Chapter 21
Design for Zero-Maintenance

M. Farnsworth, R. McWilliam, S. Khan, C. Bell and A. Tiwari

Abstract This chapter looks at the concept of zero-maintenance, in particular how it relates to design. It begins by defining what constitutes zero-maintenance, presenting current research on the themes of autonomous maintenance and self-healing and repair. A wider context of how zero-maintenance affects through-life engineering services is also discussed with a focus on the no-fault found phenomenon. Case studies are then presented for design strategies in self-healing electronics and no-fault found and the failure of design. Finally, a design for zero-maintenance process is outlined and discussed.

21.1 Introduction

Throughout the life time of products and large assets there is a persistent need for numerous interventions in the form of targeted servicing in order to maintain operation efficiency. Whether predictable by nature or triggered by unexpected events, service support occurs at all stages of product lifetime—from initial conception, design and manufacture to through-life operation and ultimately end-of-life. Integral to many of these stages are processes intended to maintain the function of the product with minimal cost overheads. These processes, whether integrated into the product or else applied in the form of planned or unplanned maintenance, are essential to maintaining sustained performance specifications from the product in question. Conception and planning for maintenance-centric processes begins at the design stage, but is all too often neglected in favour of more typical cost and performance factors. However, new design considerations based around mitigating against costs associated with product failure, maintenance and repair are emerging that need to be brought into the fold of future product design. Such considerations inevitably tie into both the cost and performance of products at some point along their life-cycle chain.

M. Farnsworth (✉) · R. McWilliam · S. Khan · C. Bell · A. Tiwari
Cranfield University, Cranfield, UK
e-mail: m.j.farnsworth@cranfield.ac.uk

L. Redding et al. (eds.), *Advances in Through-life Engineering Services*,
Decision Engineering, DOI 10.1007/978-3-319-49938-3_21

This chapter investigates the philosophy of zero-maintenance i.e., the elimination of maintenance-centric costs, and its relation to through-life engineering services for a range of high-value products and assets. The focus is centered on design, and how information gathered from the through-life services chain can be used to better inform design decisions, so as to ultimately mitigate or remove the need for maintenance whilst continuing to address the traditional goals of reducing operational costs and improving performance.

21.2 Zero-Maintenance

The availability and performance of high-value assets is the key driver of through-life engineering services (TES). Without these two factors manufacturers that undertake such contracts will quickly lose revenue and credibility. Loss in availability or drops in performance are highly correlated to system failure, either as a result of direct damage or through degradation due to environmental factors or operational use. Maintenance, either planned or unplanned is a service undertaken to reduce such system failures or return them to original design specifications. An alternative to the undertaking of maintenance is to design in capabilities for self-healing and repair within the asset itself. These capabilities either provide much needed resilience to damage, thus blunting the impact of any unforeseen event that may have otherwise lead to damage, provide processes for repairing damage autonomously or simply allow damaged functions to be replaced.

Together both standard maintenance and self-healing strategies form a powerful approach to maintaining availability and performance. The philosophy of zero-maintenance encompasses these two strategies to ultimately provide the elimination of maintenance-centric costs, and its relation to through-life engineering services for a range of high-value products and assets. The following sections discuss three main topics surrounding zero-maintenance. The first focuses on the use of autonomy and autonomous systems to perform maintenance and remove human intervention. The second on the strategy of self-healing and repair, and finally a broader look at how a zero-maintenance effects wider through-life engineering services and the phenomena of no-fault-found (NFF).

21.2.1 Autonomous Systems and Embedded Intelligence

The reliability of a mechanical system depends on its design, the quality of its components, and its maintenance. However, maintenance is typically considered during product design in an ad hoc, ineffective manner, leading to unnecessary life-cycle costs [1].

Maintenance falls into three categories, corrective, preventative and condition-based [2]. Equipment or large assets are often continually monitored with regards to performance and when signals indicate the need for some form of maintenance, action is taken. Preventative maintenance is based upon the mean-time-to-failure (MTTF) value and maintenance is scheduled to occur when required (time-based or usage-based). Occasionally failure can occur outside these bounds and corrective action is taken to repair or maintain the asset.

The traditional image of maintenance is one consisting of Dull/Dangerous or Dirty work. There is also a similarly negative connotation that maintenance is a non-productive element to manufacturing that is only now changing. Maintenance is known to have four characteristics that make undertaking it challenging. Maintenance for a start is often non-uniform because one maintenance sub-operation (removing bolt) may differ depending on the age of the asset (i.e rusted) that makes any pre-programmed autonomous behavior difficult and requires human operators to take uncertain actions outside of their possible knowledge. Second it can be non-standardised, with different actions required depending on the type of asset present, regardless of maintenance sub-operation. Therefore, removing and fitting one type of disk brake pad may vary depending on the maker of the asset (i.e., train undercarriage). Thirdly, maintenance is often irregular, with varying levels of planned and unplanned maintenance which makes determining the right operators or autonomous systems are in place at a given time difficult, particularly if such autonomous maintenance systems can only perform a specific function (inspection, manipulation) and are unable to multitask, something a human is often able to do to meet an increasing workload though only if trained correctly. Finally, maintenance is non-deterministic, because one maintenance sub-operation can change the state of the target system and invoke a different sub-operation from the one that was expected [3].

