ISSN:2249-5789

A Scrutiny of Automated Healthcare System with SFT

Jigna B. Prajapati[1], Dr.Nilesh.K.Modi[2], Ravi S.Patel[3]

[1,3]Asst.prof, Acharya Motibhai Patel Institute of Computer Studies, Ganpat University, Kherva.

[2] Prof. & Head, S.V.Institute of Computer Studies, Kadi, Gujarat, India

jigna.prajapati@ganpatuniversity.ac.in

Abstract

In today's techno savvy world, automated system is very important and contemporary issue. Automated systems are widely used at industries, appliance, automobile, undersea, space and healthcare over the past decade. The accuracy of Robots makes any system more acceptable. Here we use the Robots which assist us to manage patient's heath. We always expect the system must work under any situation. The development of Robotic software is a complex and error prone process. Most complex systems contain software, and systems failures activated by software faults can provide lessons for software development practices and software quality assurance. They must be identified and removed as early as possible. The interrelationship between software faults and failures is quite intricate and obtaining a meaningful characterization. Towards this characterization, we have investigated and classified failures observed in Robotic system. In this paper, we describe the process used in our study for tracking faults. We present the different types of faults, their impact and fault classification. The concern thing is Faults classification and proposed way to manage them. Then we propose the fault tolerance techniques as single to covenant with different faults.

Keywords

Automated Healthcare System, Software Faults, Faults classification, SFT (Software Faults tolerance)

Introduction

As technically we can say robot as "A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks"[15].

There are various types of Robots invented from last few decades. Different Robots as Mobile Robots, Industrial Robots, service Robots, Regional perspectives Robots, Military robots, General-purpose autonomous robots, Healthcare Robots, Educational Robots, Toy Robots, Washing Robots, Cleaning Robots, Gardening Robots, laundry Robots[16].

Here we would like to focus on Care robot, which is medical robot that gives care to the patients such as assisting them in doing a certain tasks and one example is helping them take their medications as to give insulin on time with proper dosage. This may be used as a substitute for taking care of the elderly to make their lives easier and more convenient.

From the user point of view, user always expects an error free soft ware in any situation. There are many factors affecting to the success of software. We start our analysis with faults occurred at particular Automated Healthcare System system. How well we test, debug and verify modularize though design errors still plague our software. The term error often is used in addition to the terms fault and failure. Often, errors are defined to be the result of faults, leading to failures [2]. We substitute the term fault for the common uses of the term error. Generally, references to the term "error" in the literature can be fitted to the context of this paper by substituting the term "fault."

Fault can be of different types as system boundaries based, creation based, time based,

action based, location based, user based, operator based and many more. Further faults can be Permanent, Temporary, physically, Intentional, Accidental, and Interaction [1]. When such faults arise within our system, that should be avoided, prevented or removed which can performed with the use of fault tolerance.

Fault-tolerance enables a system to prolong working properly in the event of the failure of some of its components. If any faults is compound or dependable to recourses and it may not able to produce result as required because of the previous faults which may cause total breakdown [3]. Fault-tolerance is particularly sought-after in high-availability or life-critical systems [3, 4]. In any applications, operational reliability is of paramount importance. Therefore, to achieve ultra-reliability in Automated Healthcare System system, it is necessary to adopt the strategy of defensive programming based on redundancy. This is referred to as fault-tolerant software.

Faults classifications

Faults may be classified based on Locality, cause, duration, phase, system state etc. Locality wise faults as atomic component, composite component, system, operator, and environment where faults reside in some specific location, the combination of more than one component, faults arise from any environmental causes or any user-operators [2] Cause wise faults as design, damage where problems arise by problematic designing of system, application or software. Physical faults can be as permanent, internal, physical faults. This class concerns those faults that have their origin within hardware components and are continuously active Temporary, internal, physical faults (intermittent faults) [1, 11].Effect wise faults as on System State crash, amnesia, partial amnesia. Boundaries wise faults internal, external where functionalities provided up to minimum and maximum, domain hardware or software, phenomenological cause, intent, and persistence. Duration wise faults can be as transient, persistent where faults occurred either temporary or permanently [1, 10].

The impact of any faults will take the system in non working state. It may cause if occurred fault within system is not breaking down the working state but it may lead another fault. Automated Healthcare System where patient is treated by Robotic action, one has to keep key eye on the action taken by the system. If any fault occurred and it will not managed at the proper time it cud be life critical matter. In such system where fault cannot be tolerant in any situating, the software fault tolerance will help a lot.

