




Fault Management Framework and Multi-layer Recovery Methodology for Resilient System


Carlo Vitucci 
Technology Management
Ericsson AB
Stockholm, Sweden
carlo.vitucci@ericsson.com

Daniel Sundmark 
Computer Science and Software Engineering
Mälardalen University
Västerås, Sweden
daniel.sundmark@mdu.se

Marcus Jägemar 
Sys Compute Dimensioning
Ericsson AB
Stockholm, Sweden
marcus.jagemar@ericsson.com

Jakob Danielsson 
Sys Architecture
Ericsson AB
Stockholm, Sweden
jakob.danielsson@ericsson.com

Alf Larsson 
Senior Specialist Observability
Ericsson AB
Stockholm, Sweden
alf.larsson@ericsson.com

Thomas Nolte 
Division of Networked and Embedded System
Mälardalen University
Västerås, Sweden
thomas.nolte@mdu.se

Abstract—Fault management is a key function to guarantee the quality of the service. Research has done a lot to improve fault supervision, and investigation is ongoing in fault prediction, thanks to the potentials of artificial intelligence and machine learning. In this study, we propose a fault management framework that puts an emphasis on fault recovery: a framework developed on multi-layer function and a fault recovery methodology distributed over several technological layers. The basic principle of our proposal is that the system’s complexity exposes it to a higher probability of temporary error. Newfound attention to the fault recovery phase is the key to keeping the service’s quality high and saving maintenance costs by decreasing the return rate.

Index Terms—Fault Management, Resilient system, Recovery methodology.

I. INTRODUCTION

Fault management is a system function that has the purpose of detecting, locating, isolating, and recovering possible fault conditions in the system [1]. In recent years, connectivity between devices with different functions has increased exponentially [2], together with the complexity of interconnected systems and the difficulty of efficient fault management. Fault management has become synonymous with reliable systems, security and performance optimization [3], [4].

The need to design a system with dedicated fault management support is well-known, and it already emerged during the ’90s. However, the attempts to add knowledge of the correctness model into the system create drawbacks, like increased resource usage, increased design complexity, and reduced maintainability [5], [6]. The research in the Artificial Intelligence and Machine Learning (AI&ML) field has aroused new attention in fault management design based on fault prediction [7]–[14]. A fault management implementation designed to maintain the impact on performance and satisfy fault prediction solutions based on AI&ML has increased the number of system requirements used in selecting hardware and software components. The main goal of matching the

new requirements is supporting the implementation of reliable systems. [15]–[17]. Nevertheless, there is still one of the main functions of fault management that needs attention to increase the product lifetime: fault recovery.

This paper introduces:

- a multi-level representation of the fault management framework showing significant fault recovery oriented functions.
- a multi-technologies fault handling approach where the fault management framework develops as recursive iterations on different technological layers to recover from a fault condition.

The multi-level functions diagram (figure 1) shows the relationship between typical fault management functions (for example, fault tolerance, logging, and reporting), and the five strategies with which it is usual to define fault management: supervision, isolation, location, prediction, and recovery. The multi-level function representation integrates the research done in three areas:

- 1) early hardware design
- 2) early system design
- 3) recovery procedures

Both multi-level function framework and multi-technologies fault handling approach focus on system resilience. The functions in the fault management framework use the hardware and software capabilities to increase recovery abilities in production and runtime. At the same time, the multi-technologies look at a holistic methodology to recover from a fault condition.

II. RELATED WORKS

Even if fault management is not a new concept, several ongoing projects are reconsidering the fault management principle due to the higher complexity of the latest networking solutions. Arfaoui et al. [18] provided a non-exhaustive

list of current security, trust, and resilience issues that are critical to be explored in 5G. They introduced two ways to ensure the security and resilience of critical nodes for a software-defined network: cross-layers fault management and metric identification. However, the cross-layers proposal is referring to specifically SDN-NFV architecture. Sanchez et al. [19] focused more on the product's service reliability and the software-defined network's contribution to the diagnostic process. They propose fine granularity templates that model network nodes and provides more detailed diagnostic suitable for SDN-NFV. However, they didn't consider how to use the diagnostic in the recovery phase. Cherrared et al. [1] propose a comprehensive state-of-the-art fault management techniques and a new classification of the recent fault management research achievements in the network virtualization environments and compare their significant contributions and shortcomings. Several studies groups ([20]–[24]) investigate the usage of machine learning and artificial intelligence to predict a possible system's faulty condition.

We can summarize that the ongoing fault management research focuses on three main areas: security in the modern networking solution, fault supervision and information propagation in the cloud or distributed system, and machine learning and artificial intelligence impacts and contributions to fault management. In this paper, we revise the fault management framework to emphasize the strategic role of fault recovery actions for the system's resilience.

III. FAULT MANAGEMENT FRAMEWORK

In a previous work, we propose a fault taxonomy as an extension to a model described by Wuhib et al. [25]. We utilize the findings of our previous study as starting point for this paper. Our extended fault management framework considers five main components: *fault supervision*, *fault location*, *fault isolation*, *fault prediction* and *fault recovery* depicted as yellow blocks in figure 1.