In the current climate there is a move towards introducing automation and intelligence into a number of products or systems. This is understandable given the benefits self-automation brings, be it the removal of humans from a particular scenario leading to reduced costs or simply their removal from dangerous situations or environments. For example in recent decades there has been an increasing use of robots within manufacturing processes. The speed, power, availability, productivity, and improved accuracy of robots have had significant impact in reducing manufacturing costs while improving production quality. A design for maintenance philosophy that looks to improve the overall maintenance process either through design of the asset itself or to aid in the design of external factors that may contribute to the process in some way, i.e autonomous system solutions is needed [4]. These can incorporate not only characteristics that aid maintenance, particularly with regards to automation, but also exhibit robustness and self-healing attributes.

A large number of maintenance tasks for example simply involve some level of inspection and analysis and if required some form of action to be taken. When considering autonomous maintenance the first step is to think on how this degree of inspection and analysis can either be removed or done in situ within the asset itself. The integration of sensors, for example and other forms of embedded intelligence can provide information to human operators without any form of interaction. Equally they can be used as a tool to aid automation of maintenance by guiding autonomous solutions or provide instruction on any steps that may be needed to complete a particular maintenance task.

21.2.2 Self-healing and Self-repair

To be successful self-healing techniques must be designed to compensate for a wide-variety of failure modes, thus overcoming some of the problems associated with uncertainty. Although specific solutions are not suggested, proposed strategies for developing a self-healing system should not focus on a finite number of underlying causes. Instead the focus should be on how these causes manifest, how they can be detected and ultimately how they can be corrected autonomously. To achieve a self-repairing system, it is clear that the system must have an element of self-awareness. Amor-Segan et al. [5] state that the ultimate aim is to develop a system with "the ability to autonomously predict or detect and diagnose failure conditions, confirm any given diagnosis, and perform appropriate corrective intervention(s)".

A general approach to self-healing applicable to most systems consists of a number of specific steps, 'Cause of Fault', 'Detection of Fault', 'Diagnosis' and 'Corrective Action' [6]. Typically the 0th step, 'Cause of Fault', begins the process, however in the scheme of things the cause is irrelevant in so much as the system should be designed that any behavioral or functional changes are compensated for through further actions. What follows next is the importance of the system to identify the fault, often spatially, with regards to a specific component or region of interest within the system. The 'Detection of Fault' step requires the ability for the system to infer through changes in behavior, function or interpretation of internal or external telemetric data the source of error. The degree of granularity will depend on the amount, accuracy and detail of information retrieved and the systems ability to uses this to come to a confident conclusion of the source fault. Next comes the ability to further analyse the system, and gathered information to bring about a 'Diagnosis'. This is important when looking to initiate some form of corrective action and therefore first be confirmed, to avoid undesirable events such as 'good' components being unnecessarily removed or routed around. Finally 'Corrective Action' will have to be taken to bring back the system behavior back to original specifications, or at the very least mitigate any effects damage may have produced [7–9].

The primary approach to self-repair in electronics involves furnishing the associated circuitry with redundant resources that either mask fault conditions or else remove fault logic, leaving only healthy logic. Redundancy may take three primary forms of spatial, temporal and information redundancy. Spatial redundancy involves the addition of spare physical resources that consume physical space. As depicted in Fig. 21.1, this typically involves copying common resources and connecting them via voting logic. The voter arbitrates between the common resources

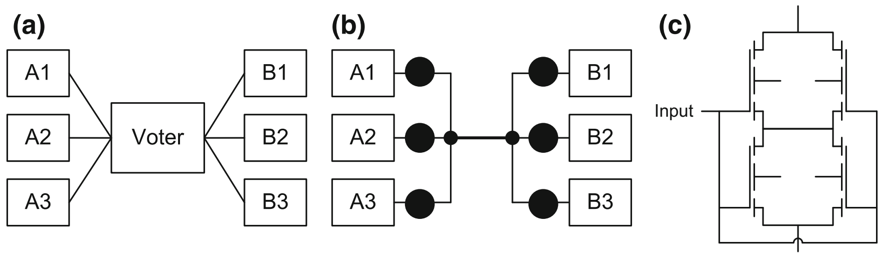

Fig. 21.1 Concept of majority signals in spatial redundancy. **a** Discrete voter. **b** Abstract signal arbitration. **c** Example of signal arbitration within electronic transistor circuit

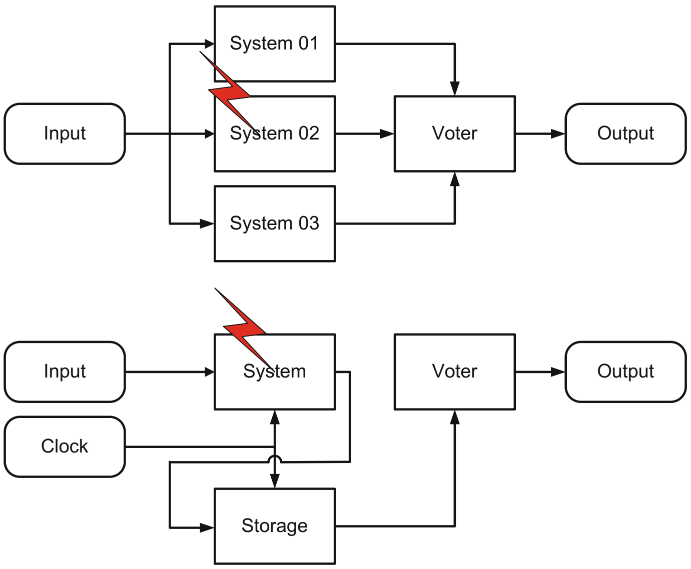

Fig. 21.2 Fault masking within **a** spatial redundancy and **b** temporal redundancy blocks. A temporary fault occurring within either design (represented by the lightning bolt) will not cause output error. However, a persistent fault will compromise the temporal redundancy approach