Scrutiny of Automated Healthcare System

Automated Healthcare System involves many routines like, serving patient, give medicine to patient at proper time, medicine dosage, insulin dosage, to check blood pressure, to check sugar level and taking care of the patient. When designing such system the designer must consider details regarding requirements, and the environment in which the healthcare system would be performed. Suppose automated healthcare System allowed 2ml but it spoiled by anyone else as below.

(1) The designer of the System did not allow for appropriate drug dosage setting. This could be: A specification fault if the XYZ department did not look forward to that more than 4ml would required need to use the boiler, or
(2) A design fault if the specification called for it being able to keep 4ml.
(3) An implementation fault if we didn't correctly follow the design.
(4) The boiler user ignored a “drug dosage Limit" sign. This would be a user fault.
(5) XYZ department posted an erroneous “drug dosage Limit" sign. This would be an operator fault.

(6) The people preparing the documentation for the Automated Healthcare System mistakenly indicated that the Automated Healthcare System would support 6ml, when in fact it was only designed to support 2 ml. The XYZ department erected a 2 ml "drug dosage Limit" sign. This would be a documentation fault, followed by an operator fault.

(7) By any natural effect, if system would damage or crashed that would be environmental faults.

As same, consider the same Automated Healthcare System with an improper blood pressure. There is no failure involved if the Automated Healthcare System continues to carry the blood pressure requested of it in spite of this fault. It may be the result of normal wear and tear. However, a thorough analyzing of the Automated Healthcare System might discover that the blood pressure in the system a faulty strut, from the point of view of the Automated Healthcare System analyzer, the strut would have failed. This component failure is an internal fault.

Scenarios like this can be generated ad infinitum. Note that a fault does not lead to a failure unless the result is observable by the user, and leads to the Automated Healthcare System becoming unable to deliver its specified service. This means that one person's fault is another person's failure. For instance, in example above, from the point of view of the department the erroneous documentation was a fault that led to an operator failure. From the point of view of the Automated Healthcare System the erroneous documentation was a documentation fault that led to an operator fault which led to a Automated Healthcare System system failure. Consider a computer system running a program to control the blood pressure of a patient. The Robotic system outputs cause the blood pressure to rise out of the normal limit, that is a component system failure and a fault in the overall system. If limit of blood pressure cross and it shows "Blue" zone that is a symptom of the system fault. We must concentrate the faults raised at the system with how to manage them. Here we use different SFT techniques to manage faults arise at Automated Healthcare System.

SFT Techniques

Techniques which are supported to software fault tolerance may detect the **faults**, prevent the faults, or remove the faults. Fault prevention: preventing the occurrence or introduction of faults for quality assurance and design methodologies. Fault removal: remove faults after the development stage is completed. This is done by exhaustive and rigorous testing of the final product [5]. Fault avoidance/prevention includes design methodologies which avoid the faults which may not have fault solution [6,10].

Fault tolerance makes the assumption that the system has unavoidable and undetectable faults and aims to make provisions for the system to operate correctly even in the presence of faults [6,7,10].

Software fault tolerance techniques are divided into two groups as Single version and multi-version software techniques [8]. Single version techniques focus on improving the fault tolerance of a single piece of software by adding mechanisms into the design targeting the detection, containment, and handling of errors caused by the activation of design faults. Single version techniques are Error detection, Exception handling, Data diversity, Process pair, etc. Multi-version fault tolerance techniques use multiple versions (or variants) of a piece of software in a structured way to ensure that design faults in one version do not cause system failures[14,13,8]. A characteristic of the software fault tolerance techniques is that they can, in principle, be applied at any level in a software system: procedure, process, full application program, or the whole system including the operating system Also, the techniques can be applied selectively to those

components deemed most like to have design faults due to their complexity. Multi-version fault tolerance techniques are as Recover block, N-version programming [9, 14].

Discussion

There are many reasons to be fail at particular system. A system fails because of incorrect specification, incorrect design, design flaws, poor testing, undetected fault, environment, substandard implementation, aging component, operator errors or combination of these causes. Though programming bugs is considered to be an important reason of the most system failures at present but the recent studies suggest that soft errors are increasingly responsible for system downtime [1]. To deal with errors in fault tolerance system classified as roll-forward and roll-back. Roll forward means to take the system to some specified location to resume the errors. Rollback means to take system to some earlier version [1, 5, 11].