Blue lines identifies logical links between the fault management's main components. Fault supervision, for instance, detects and forwards a suspected fault to the fault location. Fault location forwards the info to the fault isolation to avoid fault propagation into the system. Eventually, the fault management runs all possible fault recovery actions, using fault prediction information if available, to prevent a possible fault in advance. The five components of fault management represent the fault management macro functions, depicted as yellow blocks in figure 1. We call these components "first level function". The innovative contribution of this paper is the representation of the multi-level functions. The framework shows functions on different levels, depicted in the figure with varying colors for info-graphic purposes. The "depth" for the levels must not be confused with an assignment of priority or hierarchical order among the functions listed in the figure. Instead, the multi-level functions in the figure 1 shows an extra level of detail: the higher the function level, the lower the function abstraction. This level-view is necessary because first-level functions use many features or sub-functions, which

we call "second-level functionality". Figure 1 marks second level functionalities as green blocks. Second level functions have more significant consequences (direct or indirect) in the fault recovery mechanisms compared to the previous, first level functions. In details:

- *Error Report Support* uses our previously introduced fault taxonomy to classify and collect information on the detected fault. Our fault taxonomy definition enables us to propagate the fault information based on any possible recovery actions, as described in a recent study [26].
- *Filtering & thresholding* implements mechanisms for verifying the persistence of errors over time and frequency, such as, for example, leaky buckets or hysteresis process thresholds [27]–[29].
- *Hardware-Assisted Fault Management* is a hardware feature. Our ambition is to encourage hardware vendors to consider this feature a mandatory requirement when designing new devices. Hardware-assisted fault management enables the hardware to provide fault information at high granularity, the feature of the hardware to implement self-test mechanisms, and to automatically correct possible error conditions (in run-time or its own while performing the self-test) [30]–[33].
- Both error report support and filtering & thresholding function have a relationship with the *self-test & validation function*, depicted as a purple block in figure 1. This function requests an embedded system to perform self-test and validation. Hardware assisted fault management allows better test coverage and execution time, and reduces the impacts on the performance of the self-test. Self-test and validation procedures are also applicable throughout the overall lifetime of the product, for example:
 - The production test.
 - The "cold" test, a functional verification test performed during the system power-on phase in run time or on-demand in the case of programmed fault recovery actions.
 - The repair center test, when a board installed by the customer is sent for assistance due to a reported malfunction status.

The natural users of hardware designed assisted fault management are fault location and fault isolation. The ability to locate errors with a very detailed granularity allows, by definition, its identification and the domain where to operate the actions to avoid the propagation of the fault.

- The fault recovery component uses the *fault tolerance function*. This function is hardware (or software) capable of supporting and managing redundancy elements and compensating for local errors [34]–[36]. Various disk RAID configurations [37] are typical examples of fault tolerance mechanisms and attempts to make the system survive critical data corruption on disk. Especially with the recent development of cloud infrastructures and the consequences of deployment in a virtual environment,

