP in software reliability prediction and universal generating function (UGF) are discussed in Sects. 4 and 5 respectively. The problem statement is explained in Sect. 6. Section 7 introduces the proposed approach through a framework. The error propagation and failure analysis of SCSS are discussed in Sect. 8. Section 9 gives the case study. Conclusion and future work are discussed in Sect. 10.

2 Terminologies

A *failure* is an event that occurs when the delivered service no longer complies with the expected service of the system. An *error* is an incorrect internal state that is liable to the occurrence of a failure or another error. However all errors may not reach the system's output to cause a failure. A *fault* is active when it results in an error otherwise it is said to be inactive [3]. Nevertheless, not all faults lead to an error, and not all errors lead to a failure. *Error propagation* (EP) can be defined as a condition where a failure of a component may potentially impact other system components or the environment through interactions, such as communication of bad data, no data, and early/late data [7].

3 Related Works

In the related area of our topic, there has been a wide discussion on software error propagation and reliability prediction by many authors. Jhumka et al. [8] derived a set metrics namely error transmission probability metric, error transparency metric and influence metric during the error propagation process. They identified vulnerable modules using error transmission probability and error transparency metrics. An analytical framework was developed to reduce the inter-modular Error Propagation in software.

In [9] the error propagation was analyzed and defined using probabilistic approach and derived two more concepts namely unconditional error propagation and cumulative error propagation among components. Finally they arrived with formulas to estimate the "error propagation probability", "unconditional and cumulative error propagation". The proposed formulas are validated through fault injection experiment.

The approach presented in [10], concludes that error propagation between the components have significant effect on the system reliability prediction. The propagation of errors between functions have been discussed in [11]. They have discussed on different approaches for error propagation assessment namely probabilistic approach, model based approach and formal approach. Further, they have defined a high level strategy on preventing error propagation and established a hypothesis.

A probabilistic analysis of error propagation due to either hardware or software faults in mechatronic systems are addressed in [12]. They have proposed an abstract mathematical framework called Error Propagation Model to analyze the error propagation in system level. Each system element is characterized by three parameters namely "Fault Activation", "Error Propagation" and "Error Detection" probabilities. They have used data flow graph to determine the error propagation between elements.

Morozov and Janschek [4, 13] have used probabilistic error propagation techniques for diagnosing the system. Henceforth it aids in tracing back the path of error propagation path to the error-origin. Moreover this diagnosis helps in error localization procedure, testing, and debugging. A bottom-up approach is considered to estimate the reliability for component-based system in [2]. Foremost, the reliability of system component was assessed. Based on the architectural information, the system reliability was estimated taking into the account of error propagation probability. The system analysis was carried out through the failure model by considering only data errors across components. Authors in [2] have concluded that error propagation is a significant characteristic of each system component and defined as the probability that a component propagates the erroneous inputs to the generated output. Their approach can be used in the early prediction of system reliability.

An error befalls when there is an activation of a fault [3]. An error occurs in a module when there is a fault in the module and henceforth it cannot directly cause an error in other modules. At many times, an error in a module can lead to its failure only within that module. The reason for the module error is either due to the activation of fault in the same module or due to deviated input service (failure) from other modules. A module failure is defined as the deviation of the performance from its accepted output behavior. If the failed module is the output interface module of the system then its failure is considered as a system failure [14]. System failures are defined based on its boundary. A system failure is said to occur when error propagates outside the system.

Filieri et al. [15] stated the all errors in the component are not liable to its failure. In turn, not all component failures necessarily lead to failure of the whole system. A component failure occurs only when an error propagates within the component up to its interface, and a system failure happens only when an error propagates across components up to the generated output. While during this propagation of error, there is a probability that an error can get masked, for example an erroneous value can be overwritten before being delivered to its interface. It is probable that an error can be transformed, from one type to another. For instance, a content failure received from one component could initiate additional computations, leading to timing failure [15]. In the propagation path each propagated error need not create the same kind of failures [16]. When a component fails, it is not that all failed components will propagate error and not all components will fail due to propagated error (error masking) [15]. Some systems are critical to certain category of failures and at the same time they might be less critical to other category of failures [17]. It is not necessary that the propagated error should create the similar kind of error in other components [18, 17].

