Towards Reliability of Mechatronic Systems

-Focus on Software Reliability-

N. Jazdi
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
Nasser.Jazdi@ias.uni-stuttgart.de

C. Maga
Bosch Software Innovations GmbH
Department of Systems Engineering INST/ESY
Stuttgart, Germany
Camelia.Maga@bosch-si.com

***Abstract*— This paper discusses issues related to reliability of mechatronic systems and proposes a method to calculate it. Since the software reliability is the crucial aspect that can influence the entire system reliability, it deserves special attention. Hence, this paper presents in addition the influencing factors on the software reliability.**

Keywords-reliability, software reliability, mecatronic systems

I. INTRODUCTION

A mechatronic system is an integration of mechanical and digital electronic components that process information. It refers to the increasingly common melding of the disciplines of mechanics, electronics, electrical engineering, and information systems. The term "mechatronic system" is defined in [1] as follows: "Mechatronics refers to the synergistic interaction of the technical fields of mechanical engineering, electrical engineering, and information technology in the design and manufacture of industrial products and in process design." The rapid transformation of purely mechanical systems into integrated mechatronic systems also presents new challenges for reliability engineering.

In the current context of globalized competition the high-demanding customer requirements yield to more and more complex systems. This increased complexity usually stems from integration of suitable electronics and data processing elements into former strictly mechanical systems. First, a shift of functionality from mechanics to electronics occurs, and then an extension of enhanced and new functionalities takes place [2]. Finally, the systems gain certain intelligence and autonomous characteristics [2]. This happens, for example, by integrating modern microcontrollers, sensors and actuators into the system and implicitly providing the system's functionality on electronic-digital way.

Additionally, with regards to the trend to more efficient, more complex and nevertheless profitable systems, it has been observed that customers assign a paramount priority to reliability issues, especially in the automotive field [3].

Unfortunately, the reality differs from this wishful thinking. Since short time-to-market represents an advantage in the globalized competition, the development duration is continuously shortened; despite the increased complexity. The result is sinking of reliability [4]. Despite the fundamental importance of reliability regarding customer satisfaction, , nearly everyone has once encountered software unreliability. [4].

In order to mitigate this problem, the research on the field of reliability must find new answers for a holistic analysis of complex mechatronic systems. In this paper we consider the reliability of the software of mechatronic systems. Hence, there are big experiences on the field of hardware (we mean mechanical and electronic components) reliability.

II. TERMINOLOGY

A. Reliability

The reliability of a product is defined in [3] as follows:

"Reliability is the probability that a product does not fail during a defined duration and under given functional and environmental conditions."

Thus, the observation of functional and environmental conditions of a system gains in importance. As an analogy, in case of software, it has to be considered which functions have to be executed and which use intensity and duration are aimed [5].

The prediction of hardware reliability takes into consideration influencing factors like rugged operation conditions, temperature strokes, overvoltage or overcurrent [6]. Thus, the operation of a hardware system under different environmental conditions yields to various failure probabilities [5]. Stress-factors have to be identified and considered during the evaluation of the reliability of a system [7], [8].

B. Software Reliability

In contrast to hardware, software failures do not have physical causes. On one hand, a software failure can be caused by an inherent error made by a software developer. On the other hand, a function deviating from the specification can cause a failure during execution [10]. In any case, the number of existing errors in a software application remains constant. They have been made during implementation (implementation error) or during design (specification error). The error occurrence, however, depends on the execution environment of the software.

Since errors are made during the software development process, increasing the quality of software development directly influences the reliability of the implemented software. A quantitative determination of the software reliability is possible by using models related to empirical data about failures of the considered software. These data can be collected during the test phase or during the operation. According to [8], the software reliability is calculated as follows:

$$R_{Software} = \frac{correct\ executions}{total\ executions}$$

For this method, it has to be observed whether an inherent error has been activated during execution and consequently, an incorrect result occurs [8]. Even in this case, the reliability of the software application depends on the concrete requirements. Since empirical data are missing in the early project phases, the software reliability has to be calculated due to indirect values, like the estimated code complexity [11].