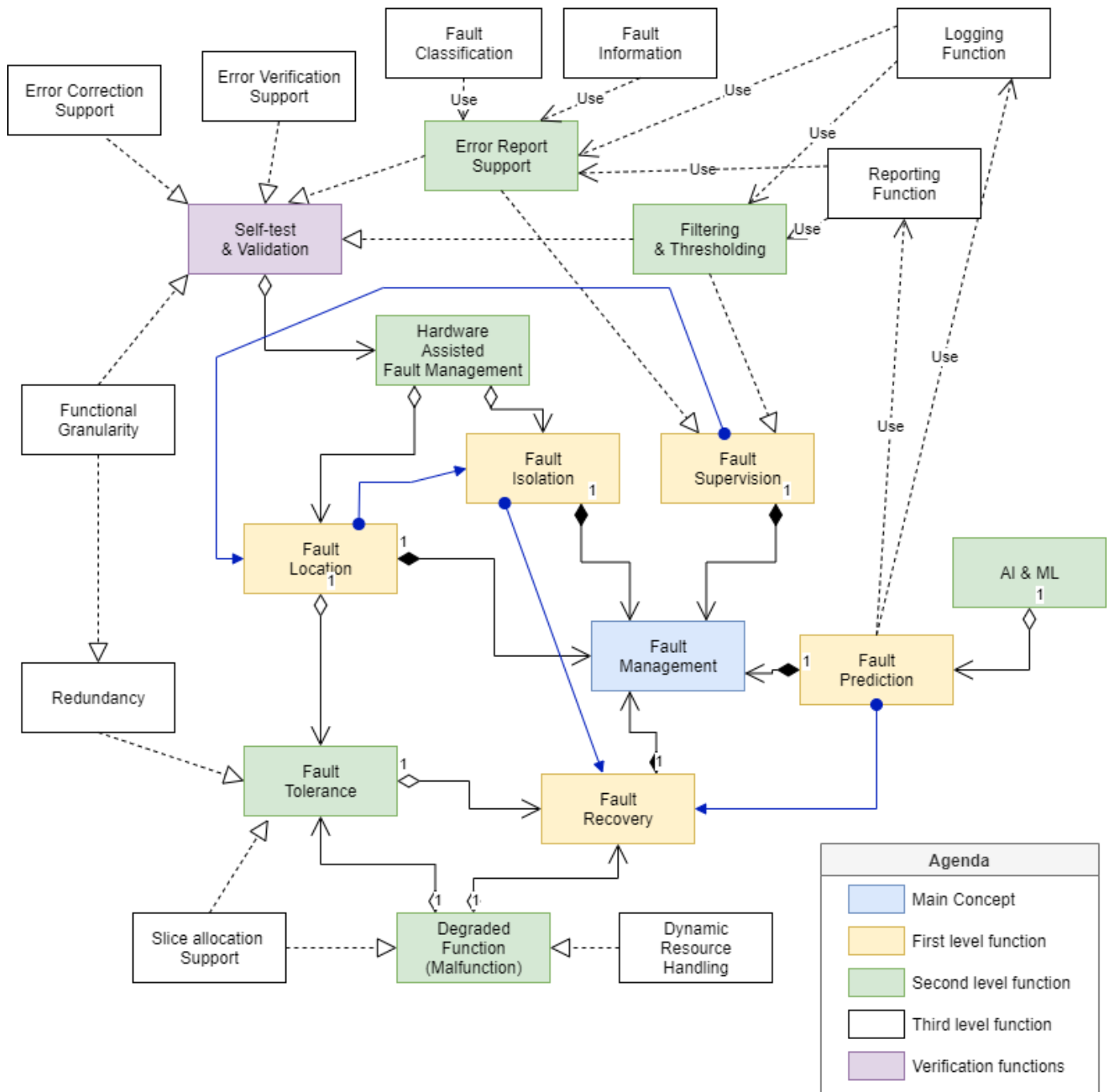


Fig. 1. Fault Management multi-levels function Framework

fault tolerance also uses the hardware and software attribute to allocate a "slice" of the available infrastructure to any (virtual) functions [2]. In case of a non-recovered fault in a part of the infrastructure, a new slice allocation (not already in use by other services) allows the virtual function to continue (or restart) its service instead of permanently suspending execution.

- As described above, the connection between the *degraded function handling* and the *fault recovery* should appear

evident: suppose redundancy characteristics or availability of infrastructural slices are not sufficient in terms of fault tolerance. In that case, dynamic management of resources by the system should identify the new allocation to be reserved for the various vertical services to allow the system to work in a state of degraded function.

- Finally, our model can incorporate elements from the emerging *artificial intelligence and machine learning* field as a means of predicting a previous functional