in order to form a majority output or condition that is passed onward to other circuitry. Thus, fault events may be masked so that their presence does not cause errors within the macro behavior.

Temporal redundancy involves the repeated utilization of resources in time in order to form a majority signal. This is depicted in Fig. 21.2, where the spatially redundant circuit is reformed as a temporally redundant circuit, the primary trade-offs being a reduction in performance and susceptibility to persistent fault conditions.

An example of spatial redundancy applied selectively to a digital NAND gate (a fundamental building block in digital electronics) is depicted in Fig. 21.3. In this

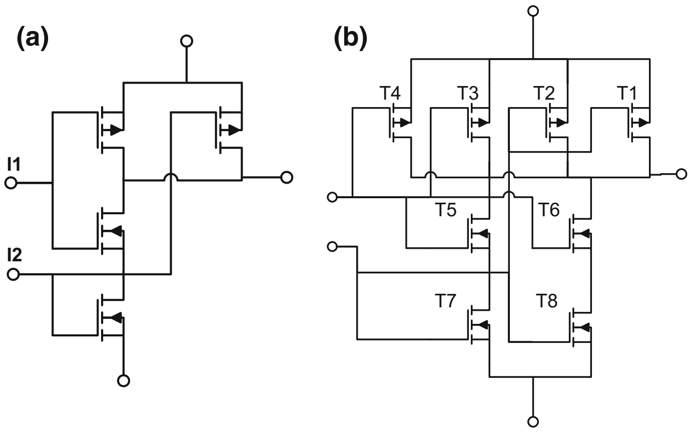

Fig. 21.3 Spatial redundancy strategy targeting ASIC circuitry. **a** Standard NAND gate schematic. **b** Alternative NAND gate schematic supporting selective fault masking and active fault repair trigger

case, the redundancy scheme is chosen to achieve fault masking of specific fault conditions (stuck off faults) while creating a current trigger in the event of stuck on faults [10, 11]. The move towards utilizing redundant resources for self-repair is a complex one due to the need for active detect and repair. However, there are further examples that operate at the transistor level [12].

Besides current off-the-shelf ASICs, future nanoscale devices are fast approaching and will require fresh approaches to fault tolerance and repair. They will be able to support new levels of spatial density but there is debate over the best strategies for achieving resilience [13]. In comparison to contemporary digital electronics design, cell-based designs include bio-inspired evolvable hardware [14] that mimics many properties of bio-cellular organisms including simple inter-cellular actions (nearest neighbor communications), cell states governed by DNA instructions (look up tables) and cellular homogeneity (massive arrays of identical cells). Though this, cellular electronics gains the desirable characteristics of self-organization and adaptation including the property of dependability [15]. The resulting hardware, governing rule sets and software configuration layers are however complex in comparison to standard ASIC design, although there are examples where resource and rule sets have been minimized where possible. Two examples of this are the Plastic Cell Architecture [16] and convergent cellular automata [17], both of which attempt to simplify cell complexity at the expense of repair capability.

21.2.3 Through-Life Engineering Services and No-Fault Found

All systems are susceptible to faults and failures during service, but those faults which occur and result in the No Fault Found (NFF) phenomenon are often unanticipated during the design, sometimes because they occur within acceptable operating tolerances. NFF can be described as a fault whose root cause cannot be found. Therefore there are subsequent inherent diagnostic difficulties. A number of mitigation strategies to combat NFF have been elaborated in literature [18–20], but most are purely that, mitigation for when a potential NFF related failure occurs—such as incorporating specialized test equipment. In truth though, the most significant solution of NFF eradication would be to design and manufacture systems that are increasingly immune to those unanticipated failures that result in NFF. This requires enhanced systems understanding, improvements to the actual design integrity and most importantly a robust mechanism for translating in-service failure (and NFF) knowledge directly back into the design process.

21.2.3.1 In-Service Feedback for TES

NFF events encompass a whole range of products in service, many of which are made up of legacy systems with well-defined operational support practices. Disregarding the fact that the root cause of a NFF will begin with the failure of a component (or unpredictable intermittent faults, which may be part of an inherent design flaw), the end result is a diagnostic failure—that is the maintenance services procedures, equipment, testing capability and guidelines for that equipment was inadequate to isolate the problem. To reduce the NFF event rate for in-service equipment, the conditions under which NFF problems occur need to be considered in depth and investigations should focus on the following areas:

- Failure Knowledge Bases, novel FMECA tools and troubleshooting guides specific for NFF to improve diagnostic success rates.
- Research to pinpoint where in the maintenance process is NFF occurring, for example at a particular maintenance line, testing station, or under specific testing equipment.
- Development of assessment tools to assess maintenance capability/effectiveness which may include:
- Introduction of integrity testing as complimentary to standard ATE (functional) testing procedures.