Here, we analyzed different faults in the mentioned embedded system which can be managed by the fault tolerance techniques. We start with specification faults, (1) which can be managed by the rechecking design specification. Design faults (2) can be managed by Design diversy (NVP or RcB). NVP use multiple versions of the same requirements, where we can divert to another version on chosen design. In NVP if one of the design fail, at least one alternate version will work [12]. Implementation faults (3) which need to check properly before getting in use. We can use Verification, error detection and check point techniques. User faults (4) are of different types but generally it will wrong range of data which can be solved by input limit checking, exception handling. If user gives improper input so it would be handled by exception or checking the limit. Operator faults (5) which may work n improper instructions which can be managed by n-self checking or data diversy[14,13]. If any hardware or part of hardware would fail because of any reason we can apply hardware fault tolerance. Hardware fault tolerance gives alternate component for failed component [12].

Conclusion

Automated system usage will increase very vastly. The faulty system cannot tolerant anywhere, especially when it associated with human life. So, SFT techniques enable system more fault free system. As we discussed faults reside and arise from requirements of the system to working of an automated healthcare system system. Few fault managed by runtime by testing or err detection techniques but the fault arise at design level must need redesign. To do so we use recovery block or N-version programming. Any wrong data instructed (user faults) to system will handle by exceptional handling. If more processes are required and any faults occurred due to such reasons then we can apply Processor pair. So, in this way we can handle different faults with fault tolerance techniques for automated healthcare system which makes the system more faults free.

References

[1] Jean-Claude Laprie. Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese, volume 5 of Dependable Computing and Fault Tolerance. Springer Verlag, Wien, 1992,16-17p

[2].http://hissa.nist.gov/chissa/SEI_Framework/framework_9.htm

[3] Randell B. “System Structure for Software Fault Tolerance” IEEE Transactions on Software Engineering 1975 SE-1(2) 220–232p.

[4] Laprie J. C. et al. Hardware and software fault tolerance: definition and analysis of architectural solutions in Proceedings of 17th International Symposium on Fault-Tolerant Computing, Pittsburgh. 1987. 116-121p.

[5]Jean-Claude Laprie. Dependability—its attributes, impairments and means. InB. Randell, J.-C. Laprie, H. Kopetz, and B. Littlewood, editors, Predictably Dependable Computing Systems, ESPRIT Basic Research Series, pages 3–18.Springer Verlag, Berlin, 1995.

[6] CSE 598D: Software Fault Tolerance Instructor: Mahmut Kandemir, 23-24p

[7] Intrusion-Tolerant Architectures:Concepts and Design byPaulo Esteves Vessimo, Nuno Ferreira Neves, Miguel Pupo Correia

[8] P. A. Lee and T. Anderson. Fault Tolerance: Principles and Practice Second Edition, Springer-Verlag. 1990.

[9] Software Fault Tolerance Techniques & Implementation by Laura L Pullan, pages 68, 72-76

[10] Malicious- and Accidental-Fault Tolerance for Internet Applications Conceptual Model and Architecture David Powell and Robert Stroud, 23-24p.Available: http://www.Joomla.org/core-features.

[11] Jean-Claude Laprie. Dependability of computer systems: from concepts to limits.
In Proc. of the IFIP International Workshop on Dependable Computing and Its Applications (DCIA98), Johannesburg, South Africa, 1998.

[12] Avizienis A. "The N-Version Approach to Fault-Tolerant Software" IEEE Transactions on Software Engineering 1985. SE-11(12) 1491-1501p.

[13] Software Fault Tolerance: An Evaluation by Anderson, T., Barrett, P.A., Halliwell, D.N. Moulding, M.R. In Software Engineering, IEEE Transactions on,2006,1502 – 1510p

[14] A SURVEY OF SOFTWARE FAULT TOLERANCE TECHNIQUES byZaipeng Xie, Hongyu Sun and Kewal at citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.1320.

[15] Dorf RC: Internntioricrl €iicyclopeditr of'Robotics: Applications andAutomatiori, John Wileyand Sons, New York, 1988

[16] www.electronicsteacher.com/robotics/type-of-robots.php