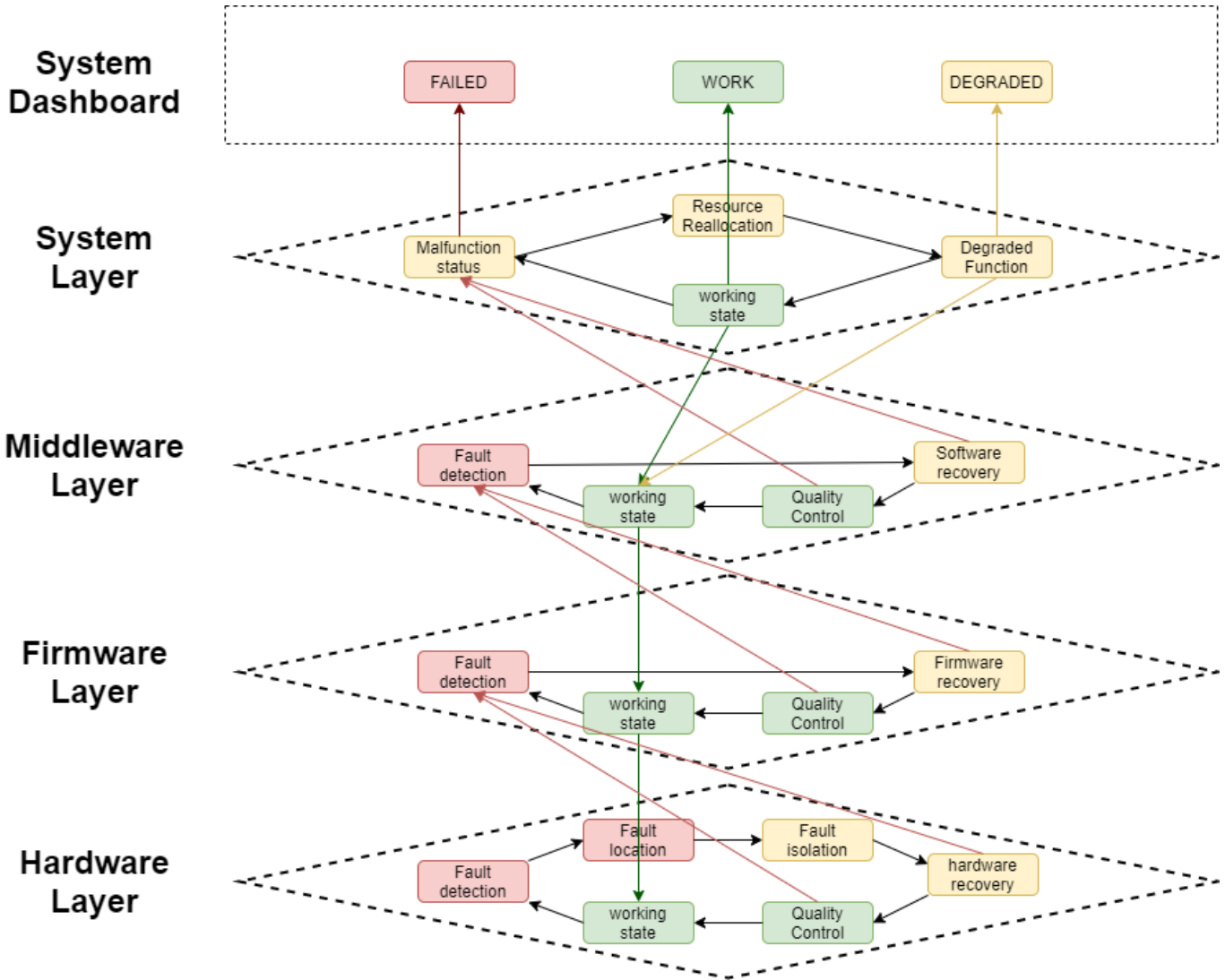


Fig. 2. Multi layers implementation approach for the fault management framework

degradation, i.e., when the fault prediction is used in turn as a fault recovery function [38], [39].

The third level functions are support functions that provide data management utilities as input to our previously explained second level functions. The third level functions are fundamental efficiency enablers. For example, the fault tolerance level can be very low or impossible without a good redundancy solution. Similarly, it is challenging to achieve acceptable levels of degraded function without the ability to manage the available resources dynamically. The third level functions are represented in figure 1 by the white blocks.

IV. THE MULTI-LAYERS APPROACH

The fault management multi-layer functions framework in the previous section describes fault management as a discipline that both detects faults and recovers the system from faulty conditions.

We depict different system events using the following colors:

Green: working state,

Red: faulty state and

Yellow: recovery state.

Figure 2 represents our multi-layers concept, including the different system states. The sequence of green-red-yellow state spreads over the multi-technology layers of the system. Any single layer tries to manage the state using its features or functions to either recover from a fault or involve a higher level to do it. Therefore, the attempt to recover from a faulty condition is iterative. In every single technological layer, the implementation of fault management admits only two possible output paths:

- 1) the return to a working condition, the green state
- 2) If the recovery attempt is unsuccessful, the passage to a higher level from a recovery condition (from a yellow state).

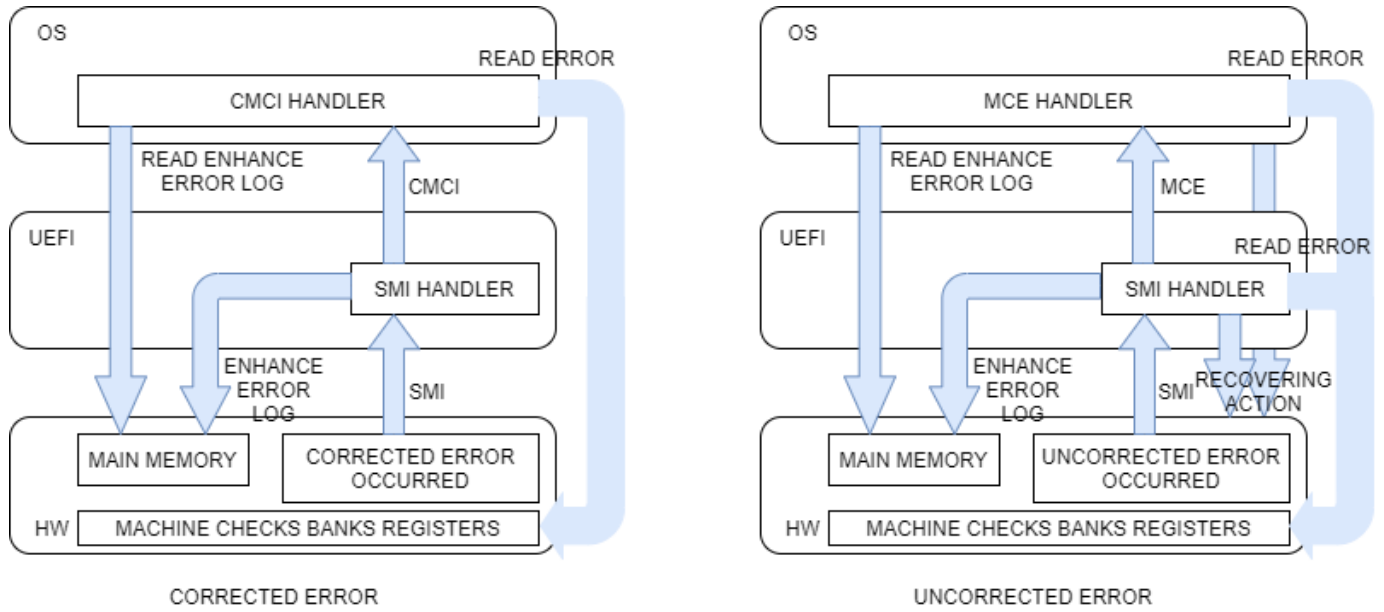


Fig. 3. Firmware-by-first model.