Although when we talk about influencing (or modifying) a complex system's design for the specific purpose of reducing the impact of NFF, designers need to be clear on what actually needs to be changed in the design. For example, does the system need to be completely redesigned with the aim of increased robustness to make it more fault tolerant? This may have a negative impact on material choices or weight/size restrictions. Or should they change the design so that any faults manifested on the system are accurately detected and located with confidence?

It is the latter of these two questions that would focus on design for testability (DFT). DFT is not a new area of research and it is well accepted that DFT could substantially reduce the effort and cost of testing and supporting a product. However, what is not being done is quantifying the burden of NFF, identifying the root causes, understanding the impact of NFF related faults on coupled systems and feeding this back into the design process with mitigation of NFF becoming a serious DFT goal. The key challenges to address here are:

- Development of design guidelines and standards to improve system designs which incorporate the reduction of NFF as a design goal for improved testability.
- Research into the relationships between system design characteristics and NFF related attributes such as rate of false alarms, fraction of faults isolated to improve design for testability.
- Modelling of complex interactions between system/subsystem/components and their physics of failure.
- Modelling of intermittent failures (identified as the number 1 root cause of NFF) from a fundamental perspective including standardised testing equipment and procedures.
- Developments of a NFF burden/rate predictor for new designs or NFF trending process for in-service systems.
- NFF specific maintenance cost models for design justification.
- In-service monitoring and feedback into design and manufacture.

The fact is that technicians around the world are discovering novel causes of failures on a daily basis. These are being labelled as novel because they have never before been experienced or observed for that particular system, and certainly were not anticipated during the design phase. The reason many failures are not predicted during the design stage is because the system is expected to be operating within a specific set of operational and environmental envelopes a breach of which would signal a predictable failure. However, these novel failures, occur in-service and within the designed operational tolerances making them unpredictable and difficult to diagnose and usually resulting in NFF events.

21.3 Case Studies

21.3.1 Self-healing Electronics

Most electronics self-repair strategies have focused on application specific integrated circuit (ASIC) design that includes the popular field programmable gate arrays (FPGAs). The general goal is to combine fault masking and active mitigation strategies on flexible programmable platforms that support online and offline hardware reconfiguration [21]. However, conventional software-configurable processors must also operate reliably in the presence of faults and their fixed-architecture is therefore enhanced using fault masking strategies involving spatial and information redundancy [22].

Perhaps the most tried and tested reconfiguration strategies involve radiation hardening for space applications. This is due to increased understanding of single event upsets (SEUs) that occur within the Van Allen belt as well as the long-term survivability of deep space and inter-planetary missions. Traditional radiation hardening methods depend upon radiation absorbing materials that add weight and cost. Instead, self-repair through functional redundancy has become desirable, especially when strategies are supported by off the shelf electronics. Prediction and quantification of survivability is a current challenge involving complex reliability modelling to understand the efficacy of redundant resources [23] as well practical experimental validation platforms [24].

A particular example that utilizes the reconfigurability of FPGAs is the Self-testing ARea (STAR) strategy [25] that relies upon creating localized, dynamically changing areas within the active FPGA fabric that quarantine the logic within and perform detailed self-test and repair algorithms. This proceeds while the rest of the FPGA fabric is functioning as normal. The principle is depicted in Fig. 21.4 although this strategy approaches the ideal of complete online self-test

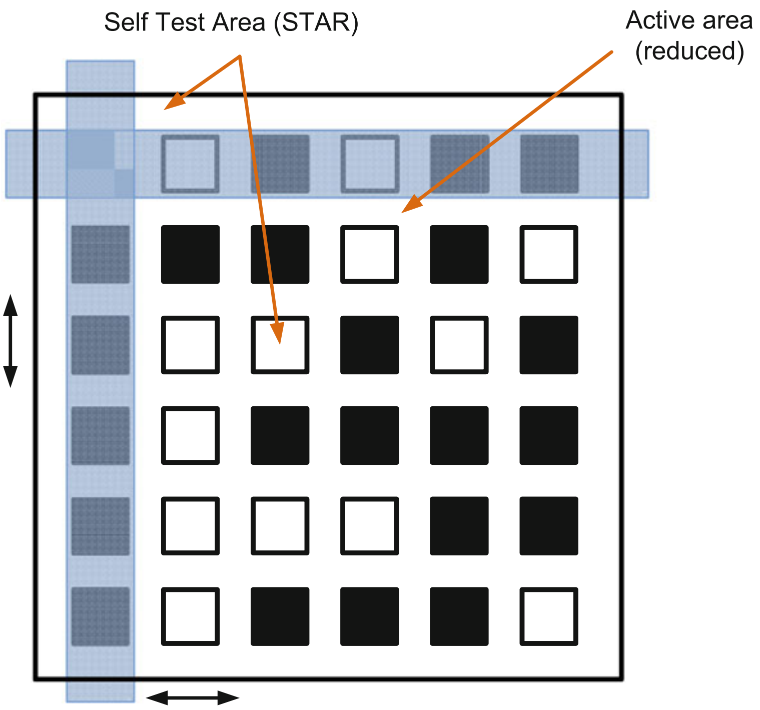

Fig. 21.4 Depiction of the STAR self-repair method for FPGAs

and repair within electronics, there are a number of tradeoffs that typify the challenges of zero-maintenance in general:

