VFAST Transactions on Software Engineering
http://vfast.org/journals/index.php/VTSE@ 2014 ISSN(e): 2309-3978;ISSN(p): 2411-6246
Volume 2, Number 1, January-December, 2014

pp. 29-38

COMPARATIVE ANALYSIS OF FAULT-TOLERANCE TECHNIQUES FOR SPACE APPLICATIONS

MUHAMMAD FAROOQ[1], MUHAMMAD WASEEM IQBAL[2] , TOQIR AHMAD RANA[3], NATASH ALI MIAN[4]

[1]Department of Computer Science & IT, The University of Lahore
alfarooq_srdc@yahoo.com

[2]Department of Computer Science & IT, The University of Lahore
muhammad.waseem@cs.uol.edu.pk

School of Computer and Information Technology, Beaconhouse National University
toqirr@gmail.com

School of Computer and Information Technology, Beaconhouse National University
natash.ali@bnu.edu.pk

Revised August 2014

ABSTRACT. *Fault-tolerance technique enables a system or application to continue working even if some fault /error occurs in a system. Therefore, it is vital to choose appropriate fault tolerant technique best suited to our application. In case of real-time embedded systems in a space project, the importance of such techniques becomes more critical. In space applications, there is minor or no possibility of maintenance and faults occurrence may lead to serious consequences in terms of partial or complete mission failure. This paper describes the comparison of various fault tolerant techniques for space applications. This also suggests the suitability of these techniques in particular scenario. The study of fault tolerance techniques relevant to real-time embedded systems and on-board space applications (satellites) is given due importance. This study will not only summarize fault tolerant techniques but also describe their strengths. The paper describes the future trends of faults-tolerance techniques in space applications. This effort may help space system engineers and scientists to select suitable fault-tolerance technique for their mission.*

Keywords: ALFTD; ALFT; Dependable Systems; Fault Tolerance; Recovery Block; Space Application; SNV; Watch-dog system;

1. **Introduction.** On-board real-time space applications (satellites) OBC are combination of hardware and software [1]. Space environment is very hard due to severe radiation effects, therefore on-board applications hardware (RAMs) and software are more vulnerable to soft error (transient faults) [2]. Strong space radiations can easily toggle/flip RAM bits which lead to incorrect on-board real-time computations and invalid data storage[3]. These situations have drastic effects on space missions. To avoid these situations experts suggest two major fault tolerance techniques for such applications/missions namely hardware redundancy mechanism and software based fault tolerance techniques[4]. For space applications hardware are very costly (space grade hardware), but are more reliable as compared to software based fault tolerance techniques[5]. For low cost micro satellites, experimental satellites efforts are made to use software fault tolerance techniques without compromising on system efficiency, functionality, performance and reliability[6]. The focus of our effort in this research is to find out and analyze software based fault tolerance techniques used in space applications. Advantages and disadvantages of these techniques, the future trends of these software fault tolerance techniques to handle soft errors, which could be used in low cost space applications and experimental applications is explored and discussed.

2. **IMPORTANT DEFINITIONS**

2.1. Dependable System:

"Trustworthy enough that reliance can be placed on the service it delivers" [7]. It has features of high availability, high reliability and also provides safe and secure operations. Dependability can be achieved by fault avoiding at system design time, fault removal during design implementation, and fault tolerance during system execution/ functioning [7] [8].

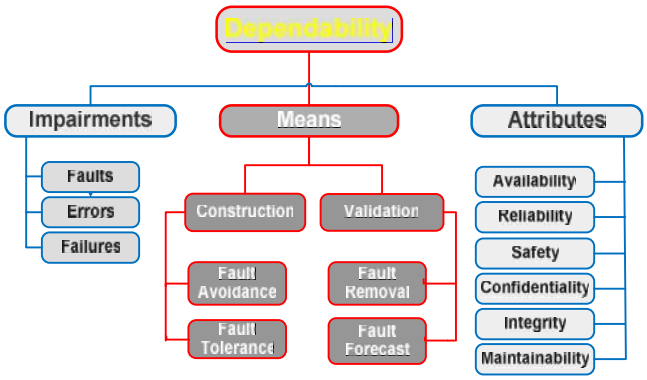

Figure 1. Dependable Systems [7]

