Int J Adv Manuf Technol
DOI 10.1007/s00170-013-5070-2

ORIGINAL ARTICLE

Self-healing and self-repairing technologies

Regina Frei · Richard McWilliam · Benjamin Derrick · Alan Purvis · Asutosh Tiwari · Giovanna Di Marzo Serugendo

Received: 29 November 2012 / Accepted: 14 May 2013

Abstract This article reviews the existing work in self-healing and self-repairing technologies, including work in software engineering, materials, mechanics, electronics, MEMS, self-reconfigurable robotics, and others. It suggests a terminology and taxonomy for self-healing and self-repair, and discusses the various related types of other self-* properties. The mechanisms and methods leading to self-healing are reviewed, and common elements across disciplines are identified.

Keywords Self-healing · Self-repair · Technologies

This work is supported by the EPSRC Centre for Innovative Manufacturing in Though-life Engineering Services.

R. Frei (✉) · R. McWilliam · B. Derrick · A. Purvis
EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services, University of Durham,
Durham, DH1 3LE UK
e-mail: work@reginafrei.ch

R. McWilliam
e-mail: r.p.mcwilliam@durham.ac.uk

B. Derrick
e-mail: b.j.derrick@durham.ac.uk

A. Purvis
e-mail: alan.purvis@durham.ac.uk

R. Frei
Intelligent Systems and Networks Group, Department of Electric and Electronic Engineering, Imperial College London,
London, SW7 2AZ UK

A. Tiwari
EPSRC Centre for Innovative Manufacturing in Through-life Engineering Services, Cranfield University,
Bedfordshire, MK43 0AL UK
e-mail: a.tiwari@cranfield.ac.uk

G. Di Marzo Serugendo
Institute of Services Science, Faculty of Social and Economics Science, University of Geneva,
1227 Carouge, Switzerland
e-mail: giovanna.dimarzo@unige.ch

1 Introduction

Mammalian skin is a perfect example of a self-healing (SH) material capable of recovery from serious injury, and in most cases, regaining a fully functional state. Also, many plants are able to self-heal when they are damaged. In fact, most natural systems and organisms are able to self-heal, which considerably contributes to their robustness and resilience. Although the idea to apply such mechanisms to engineering problems has existed for close to a century, this kind of research is still in its infancy. Material scientists have developed ways of making surface coatings self-heal [1] and have successfully applied this principles to a variety of materials and other technologies (e.g. [2, 3]). In robotics, most currently existing approaches to self-repair (SR) rely on some kind of redundancy and the replacement of failing parts, which come at high cost and may require specific repair schedules during the lifetime of the robot.

In software engineering, the implementation of self-healing and other self-* properties is less challenging and comes almost automatically when working with multi-agent systems or service-oriented architectures, although a lot of research is still needed (see Section 3 and [4]). In most other areas, however, self-healing and self-repair strategies are still the subject of intense research. In this article, we review current progress in self-healing technologies and speculate on future developments.

In three previous publications, we discussed important concepts for *complexity engineering* [5], which is the application of the findings of complexity science for engineering purposes, and reviewed advances made in complexity engineering, with a focus on computer science applications [6]. In [7], we investigated a step further and discussed complexity engineering in other research areas, including collective robotics, swarms in nano- and micro-technology, systems biology, Chem-IT, artificial chemical life, self-healing technologies, and all kinds of smart systems. This article is considerably different as it puts the focus specifically on self-healing and self-repair, considering all areas of engineering and technology.

The selection of the surveyed work mainly depended on the literature found; as it is, in all areas except for software engineering and materials, publications about self-healing and self-repair are rather scarce. Where more literature was available than could possibly be used, the authors chose those articles with the most significant contribution to the area being surveyed. This is of course a subjective assessment, and the reader may have a differing opinion.

In terms of the technologies-versus-systems being surveyed, the article was intended to focus on technologies. However, in many cases, the literature does not provide pure technologies, but rather presents them within a system with a specific purpose. Often, self-healing and self-repair only become apparent at system level, although the mechanisms are located at a lower level.

The question of self-healing or self-repair at device or unit level versus system level is also relevant when considering systems where devices or units from various manufacturers or owners come together. While unit or device level mechanisms are unproblematic because they are internal to the unit and only assure the proper functioning of the unit itself, mechanisms that act at system level may be challenging. It will be necessary to develop protocols or norms for how units interact with each other in case of self-healing or self-repair taking place, and the manufacturers of the interacting devices will have to find some consent, just the way they need to consent on communication protocols and safety regulations today. Further research is necessary in this area.

Note that the main focus of this article is on systems that can recover from damage while keeping their original body or substance, as opposed to systems which replicate themselves as originally suggested by Von Neumann [8]. His idea was to build machines that would be able to produce fully functional copies of themselves. So far, at the macroscopic scale, the RepRap[1] 3D printer probably comes closest to this vision, as it can physically reproduce all plastic parts which it is composed of, but it additionally requires metal rods, circuit boards and motors, and still need to be assembled by a human or a robot. Self-assembly, however, is the main focus of a lot of research [9]. Many projects explore self-replication at micro- and nano-scale [10], in cellular automata (CA) [11] and in computing [12]. Unfortunately, some viruses are even too perfect self-replicators. For engineering, self-replication remains a challenge [13].

[1]http://reprap.org

This article is organised as follows: Section 2 presents a terminology and a taxonomy of self-healing and self-repair to clarify the terms used throughout this article and to put those used in literature into context. Section 3 reviews existing work in self-healing software. Section 4 explains how self-healing is achieved in electronics. Section 5 presents self-healing materials. Section 6 introduces mechanical self-repair mechanisms. Section 7 details self-healing in microelectro-mechanical systems (MEMS). Section 8 looks into self-repair in robotics. Other self-healing approaches are explored in Section 9. Section 10 discusses the work reviewed in this article. Finally, Section 11 concludes and presents an outlook in the area of self-healing and self-repairing technologies.

2 Taxonomy and terminology

Over the last few years, the interest in self-repair and self-healing has constantly increased, and together with more contributions in this area of research, the need for a clear taxonomy and terminology becomes stronger. Different researchers sometimes use the same term with different meanings, and the other way round, various terms may refer to the same meaning. As an example, in the context of the BioWatch [14], self-repair is the replacement of faulty cells by functioning cells in the neighbourhood, whereas self-healing refers to the re-integration of recovered cells into the system. This is a very particular interpretation of the two terms ‘self-repair’ and ‘self-healing’, which is not generally shared by other researchers.

This section thus provides a suggestion for a terminology and taxonomy which should be considered as a basis for discussion; it does not claim to be complete nor correct under all circumstances.

2.1 Taxonomy

This taxonomy is as generic as possible. This means that it remains rather abstract but could be used to derive more specific taxonomies for sub-areas, such as self-healing software or self-repairing robotics. The intention is to provide a basis for discussion. Three main aspects are considered: the *failure* that occurs (Section 2.1.1), the *solution* (Section 2.1.2) which addresses the failure, and the *outcome* (Section 2.1.3) of the approach.

2.1.1 Failures

A variety of aspects should be considered when investigating failures.

Causes The causes of failures in technological systems are diverse. They include

- Mechanic
- Thermic
- Electrical
- Electromagnetic
- Chemical
- Software or control system-related

Failure types When failures occur, they are typically caused by one of the following:

- Accidental damage
- Fatigue
- Corrosion/degradation
- Design problem (including software or control system design)

Failure characteristics Furthermore, failures typically are either

- Permanent, which means they remain until repaired, or
- Transient, and thus recover themselves, but may occur repeatedly [4].

Damage When failures are related to damage, it may be of these types:

- Mechanic/structural: crack, rupture and deformation
- Chemical, contaminant
- Optical
- Electrical/electronic: hazardous discharge and conductivity disturbance
- Electromagnetic, photonic
- Logical (software or control system)

Damage may occur

- Locally
- Distributed over various locations
- Globally

2.1.2 Solutions

Solution characteristics Self-healing mechanisms may have any combination of the following characteristics:

- One-off or repetitive
- Intrinsic or extrinsic (e.g. capsules added)
- Autonomic or externally induced (software or user)
- With or without external power
- With or without system intelligence

Solution mechanisms Self-repair or self-healing may be based on one or several of the following principles:

- Heat/cold
- Exposure to radiation
- Reshaping (e.g. through piezoelectric effect)
- Re-establishment of equilibrium state
- Material self-healing
- Excluding faulty component
- Replacing faulty component (redundancy of components)
- Degeneracy (illustrated in Fig. 1): functional and structural redundancy and plasticity [15]
- Redundancy of information
- Stem-cell algorithm
- Cell differentiation algorithm
- Software agents (using any of their characteristics, such as adaptivity or redundancy, etc.)
- Cell growth and division, and cell death

2.1.3 Outcomes

The success of a self-healing mechanism may also have various combinations of characteristics:

- Time scale for healing to occur: from immediate to a few seconds, minutes, hours, days or weeks
- Temporary (for the system to finish its current task) or permanent
- Consuming or conserving the self-healing capability
- Restoring full functionality or maintaining the system in a degraded mode

A preliminary mapping between damage types and potential healing solutions is provided in Fig. 2.