To our knowledge, mostly researchers have discussed only on software error propagation. Embedded software is mostly used in safety critical systems to perform increasing number of safety critical functions. The fact is that these safety critical systems incorporate both software and hardware components with various mutual interactions. Hence, the occurrence of error in software due to failures caused by hardware components has to be accounted.

The error propagation analysis is an essential activity for the efficient and robust designing of safety critical software system.

In this paper, Safety Metric SM_{EP}, is considered, which has the following benefits [19]. It helps in assessing the probability of EP in the module level, subsystem and system level and to trace out the migration of error propagation from module level to subsystem level and to the system level.

4 Influence of EP in Software Reliability Prediction

Error Propagation has strong influence on system reliability. Many researchers [2, 10, 17, 18, 20, 21] have detailed the significance of EP inclusion in the architecture based reliability assessment. They have noted that many existing reliability models are based on black box approach. Moreover each researcher has different viewpoints on error propagation and characterized in several ways.

Identification of different error states and failure modes of each component or module is very essential. Each error type may have different propagation path. Probabilistic error propagation analysis is the most preferred method by many researchers. Fiondella and Gokhale [21] presented an approach to find the reliability of the software at architectural design stage through EP modeling. To achieve this each component is characterized with six parameters namely probability of correct output for a given correct input, probability of incorrect output for a given correct input, probability of no output for a given correct input, probability of correct output for a given incorrect input, probability of incorrect output for a given incorrect input, probability of no output for a given incorrect input. Through which the criticality of the components are identified such that it can be equipped with error recovery mechanisms to improve the system reliability.

Reliability at its extensive level is considered as performance measure and stated as the capability of an entity to perform a required function under specified condition for a specified period of time.

5 Universal Generating Function Technique for MS

The UGF technique is based on probability theory to assess and express models through polynomial functions. It is also called as u-function introduced by Ushakov [22] and Levitin [5] expanded and proved that UGF is an effective technique for assessing the performance of real world systems, in specific Multistate Systems. In general all traditional reliability models perceived a system as binary state systems, states being perfect functionality and complete failure. In reality, each system has different performance levels and various failure modes affecting the system performance [1]. Such systems are termed as Multistate-Systems (MS).

Let us assume an MS composed of *n* modules. In order to assess the reliability of a MS, it is necessary to analyze the characteristic of each module present in the system. A system module ‘*m*’ can have different performance rates and represented by a finite set $q_{m,}$ such that $q_m = \{q_{m1}, q_{m2}, \ldots\ldots q_{mi\ldots} q_{mk_m}\}$ [23], where q_{mi} is the performance rate of module m in the ith state and $q_i = \{1, 2, \ldots k_m\}$. The performance rate $Q_m(t)$ of module ‘m’, at time t $\geq$ 0 is a random variable that takes its value from q_m: $Q_m(t) \in q_m$.

Let the ordered set $p_m = \{p_{m1}, p_{m2}, \ldots p_{mi}, \ldots p_{mj_m}\}$ associate the probability of each state with performance rate of the system module *m*, where $p_{mi} = Pr\{Q_m = q_{mi}\}$. The mapping $q_{mi} \rightarrow p_{mi}$ is called the probability mass function (pmf) [24].

The random performance [25] of each module *m* is defined as polynomials can be termed as module's UGF denoted as "$u_m(z)$",

$$u_m(z) = \sum_{i=0}^{k} P_{mi} z^{q_{mi}}, m = 1, 2 \ldots n. \tag{1}$$

Similarly the performance rates of all '*m*' system modules have to be determined. At each instant $t \geq 0$, all the system modules have their performance rates corresponding to their states. The UGF for the MS denoted as "($U_S(z)$)" can be arrived, by the determining the modules interconnection through system architecture. The random performance of the system as a whole at an instant $t \geq 0$ is dependent on the performance state of its modules. The UGF technique specifies an algebraic procedure to calculate the performance distribution of the entire MS, denoted as $U_S(z)$, let $U_s(z) = f\{u_{m1}(z), u_{m2}(z), \ldots, u_{mn}(z)\}$,

$$U_s(z) = \nabla_{\emptyset}\{u_{m1}(z), u_{m2}(z), \ldots, u_{mn}(z)\} \tag{2}$$

where, ∇ is the composition operator and $\emptyset$ is the system structure function. In order to assess the performance distribution of the complete system with the arbitrary structure function $\emptyset$, a composition operator ∇ is used across individual u function of *m* system modules [24].