The process in figure 2 describes the propagation of a faulty condition up to the system level. In the final stage, the system manages the malfunction state. The recovery action contemplates a review of the allocation of available resources (considering as unavailable any resources whose condition remains faulty despite the several recovery attempts performed by the underlying technological levels). Malfunction handling can bring the system back to a working condition in a redundant resource system, distribute the remaining resource differently for a degraded function condition, or end up in a faulty system if neither redundancy nor resource reallocation is possible.

Fault management must pay special attention to quality control executed at each technological layer. The goal of the quality control is performing filtering of the error occurrence distribution. If the error is recoverable but continuously detected, the high occurrence frequency can be the sign of an aging issue, or it can cause performance degradation. In both cases, the resource seems compromised, and it is better to consider it a permanent fault condition.

Reliability, availability, and Serviceability (RAS) is a well-known model [40]. One of the fundamental concepts of the RAS framework is that an error indication from the hardware must be analyzed, localized, and possibly corrected by the firmware and be propagated to the operating system only if the firmware cannot recover the fault condition. The mechanism just described is called firmware-by-first, represented in figure 3 and major hardware vendors, especially processors and System-on-chip devices, and most common operating systems, support it. The firmware-by-first requires firmware (e.g., BIOS/UEFI) to receive and manage all hardware fault indications. Suppose it cannot find a suitable recovery action. In that case, firmware propagates the fault detection to the

middleware level (operating systems, e.g., Windows or Linux, can manage communications of hardware errors based on this model). The multi-layer technology implementation approach extends error management at the system level to enrich the fault recovery phase: the system can execute recovery actions requiring more complex functions and resource re-allocation than a re-initialization of a device or a board restart.

V. USE CASE EXAMPLE: MEMORY HARD-ERROR HANDLING

Soft or hard memory error means an error in the memory that cause an unwanted change of the data value. It is possible to further classify soft memory errors into chip-level and system-level. Chip-level errors are soft memory errors typically caused by radioactive decay of the proton in the chip or a bit-flip typically caused by cosmic rays. It is a rare event but the nanometer technology increase the likelihood of it. System-level soft-error refers to power supply, power distribution, clock problems, multiple software problems such as incorrect memory configuration or read/write problems which similarly can cause bit-flips. The soft-error classification refers to the physical condition of the non-persistence of the error, and it doesn't refer to the root cause of the fault. Similarly, by hard-error, we mean a permanent error in the memory, for which a fault recovery action is not possible. Recovery actions, with increasing but limited impacts for the system, are writing back the correct value, device re-initialization, and rebooting the system. The concept of persistence requires special attention. The persistence of the fault should not be confused with the persistence of the cause of the problem in the case of a software problem: a wrong configuration of the read/write cycle by the software will continue to cause soft-error, even if this should not be considered hard-error. The