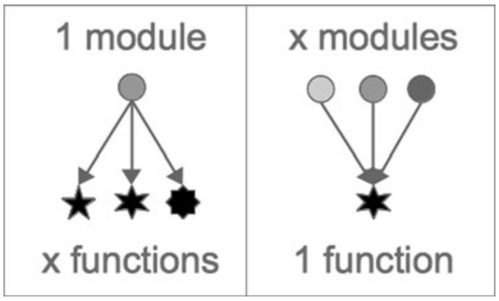

Fig. 1 Degeneracy: functional and structural redundancy and plasticity

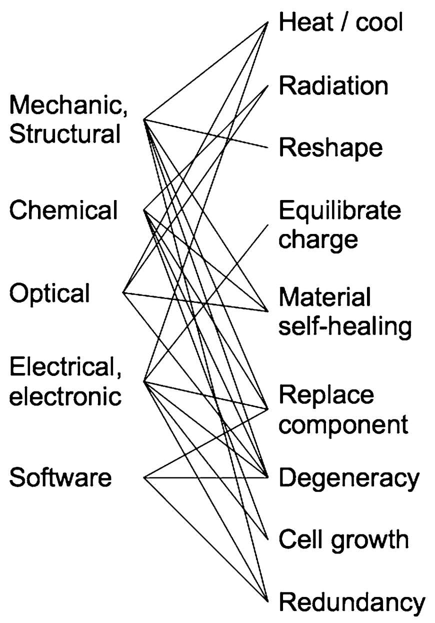

Fig. 2 A preliminary mapping between damage types and potential healing solutions

2.2 Self-healing and self-repair terminology

Various understandings of the terms self-healing and self-repair exist, some of which are contradicting each other and discussed in Section 2.3. Table 1 gives a first impression of the understanding argued in this article, and the subsequent paragraphs provide more details. The terms 'micro' and 'macro' as used in Table 1 may not apply in the case of software systems.

2.2.1 Self-healing

Self-healing is a bottom-up approach, where the components of the system heal the damage from inside. Take for example my broken fingernail; the nail will grow back by itself and gradually remove the damaged area of the nail, without needing any conscious decision or control from the brain. Typically, a failing component is rehabilitated. Not all damage can be repaired, and the design of the self-healing mechanism cannot cover every damage. Additionally, the healing mechanism may also fail.

2.2.2 Self-repair

Self-repair is a top-down approach, where the system is able to maintain or repair itself. As an example, if one of my fingernails breaks, I will use my other hand to file the broken nail into a suitable form, based on a conscious decisions of my brain. Clearly, there is no external influence or control; I execute the repair on myself, using a tool. In case of a robot, an articulated arm may execute a repair function on the robot's base, assuming that it was designed for known failure modes. In case of a self-reconfigurable robotic system, a defective module will be excluded from the system as soon as the peers detect the failure.[2] Typically, failing components will be replaced.

2.2.3 Self-maintenance

Many systems require periodic maintenance interventions. Enabling systems to execute them autonomously will reduce their dependence on the availability of technicians, and it may also reduce downtime for maintenance. Self-maintenance is most effective if the system is able to assess its own need for maintenance as well as to execute it. For instance, a machine would be able to detect the level of impurities in its hydraulic oil and when necessary, drain and refill it from a connected supply. *Design for maintenance* allows engineers to conceive systems in a way that takes their need for maintenance and repairs into account.

2.2.4 Assisted maintenance

In modes where the initiative lies with the technician, such as assisted maintenance or repair , the system is providing assistance to the technician. In cases where the initiative lies with the system, such as self-repair, self-healing and self-maintenance, the assistance comes from the technician providing help to the system.

For instance, systems with *assisted maintenance* provide the user with data about their internal state as well as information about necessary maintenance operations and potentially specific instructions about how to execute them. Assisted maintenance requires that the system designer predicts which kind of failures may appear under which conditions and how the service technician should be assisted in executing the maintenance and repair operations. Assisted healing [16] and repair mean that a system requires human intervention or the intervention of a system-external agent to recover from damage or failure.

2.2.5 Preventive maintenance

In the vast majority of industrial applications, maintenance is performed periodically, for instance, every 3 months or every 100 cycles. These periods must be chosen with a suitable safety margin for the system to never fail because maintenance has not been performed yet. The consequence

[2]Note that the mentioned self-repair mechanism in the modular robotic system is at the level of the system; the defective module is not repaired.

Table 1 Assisted repair vs. self-repair vs. self-healing

	Assisted repair	Self-repair	Self-healing
Orientation	Top-down	Top-down	Bottom-up
Strategy	Replacement of faulty component	Replacement of faulty component	Rehabilitation of faulty component
Scale	Any	Macro	Micro
Applicability	For industry now	Close future	Future
Example	Device assisting a repair technician	Robot replacing a failing module	Skin healing itself from inside

is that there is always a certain waste of resources. The alternative is to perform maintenance based on the measured condition of the systems, that is, for instance, to exchange the motor oil only once a sensor notifies that the particle concentration in the oil has reached a certain threshold.

2.2.6 Regenerative engineering

Regenerative engineering is a term coined from regenerative medicine, where organic tissue is recreated to replace a damaged one; applied to engineering, it could mean that inorganic material is given properties of organic tissues to achieve self-healing properties.

2.3 Discussion of self-healing versus self-repair

In the literature, various meanings are attributed with the introduced terminology. For instance, according to [16], a 'self-healing system is a system that is able to perceive that it does not operate correctly and, without human intervention, to make the necessary adjustments to restore itself to normality'. Based on the definitions suggested in this article, however, the definition given in [16] applies to self-repair, but not to self-healing; a differentiation that is not discussed in the cited work. In [17], self-repair refers to the detection of a fault (through self-testing) and the dynamic replacement of the faulty sub-system.

In the literature as well as common speech, the confusion of the terms self-healing and self-repair is very prevalent. For instance, the definition for self-healing in [18] corresponds to what this article suggests as a definition of self-repair: 'Self-healing systems are characterised by an automatic discovery of system failures, and techniques how to recover from these situations'.

A similar understanding is outlined in [4]: 'Self-healing implementations work by detecting disruptions, diagnosing failure root cause and deriving a remedy, and recovering with a sound strategy'. However, like in many other publications, also in [4], the term repair/self-repair is used without definition, and sometimes in a confusing way. Yet another interpretation of self-healing is used in [19], where 'the self-healing mechanisms should try to reconfigure the system to guarantee at least a limited work capability' and where 'self-healing is used to realise a graceful degradation of the system in case of failures'. Clearly, graceful degradation is not to be confused with neither self-repair nor self-healing, as it does not restore system health.

In the area of software engineering, the classification of self-healing and self-repair is tricky and does not easily suit the schema. There is, however, an argument which justifies the classification suggested in this article. For instance, software services are able to compose different systems according to the needs and the availabilities; as an example, the self-organising displays [20, 21] represented by services create ad-hoc compositions of available multimedia displays to serve a mobile user. The compositions may change in time, and thus, if a failure occurs in a display device—possibly detected electrically or optically—either the health of the problematic device will be restored or another composition will be formed using functional displays.

In such a scenario, where a user is streaming audio using a mobile ad-hoc network, which means that the music is saved on a device not directly connected with the device playing the music, the following could occur: One of the two devices might start to send messages continuously; thus the number of message collisions could prevent the user from hearing the audio with good quality. The system will have a diagnosis service which recognises that the music delivered to the user is below quality standards.

This problem could then be fixed in two ways—either by self-healing or self-repair: self-repair would mean that the system replaces the faulty devices by well-functioning ones. Self-healing, however, would mean that the playback service has a feature that automatically suppresses disturbing message streams. The existing problematic devices are rehabilitated.

Another difficult example is self-reconfigurable robots, where modules can replace each other and exclude failing ones. This may, at first view, look like self-healing, as it may be seen as a process that happens automatically and intrinsically. However, the self-reconfiguration requires the detection of a problem as well as an intelligent strategy to solve it and, thus, should be called self-repair.

2.4 Related terms

A set of terms are particularly relevant in the context of systems that are able to cope with failures. The terms *robustness*, *dependability*, *resilience*, *redundancy*, *degeneracy* and *graceful degradation* are often used in the same context: they all refer to how a system copes with failures and perturbations, which includes systems that are able to self-heal or self-repair.

2.4.1 Robustness [5]

A system does not easily get disturbed in its normal functioning. It can cope with failures, changing conditions, and is able to remain usable.

2.4.2 Maintenance-free

A variety of technologies are being developed with a view towards becoming completely maintenance-free [22]. This implies that they must not require any human interaction while in normal operation.

2.4.3 Degradation-free

Degradation is observed in many areas, with system performance decreasing over time. However, in electronics [23] and spectrometry [24], the possibility of degradation-free systems has recently appeared.

2.4.4 Failure tolerance

A system is able to continue its normal operation in spite of failures occurring. The desire to build failure-tolerant systems goes back a long way [25].

2.4.5 Graceful degradation [5]

A damaged or perturbed system does not totally break down. It maintains at least part of its functionality, even if with reduced performance. This means that the process of self-healing/self-repair is either only very temporary or not complete.

2.4.6 Resilience [5]

The original meaning of resilience refers to the maximal elastic deformation of a material. In the context of computer science and robotics [26, 27], resilience means *dependability when facing changes*, or in other words, its ability to maintain dependability while assimilating change without dysfunction. In the case of MetaSelf [28], a key feature for dynamic resilience is the availability of dependability metadata at runtime. For instance, for dynamically attributing a new server, it is necessary to know the dependability values of the servers in question.

Dynamic resilience is a system's capacity to respond dynamically by adaptation in order to maintain an acceptable level of service in the presence of impairments' [27], whereas *predictable dynamic resilience* refers to the capacity to deliver dynamic resilience within bounds that can be predicted at design time [29]. Accordingly, for MetaSelf, *resilience metadata* is information about system components, sufficient to govern decision-making about dynamic reconfiguration. *Resilience policies* serve as guidelines for reconfiguration.

2.4.7 Dependability [5]

Dependability is the ability of a system to deliver a service that can justifiably be trusted [30]. For instance, a cash machine must always provide the same service, and one must be sure that nothing else happens when we are requesting a certain amount of cash. Central to this definition is the notion that it is possible to provide a justification for placing trust in a system. In practice, this justification often takes the form of a dependability case which may include test evidence, development process arguments and mathematical or formal proof. This is a key issue for the industrial take up of autonomous systems as systems must be completely deterministic to satisfy traditional qualification requirements, especially for (military) aerospace.