3. Software Fault-tolerance Techniques For Space Applications

There are two main levels/types of fault tolerance mechanism/techniques for space applications; hardware fault tolerance and software fault tolerance. Hardware fault tolerance for space applications follows space grade hardware replication and it is not in the scope of this paper. Our focus is on software fault tolerance techniques for space applications especially satellites. Software fault tolerance mechanism is applied through software programming to recover the on-board system from program faults, data faults, and transient faults (soft errors due to space environment). In space applications OBC software fault tolerance design is implemented by three ways i.e. 1) implementation through single version of OBC software programming 2) implementation through multiple versions of OBC software programming 3) implementation through whole code uploading from ground station (in case of complete OBC software failure).

1)Single Version programming (SVP): Single version of the satellite on-board software is used. In this programming software fault tolerance implemented by applying exception handling, check point & restart, input data repression mechanisms. The main technique applied is backward recovery. In backward recovery the on-board satellite software faulty state is corrected by restoring the system to its previous correct state. The backward recovery scheme is implemented by using recovery blocks (RcBs). Recovery block has an executive, acceptance test, primary and alternate mechanisms/ algorithms. In embedded systems of satellites the recovery block also have watchdog timer (WDT). This scheme is good for recovering soft errors that arises in satellites OBC software due to space environment [7][10].

2)Multi-Version programming (MVP): In this scheme of on-board satellite software programming, two or more software versions are executed sequentially/concurrently to achieve high reliability. These versions are created using some kind of design diversity e.g. with different design teams or algorithms. In OBC of satellites, this fault tolerance technique is implemented using N-version programming (NVP) technique and forward errors correction mechanism is adopted in NVP. In forward error correction mechanism different design diverse/algorithms for the same process are executed simultaneously [7][10].

In real-time OBC of satellites NVP design have an executive, n variant (versions), and a decision mechanism (DM) and acceptable results obtained through DM are forwarded to be used in further on-board processing. Various forms of implementation of NVP in real-time OBC of satellites are a) all replicas on a single HW component b) NVP Replicas on multiple HW components c) NVP adjucator on separate HW component d) NVP complete program replicas vs program segment replicas.

4. SOFTWARE FAULT TOLERANCE TECHNIQUES USED IN SATELLITES

In satellites high reliability is achieved by using combination of hardware and software. The satellite on-board software design is very critical therefore it is not make public. Any space agency's documents are highly classified therefore not much data is available with respect to on-board software, specific to spacecraft missions. However following are the few examples of on-board software fault tolerance techniques used in experimental/micro satellites.

4.1 WATCH-DOG SYSTEM & ACCEPTANCE TESTS FOR PANSAT:

This on-board software fault tolerance technique was used in PANSAT (an experimental satellite). The main objectives of the satellite were to provide email, binary file transfer and radio services The OBC hardware of PANSAT consists of two space grade Intel 80186 CPUs, one 512 Kb working RAM having hamming code for detecting and correcting single bit errors. Hamming code can also detect two bit errors but could not correct them; however three or more bit errors cannot be detected by this code. Two 04 MB memories for email and binary file storage data. These memories are redundant for each other and do not have error correcting hamming code. Two 512 Kb non-volatile memories for storing telemetry and other important data. The main on-board operating system and application software was uploaded after the launch from ground. This code uploading mechanism provides the benefit of uploading the new enhanced versions of on-board software which also allows error rectification in the existing on-board software and adding additional features. Following figure shows the error classification/categories that PANSAT on-board software fault tolerance mechanism had to handle [11].