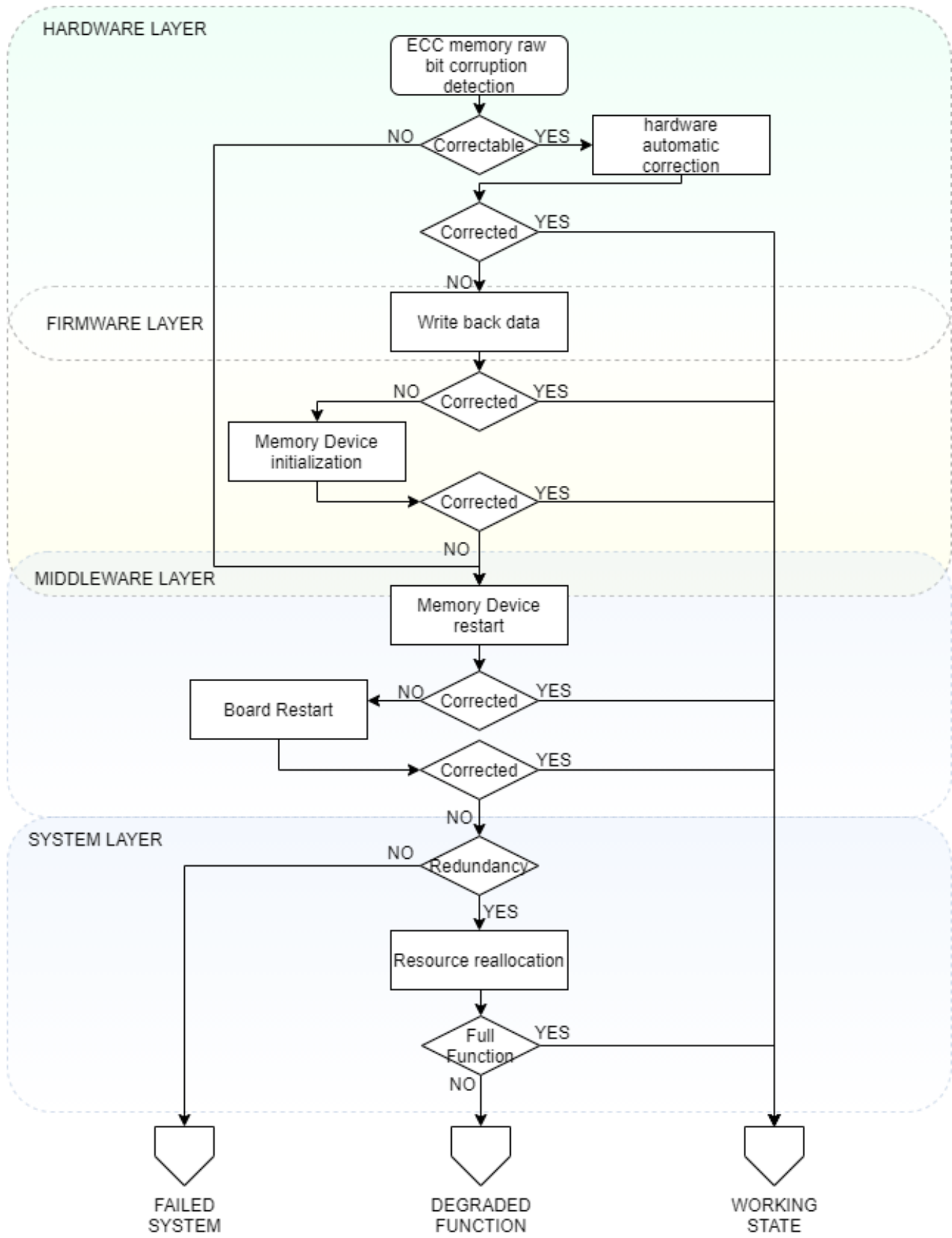


Fig. 4. Multi layers implementation approach Memory Error use case

use case describes the application of the multi-layers approach for memory bit-flip management. Figure 4 shows the logical flow. we take into consideration the single-bit error case. The description doesn't consider the possibility of an error on several bits. Still, the implementation flow of the recovery action is not very different if we exclude the impossibility for the memory controller to write back in the memory the exact value. Whenever a recovery action has been successful and the flow ends in the working state, fault management considers the memory error corrected.

One of our previous studies [26] uses the typology of memory fault detected by the hardware to apply defined topology between fault and recovery actions domains. This paper extends the case described in the previous study: it distributes the recovery actions on different technological levels and considers the handling of the hard-error and degraded function.

In the lower layer, the control mechanism available in the hardware reveals memory corruption. One of the most popular algorithms is ECC (Error Correction Code). According to the implemented algorithm, the hardware can detect an error condition on a single bit or multiple bits. If the hardware can autonomously correct the error, fault management only propagates the error as corrected for statistics purposes. Otherwise, it communicates the error as correctable to the firmware if the value before corruption is known. The firmware is the first to receive error information in memory. If feasible, the first recovery attempt is to write the correct value into memory and check that the write was successful. If the error is hard-error, writing the correct value will fail. Recovery actions are available on different layers. Using multiple recovery actions minimizes the risk that software issue misleads to a hard-error conclusion: new initialization of the memory, device restart (but this action is not always possible without a restart of the entire board) and board restart. As figure 4 shows, both the restart of the single device and the restart of the inter-board is a possible recovery action defined in the middleware layer. In the majority of soft-error cases, these recovery actions are very effective (see benchmark in [41]). In case of hard-error, the above recovery actions are not enough to fix the problem, and the fault management flow continues at the system level. If there are spare memory arrays, Fault management can use this redundancy capability instead of the faulty one. Final status is "working state" because the final result does not alter the service performance. If, on the other hand, the reallocation could not count on available spare resources, then service can count on fewer resources. Therefore the service is working in a degraded function condition.

VI. CONCLUSION AND FUTURE WORK

With a multi-level functions vision, the development of fault management solutions is more attentive to the completion of its final goal. Fault management must not limit itself to detecting a defect or faulty condition but holistically manage a fault by:

- Collecting the information necessary to manage the fault;
- Finding the best localization possible of a fault thanks to the optimal usage of the hardware features;

- Preventing the error propagation through the system;
- Constantly gathering system statistics to react in advance to a fault condition, with the great advantage of significantly limiting possible performance drawbacks on the service;
- Performing one or more actions to recover from a faulty condition and return to an operational state.
- In case of a non-recoverable fault, guaranteeing working condition even if under degraded function state.