2.4.8 Degeneracy

According to [31], degeneracy is an mechanism which can be observed in natural systems such as the brain. In case of a lesion, structurally different brain regions can adapt to take over the tasks of the damaged area. The same can also be achieved in technological systems. For instance, a robot may request new coalition partners to form composite skills, which allow them to take over the task of a failing original robot. Degeneracy is a combination of structural as well as functional redundancy and plasticity [15].

2.5 Self-* terminology

Based on the working definitions published in [32] as well as those referenced and analysed in [33], and taking into account the focus of the current work on self-healing and self-repair, the top-down hierarchy of self-* characteristics we suggest may be as follows:

2.5.1 *Self-configuration*

The system is able to put itself into an operating mode [5], obtain its configuration parameters and initialise itself to provide the expected services [34], or to readjust itself 'on the fly' [4].

2.5.2 *Self-adaptation*

Systems are able to make small changes to maintain or improve their performance under potentially changing conditions [32]. Self-adaptive software evaluates its own behaviour and changes behaviour when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible [35].

2.5.3 *Self-optimisation*

Self-optimisation means maximising resource utilisation to meet user needs [4], reconfiguring the system depending on monitoring data and application-dependent metrics [19].

2.5.4 *Self-testing*

Self-testing means that the system is able to determine its its own state [32] and to determine the presence of a fault [17].

2.5.5 *Self-monitoring*

Using system-internal sensors, the system is able to monitor and keep track of its own performance. This may be connected to self-adaptivity in a feedback loop [32]. Self-monitoring also refers to the logging of process data and its analysis [36].

2.5.6 *Self-knowledge*

The system knows its own functionalities and characteristics [32], or its inner states [37].

2.5.7 *Self-inspection*

Using sensors and self-knowledge, the system investigates its own internal state [32] or compares the actual state with the intended one [38].

2.5.8 *Self-awareness*

The system has prior knowledge of itself [39] and its internal as well as external states, where the latter are defined by the system being situated in its environment [32].

2.5.9 *Self-protection*

The system anticipates, detects, identifies and protects itself from attacks [4, 39]. If protection implies anticipation, it requires some awareness of external states; otherwise, self-protection can be a passive or a reactive process.

2.5.10 *Self-diagnosis*

Besides self-awareness, the system also knows about its failure modes and can determine when one of them has occurred [32], or it can detect and locate faults [40].

2.5.11 *Self-reconfiguration*

The system is capable of autonomously re-arranging its components to fit changing circumstances [32, 41].

2.5.12 *Self-organisation*

Self-organisation is the dynamical and adaptive mechanism or process enabling a system to acquire, maintain and change its organisation without explicit external command during its execution time. There is no centralised or hierarchical control. It is essentially a spontaneous, dynamical (re-) organisation of the system structure or composition [42, 43]. For a thorough discussion of self-organisation, refer to [32].

2.5.13 *Self-management*

At the very core of the autonomic computing paradigm, it includes self-configuring, self-healing, self-optimising and self-protecting [44, 45].

Notice that the above descriptions are working definitions; they may very well vary depending on the domain of application, and they certainly do require further discussion. Many authors use self-* terminology relying on the reader's intuitive understanding of the terms, but do not make the intended meaning explicit; examples include [46–48]. Furthermore, there is no limit to the variety of self-* terms that can be created; for instance:

- Horn [49] also uses *self-regulation, self-evolving, and self-governing/self-governance*.
- Sterritt and Bustard [39] mentions *self-adjustment and self-learning*.
- Hinchey and Sterritt [50] intends to create *selfware* and write about *self-situation/self-situatedness and self-destruction*.
- Ibanez et al [38] refers to *self-recognition*.
- Goldberg [37] explains *self-deception*.

2.6 Technology readiness level

Throughout the following survey sections, the commonly known technology readiness level (TRL) will be used to indicate how close the self-healing and self-repair mechanisms are to being used in an industrial context or another customer-oriented application.

The European Space Agency (ESA) defines the levels [51] as listed in Table 2 in the left column; our own interpretations are in the right column.

3 Self-healing software

3.1 Introduction

Software systems which consist of many interacting entities—such as agents or services—often have an intrinsic ability to adapt and to recover from failures and disturbances; adaption is the 'norm', and they are thus self-healing by definition. Agents are self-contained units that have a certain autonomy. They have knowledge, skills and interfaces; they interact with their peers and with the environment. Their behaviour follows certain rules or policies. Multi-agent systems typically contain a redundancy: if one agent fails, another or several others will take over the task. Agents may collaborate and/or compete. Their behaviour will always depend on the rules of behaviour specified in their code, and thus, any desired self-* capability may be designed or provided through evolutionary techniques. This is not to imply that creating self-* properties in multi-agent systems is easy; numerous researchers all around the world are working on it, but there is certainly a lot of potential, and it may be easier to make a software system be self-* than a purely mechanical system.

Self-* software systems arise from the result of two originally distinct research efforts both based on software agents to help software adapt to changes in their environments: *self-managing* software [52] and *self-organising* software [53]. Self-managing software addresses large-scale distributed systems that configure, heal, protect and optimise 'on their own', usually in a hierarchical way, in order to alleviate human administrators' work. Typical applications include management of corporate data centres, or more recently grid and cloud computing infrastructures. Self-organising software addresses distributed systems able to (re)-organise themselves without external control, and usually work in a decentralised way. Typical applications involve swarm robotics, optimization problems, agent coordination or services ecosystems. Recent work also tries to

Table 2 Technology readiness levels according to ESA and this article

Level	ESA definition	Definition suggested in this article
Level 1	Basic principles observed and reported	(same)
Level 2	Technology concept and/or application formulated	(same)
Level 3	Analytical and experimental critical function and/or characteristic proof-of-concept	(same)
Level 4	Component and/or breadboard validation in laboratory environment	(same)
Level 5	Component and/or breadboard validation in relevant environment	(same)
Level 6	System/sub-system model or prototype demonstration in a relevant environment (ground or space)	(same)
Level 7	System prototype demonstration in a space environment	System prototype demonstration in the end user environment
Level 8	Actual system completed and 'flight qualified' through test and demonstration (ground or space)	Actual system completed and qualified through test and demonstration
Level 9	Actual system 'flight proven' through successful mission operations	Actual system successfully operating in real-world application

bring together those two research trends combining self-managing techniques with self-organising and decentralised ones together.

It might be argued that software cannot heal itself because code can rarely write itself or correct itself when broken. It often comes down to self-management with dynamic reconfiguration. Self-healing is, however, a term used in this context in published literature about self-healing.

3.2 Survey

Self-managing software is an answer to software systems getting increasingly complex and difficult to manage by human administrators. In this category, *autonomic computing* [49, 52] was proposed, inspired from the autonomic nervous system. It tackles specifically on four areas: self-configuration, self-optimisation, self-healing and self-protection. Software should actively manage itself instead of passively being managed by a human administrator. Most self-* properties can be achieved under the responsibility of a single autonomous entity (a manager) which controls a hierarchy of other autonomous entities. The *autonomic manager* consists of a central loop which handles all upcoming events within the system. The autonomic manager follows the *MAPE loop* [66], which stands for *monitoring, analysis, planning and execution*, supported by a knowledge base. An alternative to this centralised approach is *decentralised autonomic computing* [67], where interacting and fairly autonomous individuals replace the manager.

High-level *policies*, specified by human administrators, are at the heart of self-managing systems. The adherence of autonomic managers to those policies, as well as their interactions globally ensures the policies. Policies may be of different types, from action-based (e.g. what to do in a given state) to goal-based (e.g. what the desired state to reach next) or to utility function-based (e.g. computation of the desired state to reach based on utility function to optimise) [68, 69].

Concepts mirroring human mechanisms, such as reflex reactions and the use of vital signs to assess operational health, may be used to design and implement a personal autonomic computing architecture [55]. This was implemented on personal computing devices as a self-healing tool using a pulse monitor and a vital signs health monitor within the autonomic manager. The presented support architecture for multi-platform working is based on autonomic computing concepts and techniques and enables collaboration among personal systems to take a shared responsibility for self-awareness and environment awareness.

Autonomic computing is also used to manage the security of knowledge management systems in organisations [54]. A partial autonomic system with self-healing mechanisms can provide a stable environment for securing enterprise knowledge assets and can reduce hacking. The model of the self-protection and self-healing configuration attributes in autonomic systems uses game-theoretic approaches. The proposed modelling approach progressively moves from a manual intervention-oriented security setup to an autonomic security setup using game-theoretic approaches. The authors provide insights on the applicability of these approaches to different security environments. An autonomic approach is especially attractive when it is difficult to impose penalties on malicious users.

A grid platform exhibits the autonomic characteristic of self-healing to ensure workflow execution [59]. The work enables a mobile agent workflow management system to optimally configure its fault-tolerance mechanisms through awareness of the computational environment. The authors develop a model for dynamic fault-tolerance technique selection. This can be embedded in a mobile agent workflow management system, allowing the system to optimally configure its fault-tolerance mechanisms through awareness of the computational environment.

Self-organising software [53] targets decentralised systems made of numerous entities evolving together to produce some global result. Self-organisation is often closely related to self-healing, as adaptation is part of the regular behaviour of the system. So, in case of a failure, self-organisation will lead to the system maintaining its functionality in an alternative way. Self-organising software usually takes inspiration from nature and is composed of relatively simple autonomous components or agents, each applying specific *rules* with only a local knowledge of their environment. Through the joint work and local coordination of the different components, some global result 'emerges'. The different elements of design of a self-organising system include the (active) agents themselves, the self-organising mechanisms (or rules) to which they abide, the environment in which they evolve and the (static) artefacts on which agents and environment act upon [70]. A description of types of faults for self-organising systems linked to each design element is also provided in [70].

Crucial to the decentralised coordination among agents are the *self-organising mechanisms*, usually inspired by nature. Among the most popular ones, we can cite: stigmergy (indirect communication through the environment like ant coordination through pheromones [65]), bird flocking or fish schooling, gossiping or immune systems, as well as evolutionary techniques [71]. Engineering efforts led researchers to express those self-organising mechanisms under the form of design patterns [72, 73]. Recent work in this area provides a comprehensive classification and description of design patterns for self-organising mechanisms together with a detailed description of their interrelations [74]. This work is completed by a preliminary software architecture proposal for using the self-organising mechanisms as basic primitives provided by the environment [75].