- The STAR overhead is considerable and includes the roving configuration logic, test and reconfiguration logic and coordination logic that typically takes the form of an external processor.
- Fault response time is dependent upon the time taken to perform a full test and repair sweep. In [26], Parris estimates the typical sweep time to be the order of 1 s for a modest FPGA device. However, the estimation also predicts sweep times in excess of 17 s for larder state of the art FPGAs. This may represent an unacceptable latency period for some applications.
- Overall throughput of the device is reduced due to the spatial redundancy overhead. In the example given in [25], this amounted to 16% reduction that would impact overall application performance. Given that zero-maintenance extends to applications where performance versus cost is crucial (e.g., the automotive industry), performance throughput represents an additional design overhead that must be factored.

A further complexity is the need for verification of fault repair circuitry of any self-repair strategy. This presents unique challenges, since the current state of the circuit depends on the history of fault events and thus becomes difficult to predict. To help ascertain valid behavior, deterministic fault injection and testing may be performed [27] to quantify the circuit response to specific fault conditions and complemented by probabilistic models where possible [23].

21.3.2 No-Fault-Found and the Failure of Design

One key area that is related to system's design and NFF that is often overlooked is how that system is utilized by the user. For example, is the user operating the system incorrectly because of inadequate training or has the system been designed without the end user in mind? This design issue is a major contributor to the NFF issue across multiple industries. To highlight the case we can consider the consumer electronics industry where the annual bill for NFF returns is in the billions of dollars.

The major impact area for operators due to no fault found device returns is that when customers buy a smartphone, it's increasingly likely that it is a replacement for an existing device. Their expectation is that the new device will perform at least as fast as their old device and also other devices owned by friends and family. If those expectations aren't met because that particular combination of device, application and OTT suffers from wasted data, the customer will return the device as "faulty". The operator then spends time and money testing the device without discovering any faults, and has to resell the device as "refurbished" at a lower margin. This increases their support costs and makes the device less profitable for them.

If a device is not functioning as expected, even with a user error fault, the reason for not obtaining the expected functionality must be identified. One of the underlying reasons for devices being returned as NFF is that the user often has the device configured incorrectly, has misunderstood the device's capabilities or functionality, or there is an underlying hardware/software design problem that is having a secondary 'fault' affect—but that design root cause is not obvious. The user will be required to often contact a service representative who will talk them through a troubleshooting process; however, this can create a frustrated customer when the help and advice provided is not solving the issue. The service representative is not aware of how the device is currently being used and in what conditions, or if the customer is providing the correct information—although this could be rectified through real-time analysis of available data. The cost of service representatives could also be significantly reduced if this real-time analysis for use in device troubleshooting could be delivered directly to the device, allowing the user to identify the nature of the problem.

There is no doubt that the very nature of the NFF problem has its roots firmly embedded within inadequacies present in the design process. This is in no way laying the NFF blame on designers, they will design to the required specifications which almost certainly will not include any reference to NFF—in fact most may not even be aware of any in-service issues that could enhance their designs. In order to eradicate NFF at the design stage this needs to change. Designers need to be provided with information and knowledge captured in the field and more emphasis placed on improved predictability of system usage and operating environments in order to reduce the probability of unanticipated faults occurring. This is no easy task and many of the challenges highlighted in this chapter still remain far from resolved. In addition to this, designers need to be turning their attentions to enhancing the testability of systems. This will ensure that access to test points is easy, appropriate test equipment for the task is identified and the overall test coverage of the system is enhanced. If you cannot test a system then you cannot diagnose it, and NFF will always prevail. Finally, the often overlooked contributor to NFF is how the interaction between the user and system is managed. Systems should always be designed with the user in mind with full training and support to avoid confusion and incorrect operation resulting in perceived faults that again lead to a category of NFF.

21.4 Design for Zero-Maintenance

Systems fail inevitably, in particular complex systems that work across long time scales and within harsh environments. There is no getting around this fact, and as a result mitigation strategies have to be put into place in order return the system to preferably an optimal state. These strategies cover a wide range of fields, services and technical solutions. The most typical comes in the form of maintenance, with processes and actions designed to return function, reduce further degradation and preemptively look to halt future failure. Design also plays a significant role, with a complex struggle in designing solutions that meet cost and performance requirements but also are resilient to damage and robust enough to perform to specifications over the lifetime of the system. The design choices made ultimately affect the performance of the system over its lifetime and how much maintenance is required to sustain this level of performance. The probability of failure of the system or components within the system is naturally also strongly correlated with their design.

Therefore, it is crucial that at the design stage all possible sources or events that lead to failure are understood for the targeted solution. It is also important to understand what the maintenance requirements will be for the targeted design so informed decisions can be made so as to improve its maintainability and resilience to failure. Figure 21.5 outlines a process wheel for the designing in of zero maintenance strategies. The challenge for the designer begins with an understanding of the initial problem or set of problems they wish to overcome.