None of the components of fault management has a higher priority than the others. Each of them is fundamental to the economy of the product. In particular, the fault recovery component directly impacts the maintenance cost. The above considerations allow us to outline interesting fields for future studies.

We think that an analysis of the distribution of the cost of a fault will show that investments to improve fault management during the development and design phase of a product has a cost in the order of thousands of times lower than the management of a fault during the maintenance phase.

For the same reason, it will be beneficial to identify a series of hardware and software requirements aimed at realizing the support functions of the fault management with a minimum impact on functional performance.

Among the possible functions to support fault management, this study mentioned using AI&ML in a cloud-based environment. The extension of the multi-layer functions approach in a cloud environment is essential for developing fault management in the latest generation (5G and 6G) networking and telecommunication systems.

REFERENCES

- [1] S. Cherrared, S. Imadali, E. Fabre, G. Gossler, and I. G. B. Yahia, "A survey of fault management in network virtualization environments: Challenges and solutions," *IEEE Transactions on Network and Service Management*, vol. 16, pp. 1537–1551, 12 2019.
- [2] C. Vitucci and A. Larsson, "Flexible 5g edge server for multi industry service network," *International Journal on Advances in Networks and Services*, vol. 10, pp. 55–69, 12 2017. [Online]. Available: https://www.researchgate.net/publication/322423817_Flexible_5G_Edge_Server_for_Multi_Industry_Service_Network
- [3] A. Dusia and A. S. Sethi, "Recent advances in fault localization in computer networks," *IEEE Communications Surveys and Tutorials*, vol. 18, pp. 3030–3051, 10 2016.
- [4] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, pp. 165–194, 2004.
- [5] E. Marilly, A. Aghasaryan, S. Betgé-Brezetz, O. Martinot, and G. Delègue, "Alarm correlation for complex telecommunication networks using neural networks and signal processing," *2002 IEEE Workshop on IP Operations and Management, IPOM 2002*, pp. 3–7, 2002.
- [6] M. Miyazawa and M. Tsurusawa, "Fault localization mechanism using integrated resource information model in next generation network," *Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, NOMS 2010*, pp. 734–747, 2010.
- [7] N. Lashkari, H. F. Azgomi, J. Poshtan, and M. Poshtan, "Asynchronous motors fault detection using ann and fuzzy logic methods," *ECCE 2016 - IEEE Energy Conversion Congress and Exposition, Proceedings*, 2016.
- [8] —, "Robust stator fault detection under load variation in induction motors using ai techniques," *Proceedings - 2015 IEEE International Electric Machines and Drives Conference, IEMDC 2015*, pp. 1446–1451, 2 2016.