Shifters [60] are software agents that are similar to stem cells. In the beginning, they are neutral elements and then evolve towards having a specific function. This emergent adaptation is based on the needs of the system as well as adaptation pattern [76] and caused by the agent's interactions with its peers. By transforming initially neutral elements into any type of agent, just as required, the system has the ability to cope with failing agents by generating new ones. This is a type of redundancy strongly inspired by the way our body replaces dying cells by new ones.

Another cell-based programming model [61] serves as the basis for further programming abstractions which will inherit the biological characteristics of intrinsic robustness, scalability and self-healing. The computational cells go through a set of states and are able to reproduce through cell division in case some cells become defective. The intended functionality of this cell-based self-healing software model is the same as described above for shifters, although the mechanism used to achieve this is different. In order to take advantage of both worlds, research efforts are now *combining both self-managing and self-organising* characteristics within software systems.

Organic computing (OC) brings together neuroscience, molecular biology and software, with particular emphasis on self-* properties. Other areas of influence [77] are cybernetics, general systems theory, artificial neural networks, evolutionary algorithms, systems biology, computational intelligence, cognitive systems, behaviour-based robotics and multi-agent systems.[3] OC aims at a favourable trade-off between autonomy and controllability, which machines may never lose, i.e. freedom and safety. A similar trade-off is required between robustness versus sensitivity [77, 78]. One of the main goals of OC is to find ways of controlling emergence [79]. OC is oriented towards the user's needs: to be more user-friendly, systems are designed to be *life-like*. They dynamically adapt to the changing environment. To achieve this, OC previews the *controller/observer architecture* [80], which is very generic. It consists of three layers: a top layer (high level) with reasoning, simulations and observation capabilities, which can give feedback; a middle layer; and a bottom-up layer (low level) with reflexes. The low level assertions [79] are similar to action policies (if-then rules). The counterpart in the system which receives violation messages is the *observer* (acting as a sensor or detector); the one which takes measures accordingly is the *controller* [56]. When the observer decides that reconfiguration is necessary, the controller will execute it. Notice that the above-mentioned self-managing MAPE architecture can be considered as a type of observer/controller architecture. The architecture does, however, have some centralised and top-down structures, as management and supervision functionalities are bundled in the *framework component*, which is the control and configuration manager. But this depends on the actual implementation, which can also be decentralised.

Taking inspiration from chemical reactions, the MYRIADS project [62] designs and implements systems and environments for autonomous service and resource management in distributed virtualised infrastructures, such as grids or cloud computing. The focus is on creating dependable applications and efficiently managing resources in the future *internet of services*. Computations happen according to a set of rules, similar to how chemical reactions happen between molecules in a solution; this is described in the *chemical reaction model* [81]. The computational 'molecules' are stored in a multi-set, and their reactions occur in parallel and in an autonomous way. Hence, services may be composed to fulfil tasks that are specified as workflows [82].

The SAPERE project [63] tackles both self-management and self-organisation for ecosystems of services, exploiting the chemical reaction metaphors to accommodate both self-management and self-organisation in a unified framework. Actual implementation involves a middleware deployed on intelligent screens and tablets communicating with each other, and supporting self-management and self-composition of context-aware services.

A distributed layered architecture is used for multi-robot cooperation with self-* properties [64]. The upper layer uses ant behaviour (based on stigmergy) to self-regulate the regional distribution of robots in proportion to that of the moving targets. The lower layer uses self-organising neural networks to coordinate their target tracking; self-configuring, self-optimising, self-healing, and self-protecting are emergent properties resulting from this approach. In the same mindset, through a joint use of *rules for self-organising mechanisms* and *policies*, the MetaSelf framework [28] provides a proposal to accommodate both self-management and self-organisation in a unified framework. Other policy-based approaches [69] lead to self-healing systems, where artificial intelligence principles provide guidance for autonomic systems to choose actions that move it into desirable states. Meta-rules [83] allow systems to adapt their rule-guided behaviour to dynamically changing conditions.

Additional works targeting specific applications or new concepts include [57], where the term 'self-healing' is used in an argumentation for applying the principles of

[3] Organic Computing may be considered as an area of natural computing, which provides an umbrella for many ways of bringing nature and computing together.

complex adaptive systems to the engineering of self-repairing energy systems, telecommunications, transportation, financial and other infrastructures. The suggested approach consists mainly of using multi-agent software systems to obtain robustness and adaptivity, which are intrinsic characteristics of self-healing and self-repairing systems. To realise the self-healing function in computing software and systems under real-time requirement, a new concept defined as consequence-oriented diagnosis and healing is introduced [84]. The hybrid diagnosis tool is based on the multivariate decision diagram, fuzzy logic and neural networks, achieving efficiency in diagnosing and preventing software and system failures.

More generally, to achieve self-healing in software systems, an architectural style needs to meet certain requirements [85], which include adaptability, dynamicity, awareness, autonomy, robustness, distributability, mobility and traceability. These requirements support the four phases of a self-adaptive software life cycle which consists of monitoring, planning the necessary changes, deploying change descriptions and enacting the changes. How well an architectural style meets the requirements for self-healing can be measured along five features of software architectures: external structure, topology rules, behaviour, interaction and data flow.

Further details and reviews of autonomic communications discussing issues linked with self-organisation, decentralisation, context-awareness and stability are provided in [86]. A survey of bio-inspired networked systems technologies reviewing both self-managing and self-organising systems is provided by [87].

A survey and synthesis on self-healing software [16] provides references to further work and categorises the contributions as focusing on maintaining system health, detecting failures and system recovery. Especially in terms of software systems, it is important to make the difference between self-healing systems and systems that are self-adaptive, robust, resilient or dependable [16]; for a discussion of these terms, see Section 2.4.

Self-healing systems can be seen in the context of work on *fault-tolerant computing* [88] and *resilience* [29], where the goal is to enable a system to maintain its intended function even when facing faults, and *recovery oriented computing* [89] which contributes by providing dependable internet services.

Finally, a recently published survey on self-healing software systems [4] similarly points out that these systems originate both from research on fault-tolerant [25], self-stabilising [90] and survivable [91] systems, as well as in autonomic computing [52] and self-adaptive [35] systems. A lot of importance is given to feedback loops in self-healing systems, which can be implemented in various ways and at various levels, as well as to the system behaviours being guided by adaptable policies. A particular challenge is the fact that there is often a grey zone between desired or healthy and undesired or faulty system behaviour. The survey [4] provides an overview of failure classes, based on the failures of fault-tolerant systems [92], as well as a broad survey on implemented approaches and applications in the areas of embedded systems, operating systems, architecture-based as well as cross or multi-layered approaches, agent-based systems, reflective middleware, legacy applications, discovery systems, web services and quality of service-based systems. The approaches are analysed in terms of their strategies for detection, diagnosis and recovery.

3.3 Analysis of mechanisms

Table 3 provides an overview on the technology readiness levels of the mechanisms for self-healing used in the work cited above.

4 Self-healing electronics

4.1 Introduction

Electronics is an area with clear and achievable cases for demonstrating self-healing: micro- and nano-scale interconnects and semiconductor structures are extremely sensitive to electromagnetic, chemical and mechanical influences, and are therefore susceptible to a variety of fatigue mechanisms and external upset events. Self-contained reliability is most often achieved by one of two approaches: fault-tolerance and self-repair. Fault-tolerant methods enable the system to absorb one or more failure events while continuing normal operation. On the other hand, self-repair strategies are reactionary, i.e. they activate after the failure event has occurred and seek to restore operation by some internal reconfiguration mechanism.

4.2 Survey

A feature of all self-restoring systems is an algorithm, process or chemical reaction that is trying to converge on a final state of wholeness, healthiness or perhaps minimum energy or entropy. To emulate this in electronic systems, it is easiest first to think of the software algorithm that will act to restore a data pattern. Simply keeping a copy of the correct pattern to cross check is one solution, but the risk of the master copy being corrupted leads quickly to the use of *triple modular redundancy* (TMR) or *N-modular redundancy* strategies. This is a huge overhead in resource: a 300 % increase in ICT cost to protect one circuit's output. Alternative strategies have been suggested, whereby a finer-grained redundancy

Table 3 Technology readiness levels in mechanisms for self-healing software

Mechanism	TRL	Reference	Reasoning
Autonomic computing principles	L5	[52, 54]	Autonomic computing principles are being applied to an increasing number of problems but have not become widely used in industry yet
SW and HW condition monitoring	L5	[55]	Being applied and validated
Organic computing principles	L3	[56]	Principles of functionality are being developed in the lab
Agent technology	L4–L9	[57–59]	Agent technology has, in some cases, already reached industrial deployment, but other developments are still at earlier stages, depending on other technology aspects
Stem cells	L2	[60, 61]	The concept is formulated and simulated in a lab experiment
Chemical reactions model	L1–L4	[62, 63]	Application of the chemical reaction principles to computing and system behaviour is being observed. Deployments on mobile devices are provided
Rules and policies	L3	[28]	Principles of functionality are being developed in the lab and demonstrated in simulation
Artificial neural networks	L4	[64]	Validated in the lab
Stigmergy	L7	[64, 65]	Numerous experimental systems use stigmergy and are shown to be functional in the application environment; evidence of actual industrial use remains difficult to find

can lead to fractional overheads by exploiting the reconfigurability of field programmable gate array (FPGA) devices to reroute around localised faults [105]. This granularity provides a tuning slide, where the overhead is in proportion to the protection required. In [93, 94], the mapping of this software approach is mapped to the hardware structures themselves.