The first step is to identify the requirements of the system to maintain its function over its life cycle. This is in relation to known faults and possible failure modes, and any processes for maintenance that are necessary for the system to maintain its function to original specifications. It is likely that future design choices further own in the design cycle will introduce additional requirements, however for now it can be ignored. Once the designer has available to him a list of requirements they can begin to reassess which of these requirements can be removed or avoided through a simple design change. In step 2 a designer can begin to prune away certain requirements. It may not be necessary to perform certain inspection practices

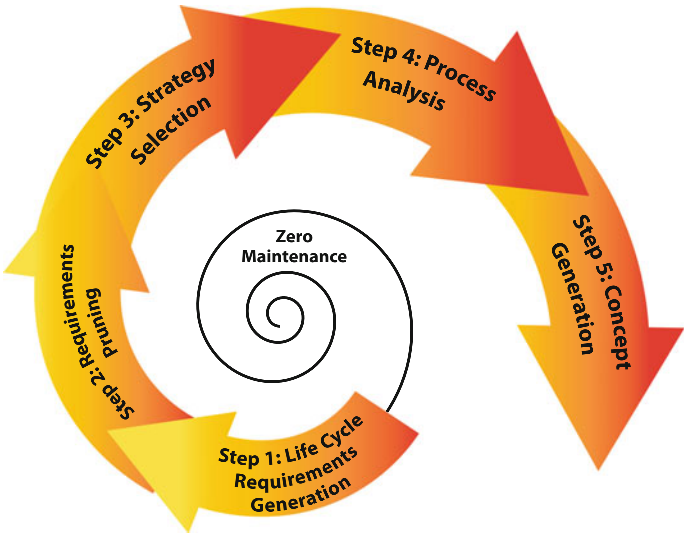
Zero Maintenance
Step 1: Life Cycle Requirements Generation
Step 2: Requirements Pruning
Step 3: Strategy Selection
Step 4: Process Analysis
Step 5: Concept Generation

Fig. 21.5 Design for zero-maintenance process wheel

for example if a sensor can replace the role of a human engineer. Careful consideration should occur during this step as it can greatly reduce the number of mitigation strategies integrated into the system. Step 3 forces the designer to make a decision on what direction they wish to take with regards to particular mitigation strategy, whether it is some form of self-healing or repair, or whether outside agents in the form of autonomous systems will provide a solution. Often this will be dictated by the particular failure mode or source of fault, and the component within the system it is related to. Once a strategy has been decided the next step is an analysis of the likely fault process or in the case of maintenance the steps required performing the specific task. Understanding the functional requirements necessary to perform a specific strategy can allow designers to identify alternative design choices to aid the fault recovery or maintenance process. The use of autonomous solutions for example mobile robots can be greatly aided if thought is given into how they may undertake certain maintenance or fault reduction tasks, for example inspection can be helped if diagnostic information is easy to access, or maintenance sites that do not require manipulation to open. Step 5 brings us to concept generation, where for each particular lifecycle requirement, the chosen strategy and functional requirements a designer can look to develop a number of solutions. Finally the designer has to look at the integration of all solutions into a single system and evaluate how such strategies may or may not affect each other. This may result in a different strategy or concept being pursued, or alternatively it may help identify shared functional requirements between mitigation strategies.

When looking to design in electronics self-repair capability its impact is felt not only in the electronic modules, but also the wider system (or sub-systems) given the pervasiveness of electronic and electrical circuits within modern engineering products. In the most extreme cases this brings survivability concepts into the fray across many design levels as shown in Fig. 21.6. Key metrics become fault capacity; latency and repair time become central in the role of electronic sub-systems to assist in securing continued operation in the presence of failures.

With respect to the above example, the performance metrics that need to be considered in design for self-repair are summarized in Table 21.1. Importantly, all such metrics must be considered at the various design stages and fully understood with respect to the strategies under consideration.

Not included in this particular design wheel is an important topic discussed previously, identifying design choices linked with through-life engineering services and their impact. During manufacturing the impact that process variations have on final product quality may cause unanticipated system failures and, in order to address the root causes of such service failures, it is necessary to develop analytical methods based on inter-loop modelling which integrates data from different phases of lifecycle with product and process models. Such a method exists [28] and is summarised below.

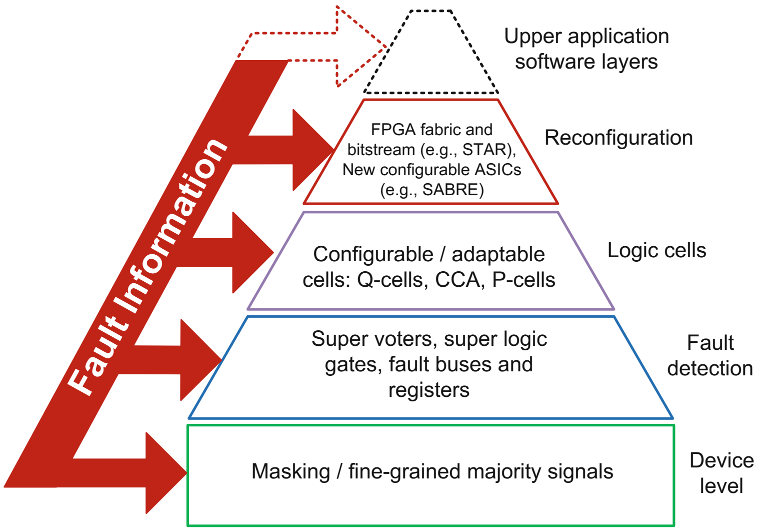
Upper application software layers
Reconfiguration
FPGA fabric and bitstream (e.g., STAR), New configurable ASICs (e.g., SABRE)
Logic cells
Configurable / adaptable cells: Q-cells, CCA, P-cells
Fault detection
Super voters, super logic gates, fault buses and registers
Device level
Masking / fine-grained majority signals
Fault Information