- [9] K. Michail, K. M. Deliparaschos, S. G. Tzafestas, and A. C. Zolotas, "Ai-based actuator/sensor fault detection with low computational cost for industrial applications," *IEEE Transactions on Control Systems Technology*, vol. 24, pp. 293–301, 1 2016.
- [10] H. A. Shiddiqy, F. I. Hariadi, and T. Adiono, "Power line transmission fault modeling and dataset generation for ai based automatic detection," *Proceeding - 2019 International Symposium on Electronics and Smart Devices, ISESD 2019*, 10 2019.
- [11] R. Jothi, "A comparative study of unsupervised learning algorithms for software fault prediction," *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, pp. 741–745, 3 2019.
- [12] T. Zheng, Z. Wang, C. Tan, and R. Wang, "Research on fault prediction and diagnosis method of pcb circuit," *Proceedings - 2020 International Conference on Artificial Intelligence and Computer Engineering, ICAICE 2020*, pp. 387–390, 10 2020.
- [13] F. Karpat, A. E. Dirik, O. Dogan, O. C. Kalay, B. Korcuklu, and C. Yuce, "A novel ai-based method for spur gear early fault diagnosis in railway gearboxes," *Proceedings - 2020 Innovations in Intelligent Systems and Applications Conference, ASYU 2020*, 10 2020.
- [14] L. Junyong, T. Yinyin, Z. Delin, Y. Feifei, and Z. Yufeng, "Fault prediction of electromagnetic launch system based on knowledge prediction time series," *IEEE Transactions on Industry Applications*, vol. 57, pp. 1830–1839, 3 2021.
- [15] N. Bidokhti, "The impact of reliability requirements on development life cycle," *Proceedings - Annual Reliability and Maintainability Symposium*, pp. 307–311, 2008.
- [16] A. Aljer and P. Devienne, "Formal fault tolerant architecture," *3rd Int. Conf. on Communication Theory, Reliability, and Quality of Service, CTRQ 2010, Includes MOPAS 2010: 1st Int. Conf. on Models and Ontology-Based Design of Protocols, Architecture and Services*, pp. 73–78, 2010.
- [17] S. Chattopadhyay, N. Sett, S. Nandi, and S. Chakraborty, "Flipper: Fault-tolerant distributed network management and control," *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*, pp. 421–427, 7 2017.
- [18] G. Arfaoui, J. M. S. Vilchez, and J.-P. Wary, "Security and resilience in 5g: Current challenges and future directions."
- [19] J. M. Sánchez, I. Grida, B. Yahia, and N. Crespi, "Self-modeling based diagnosis of software-defined networks."
- [20] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, pp. 393–430, 1 2019.
- [21] Z. Ding, J. Zhou, B. Liu, and W. Bai, "Research of intelligent fault diagnosis based on machine learning," *Proceedings - 2021 International Conference on Computer Network, Electronic and Automation, ICCNEA 2021*, pp. 143–147, 2021.
- [22] K. Moloi and A. O. Akumu, "Power distribution fault diagnostic method based on machine learning technique," *IEEE PES/IAS PowerAfrica Conference: Power Economics and Energy Innovation in Africa, PowerAfrica 2019*, pp. 238–242, 8 2019.
- [23] Z. Xiao, Z. Cheng, and Y. Li, "A review of fault diagnosis methods based on machine learning patterns," *2021 Global Reliability and Prognostics and Health Management, PHM-Nanjing 2021*, 2021.
- [24] H. Won and Y. Kim, "Performance analysis of machine learning based fault detection for cloud infrastructure," *International Conference on Information Networking*, vol. 2021-January, pp. 877–880, 1 2021.
- [25] F. Wuhib, C. Fu, and M. Soualhia, "A look at automated fault management with machine learning - ericsson," 2019. [Online]. Available: <https://www.ericsson.com/en/blog/2019/6/automated-fault-management-machine-learning>
- [26] C. Vitucci, D. Sundmark, M. Jägemar, J. Danielsson, A. Larsson, and T. Nolte, "A reliability-oriented faults taxonomy and a recovery-oriented methodological approach for systems resilience," *Proceeding IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 48–55, 6 2022.
- [27] B. Solomon, D. Ionescu, C. Gadea, S. Veres, and M. Litoiu, "Leaky bucket model for autonomic control of distributed, collaborative systems," *SACI 2013 - 8th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, pp. 483–488, 2013.
- [28] V. Vakkalanka and P. Mohan, "Fault tolerance using leakybucket and network border patrol," *Proceedings of the 6th International Conference on Communication and Electronics Systems, ICCES 2021*, pp. 1691–1696, 7 2021.
- [29] R. S. Srivatsa and W. K. Jenkins, "Learning characteristics of adaptive fault tolerant filters in the presence of transient errors," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 1, 2002.
- [30] R. Kothe, C. Galke, S. Schultke, H. Fröschke, S. Gaede, and H. T. Vierhaus, "Hardware/software based hierarchical self test for socs," *2006 IEEE Design and Diagnostics of Electronic Circuits and Systems*, vol. 2006, pp. 157–158, 2006.
- [31] S. Dikic, L. J. Fritz, and D. Dell'Aquila, "Bist and fault insertion reuse in telecom systems," *IEEE International Test Conference (TC)*, pp. 1011–1016, 2001.
- [32] S. Shamshiri, H. Esmaeilzadeh, and Z. Navabi, "Instruction level test methodology for cpu core software-based self-testing," *Proceedings - IEEE International High-Level Design Validation and Test Workshop, HLDVT*, pp. 25–28, 2004.
- [33] H. Hulvershorn, P. Soong, and S. Adham, "Linking diagnostic software to hardware self test in telecom systems," *IEEE International Test Conference (TC)*, pp. 986–993, 1995.
- [34] N. Ya, X. Wang, S. Zhang, and M. Huang, "Multipath fault-tolerance routing mechanism in data center network," *Proceedings - 2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science, DCABES 2018*, pp. 246–249, 12 2018.
- [35] T. Johnson and H. Lam, "Incorporating fault-tolerance awareness into system-level modeling and simulation," pp. 829–830, 10 2021.
- [36] P. Das and P. M. Khilar, "Vft: A virtualization and fault tolerance approach for cloud computing," *2013 IEEE Conference on Information and Communication Technologies, ICT 2013*, pp. 473–478, 2013.
- [37] J. Yao, H. Jiang, Q. Cao, L. Tian, and C. Xie, "Elastic-raid: A new architecture for improved availability of parity-based raids by elastic mirroring," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 1044–1056, 4 2016.
- [38] M. K. Das and K. Rangarajan, "Performance monitoring and failure prediction of industrial equipments using artificial intelligence and machine learning methods: A survey," *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020*, pp. 595–602, 3 2020.
- [39] B. Shayesteh, C. Fu, A. Ebrahimzadeh, and R. Glioth, "Auto-adaptive fault prediction system for edge cloud environments in the presence of concept drift," pp. 217–223, 11 2021.
- [40] S. Scargall, "Reliability, availability, and serviceability (ras)," *Programming Persistent Memory*, pp. 333–346, 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-4932-1_17
- [41] V. Sridharan, N. D. Bardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems the good, the bad, and the ugly," *ACM SIGPLAN Notices*, vol. 50, pp. 297–310, 4 2015.