$U_S(z)$, is U-function representation of performance distribution of the whole MS software system. The composition operator ∇ determines the U function of the whole system by exercising numerical operations on the individual u functions of the system modules. The structure function $\emptyset(\cdot)$ in composition operator ∇ expresses the complete performance rate of the system consisting of different modules in terms of individual performance rates of modules. The structure function $\emptyset(\cdot)$ depends upon the system architecture and nature of interaction among system modules.

Reliability is nothing but continuity of expected service [3] and it is well known that, it can be quantitatively measured as failures over time. The UGF technique can be used for estimating of software reliability of the system as a whole consisting of *n* module. Each of the modules performs a sub function and the combined execution of all modules performs a major function [24]. An assumption while using the UGF technique is that the system modules are mutually independent of their performance.

In MS, the reliability (R_{MS}) at instant 't' can be defined as the probability that a system as whole can operate and perform the required service "S". Hence the performance rate of the MS at instant ‘t’ can be represented, as ‘B’ and that should be greater than or equal to “S”.

$$R_{MS}(t, S) = \Pr\{B \geq S\} \tag{3}$$

To assess the reliability of the MS system, we need to estimate the performance of the system as whole as shown in Eq. 2.

6 Problem Statement

Error Propagation (EP) is defined as the condition where there is a probability of an error (or failure) propagates across various modules or components in the SCSS [26]. Our approach focuses on quantifying the propagation of error between modules in safety critical software system using UGF, because it provides a practical adaptation concept to facilitate appropriate actions in complex and changing environments [1].

7 Proposed Approach

The analysis proposed in this research is explained through a framework as shown in Fig. 1. To begin, the performance distribution of system modules called PD_{MOD} is determined using u-function.

The probability of error propagation at module level ($PD_{MOD} + SM_{EP}$) is quantified in the second step. As third step, the performance distribution of subsystems is arrived through composition operator ∇ having a structure function $ø(\cdot)$. As the final step the failure prediction is achieved through recursive operations for quantifying the error propagation throughout the system. During software development, this framework would be helpful to demonstrate the probability of error propagation to identify the error prone areas. The estimated performance of the system helps to assess the reliability.

8 EP and Failure Analysis

The EP and failure analysis model is a conceptual framework for analyzing the occurrence of error propagation in SCSS [19]. The system considered is broken down into subsystem, and each subsystem in turn is subdivided into modules called elements. A module is an atomic structure, which performs definite function(s) of a complex system.

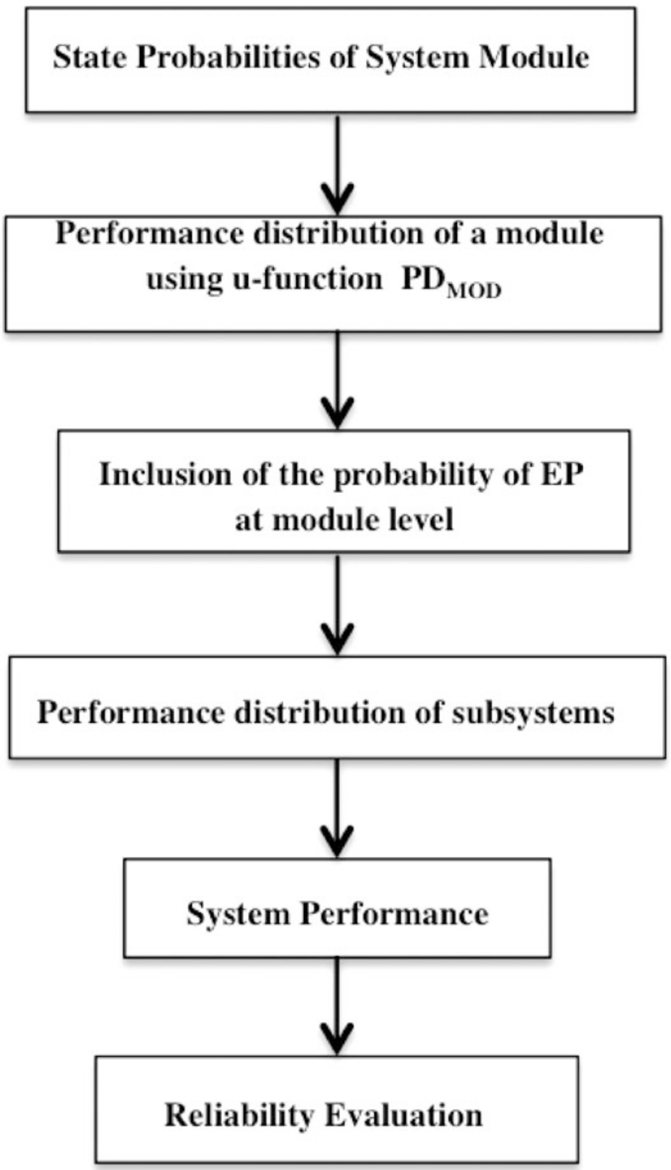
State Probabilities of System Module
Performance distribution of a module using u-function PD_{MOD}
Inclusion of the probability of EP at module level
Performance distribution of subsystems
System Performance
Reliability Evaluation