Fig. 21.6 Example hierarchical approach for self-repair focusing on electronic design

Table 21.1 Important performance metrics pertaining to self-repair capability in electronic systems

	Metric	Description
Response	Fault coverage	Refers to fractional area of overall circuit that protected and diversity of faults that can be handled
	Fault granularity	Minimum design layer at which faults can be detected/addressed
	Fault capacity	Number of remaining faults that can be sustained by mitigation strategy
	Performance reduction	Loss of application performance due to self-maintenance operations
	Latency	Timed required for recovery
Resources	Resource overhead	Number of additional components needed over and above basic design
	Resource re-use	Achieving efficient consumption of redundant resources (active methods)
	Energy usage	Energy consumed during recovery and consumed by additional resources overall
Diagnostic	Reporting	Discrimination and reporting of fault events at multiple system levels
	Remaining lifetime	Indication of remaining operational hours after which faults will no longer be handled
	Logging	Capacity to store a log of fault history and classification

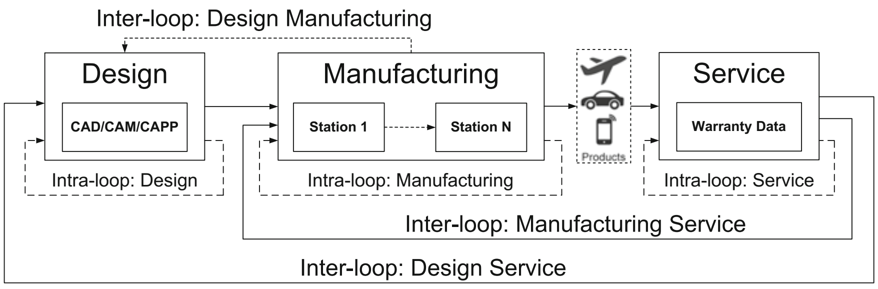
Inter-loop: Design Manufacturing
Design
CAD/CAM/CAPP
Intra-loop: Design
Manufacturing
Station 1
Station N
Intra-loop: Manufacturing
Products
Service
Warranty Data
Intra-loop: Service
Inter-loop: Manufacturing Service
Inter-loop: Design Service

Fig. 21.7 Framework for a closed loop lifecycle model [28]

The loops of a self-resilient production system are classified as intra-loop and inter-loop based on the availability of data from same or different phases of the product lifecycle respectively. Figure 21.7 shows the closed-loop framework. The intra loop refers to integration of data with product and process models from the same phase of the product lifecycle such as SPC that uses manufacturing data for monitoring purposes. The inter-loop refers to integration of data with product or process models obtained from more than one phase of the lifecycle such as addressing service failures.

Within the design phase, product simulation generates data on design parameters that satisfy a set of pre-defined functional requirements. Design changes and optimization is then achieved by modelling the relationship between critical design parameters, functional requirements and critical process variables. During the manufacturing phase, the intra-loop consists of continuous data on the design parameters and process variables obtained using in-line and/or off-line measurements of products and processes during production. The intra-loop in manufacturing is used to address out-of tolerance failures. The monitoring capability can be further integrated with process models to enhance the intra-loop capability of the production systems for fault diagnosis and adjustments. An intra-loop in service consists of warranty data and failure data which are analysed to send feedback to OEMs for setting economic warranty reimbursements to customers, estimating field reliability of products and changing design to address service failures. Warranty data is also used to improve performance of service centres by generating pre-alerting rules to diagnose product failures from customer complaints.

The Design-Manufacturing inter-loop integrates information from manufacturing with design to evaluate and improve diagnosability. By integrating manufacturing and service information together, design parameters can be defined to identify and isolate in-tolerance fault regions. In the Manufacturing-Service inter-loop, the Functional manufacturing and service information is integrated to identify and isolate in-tolerance fault regions. For the Design-Service inter-loop, there is the need for an analytical method capable of identifying root causes of service failures by integrating warranty failures with design models.