Another possibility to calculate the software reliability is the usage of qualitative methods like FMEA or FTA. There are different classifications of methods to calculate the software reliability. For instance, [4] proposes the following classification criteria:

- Distinction between micro and macro models
- Usage in a certain development phase
- Classification of architecture-based models in path-based, state-based and additive
- Classification in models for time intervals or models for value ranges
- Classification concerning probabilistic characteristics
- Further classifications, e.g. after Goel or Pham.

Reference [5] describes a method to calculate the reliability of industrial automation systems. The approach is based on the idea of propagation of experiences from operation of electronic parts and from execution of software components. In doing this, information related to reliability is selected from catalogues or gained from empirical data, then saved together with the concerned components. These data are then applied to the new industrial automation system that contains these components. This step is supported by numerical methods and dedicated software tools.

In order to determine the reliability of a system, empirical data describing the failure of the considered software component have to be available. These data are collected from reliability tests. Subsequently, the reliability is calculated by statistic evaluation. The obtained value is checked by computing the confidence interval and is saved together with the related software component. When this software component is used in a new industrial automation system, the data related to reliability will already exist.

In analogy, reliability data of electronic parts can be selected from catalogues. The data are captured by the developer and are independent on a reliability evaluation.

III. An Iterative Method to Increase Software Reliability

This section presents an iterative method that allows a quantitative estimation of software reliability. Thus, software reliability becomes a metric for the software quality.

Let us consider the definition of the software reliability, which is given in the previous section. From this definition it can be induced that any software reliability computations depend on the error frequency and the related number of executed test cases. The main idea, which is followed in this approach, is the assertion that high-quality software shows fewer errors and has a high reliability. Hence, the goal is to develop software by combining high-quality software components. One of the current research areas at the Institute of Industrial Automation and Software Engineering of the University of Stuttgart focuses on identification of influencing factors related to software reliability. At present, the following influencing factors have been identified:

A. *Usage of a Procedure Model during Software Development*

In order to control the complexity of software, procedure models are used. The goal is to apply engineer-like methods during the software development, so that on one hand, high-quality software can be implemented, and on the other hand, time and costs can be mitigated [12].

Procedure models divide the development process into accurately defined phases, which are grouped according to temporal aspects. The software is developed stepwise. The results are evaluated after each phase and the next phase is started only after release. The entire process is accompanied by project management and quality assurance. They define the mandatory aspects of the logical and the temporal structure of the development. In addition to procedure models, there are further factors that have an impact on the software quality. Such factors are, for example, the technical background and the programming skills of the developers, but also the utilized technology [14].

There are different procedure models, like the waterfall model or the spiral model. They differ in their level of detail and the temporal aspects of the described activities. In the past years, different assessment methods of the development process have been conceived. They aim to evaluate and quantify the maturity level of the development process. Typical assessment methods are Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI) and Spice [13].

For measuring the influence of this factor on the reliability of the software we have to quantify it [15]. For the quantification, the use of the 5 levels of the CMM is suggested, where 1 means “initial” and 5 corresponds to the level “optimizing”. Up to now, we have been also employing this suggestion.

B. *The Test Factor*

Testing is a fundamental method to increase the software reliability and implicitly, the quality of the software [17]. The goal of testing is to find as many errors as possible within a test unit. In case no errors are found, the test is not successful.

Unfortunately, complete tests become impossible with the increasing complexity of software. The rise of the number of all possible test cases is so gigantic that their complete execution would take several billions years. Hence, it is necessary to select and to prioritize test cases. Generally it is aimed to select especially those test cases, which would reveal as many severe errors as possible.

Even in the case of a fully-tested software component, it cannot be guaranteed that its behavior will not cause a failure. The reason for this is the already described dependence on the execution context.

The test factor can be quantified by regarding different metrics like efficiency, performance and amount of considered test cases. Level 5 corresponds thereby to an optimal test execution.

Let us now assume that the higher the level for the described factors determined for a software component, the higher is its reliability. This assumption is based on literature research and on experiences from former projects.