Achieving fault-tolerant electronics has been the focus of mission critical systems for several years. Of the possible strategies, TMR has been applied most extensively for the case when a single event upset (SEU) is predicted to affect a particular system or sub-system [106]. Alternative strategies have been proposed, where the functional electronics behaviour is implemented by a CA, the global output state of which is determined by a special set of rules and by state mapping logic. In addition to such ultra-reliable applications, this is perhaps equally relevant to high-volume electronics markets that exhibit underlying reliability issues.

A fault-tolerant parallel processor based on cellular automaton models was proposed [107], in which the behaviour of processing element arrays is governed by CA neighbourhood state-transition rules including the flow of data. When a faulty interconnection or processing element is detected, reconfiguration to healthy state is achieved by masking and re-mapping between the faulty and redundant elements. This concept has been advanced by considering the processing unit as a convergent cellular automata [94], i.e. a set of CA rules is derived deterministically which are driven by nearest-neighbour communication and which are independent of the current state of the CA. The next-state rules drive the CA to relentlessly converge upon a specified global state. If cellular states are mapped to logic functions (via selectable bitstreams), the CA behaves as a robust logic unit and redundant cells replace faulty ones in the event of failure.

A working full adder was demonstrated by this principle using an FPGA (a custom ASIC design would lead to a more efficient implementation), wherein spare cells are included in the design which are used when a faulty cell is detected, the rule set being copied to the replacement cell. A reliability analysis compared the CA implementation to that of an N-modular redundant system consuming equivalent FPGA resource and showed that for many situations, the reliability of the CA implementation was considerably higher especially for ultra-reliability scenarios. Besides rules, the behaviour of a CA is driven by boundary values; by following the same rules but instead new boundary values, it was shown that the CA converges to new behaviour in the form of other logic units.

Another approach to self-healing electronic hardware is inspired by eucaryotes and procaryotes [95]: Memory cells contain the equivalent of DNA fragments which describe the cell's characteristics and functionalities. Faulty genes can then be extracted from neighbouring cells and using correlation mechanisms, allow the damaged cell to self-heal and establish its original state. The system is hierarchical, with the logical block corresponding to a biological molecule, up to an electronic array representing a biofilm formed of bacteria, and a bus standing for a cytoskeleton.

To make electronic circuits self-heal, carbon nano-tubes have been encapsulated inside polymer spheres. Carbon nano-tubes have high electrical conductivity, and their elongated shape is ideal for lining up to bridge gaps [96]. When the electronic circuits experience a mechanical impact—for instance, because a phone is dropped on a hard floor—tiny gaps may appear in the structure and impair the functioning. The carbon nano-tubes may then repair the circuit and re-establish its function. Alternatively, the same effect is reached through the inclusion of micro-capsules that are filled with a liquid metal (gallium–indium) [97, 98]. A combination of bigger and smaller micro-capsules may optimise both reliability and conductivity after repair.

A different type of self-healing in electronic chips relies on electrical fuses [99, 100]: after self-diagnosis with the help of an on-chip built-in self-test, the chip is able to 'blow' fuses to reroute its circuit to avoid defective areas. The so-called *eFuse* is a micro-metric strip of polycrystalline silicon and is operated through *electromigration*, which is a process where current pushes atoms out of place and thus changes their electro-conductivity, resulting in a logical rerouting of the circuit.

Related approaches use insulator layers, for instance aluminium oxide, which melt away when a voltage is applied that is greater than the insulator's breakdown voltage. Application examples include street lamps and christmas lights; their bulbs are serially connected. If one fails, all of them would stop working. Thanks to the melting fuse, however, the remaining bulbs continue to function even in the presence of failing ones.

The *BioWatch* [14] was part of the Embryonics project,[4] which also developed multicellular automata with self-repair and self-replication [108, 109]. The BioWatch is an application example of the *BioWall*, a bio-inspired electronic wall covered with 'reconfigurable computing tissue' that can display the time and has touch-sensitive buttons for the human to interact with the wall. Indeed, touching a cell causes a temporary 'cell death'. The self-repair mechanism then replaces the dead cell by a peer, whereas through self-healing, a recovered cell is re-integrated into the system. Each cell contains a compete blueprint of the system and fulfils its specific function in time counting depending on its position; the system is thus highly redundant. This electronic tissue models the biological processes of *ontogeny* (growing, here including learning), *epigenesis* (development) and *phylogeny* (evolution); this is also called the POE model, and accordingly, POEtic tissue [110].

In the scope of the PERPLEXUS project, a bio-inspired approach to electronics named THESEUS was developed [10]. The biological hierarchy of

$$molecule - organelle - cell - organism$$

corresponds to the electronic hierarchy of

$$logic\ cell - programmable\ logic\ cell\ array - chip - device.$$

The logical cells all contain the full genome and express different functionalities depending on their place in the organism. Faulty logical cells are replaced by others, which are then differentiated accordingly. The configuration of the system spreads through the cell array by a serial propagation mechanism. Through to the system's capability to self-inspect and to self-reconfigure, a form of self-repair can be achieved.

A similar approach, also based on the POE model, implements the embryonic approach on an FPGA-based artificial cell network [101]. Due to cell redundancy and information redundancy, the system exhibits robustness and fault tolerance when facing cell failures.

In microprocessors, soft errors can be eliminated by various error detection and correction methods such as parity and error-checking codes which are able to correct one or more bit errors occurring in a data stream. The soft error rate for this kind of SEU is expected to increase in the future due to diminishing transistor size [111]. These self-correcting mechanisms are routinely applied to strategic points within the processor system such as the memory controller.

Recovery from hard errors in microprocessor systems is demonstrated by the IBM POWER high-performance critical server platform [111, 112]. These microprocessors are capable of restoring operation following permanent hardware failures in memory and clock signal sub-components by invoking redundant hardware in an autonomous fashion that would otherwise require system downtime. Concern over the reliability of external bus interconnections is addressed in the dynamic random access memory (DRAM) interface, which is capable of sustaining failure in one or more data bit lines by dynamically re-mapping the faulty bit line to a redundant bit line. This is performed either during power-on self-test or during execution and requires co-operation between the CPU and memory controller. The faulty condition is initially detected by the occurrence of parity errors and failed re-send requests. In addition to external faults, the CPU also runs continuous memory checks during either execution or idle time.

Advanced error correction is implemented in external Dual Inline Memory Modules (DIMM), such that in the event of a faulty DRAM chip being detected, the DIMM is capable of sustaining further data errors and, therefore, does not require immediate replacement. By further including redundant DIMM modules, memory self-healing is enhanced by re-mapping troublesome DIMM ranks to corresponding ranks of redundant modules [113]. Memory

[4] http://lslwww.epfl.ch/pages/embryonics

checking in idle time can be performed via data scrubbing to systematically check all memory cells. If during this procedure an irrecoverable failure is detected, a chipkill event is issued, whereby the processor marks the offending memory location as defect and remaps to another location. Memory modules with the IBM Chipkill were combined with RAID functionality for the NASA Pathfinder Mars mission [114].

A self-repairing mechanism is also applied to the master oscillator of the main board of IBM POWER and Intel Itanium processors [112, 115–117], whereby a redundant oscillator is immediately initiated in the event of failure. This requires dynamic self-synchronisation between the two oscillators to ensure seamless switchover.

A formal method for the analysis and synthesis of a digital system with inherent fault tolerance [102] uses Petri nets to determine the necessary level of redundancy. The technique has ramifications for designing fault tolerance into chips using high-level synthesis tools rather than at the gate level. High-level analysis leads to low-level analysis, where error checking, built-in self-tests, redundancy and other self-repair concepts can be added.

An early approach towards self-* electronics was the fault-tolerant architecture [103] for self-maintaining AGV, based on vision involving direct replacement or rippling out of modules. The challenge in image processing is to balance processing requirement, error correction and real-time constrains (i.e. speed vs. reliability); it is addressed through electronic processor scheduling.

Also, fault diagnosis for electronic systems [104], applied to a distributed vehicle system, contributes to self-* properties becoming more realistic. The sharing of fault diagnosis information among different levels assists the repair process.

4.3 Analysis of mechanisms

Table 4 provides an overview on the technology readiness levels of the mechanisms for self-healing used in the work cited above.

5 Self-healing materials

5.1 Introduction

Self-healing materials are increasingly appearing in the media lately, for instance, with some manufacturers promising self-healing paint on their cars or phone protection cases, or with researchers developing resins that can be integrated in aircraft to show micro-cracks. Self-healing materials is clearly the area with the most directly applicable research, where industry shows the least resistance to deployment. Today, self-healing materials are best known to the general public, although first advances were made almost a hundred years ago.

5.2 Survey

Probably one of the earliest examples of self-repairing technology are car tires invented around 1934: the first tires that could temporarily self-heal in case of damage had been invented for military purposes—to resist gun shots—and were then also used on commuter trains and trolleys [118]. Since then, various versions of the idea have appeared and been sold to customers who are ready to pay the higher price, both for self-healing and assisted healing. [5]

Self-healing coatings or paint have hit the headlines several times over the last few years and created great expectations among car owners [131]. An 'early stage' company called *Autonomic Materials* have started to market self-healing coatings from Champaign, IL, USA, under an exclusive licence.

The polymer coatings contain micro-capsules of repairing agents as well as catalysts which are set free when the coating is damaged. The healing polymerisation process takes energy from ultra-violet light and re-creates an even surface. By treating micro-cracks in their early phases, bigger cracks are prevented from spreading [119, 120]. It remains, however, unsolved at present if larger-scale damage caused by mechanical influence can be healed as well.

In polymers and composites, three types of self-healing have been observed so far: capsule-based healing systems, vascular healing systems (inspired by the caoutchouc tree) and intrinsic healing polymers [1]. The self-healing mechanisms may require an external source of energy or pressure.

Vascular systems may use different technologies: In one case, hollow glass fibres inside laminate layers provide a separated space for the healing resin to flow, at the cost of high brittleness. In the other case, tin solder is included in the laminate to create the vascular network and is then melted at high temperatures [121, 122]. Modelling such composite laminate structures, it can be determined where damage is most likely to occur and, thus, where the healing resins must be located ideally [132].