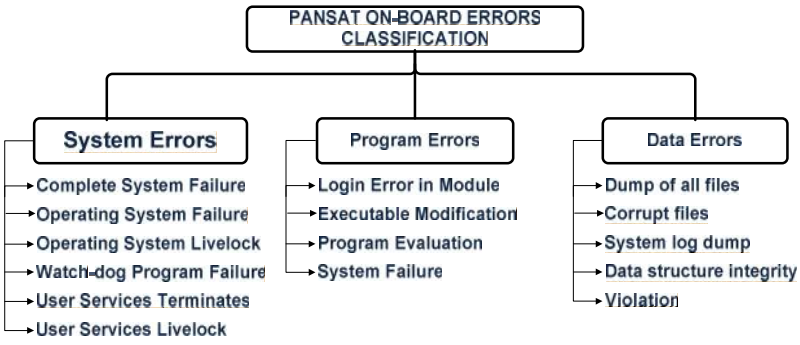

Figure 9: PANSAT Software Fault Tolerance Error Classes [11]

N-version and recovery block scheme could not be used for handling PANSAT on-board software faults mentioned above. Instead a strategy of watch-dog system along with acceptance tests to identify errors in each class was developed for PANSAT on-board software fault tolerance. Watch-dog function send query to each on-board software module after every 10 minutes and verify their health status. If no-reply or bad-reply received watch- dog system isolate the error and rectify it. Watch-dog function just provides remedies for the faults, occurred in the on-board software, instead of masking faulty results [11].

For system errors complete system, operation system failure and operating system live look ground station intervention is required to make the PANSAT on-board system fully functional, however for rest of the system errors, the system itself has the capability to rectify them on-board. For those system errors watch-dog function restart the relevant processes in case of errors, even if the problem not solved ground intervention is required. For program errors and data errors firstly on-board system itself try to rectify the error by resetting the faulty process and even if the problem does not solved the ground operator correct it by analyzing the error log file. These types of errors have least impact on the system and system continues to perform its operations. Error in the log file is also removed from ground, by uploading the new code. For all types of the above mentioned errors, acceptance test is the only mechanism to confirm them as error. Four major types of acceptance tests were designed and implemented for PANSAT on-board software fault tolerance scheme [11].

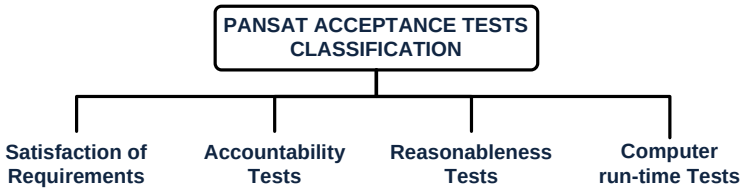

Figure 10: PANSAT Software Acceptance Tests Classifications [11]

Satisfaction of requirements acceptance tests were used to confirm that all subsystems are functioning properly and their output is according to the fixed criteria. For such types of acceptance test a predefined table for each subsystem is maintained in non-volatile memory and when the watch-dog function query for their status, the output of each system is verified against this table and correct/ incorrect status is communicated and rectified as explained above. In acceptance test each procedure take data, process it and check it against the table entries, as it is not possible to check all outputs against inputs, therefore only one or two test for each subsystem are performed. Satisfaction of requirements tests were used to confirm/check on-board data packetization process, on-board data compression/de-compression process/algorithms, position determination routines, correctness of file size and names, correct entry in the log file, and correctness of file defragmentation process. Second types of acceptance tests "Accounting tests" were used to 1) verify version number of user services by matching the counter of user service and file system. Mismatch counter indicate the error in user services and should be rectified. 2) To verify data packet numbers and their size, as both are fixed in the system and variation in number and packet size is error. 3) to verify clock errors, as time stamps are placed by the system clock on user services records, packet transmission and position checking, if the time stamps are not in proper order then there is problem with on-board system clock [11].
To verify system results reasonable tests were used. These tests were used to verify that sensor data is within nominal range, satellite orbit is correct, acceptable commands are being executed. Violations in all these are errors and rectified accordingly [11].
Fourth types of acceptance test used for PANSAT on-bard software fault tolerance design were computer run-time tests. These are basically exception handling against programming/ logic errors like division by zero, stack over flow, and stack under flow and save the system from crashing during normal satellite operations [11].