A systematic software development according to a procedure model yields, on one hand, to a precise documentation and tracking of the project requirements. On the other hand, it enables the error recognition at the early stages by prescribing mandatory test activities [16]. In consequence, the number of specification and implementation errors declines.

The goal is to develop the software by deploying software components with high reliability values for all presented factors. However, the issue of interaction of the software components within an application still needs intensive research. The mutual interaction between software components can lead to undesired effects.

C. *The Human Factor*

A significant influencing factor on the software reliability is the human factor. In case an experienced developer implements the software, it can be assumed that this contains fewer errors than that of an inexperienced developer. Hence, solid training and self-motivation could reduce the number of errors decisively. Development methods like cleanroom software engineering or agile methods admit the special role of the software developer and aim to increase one's productivity and efficiency [16].

There are numerous factors that have an impact on the productivity of software developers, like work conditions, difficulty of the software to be implemented, project basic conditions and several individual factors. These factors can influence the error amount directly or indirectly [14]. The developers have to be able to follow the development process and to use the existing technologies. Similar to the other sections, levels from 1 to 5 are suggested, where 1 represents the lowest and 5, the highest productivity.

IV. Calculation of System Reliability for Mechatronic Systems

A mechatronic system contains mechanical, electrical and information processing components. Hence, the system reliability has to be calculated by considering both hardware and software reliabilities.

In case given requirements regarding the system reliability have to be fulfilled, one must first analyze the composing elements (mechanical, electrical and software components). Then, the hardware components are selected, given that their reliability values are already determined. They are stemming from empirical data. Only the software components permit to influence the system reliability, since they are variable and their quality characteristics can be modified. This statement is based on two assumptions. First, it is assumed that no reusable software components with a defined failure rate are available. Second, it is assumed that high-quality software components have lower failure rates than other software components. Hence, if the mechatronic system contains high-quality software components, its failure rate will be low. The following table represents this idea.

TABLE I. Evaluation Metrics

Influencing Factor	Example	Min Value	Max Value
Usage of a procedure model	No procedure model / CMMI Level 5	1	5
Test execution	Random test / C1-coverage, specification techniques and white-box-tests	1	5
Developer experience	New or inexperienced developer / developer with 10 years experience in the domain	1	5
Lines of code	> 10.000 LOC / < 1000 LOC	1	5
Cyclomatic complexity	G > 50 / G < 10	1	5

In case a software with a low failure rate is required, e.g. for diagnosis systems, then this software has to be developed in a way that allows high values of the metrics presented in the table above. The system reliability can be affected positively so that the overall reliability requirements can be fulfilled. However, this implies high costs for the software development companies.

According to the presented factors, which have an influence on software reliability, we suggest a method for considering these factors during the development. The following flow graph shows the steps. First of it should be investigated whether the project could be built out of reusable components. In this case we could calculate the reliability of the software within existing methods. In the case that we built the software from scratch, we have to consider the five suggested metrics for calculating the reliability of the software.