An application of such technology, Firetec[6] has a range of self-* fire extinguishers with self-healing tubes and system level self-repair capabilities. They address the automatic targeted sensorless extinguishing of fires.

Self-healing not only helps against corrosion and mechanical damage but also thermic damage. Besides the previously cited approaches to self-healing, also other

[5]http://www.slime.com

[6]http://www.firetecuk.com

Table 4 Technology readiness levels in mechanisms for self-healing electronics

Mechanism	TRL	Reference	Reasoning
Data-restoring algorithm	L2	[93, 94]	Development of algorithm; laboratory hardware demonstration
DNA approach	L1	[95]	Theoretical development awaiting hardware implementation
Healing nano-tubes	L2	[96]	Nano-tube suspension in capsules demonstrated; healing of electronic material to be demonstrated
Healing capsules	L3	[97, 98]	Laboratory demonstrations showing healing of electronic material
Melting fuse	L9	[99, 100]	In active use for several years for production repair and in-use self-repair
Embryonics, POE model	L2–L4	[10, 14, 101]	Hardware/software at conceptual stage; some instances of hardware validated in a laboratory environment
Software and formal methods for redundancy	L2	[102, 103]	Technology concept formulated
Sharing of local/global diagnosis information	L2	[104]	Technology concept formulated

topics are being investigated [133]: In *ballistic* self-healing, a bullet penetrating a plastic membrane generates enough heat for the material to intrinsically re-seal the hole; so-called *nano-clay beams* are healing elements in elastomers that will absorb mechanical energy by opening up and heal again once the stress is released.

In self-healing bearings, one of the usual steel balls is replaced by a ceramic ball, which will constantly polish the raceways and maintain them in a super-finished condition. This considerably increases the bearing's wear resistance while reducing noise and vibrations [124].

Alternatively, the same principle as in self-healing paint can also be used for bearing surface coatings: lubricant capsules will burst when surface wear starts to appear, and can–at least for as long as the additional lubrication lasts—prevent further damage [134]. Recent related research investigates the possibility to produce coatings containing capsules with two types of substances which may act as a two-component adhesive.

To achieve the maximal material strength, the concentration of capsules should be minimal, but more capsules means more self-healing. A solution to his may be the integration of micro-wires made of shape memory alloy in the composite material, as it was done for the Alinghi hull [2, 135]. Electric current brings deformed wires back into their original shape, and thus the few self-healing capsules are sufficient to heal the smaller crack. Additionally, the local heating due to the electricity facilitates the healing polymerisation effect. The challenge is now for the material to become able to sense where it has been damaged and to send current to these locations only.

Concrete—as used in construction—develops self-healing properties if micro-capsules with a sodium silicate solution are included [3]. Corrosion is effectively inhibited, and cracks are healed. Also, structures built of traditional lime mortar are known to exhibit such self-healing properties through natural recristallisation when exposed to air, but concrete has many other properties which are needed by today's construction industry, such as a higher strength and being easier to work with than mortar.

A step further towards biotechnology goes the inclusion of bacteria in concrete or asphalt to repair cracks [125]. As soon as water is intruding, the alkali-resistant spore-forming bacteria create minerals that fill up the crack and thus re-seal the material. A related approach is to use *protocells* [126] to build 'living' structures that dynamically reinforce the existing old ones, such as walls and piles. In the *Future Venice* project,[7] the idea is that the protocells use substances available in the water or air to self-assemble and build structures.[8]

On-the-fly repairs (in the literal sense) occur in self-healing aircraft: the hollow parts of composite-based plane are filled with a hardening epoxy resin, which 'bleeds' out of any hole or crack that forms during a flight and patches it up. This assures a safe continuation of the journey, until the aircraft can be properly repaired on the ground. The idea is that the healing liquid might be flowing through a vascular system in the plane, similar to the way blood circulates in our body [122, 123].

The latest advances in terms of artificial skin combine pressure and flexion sensitivity with mechanical self-healing [137]. It is a composite material composed of a supramolecular organic polymer with embedded nickel nano-structured micro-particles. When ruptured, the material can re-establish both electrical conductivity and mechanical strength.

[7]http://www.futurevenice.org

[8]Protocells represent the body of research in chemistry and bottom-up synthetic biology [136], which intends to investigate how life could have emerged. Chemical compositions have been found, which exhibit life-like characteristics including 'cells' (or rather, blobs or bubbles) separating, merging, interacting, pulsating and moving through chemotaxis.

5.3 Analysis of mechanisms

Table 5 provides an overview on the technology readiness levels of the mechanisms for self-healing used in the work cited above.

6 Self-healing and self-repairing mechanics

6.1 Introduction

Purely mechanical self-healing or self-repair is very limited and rare. Most often, the phenomenon of self-repair manifests itself in a mechanical system, although strictly speaking, its origin may be in the material, the control system or in redundancy. Nevertheless, the manifestation makes it count as mechanical, because the self-repair is crucial for the system to remain mechanically functional.

6.2 Survey

In the agricultural industry, self-sharpening ploughshares were developed because the traditional ploughshares would become dull after a few hours' work. The principle was patented in 1785 by Robert Ransome from Ipswich, UK [127]. The principle was that the lower surface would be cooled more than the upper one. With iron being harder at lower temperatures, the lower surface would thus be harder, and abrasion would be slower than on the upper side. This simple mechanism assured that the cutting edge would always be sharp, and today's ploughshares still use the same principle. Certainly, the calculations of abrasion laws have become more precise, and the materials more sophisticated, e.g. multi-layered [140]. However, self-sharpening ploughshares are still observed to perform considerably better than traditional ones [141].

Self-sharpening knives [128] work with the same principle, but the increased hardness of one surface is due to Hardide^{TM}, which consists of tungsten carbide nanoparticles dispersed in a metal tungsten matrix. Nanostructured materials show unique toughness and crack and impact resistance. Thus, coating only one side of tools for cutting paper and plastics, while leaving the other side uncoated; therefore, more vulnerable to abrasion leads to self-sharpening effects [128]. Indeed, the thin hard coating on one side of the blade will serve as the sharp cutting surface, while the softer metal which carries the coating will get abraded and never let the blade become blunt.

While the development of a self-sharpening form of the ploughshares and knives qualifies as a mechanical strategy for self-healing, the use of a multi-layered material would, strictly speaking, make the example belong to Section 5; however, the effect of the self-sharpening does manifest itself, critically maintaining a mechanical functionality. If it was not for the material keeping the knife sharp, it would lose its use.

Recently, another type of self-sharpening knife has appeared on the consumer market. Their 'self-sharpening' effect is reached by sliding the blade over an abrasive surface every time the knife is taken out of the holder. In this sense, the cutting surface does not really self-sharpen, but is sharpened by the unaware user; in other words, the user is considered as being inside the borders of the considered system here. This is a concrete illustration of how the definition of system boundaries is crucial for defining a system as being self-* or not.

While usually not referred to as self-healing or self-repairing, shape memory alloys or *smart metals* are capable of re-establishing an initial cold-forged state after deformation. Shape memory materials [129] are used in numerous applications across many fields, including aircrafts, automotive industry and medical engineering, to name but a few. Therefore, by exposing a deformed part to the necessary stimulus (typically a change of temperature, electrical current or a magnetic field), the original shape would be regained, and the system might resume its functionality. Shape memory materials may thus be considered as offering a mechanism that is predestined to be used for self-healing or self-repair.

Under the title of 'self-repairing mechanical systems', the principles used in reconfigurable robotics (Section 8)—

Table 5 Technology readiness levels in mechanisms for self-healing materials

Mechanism	TRL	Reference	Reasoning
Liquid inside tires	L9	[118]	Well known and widely used technology, available in the market
Intrinsic polymer healing	L4	[1]	At the lab test stage
Capsules	L4–L5	[3, 119, 120]	Being tested both in labs and relevant environments
Vascular layers	L4	[121–123]	At the lab test stage
Ceramic ball bearings	L9	[124]	Being used in industry
Bacterial residues	L3	[125, 126]	Functionality being explored

namely self-assembly and self-repair through module replacement—were discussed [130]. This requires component redundancy and functional redundancy, meaning that the system includes several of the same modules and that a function can be achieved through various module combinations. [9] The idea is that faulty components are excluded from the system and replaced by peers.

A survey of self-assembly at macro-scale [9] (for surveys on self-assembly at micro-scale see [142, 143]) includes robotic and other systems, all composed of modules which are either self-propelled or moved by an external force. Generally, self-assembly processes depend on the characteristics of the components and their possibilities for selectively bonding with peers. Most systems are homogeneous, and in the few cases with heterogeneous modules, the distinct module types are only very few. In some cases, a seed is necessary to trigger the self-assembly process. Self-repair is an intrinsic capability of most self-assembly systems because failing modules can easily be replaced by other modules which are either already part of the system or which are acquired from the environment [9].

The concept of self-assembly is also found in irregular cellular automata where cells, driven by rules that govern not only their next state mapping but also which neighbouring cells are used to determine the next state, arrange themselves into complex shapes [93]. The desired global pattern is partitioned into sub-regions, each denoted by different neighbourhood functions. Self-assembly originates from the origin cell, and new cells attach themselves to existing cells that broadcast their cell state. By further arranging for the CA rules to cause deterministic convergent behaviour, self-assembly is regulated by convergence, i.e. once the CA has reached convergent state assembly is complete. The presence of convergent rules also makes the system self-healing; the pattern will automatically reconfigure to the convergent state in the event of an external disturbance. The method is extensible to 3D patterns and complex designs, for instance, to mechanically adjust brushes for wear and so prolong motor life.

6.3 Analysis of mechanisms

Table 6 provides an overview on the technology readiness levels of the mechanisms for self-healing used in the work cited above.

[9]The principle of degeneracy as discussed in [15] additionally includes structural and functional plasticity, meaning that a component can fulfil more than one function.