Fig. 1 Framework for EP and failure analysis

8.1 Performance Distribution of a Module

The performance rate of a module can be measured in terms of levels of failure [19].

Let us assume that the performance rate of a module *m* with 0% failure is q_{m1}, 10% failure is q_{m2}, 30% failure is q_{m3}, 50% failure is q_{m4} and 100% failure is q_{m5}.

The state of each module m can be represented by a discrete random variable Q_m that takes value form the set,

$$Q_m = \{q_{m1}, q_{m2}, q_{m3}, q_{m4}, q_{m5}\} \tag{4}$$

The random performance of a module varies from perfect functioning state to complete failure state.

The probabilities associated with different states (performance rates) of a module *m* at time t can be represented by the set,

$$P_m = \{p_{m1}, p_{m2}, p_{m3}, p_{m4}, p_{m5}\}, \text{where}, P_{mh} = \Pr\{Q_m = q_{mh}\}.$$

The module's states is the composition of the group of mutually exclusive events,

$$\sum_{h=1}^{5} P_{mh} = 1 \tag{5}$$

The performance distribution of a module *m* (pmf of discrete random variable G) can be defined as

$$u_m(z) = \sum_{h=1}^{5} P_{mh} z^{q_{mh}} \tag{6}$$

The performance distribution of any pair of system modules *l* and *m*, connected in series or parallel [25] can be determined by,

$$u_l(z) \nabla u_m(z) = \sum_{h=1}^{5} P_{lh} z^{q_{ih}} \nabla \sum_{h=1}^{5} P_{mh} z^{q_{mh}} \tag{7}$$

The composition operator ∇determines the u function for two modules based on whether they are connected in parallel or series using the structure function ø. The equation arrived in Eq. (7) quantifies the performance distribution of combination of modules. Levitin et al. in [6] have demonstrated the determination of performance distribution when the modules are connected in series or parallel.

8.2 Safety Metric SM_{EP}

The probability of occurrence of EP in a module can be defined by introducing a new state in that module [6]. Assuming that the state 0 of each module corresponds to the EP originated from this module [5]. The Eq. (6) can be rewritten as,

$$u_m(z)_{ep} = P_{m0} z^{q_{m0}} + \sum_{h=1}^{5} P_{mh} z^{q_{mh}} \tag{8}$$

$$u_m(z)_{ep} = P_{m0} z^{q_{m0}} + u_m(z) \tag{9}$$

where p_{m0} is the probability state for error propagation and q_{m0} is the performance of the module at state 0. $u_m(z)$ represents all states except the state of error propagation.

The performance distribution of a module *m* at state 0 is the state of error propagation is given by $P_{m0} z^{q_{m0}}$ is termed as Safety Metric SM_{EP}, used to measure the probability of error propagation. The Safety metric SM_{EP} [19] of each module will carry a weightage based on the probability of propagating error. If the module does not propagate any error, the corresponding state probability should be equated to zero [6].

$$p_{m0} = 0 \tag{10}$$

By substituting Eq. (10) in Eq. (8), the SM_{EP} is quantified as zero. Therefore Eq. (8) becomes,

$$u_m(z)_{ep} = \sum_{h=1}^{5} p_{mh} z^{q_{mh}} \tag{11}$$

If the module that does not have error propagation property or state then Eq. (11) will be reduced to Eq. (6).