References

1. Hernandez G, Seepersad CC, Mistree F (2002) Designing for maintenance: a game theoretic approach. Eng Optim 34:561–577
2. Mobley RK (2002) An introduction to predictive maintenance. Butter-Heinemann, pp 2–5
3. Akrout H, Anson D, Bianchini G, Neveur A, Trinel C, Farnsworth M, Tomiyama T (2013) Maintenance task classification: towards automated robotic maintenance for industry. Procedia CIRP 11:367–372
4. Farnsworth M, Tomiyama T (2014) Capturing, classification and concept generation for automated maintenance tasks. CIRP Ann Manuf Technol. http://dx.doi.org/10.1016/j.cirp.2014.03.093
5. Amor-Segan ML, McMurran R, Dhadyalla G, Jones RP (2007) Towards the self-healing vehicle. Automotive electronics. In: 3rd institution of engineering and technology conference. IET, pp 1–7
6. Farnsworth M, Bell C, Tomiyama T, Khan S (2014) Autonomous maintenance for through-life engineering. In: Redding L, Roy R (eds) Through-life engineering services, decision engineering. doi:10.1007/978-3-319-12111-6_23
7. Farnsworth M, Tiwari A, Dorey R (2014) Modelling, simulation and optimisation of a piezoelectric energy harvester. Procedia CIRP 22:142–147
8. Farnsworth M, Tiwari A (2015) Modelling, simulation and analysis of a self-healing energy harvester. Procedia CIRP 38:271–276
9. Bell C, Farnsworth M, Knowles J, Tiwari A (2015) Self-repairing design process applied to a 4-bar linkage mechanism. Proc Inst Mech Eng Part B: J Eng Manuf
10. Schiefer P, *McWilliam R, *Purvis A (2014) Fault tolerant quadded logic cell structure with built-in adaptive time redundancy. Procedia CIRP 22:127–31
11. McWilliam R, Schiefer P, Purvis A (2015) Experimental validation of a resilient electronic logic design with autonomous fault discrimination/masking. Procedia CIRP
12. Koal T, Ulbricht M, Vierhaus HT (2013) Virtual TMR schemes combining fault tolerance and self repair. In: 2013 Euromicro conference on digital system design (DSD), pp 235–242
13. Han J, Leung E, Liu L, Lombardi F (2014) A fault-tolerant technique using quadded logic and quadded transistors. IEEE Trans Very Large Scale Integr VLSI Syst PP(99):1–1
14. Tyrrell AM (2016) Fault tolerant applications. In: Evolvable hardware [Internet]. Springer, Berlin [cited 2016 Apr 22], pp 191–207. (Natural Computing Series). http://link.springer.com/chapter/10.1007/978-3-662-44616-4_7
15. Tyrrell AM, Greensted AJ (2013) Evolving dependability. J Emerg Technol Comput Syst [Internet] [cited 2013 Feb 20] 3(2). http://doi.acm.org/10.1145/1265949.1265953
16. Nagami K, Oguri K, Shiozawa T, Ito H, Konishi R (1998) Plastic cell architecture: towards reconfigurable computing for general-purpose. In: IEEE symposium on FPGAs for custom computing machines, 1998 proceedings, pp 68–77
17. McWilliam R, Schiefer P, Purvis A (2015) Creating self-configuring logic with built-in resilience to multiple-upset events. Proc Inst Mech Eng Part B J Eng Manuf [Internet] [cited 2015 Oct 26]. http://pib.sagepub.com/content/early/2015/10/01/0954405415611607
18. Khan S, Phillips P, Jennions I, Hockley C (2014) No fault found events in maintenance engineering part 1: current trends, implications and organizational practices. Reliab Eng Syst Saf 123:183–195. http://dx.doi.org/10.1016/j.ress.2013.11.003
19. Khan S, Phillips P, Hockley C, Jennions I (2014) No fault found events in maintenance engineering part 2: root causes, technical developments and future research. Reliab Eng Syst Saf 123:196–208. http://dx.doi.org/10.1016/j.ress.2013.10.013
20. Soderholm P (2007) A system view of the no fault found (NFF) phenomenon. Reliab Eng Syst Saf 92(1):1–14
21. Cheatham JA, Emmert JM, Baumgart S (2006) A survey of fault tolerant methodologies for FPGAs. ACM Trans Autom Electron Syst. 11(2):501–533

22. Patel A, Prakash K (2010) Fault-tolerant features of modern processors—a case study. University of Wisconsin-Madison
23. Liu Y, Trivedi KS (2006) Survivability quantification: the analytical modeling approach. Int J Perform Eng 2(1):29–44
24. Kastensmidt FL, Reis R (2010) Fault-tolerance techniques for SRAM-Based FPGAs. Softcover reprint of hardcover 1st edn. Springer, 198 p
25. Emmert J, Stroud C, Skaggs B, Abramovici M Dynamic fault tolerance in FPGAs via partial reconfiguration. IEEE Comput Soc [cited 2012 Apr 25], pp 165–174. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=903403&tag=1
26. Parris MG, Sharma CA, Demara RF (2011) Progress in autonomous fault recovery of field programmable gate arrays. ACM Comput Surv 43(4):31:1–31:30
27. Validation resilient logic units by automated fault injection–solutions—national instruments [Internet] [cited 2016 Jun 26]. http://sine.ni.com/cs/app/doc/p/id/cs-17146
28. Pal A, Franciosa P, Ceglarek D (2014) Root cause analysis of product service failures in design—a closed-loop lifecycle model approach. Procedia CIRP 21:165–170
29. de Novaes Kucinskis F, Ferreira MGV (2010) Taking the ECSS autonomy concepts one step further. In: SpaceOps 2010 conference "Delivering on the Dream" Hosted by NASA Mars [Internet] [cited 2015 Apr 14], pp 25–30. http://arc.aiaa.org/doi/pdf/10.2514/6.2010-2364
30. Lee J, Ni J, Djurdjanovic D, Qiu H, Liao H (2006) Intelligent prognostics tools and e-maintenance. Comput Ind 57(6):476–489