The fault tolerance strategies adopted for PANSAT on-board software is comprehensive and cover almost every types of errors/ anomalies that could arise due to space environment and on-board software design and data problems.
The main drawback seen in this strategy in our analysis is; it requires most of the time ground operator's intervention for on-board software anomaly rectification.

4.2. STEP WISE NEGOTIATING VOTING (SNV) FOR HITEN:

This on-board software fault tolerance technique was used in the engineering satellite "Hiten". Two main objectives of the satellite were on-board software fault tolerance mechanism verification against space environment faults and high-efficient satellite telemetry data transfer, utilizing packet format. The commercial grade hardware (micro processor HD68HC000, 64 Kb memory having error correcting codes) were used for the OBC of HITEN satellite.
In SNV the probability of fault free operations of each redundant variant is calculated and the outputs of the modules having highest reliability are selected as final output of the system. This reliability of a module is determined by its capability/efficiency of handling data during data comparison and self-checking tests. The data and self-checks are independent. Following figure describe the SNV mechanism [12].

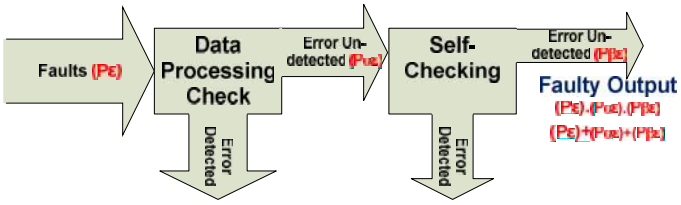

Figure 11: SNV Reliability determination mechanism [12]

In HITEN satellite the SNV on-board fault tolerance design, the variants are declared as cells. The OBC contains three cells A,B,C. Following are the main steps of SNV algorithm.

- The results of data processing variants (cells) and self checking variants (cells) are mutually exchanged using a proper inter-cell communication mechanism. The exchanged data is passed through adjucator for comparison (acceptance tests).
- Reliability of data processing in each cell is calculated based on self-checking results and adjucator (mutual exchanged) results as follows

Sr. No.	Self Check (Pβε)	Data Compare (Pαε)	Reliability (Rd)
1	Good	Agree	1- Pε.Pβε.Pαε
2	Error Detected	Agree	1 - Pε.Pαε
3	Good	Disagree	1 - Pε.Pβε
4	Error Detected	Disagree	----------

Table1: SNV Reliability determination [12]

- Data from the cell having highest reliability is selected as final output of the system
- For more safety/ reliability these reliability calculations are done separately in each cell also, and both the selected system data and this calculation are further compared in modified Voter Hardware (MV) for avoiding of selecting wrong data.
- All the above mentioned process is explained in the following figure [12].

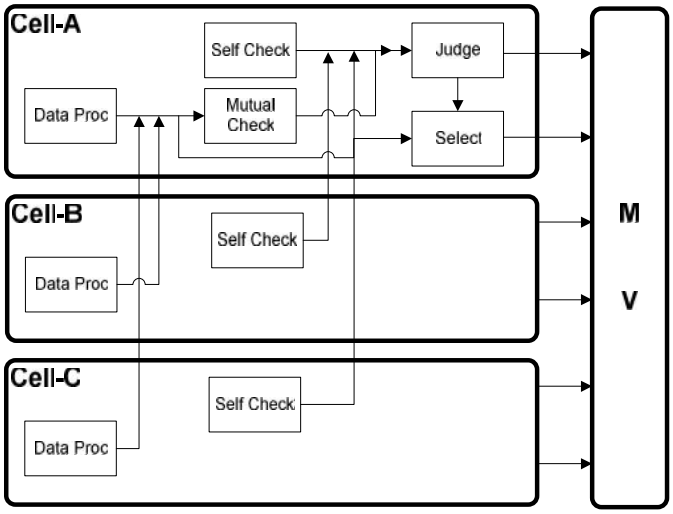

Figure 12: SNV On-board Software Fault Tolerance Mechanism [12]

The experiment shows the NSV on-board fault tolerance mechanism worked successfully and it detected and corrected 1-bit errors (all SEUs) occurred in various devices of OBC. This technique is derived from traditional NVP fault tolerance mechanism. It follows on-board software variants on single hardware and provide autonomous successfully

operations without interference from ground operators. We can conclude that this technique is comparatively better that the PANSAT on-board fault tolerance technique.