If the module can cause error propagation, then the performance of the module in that state of error propagation is

$$q_{m0} = \alpha \tag{12}$$

The value of α in Eq. (12) can be any random performance q_{m1} or q_{m2} or q_{m3} or q_{m4} or q_{m5}. The conditional pmf of any operational module *m* that will not fail due to error propagation can be represented by u-function [6],

$$u_m(z)_{ep} = \sum_{h=1}^{5} \frac{p_{mh}}{1 - p_{m0}} z^{q_{mh}} \tag{13}$$

Because the module can be in any one of the five states as defined in Eq. (6). The Safety metric SM_{EP} depends on the performance of each module in the multistate system. As per [19], this safety metric helps to measure whether the module or the subsystem has the influence of EP or not. Hence the performance distribution of module, subsystem and system is dependent on this safety metric.

8.3 *Module Definition in Terms of PD_{MOD} and SM_{EP}*

The safety metric SM_{EP} depends upon the performance of each module. This is the second step of our EP and failure analysis framework as depicted in the Fig. 2. The safety metric SM_{EP} of each module will carry a weightage based on the level of interaction with other modules of the system and the impact of error propagation to the other modules and within itself.

SM_{EP} = Function (Performance of module w.r.t propagation of error)

In this aspect, each module called MOD, can be defined by the following tuple,

MOD = $<PD_{MOD}, SM_{EP}>$
PD_{MOD} = Performance distribution of module in terms of levels of failure
SM_{EP} = Module Safety Metric.

The estimation of safety metric SM_{EP} depends upon the probability of occurrence of EP among modules in the system as described in the Eqs. (9–13).

8.4 *Performance Distribution of Subsystem PD_{SS}*

The subsystem performance depends upon the performance of all modules present in the subsystem. Because there is probability that migration of error occurs from

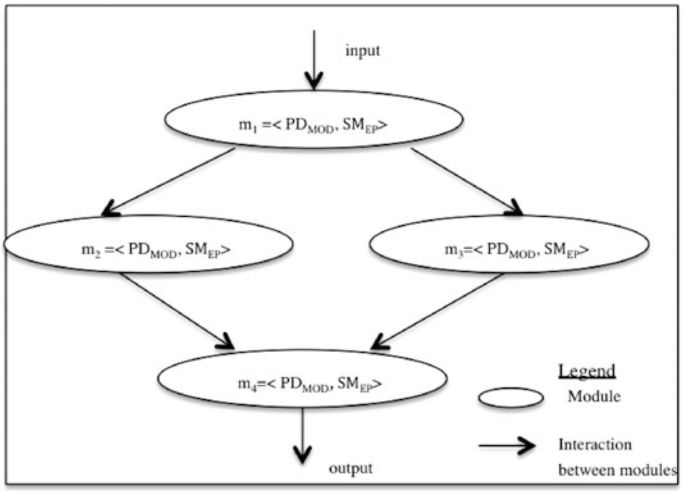
input
$m_1 =< PD_{MOD}, SM_{EP}>$
$m_2 =< PD_{MOD}, SM_{EP}>$
$m_3 =< PD_{MOD}, SM_{EP}>$
$m_4 =< PD_{MOD}, SM_{EP}>$
output
Legend
Module
Interaction between modules

Fig. 2 An example subsystem

modular level to subsystem level. Hence the performance of subsystem depends on the module performance PD_{MOD}, module safety metric SM_{EP} and subsystem structure function $\emptyset(\cdot)$. The function $\emptyset(\cdot)$ depends upon the nature of modules interaction in the subsystem. Dependency Graph (DG) is established method for conveying the architectural dependency between modules [27] and indicating the possible execution of modules. An example subsystem is depicted in the Fig. 2.

Each subsystem is defined by a quadruple $<N_m, \emptyset(\cdot), PD_{SS}, SM_{EP}>$, where,

N_m	Number of modules in the subsystem
$\emptyset(\cdot)$	Structure function determines the type of connection and nature of interaction among modules in the subsystem
PD_{SS}	Performance distribution of subsystem in terms of levels of failure
SM_{EP}	Safety Metric.

Depending upon the subsystem architecture, the u function of each subsystem can be quantified by applying the composition operator $\nabla_{\emptyset}$. In a subsystem, if a failed module is the output interface module, then its failure is considered as subsystem failure. Hence there will be a probability of error propagation, outside this subsystem. Then the recursive approach is used to obtain the entire u-function of safety critical software system.