7 Self-healing MEMS

7.1 Introduction

MEMS devices are usually very cheap because they are mass fabricated. The interest of spending time and effort to make them self-healing or self-repair may thus seem without avail. However, most MEMS devices are integrated into bigger and more expensive systems, be it a mobile phone, an airbag or something else. The bigger system needs high levels of availability and reliability. After all, we do not want to bring our phone to the repair centre to get the gyroscope MEMS exchanged repeatedly, even if the cost of the gyroscope is insignificant. There is thus reasonable motivation to design MEMS that can autonomously recover from incurred damage, although the advances made so far are very scarce.

7.2 Survey

MEMS exhibit a large set of different typical failures [154, 155]. They include: stiction (static friction), wear, fracture, crystallographic defects, creep, degradation of dielectrics, environmentally induced failures, electric-related failures and packaging problems. To the best knowledge of the authors, these problems are mostly addressed through improved design, improved materials and adapted handling.

A rare exception is the self-repairing MEMS accelerometer [138], which has redundant gauging finger modules. Only four of the six modules are being used at once; when a module becomes defective, a circuit connection control mechanism replaces the faulty element by a redundant one. Thanks to this so-called *built-in self-repair* strategy, the reliability of the accelerometer is considerably improved [156, 157].

An example of a strategy to prevent MEMS failure is the use of a surface lubricant [139]: silicon oxide surfaces which rub against each other get easily damaged, and a lubricant could protect them. Three ways for applying polymer lubricants are investigated: hard coatings, vapour phase lubrication and boundary film lubrication. They all reduce friction and thus wear on silicon surfaces and thus prolong MEMS life cycles, but they are not self-healing mechanisms in the proper sense. Their relevance is due to the potential of using surface lubricants in new self-healing MEMS designs, which will require such mechanisms and ways of manipulating micromechanical systems.

7.3 Analysis of mechanisms

Table 7 provides an overview on the technology readiness levels of the mechanisms for self-repairing used in the work cited above.

Table 6 Technology readiness levels in mechanisms for self-repairing and self-healing mechanics

Mechanism	TRL	Reference	Reasoning
Mechanical self-sharpening	L9	[127, 128]	Widely used in agriculture and somewhat in households
Shape memory materials	L9	[129]	Used in industry
Functional and structural redundancy (achieved autonomously)	L4	[130]	Demonstrated in the lab

8 Self-repairing robotics

8.1 Introduction

Close to all robotic systems that currently exhibit some form of self-repair are self-reconfigurable, which means that they consist of a few to a few hundred modules that can connect to their peers and thus collectively form a bigger whole. Modules can be added, removed or exchanged at any time. Redundancy thus assures that the system as a whole may be considered as self-repairing. A few other approaches exist as well, as cited towards the end of the following paragraph. Note that the use of self-healing materials or self-healing electronics on a robot does not, according to the view taken by this article, qualify the robot as self-healing. This section looks into robotic systems that have additional characteristics of self-healing or self-repair that critically contribute to the 'robot functionality' itself.

8.2 Survey

Most robotics projects investigated at a scientific level approach the problem of self-repair through modularity and redundancy; a failing module or robot will be replaced by a functional one. Usually, the robots or modules used in these approaches are physically and functionally homogeneous, which means that they all have the same physical and functional characteristics. The robotic modules can connect with each other through some mechanism. Often, they exhibit some degree of self-adaptivity and self-organisation. A wide range of self-reconfigurable robots exists [41]. This article will only mention a small selection of them, due to their similarities in terms of system-level self-repair. Self-reconfigurable robots are sometimes also referred to as being *polymorphic* because of their ability to change their shape and move in various ways.

S-bots are individual identical mobile robots which collectively compose a *Swarm-bot* [145]. Swarm-bots are strongly inspired by social insects (ants), can autonomously assemble into 2-dimensional structures and collectively transport objects. Teamwork [162] is possible thanks to the emergence of higher-order entities composed of several robots. The robots solve a foraging task including exploration, path formation, recruitment, self-assembly from S-bots into Swarm-bots and group transport without identifying their peers (the robots are interchangeable). In certain robotic swarms, the robots perform *trophallaxis* [146], which means that they are able to share energy with their peers and thus 'repair' a robot whose battery has gone flat. Given that both robots are part of the same system, this qualifies as self-repair. Furthermore, swarm members have also been observed to imitate each other's behaviours [147], which may prove to be useful to deal with failures, either through re-training a problematic peer or by replacing it by a new one that must be familiarised with the swarm behaviour.

The successor of Swarm-bots are *Swarmanoids* [163, 164], where heterogenous mobile robots collaborate in 3D to collectively achieve a common task. The autonomous robots come in three types: *eye-bots*, *hand-bots*, and *foot-bots*. These robots are used for a multitude of different experiments in the spirit of ant-inspired collective intelligence. Both in case of the Swarm-bots and the Swarmanoids, their redundancy and self-organisation leads to self-repair at the swarm-level.

Molecubes [144] are cubic robots that form versatile shapes in a 3D lattice. Their swivelling movement along the (1,1,1) plane gives them a different type of mechanical flexibility than most other modular robots have. Each Molecube can carry numerous peers. Through evolved self-reconfiguration sequences, the robots can transform themselves from any initial shape to any target shape. Failing modules are either carried along or excluded and replaced by a peer.

The concept of *robotic sand* [165] or *robotic pebbles* [152] takes the idea of self-reconfiguration to the extreme by miniaturising the robotic modules. They connect to their peers through electromagnets and are able to take any desired body form through self-disassembly and

Table 7 Technology readiness levels in mechanisms for self-repairing MEMS

Mechanism	TRL	Reference	Reasoning
Redundancy with rewiring	L3	[138]	Critical function observed in the lab
Surface lubrication	L1	[139]	Basic principles observed

re-assembly. Due to the very small size, their distributed intelligence relies on minimal computing power. The concept of robotic sand is related to the earlier project on *smart sand* [166], where tiny MEMS components—sensors, robots, power sources, networking devices and others—interact wirelessly and execute a sensing task, as well as the follow-up project on *smart specks*.[10]

A different type of project, but also following the idea of redundancy, the solar-powered *Odysseus* [148] also executes self-repair on-the-fly: the aircraft will be able to autonomously modify its body and thus to exclude failing modules. Odysseus is a project in the scope of the *DARPA Vulture* programme, aiming at aircraft which can remain airborne over a duration of five years.

Space robotics also needs systems that are able to cope with failures. A decentralised autonomous control algorithm using parallel processing with low-performance processors was developed to control hyper-redundant manipulators [167]. The system is thus able to position itself correctly even if some joints are blocked.

Designed for remote situations like a mission to Mars, modular toy robots are able to assemble identical copies of themselves [149]. This possibility also allows for a basic form of self-repair: the robots replace failing modules with new functional ones. In this case, the modules are heterogeneous in form and function. The robot in search for modules identifies them through a barcode.

A rather amusing example of self-repair or self-reassembly is the robotic chair [150]. Every part of the chair has a robotic module inside which is able to locomote and find its neighbours after the chair has disassembled and dropped to the floor. While the usefulness of this chair is questionable, it is an impressive demonstrator of what basically any product could do if its components had computational and locomotive capabilities.

Commercially available robotic systems such as LEGO Mindstorms and others inspire children as well as adults all around the world to create systems with a multitude of characteristics, including a robot which can autonomously exchange its own wheel [151]; a video is available online.[11]

A completely different approach to self-healing consists of the robot modifying its internal model of itself to the changing state of its body, and thus to find alternative ways of maintaining its functionality. For instance, a walking robot which loses a limb will modify its gait to still be able to walk [26]. Most of this work was done through software simulations, but practical experiments were done as well.[12]

A recurrent cerebellar model articulation controller was developed for the fault-tolerant control of a two-legged robot [153]. An online fault estimation module provides approximation information in case of system failure and modelling error. The controller then stabilises the system to compensate for the effects of the failure and to achieve fault accommodation.

8.3 Analysis of mechanisms

Table 8 provides an overview on the technology readiness levels of the mechanisms for self-healing used in the work cited above.

9 Other self-healing technologies

9.1 Introduction

A few self-healing and self-repairing technologies do not seem to match the above categories or are too interdisciplinary to be associated with any of the specific areas and are thus described subsequently.

9.2 Survey

The use of fault-tolerant sensors [168] for systems to monitor themselves in combination with the availability of actuators in feedback loops enable systems to self-adapt, which is a step on the way to self-healing. Especially in industrial environments, self-adaptation is often a first achievable step. It allows systems to run more independently and to correct minor deviations automatically and autonomously. This has two consequences: on the one hand, some potential failures may be avoided by correcting deviations early, and on the other hand, machines which self-monitor and self-adapt are predestined for receiving further self-* capabilities in the future, and their operators are already used to handling their partial autonomy. For a survey and discussion, refer to [7].

A self-healing house—currently being investigated in a project named *Intelligent Safe and Secure Buildings*[13]—includes nano-polymer particles which convert into liquid when under pressure, flow into cracks, and solidify. Sensors in the walls and building structures collect data about vibrations and stress, and allow the inhabitants to be warned early in case of danger.

Naval shipboard power systems may be equipped with an automated self-healing strategy based on reconfiguration [158]. The focus is on problem diagnosis and reconfiguration of power systems on a typical surface

[10] http://www.specknet.org

[11] http://www.youtube.com/watch?v=f1iHi8Pgb90

[12] http://www.eurekalert.org/pub_releases/2006-11/uov-rht111506.php

[13] http://cordis.europa.eu/fetch?CALLER=EN_NEWS&ACTION=D&SESSION=&RCN=27445

Table 8 Technology readiness levels in mechanisms for self-repairing robotics

Mechanism	TRL	Reference	Reasoning
Homogeneous redundancy with self-reconfiguration	L6	[41, 144]	Systems have been demonstrated to function in real-world environments, but due to the frequent lack of concrete application case, many systems have not been assigned an end user environment yet
Coordination mechanisms	L3	[145–147]	Observed in the lab
Air-borne homogeneous self-reconfiguration	L2	[148]	The real state of the project remains undisclosed, but it is assumed that a functioning demonstrator will be announced as soon as it exists
Autonomous replacement of heterogeneous modules	L3	[149–152]	Observed in the lab
Trophallaxis (sharing energy)	L3	[146]	Observed in the lab
Adaptive self-modelling and failure compensation	L3	[26, 153]	Observed in the lab

combatant ship for service restoration. Faulty components are quickly isolated and alternative power supply arranged.