4.3. APPLICATION LEVEL FAULT TOLERANCE AND DETECTION (ALFTD) FOR OTIS:

This on-board fault tolerance technique was used in Orbital Thermal Imaging Spectrometer (OTIS) mission of NASA. This ALFTD is an enhanced/modified version of Application Level Fault Tolerance Technique (ALFT); which itself is a scale down version of algorithm based fault tolerance technique. ALFT technique is implemented by developing two procedures/ functions for various/selected important on-board application tasks. The primary routine for the task contain full functionality. The secondary routine is a scale down version of primary routine. It is implemented through different algorithm/ technique; however it provides acceptable results/ output in-case of primary routine fails to perform its function. The design scheme of ALFT is described in following figure [13].

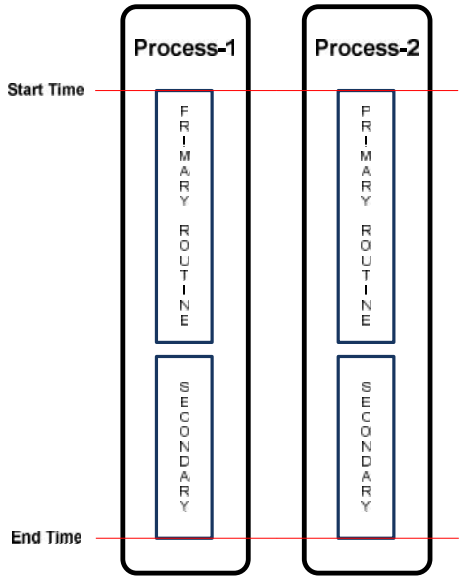

Figure 13: ALFT Software Fault Tolerance Mechanism [13]

As described above, the secondary variant/routine is a reduced version of primary function. It focuses only on important data and estimation techniques are applied to produce near accurate acceptable results. The main problems with this technique are a) many data errors remain undetected as implementation of all possible data checks is not possible b) the secondary routines have to execute continuously/rigorously for effective fault tolerance [13].

The ALFTD was designed to overcome the shortcoming of ALFT. The secondary routines are executed only after the error/fault in the data/ primary routine has been established. A well defined/reliable fault detection mechanism is designed and implemented to detect errors in the data and system. The efficiency of the ALFTD depends on the frequency of faults detection; less fault detection will definitely results in less execution of secondary variants. Therefore the fault detection mechanism incorporated in the scheme give more attention in terms of its accuracy. To make this mechanism more reliable filtering-bands concepts is introduced. If the output of primary procedure is not with in nominal ranges, the secondary procedure is used to perform desired calculations and if the calculation results are acceptable, only then they are used by the system otherwise both routines are considered faulty and relevant alarms are generated. ALFTD was designed and implemented for catering two types of problems 1) the crash of main process 2) fault in one data pixel and its propagation over the network. The probability of each pixel being faulty is Pf and the errors due to this faulty pixel are unknown [13].

The ALFTD finds the erroneous pixel and transfer it to secondary function. For this purpose two filters natural bound and special locality were defined in this fault tolerance scheme to handle missing of fault detection and generation of false problems. Handling of both these types of problems is very important for achieving good results. Both these filters were applied standalone and combined to determine the probability Pmiss and Pf.a. The combined application of filters gives good results [13].