9 Case Study

To illustrate our proposed approach on EP and failure analysis, we have taken Insulin Infusion Pump (IIP) as a case study. IIP is software intensive medical device for treating diabetic patients. The main function of this pump is to infuse calculated amount of insulin at correct time. Every year FDA [28] receives reports on adverse events with IIP including many injuries and deaths. The most frequently received problems are due to software defects, user interface issues and hardware problems [28]. The most hazardous situation in IIP are insulin overdose and insulin under dose which are insulin delivery errors, which occurs at the system output. In all modern pumps, software is responsible for insulin dosage control, mitigation of hazards through alerts, input interface and display. Hence, laborious hazard analysis and software development with safety requirements must be carried out and validated for safety usage. FDA [28] have listed out reasons for hazards, to name a few, random failures, systematic failures, failures caused by component, subsystem or system interactions, operating environment etc. The generic system architecture for insulin infusion pump model is shown in Fig. 3.

This model consists of pump controller, user interface, pump delivery mechanism, and drug reservoir and infusion set and each one is described with stated

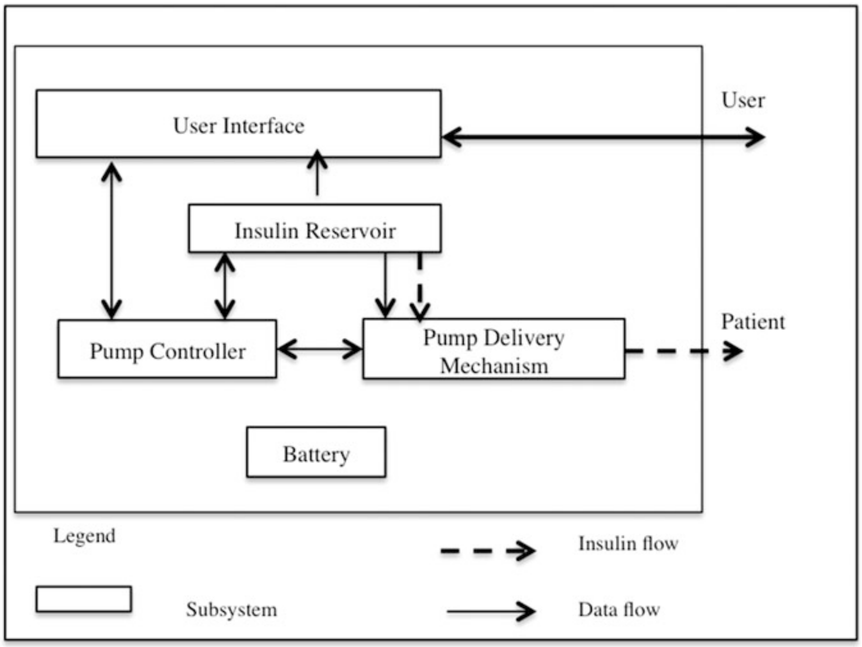

Fig. 3 Generic block diagram of insulin infusion pump

scope. The "Pump controller" is a subsystem responsible for "Computation" for the insulin infusion pump as a whole. The pump controller computes the dose of insulin for administrating the patient. Four types of insulin can be administered to a patient, namely basal insulin, temporary insulin, bolus and extended bolus insulin. In order to determine the IIP is adequately safe, a rigorous failure analysis has to be conducted. The performance distribution of the pump can be assessed using our proposed safety metric $SM_{EP.}$ Numerical calculations to assess the performance of the pump through our EP will be discussed in our subsequent work. An elaborate exercise has to be carried on all possible failure behavior of the pump, to locate the occurrence of error and subsequently the error propagation.

10 Conclusion and Future Work

The EP and Failure Analysis framework helps to analyze the failure of multistate safety critical software. The proposed metric SM_{EP} has the application in finding the failure probability of each module, the migration of error propagation from modular level to subsystem and then to system level. Subsequently it helps in the process of identifying the most critical module across the safety critical software system and the impact of error propagation in the performance of SCSS. Hence this method provides the base for the reliability evaluation, since the occurrence of error propagation across the modules has a significant effect on the system behavior during critical states.