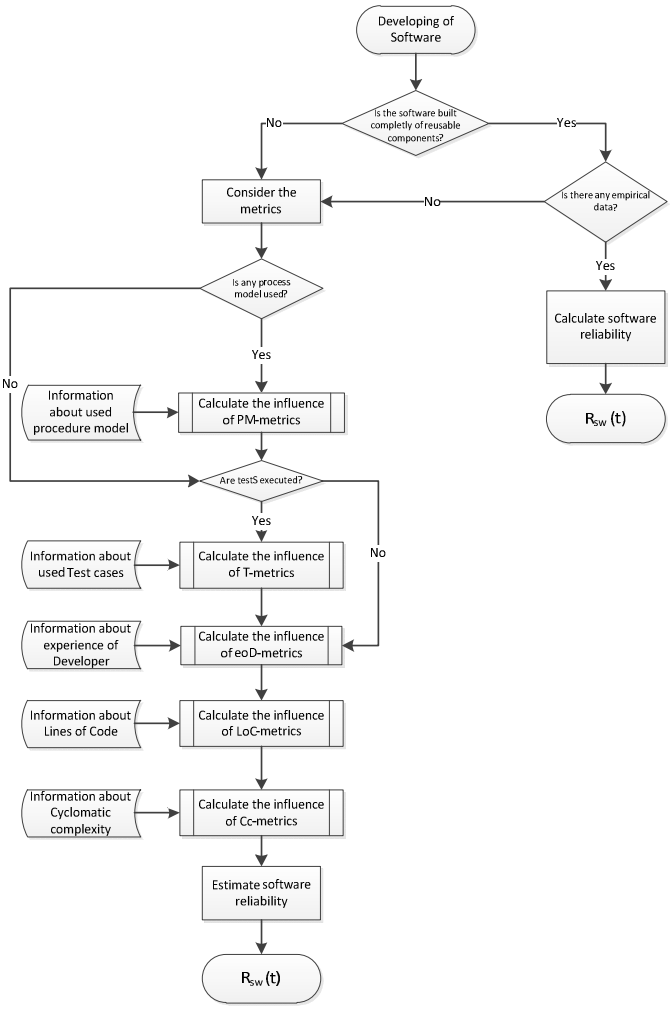

Figure 1: Flow graph for considering the metrics

V. CONCLUSION AND OUTLOOK

This paper presented a method to calculate the reliability of mechatronic systems. Since the software reliability is the crucial aspect that can influence the entire system reliability, it deserves special attention. For this reason, this paper discussed the influencing factors on the software reliability as well.

However, there are still two aspects that have not been thoroughly considered in today's research. First, it is necessary to quantify the quality characteristics in more detail. In order to do this, it is suggested to use existing experiences from literature and from finished projects. Second, it should be possible to derive the reliability of software components from the presented quality metrics. These two issues are the focus of our further research.

REFERENCES

[1] W. Heise, "The small 1 x 1 book of reliability and LCC," Lulu.com, 2009

[2] R. Isermann, "Mechatronic systems: fundamentals," Springer Verlag, London Limited, 2005

[3] B. Bertsche, "Reliability in automotive and mechanical engineering," Springer Verlag, 2008

[4] B. Bertsche, P. Goehner, U. Jensen, W. Schinkoethe, and H. J. Wunderlich, "Reliability of mechatronic systems -Principles and evaluation in early development phases," Springer Verlag, Berlin Heidelberg, 2009

[5] W. Heise, "The small 1 x 1 book of reliability and LCC," Lulu.com, 2009

[6] F. Saglietti, N. Oster (eds.) "Computer safety, reliability, and security," Proc. 26th International Conference SAFECOMP, 2007

[7] O. Koller, N. Jazdi, P. Goehner, U. Hipp, T. Liedtke, and A. Meyer, "Calculating failure rates rgarding field conditions," atp edition, 10-2011

[8] N. Jazdi, "Lecture notes reliability and safety of automation system," Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2011

[9] M. Wedel, "Assessment of the reliability of automation systems in the early development phases," Shaker Verlag, 2/2011

[10] J. D. Musa, "Software reliability engineering - an overview, more reliable software faster and cheaper," Software Reliability Engineering and Testing Courses, 2004

[11] J. D. Musa, A. Iannino, and K. Okumoto, "Software reliability: measurement, prediction, application," Mcgraw Hill Software Engineering Series, 1987

[12] P. G. Neumann, "The Risks Digest," ACM Committee on Computers and Public Policy, http://catless.ncl.ac.uk/Risks/

[13] B. P. Gallagher, M. Phillips, K. J. Richter, and S. Schrum, "CMMI-ACQ. Guidelines for improving the acquisition of products and services,"Addison-Wesley, 2009

[14] J. Ludewig, H. Lichter, "Software engineering – basics, people, processes, methods," 2. Add., dpunkt.verlag Heidelberg, 2010

[15] Ch. Ali Asad, et al, "An Approach for Software Reliability Model Selection," Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04) 0730-3157/04, IEEE 2004

[16] P. E. McMahon, "Integrating CMMI and agile development: case studies and proven techniques for faster performance improvement," SEI Series in Software Engineering, 2010

[17] N. E. Fenton, "A Critique of Software Defect Prediction Models," IEEE Transactions on Software Engineering, Vol. 25, No. 5, September /October 1999.