Although this scheme by using two types of above mentioned filters simultaneously give good results, however the computational overhead increased many times. To choose a scanning range for these filters is also a challenging task, choosing a very small range will give more accuracy in catching errors, at the cost of more alarms and it will consume more time to handle those alarms by using secondary variants. Choosing a large scanning range is vice versa and also consumes more time in similar way. If a suitable range (between too large and small) is chosen intelligently very good results in terms of high probability of catching faults and less false alarms can be achieved with reasonable computational overhead (cost).

Following table summarizes the three above mentioned satellites fault tolerance techniques

S.No.	Satellite	On-board SFT Scheme	OBC HW	On-board SFT Features	Remarks/Issues
1	PANSAT	1) Watch-dog system 2) Acceptance Tests 3) Hamming Code	1) Two space-rated Intel 80186 central processing unit. 2) One 512 Kb volatile working RAM having hamming code 3) Two 512 Kb non-volatile RAMS without hamming code. 4) Two 04 MB memory banks	A satisfactory approach for catring a variety of system, program and data errors that occure in the on-board system due to space environment.	1) Not autonomous, moslty require ground operators intervention for errors rectifications 2) Does not provide 100% accuracy
2	HITEN	1) Stepwise Negotiating Voting (SNV) 2) Hamming Code	1) MPU HD68HC000 2) One 64 Kb RAM having hamming code	A reliable approach for rectifying soft errors (SEUs) occurred in various devices/hardware of OBC due to space environment.	1) Autonomous scheme, OBC automatically handles some of the issues. 2) Ground operators intervention will be required incase of sever environment 3) Does not provide 100% accuracy
	OTIS	1) Application level fault tolerance detection (ALFTD)	Not Known	A suitable approach for low cost satellite missions. It is based on primary and secondary (low scaled) varients for handling various types of errors occured in space environment.	1) Although this scheme by using two types of above mentioned filters simultaneously give good results, however the computational overhead increased many times. 2) To choose a scanning range for these filters is also a challenging task

Table2: On-board software fault tolerance Techniques

5. FUTURE TRENDS OF FAULT TOLERANCE TECHNIQUES FOR SPACE APPLICATION

5.1. PARTIAL SOFTWARE PROTECTION

This software fault tolerance technique is useful for static codes. In this technique only important code is protected by writing redundant code and the code which is less important and its probability of execution is very low are left as it. This approach reduces the size of code by avoiding the unnecessarily writing of duplication code and increases the reliability and performance of the space applications. The following figure shows the implementation of this fault tolerance scheme.

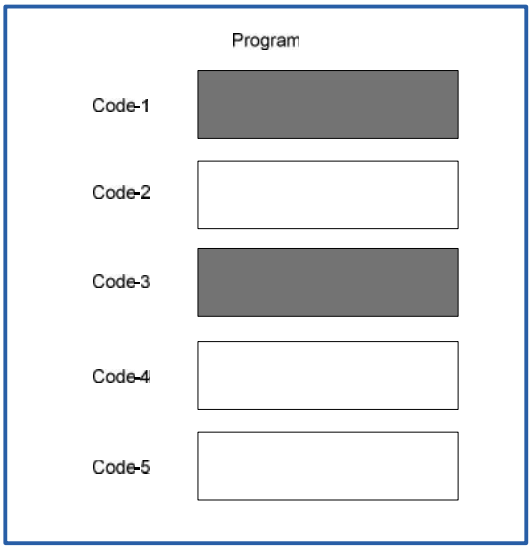

Figure 14: Partial software protection approach [14]

The dark code represents it is made fault tolerant and the clear code is it (not made fault tolerance). To make a code fault tolerant the decision is made by developing control flow chat and also by selecting a probability (p) for its execution. If the execution probability is less than p, the code is left unprotected and if it is greater than p it is made fault tolerant by generating redundant lines of code [14].

The performance evaluation of this technique shows that it give better results in terms of enhanced reliability of the application against various types of errors.