References

1. G. Levitin, A universal generating function in the analysis of multi-state systems, in *Handbook of Performability Engineering* (Springer, London, 2008), pp. 447–464
2. V. Cortellessa, V. Grassi, A modeling approach to analyze the impact of error propagation on reliability of component-based systems, in *International Symposium on Component-Based Software Engineering* (Springer, Berlin, 2007)
3. A. Avizienis et al., Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
4. A. Morozov, K. Janschek, Probabilistic error propagation model for mechatronic systems. Mechatronics **24**(8), 1189–1202 (2014)
5. G. Levitin, Block diagram method for analyzing multi-state systems with uncovered failures. Reliab. Eng. Syst. Saf. **92**(6), 727–734 (2007)
6. G. Levitin, L. Xing, Reliability and performance of multi-state systems with propagated failures having selective effect. Reliab. Eng. Syst. Saf. **95**(6), 655–661 (2010)
7. P.H. Feiler, J.B. Goodenough, A. Gurfinkel, C.B. Weinstock, L. Wrage, Reliability validation and improvement framework. Carnegie-Mellon University, Pittsburgh, PA, Software Engineering Institute, 2012
8. A. Jhumka, M. Hiller, N. Suri, Assessing inter-modular error propagation in distributed software, in *Proceedings. 20th IEEE Symposium on Reliable Distributed Systems, 2001* (IEEE, 2001)
9. D.E. Nassar, W.A. Rabie, M. Shereshevsky, N. Gradetsky, H.H. Ammar, S. Bogazzi, A. Mili, Estimating error propagation probabilities in software architectures 1 (2004)
10. P. Popic et al., Error propagation in the reliability analysis of component based systems, in *16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)* (IEEE, 2005)
11. R. Fredriksen, R. Winther, Challenges related to error propagation in software systems, in *Proceedings of Risk, Reliability and Societal Safety (ESREL),* 2007, pp. 83–90
12. A. Morozov, K. Janschek, Case study results for probabilistic error propagation analysis of a mechatronic system. TagungsbandFachtagungMechatronik 229–234 (2013)
13. A. Morozov, K. Janschek, Dual graph error propagation model for mechatronic system analysis. IFAC Proc. **44**(1), 9893–9898 (2011)
14. A. Mohamed, M. Zulkernine, Failure type-aware reliability assessment with component failure dependency, in *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI)* (IEEE, 2010)
15. A. Filieri et al., Reliability analysis of component-based systems with multiple failure modes, in *Component-Based Software Engineering* (Springer, Berlin, 2010), pp. 1–20
16. N.A.M. Alzahrani, D.C. Petriu, Modeling component erroneous behavior and error propagation for dependability analysis, in *SDL 2013: Model-Driven Dependability Engineering* (Springer, Berlin, 2013), pp. 124–143
17. A. Mohamed, M. Zulkernine, Architectural design decisions for achieving reliable software systems, in *Architecting Critical Systems* (Springer, Berlin, 2010), pp. 19–32
18. A. Mohamed, M. Zulkernine, On failure propagation in component-based software systems, in *The Eighth International Conference on Quality Software, 2008. QSIC'08* (IEEE, 2008)

19. R. Selvarani, R. Bharathi, A novel safety metric SM_{EP} for performance distribution analysis in software system. in *Strategic Engineering for Cloud Computing and Big Data Analytics*, Springer, 2017 (in press).
20. T.-T. Pham, X. Défago, Q.-T. Huynh, Reliability prediction for component-based software systems: dealing with concurrent and propagating errors. Sci. Comput. Program. **97**, 426–457 (2015)
21. L. Fiondella, S.S. Gokhale, Architecture-based software reliability with error propagation and recovery, in *2013 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)* (IEEE, 2013)
22. I.A. Ushakov, A universal generating function. Soviet J. Comput. Syst. Sci. **24**(5), 118–129 (1986)
23. G. Levitin, A. Lisnianski, Multi-state system reliability: assessment, optimization and applications (2003)
24. G. Levitin, *The Universal Generating Function in Reliability Analysis and Optimization* (Springer, London, 2005)
25. G. Levitin, T. Zhang, M. Xie, State probability of a series-parallel repairable system with two-types of failure states. Int. J. Syst. Sci. **37**(14), 1011–1020 (2006)
26. S. Sarshar, Analysis of error propagation between software processes. Nucl. Power-Syst. Simul. Oper. 69 (2011)
27. S. Yacoub, B. Cukic, H.H. Ammar. A scenario-based reliability analysis approach for component-based software. IEEE Trans. Reliab. **53**(4), 465–480 (2004)
28. http://www.fda.gov/downloads/Training/CourseMaterialsforEducators/NationalMedicalDeviceCurriculum/UCM404248.pdf