5.2. Model Driven Software Development (MDSD)

Another approach for developing fault tolerance applications is MDSD. In MDSD fault tolerance framework N-versions modeling (NVM) techniques and specifically developed design patterns are utilized to make applications fault tolerant. This technique is similar to V-version programming (NVP); however its reliability will definitely be high if the developer succeeded to create independent models. Greater independence means greater reliability and fault tolerance. Following are the different NVM FT frameworks [15]

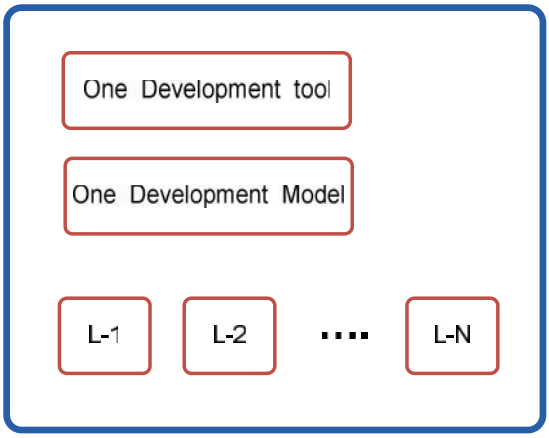

Figure 15: NVM FT Framework-1 [15]

The above figure shows the NVM FT framework developed using 1-tool, 1-development model and many languages.

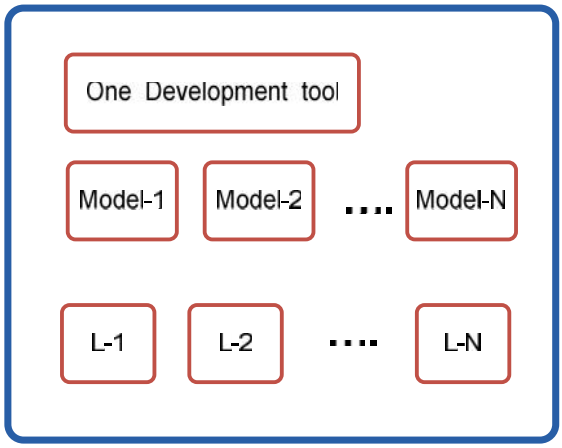

Figure 16: NVM FT Framework-2 [15]

The second variant of NVM, 1- development tool, many development models, and many development languages.

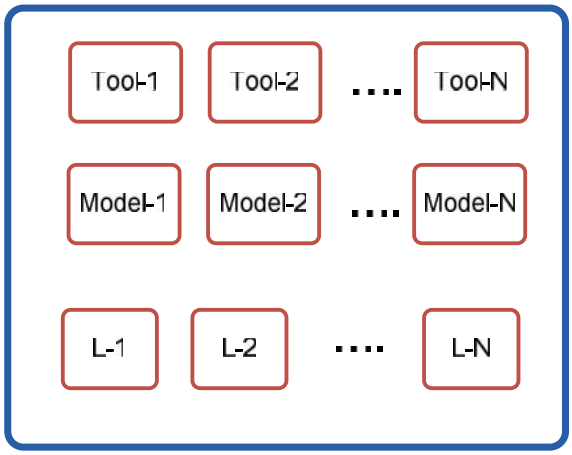

Figure 17: NVM FT Framework-3 [15]

The third variant of NVM, many development tool, many development models, and many development languages.

This technique seems very useful, however it is still in experimental stages and will require a lot of time achieve certain maturity level. The questions regarding its development cost and levels of independence at various levels and in different variants achieved for reliability are still to be answered.

The other potential candidate approaches that can be used for on-board software fault tolerance of satellite are “FTOS: Model-Driven Development of Fault-Tolerant Automation Systems ” [17], “SES-based Framework for Fault-tolerant Systems [18]”, “System Level Hardening by Computing with Matrices [19]”

6. **Conclusion**. Life of space applications (satellites) varies from few days to many years (up to 15 years). Therefore, depending upon satellite mission requirements and recourse availability, a proper comprehensive, efficient, high performance software fault tolerance scheme can be chosen /designed. It is also essential because once the mission is launched there are minimal chances of correction or maintainance. In case of OBC software error, only proper fault tolerance technique can help to avoid failure. To make the low cost satellite missions successful by using less expensive, commercial grade OBC hardware, a comprehensive on-board software fault tolerance technique must be designed and implemented to handle various types of errors caused by space environment. Hybrid on-board satellite software fault tolerance approaches (combination of H/W and S/W and combination of various S/W fault tolerance schemes) will produce better results.

.

REFERENCES

[1] Peter Popov, Lorenzo Strigini, (2010). Assessing asymmetric fault-tolerant software. *In Proceedings of the IEEE-Twenty-First International Symposium on software reliability engineering.*

[2] Domenico Cotroneo, Antonio Pecchia, Roberto Pietrantuono, Stefano Russo, (2011). Architecture-Based Criticality Assessment of Software Systems. *In Proceedings of the Latin-American Symposium on Dependable Computing.*

[3] Y. Bentoutou, (2011). Performance Comparison of Real Time EDAC Systems for Applications On-Board Small Satellites. *World Academy of Science, Engineering and Technology.*

[4] Jaynarayan H. Lala, Linda S . Alger, (2004). Hardware and Software Fault Tolerance: A Unified Architectural Approach. *In Proceedings of the IEEE-18th International Symposium on software reliability engineering.*

[5] L. Anghel, R. Leveugle, P. Vanhauwaert, (2005). Evaluation of SET and SEU effects at multiple abstraction levels. *In proceeding of the 11th IEEE IOLTS Symposium.*

[6] Y. Bentoutou and M. Djaifri, (2008). Observations of Single-Event Upsetsand Multiple-Bit Upsets in Random Access Memories On-Board the Algerian Satellite. *In procedi ngs of the IEEE Nuclear Science Symposium.*

[7] Laura L, Pullum, (2001). Software Fault tolerance techniques and Implementation, ARTECH HOUSE INC. ch-1.

[8] Kanoun, K. and L. Spainhower, eds. (2008). Dependability Benchmarking for Computer Systems., Wiley - IEEE Compute Society Press Hoboken, New Jersey.

[9] Soren Abildsten Bogh, Mogens Blanke (2004). Fault Tolerant Control "A case study of the OSTED satellite".*In Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing.*

[10] Zaipeng Xie, Hongyu Sun, & Kewal Saluja (2005). A Survey Of Software Fault Tolerance Techniques.

[11] G. Ken Hunter and Neil C. Rowe (2000). Software Design For a Fault-Tolerant Communications Satellite.

[12] Tadashi Takano, Takahiro Yamada, Tohshiroh Shutoh, Nobuyasu Kanekawa (2004). Fault-Tolerance Experiments of the "Hiten" Onboard Space Computer. *In Proceedings of the IEEE-Twenty-First International Symposium.*

[13] E. Ciocca, I. Koren, C.M. Krishna, D.S. Katz (2004). Application-Level Fault Tolerance in the Orbital Thermal Imaging Spectrometer . .*In Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing.*

[14] Lei Xiong, Qingping Tan, (2011). A Configurable Approach to Tolerate Soft Errors via Partial Software Protection. *In proceedings of the Ninth IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops.*

[15] Craig A. Lewis, Ronald W. Smith, Alain Beaulieu (2011). A Model Driven Framework for N-Version Programming.

[16] Lei Xiong, Qingping Tan (2011). An Approach to Analyze Effects of Soft Errors from Dynamic Software. *In proceedings of IEEE International Conference on High Performance Computing and Communications.*

[17] Christian Buckl, Dominik Sojer, Alois Knoll (2010). FTOS: Model-Driven Development of Fault-Tolerant Automation Systems.

[18] Michael Steindl, Juergen Mottok and Hans Meier (2010). SES-based Framework for Fault-tolerant Systems.

[19] Ronaldo R. Ferreira, A´ lvaro F. Moreira, Luigi Carro (2010). System Level Hardening by Computing with Matrices. *In proceedings of 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools.*