Airy beams are a particularly robust type of waves which can maintain their shape and structure in turbulent and scattering environments. They have been observed to exhibit self-healing behaviour [159] when reforming during propagation under perturbations. Airy beams move along parabolic trajectories, whereas their centre of gravity follows a straight line. As opposed to screening or photovoltaic spatial *solitons*, this new class of self-localised beams owes its existence to carrier diffusion effects [169].

Self-healing photovoltaics [160, 161] are at a border-line between self-healing and self-replicating: the molecules of which the photovoltaic substance consists of constantly disassemble and re-assemble. This way, the substance never remains in a static state, which would lead to the substance's degradation due to the solar radiation.

9.3 Analysis of mechanisms

Table 9 provides an overview on the technology readiness levels of the mechanisms for self-healing used in the work cited above.

10 Discussion

Having studied the work being done on self-healing across many different fields, it is interesting to identify common elements. Is there something in common to all the approaches to self-healing, or only to some?

Some strategies for self-healing and self-repair are used across more than one field, as shown in Tables 10, 11, 12 and 13 covering solutions implemented by physical effects, material changes, switching in redundant elements and smart computing strategies respectively. The tables have been arranged to show how a given strategy may be applicable across many target areas and how the solution has been inspired or is similar to various biological mechanisms. The tables can be used to find a particular failure mode in the left hand column and gradually explore self-* solutions in adjacent areas published in recent literature. Many solutions are inspired by nature [170], although to differing degrees. Figure 1 takes this basic self-healing categorization and provides a different and more graphical mapping linking the mechanism directly to the application area. For example, related concepts to intrinsic polymer healing are grouped under the strategy 'organise new resources', providing a tool for the researcher to broaden his search in a structured way.

Figure 3 represents a classification of mechanisms for self-healing and self-repair describing the type of mechanism used. Four categories were identified: reorganising available resources, organising new resources, inhibiting

Table 9 Technology readiness levels in mechanisms for diverse self-healing systems

Mechanism	TRL	Reference	Reasoning
Smart systems (sensors, intelligence, and actuators)	L9	[7], Section 2.8	Widely used
Isolation of faulty component and reconfiguration	L4	[158]	Validation in the lab
Self-focusing (airy beams)	L1	[159]	Principle observed and reported
Molecular self-reassembly	L4	[160, 161]	Component validation in the lab

Table 10 Materials related solutions

Failure	Solution	Outcome	Application areas	Nature inspiration	Remarks and limitations
Open-circuit	Healing nano-tubes	Reorganise to maintain circuit	Materials, electronics	–	Re-establishes broken conductive links
Open-circuit	Melting fuse	Reorganise to maintain circuit	Materials, electronics	–	Makes new connective links when others are broken
Mechanical rupture (surface)	Healing capsules	Inhinit rupture	Materials, electronics	–	Heals cracks; one-off
Mechanical rupture (internal)	Vascular layers	Inhibit rupture	Materials	Mammal skin	Heals cracks; repetitive action
Mechanical rupture (surface)	Sealing liquid inside	Inhibit rupture	Materials, mechanics	–	Heals cracks and holes
Mechanical shock	Intrinsic polymer healing	Regenerate using collision energy	Materials	Plants reconnecting disrupted tissue	Reconnects broken fibres
Wear:friction	Ceramic ball bearings	Inhibit wear	Materials, mechanics	–	Could be used for any component subject to friction
Wear:friction	Surface lubrication	Inhibit wear	MEMS	Mucous membranes	Similar to the surface effects of ceramic ball bearings
Wear:material breakdown	Bacterial residues	Regeration of material	Materials, mechanics, architecture	Social insects, bacteria	Theoretically, bacteria could be lead to build any kind of structure

Table 11 Redundancy-related solutions

Failure	Solution	Outcome	Application areas	Nature inspiration	Remarks and limitations
One-off failure involving a single sub-system or component	Simple redundancy	Inherent redundant sub-system/component automatically takes over; subsequent failures usually lead to overall failure	Any	–	Redundancy alone may not be enough; intelligence may be required from the system or otherwise the user
Multiple successive failures involving a single sub-system or component	Isolation of faulty component and reconfiguration	Redundant sub-system/component actively takes over; capacity for greater fault-tolerance using several redundant copies	Mechatronics, robotics	Immune system	Requires redundancy and intelligence
Multiple successive failures involving one or more sub-systems/components	Homogeneous or heterogeneous redundancy with self-reconfiguration	Strategic placement of redundant copies permits recovery from a variety of failure scenarios; prediction of remaining capacity for recovery may be difficult	Robotics	–	Requires redundancy and intelligence
Error in sub-routine that has been anticipated by formal design	Software and formal methods for redundancy	Formal redundancy/self-repair routine executed for specified failure event	Software	–	
Sub-system/component fails in a predictable manner	Autonomous functional/ structural redundancy	Inherent activation of self-restoring mechanism triggered by the failure event	Robotics, mechatronics, MEMS	–	Requires sensors, actors and some kind of intelligence or mechanism for mechanical rewiring when triggered

Table 12 Physical solutions

Failure	Solution	Outcome	Application areas	Nature inspiration	Remarks and limitations
Degradation of beam	Self-focusing (airy beams)	Self-reconstruction of beam	Physics, optics	Physical phenomenon	Only for very specific systems
Dynamic failures, changes of requirements	Chemical reactions model	Available resources reorganised	Software	Molecules	Modelling the chemical reality
Unpredictable adaptive behaviour	Rules and policies	Constrain to predictable behaviour	Software	Chemistry, sociology, economy	Abstraction of chemical principles (and others)
Degradation of photovoltaic cells	Molecular self-reassembly	Regeneration of photovoltaic cells	Chemistry, materials	Molecules	Reality of what is modelled in above mechanism; could work for self-reassembly at macro-scale as well

further failures and adapting the functional behaviour. This classification is different from the one used in Table 12 (physical solutions), Table 10 (materials related solutions), Table 11 (redundancy-related solutions) and Table 13 (computing-related solutions).

The potential of self-healing and self-repair in terms of improving current engineering is very promising. At first, it may look as if technologies become more complex and more expensive. However, once the self-* mechanisms are in place and working reliably, the product life cycles will become longer and their functioning cheaper due to reduced or eliminated maintenance and repair. For future technologies, this means that their design can be more integrated, putting the focus more on the self-* system as a whole. A detailed *cost–benefit analysis* across all self-healing technological areas is beyond the scope of this paper; however, taking an illustrative example from electronics, the use of memory row- and column-redundant cell blocks has traditionally allowed the restoration of otherwise faulty memory chips when the testing was done by external equipment, and the fixing was done by the re-patching of functional blocks for faulty ones [171]. Now, built-in self test can diagnose the faulty blocks, and built-in reconfiguration can make new working chips from failures [172], improving yield substantially and directly increasing the profit as the self-repair overhead does not require new infrastructure, only improved design. We are seeing this approach spread to logic elements [173] and foresee its drift into MEMs and mechatronic systems. Another aspect is that today, engineers increasingly pay attention to assuring that products can be serviced comfortably and cost-effectively. With self-* properties becoming more available and more reliable, this focus on serviceability will again become less important, because maintenance may mainly consist of assuring that everything is working correctly.

10.1 Challenges and limitations

One important aspect—as with all new technologies—is not to expect miracles. Self-* technologies need to be introduced step-by-step, giving the technology time to mature and the users time to adapt to the technology's new capabilities. Other limitations, from the technology's side, may consist in the currently limited possibilities for introducing biological, chemical, electronic or mechanisms to mechanic or mechatronic systems. For instance, it is currently not possible yet to build MEMS which consist of silicon that is composed of electronic stem cells for self-healing—but maybe in the future. Currently, robot grippers are not composed of intrinsically self-healing polymers, but what if we tried to realise this? Ethical issues may be raised once scientists succeed in creating artificial life forms or when they manipulate living tissues to be integrated in

Table 13 Computing-related solutions

Failure	Solution	Outcome	Application areas	Nature inspiration	Remarks and limitations
Component wear	Autonomic computing principles	Organise available limited resources	Software, robotics	–	Gives the system some autonomy, which may not be desired in safety critical systems
Component wear, creep	SW and HW condition monitoring	Protective measures to prevent failure	Any system with intelligence	–	Closer to traditional engineering than other approaches
Wear, upset events between coupled systems	Sharing of local/global diagnosis information	Advanced protective measures	Any system with intelligence	Nervous systems	Closer to traditional engineering than other approaches
Dynamic environment upsets	Agent technology	Adaptation and reorganisation	Software, robotics	Social insects	Gives the system some autonomy, which may not be desired in safety critical systems
Degradation/failure of group member	Coordination mechanisms	Adaptation to restore member, continue overall goal	Systems with intelligence	Social insects	Systems need to be composed of interacting entities
Degradation/failure of group member	Trophallaxis (sharing energy)	Redistribute available resources	Systems that interact with each other and the environment	Social insects	Systems need to be composed of interacting entities
Loss of sub-component	Adaptive self-modelling and failure compensation	Adapt to survive without sub-component	Software, robotics	Brain	Requires sophisticated software or artificial neural network
Traceable upset, pre- or intra-fault	Smart systems (sensors, intelligence and actuators)	Pre-emptive, reactionary measures	Interdisciplinary	Sociology	Requires a suitable control system and some intelligence
Transient error, hard error	Stem cells	Reorganise available resources	Software, electronics	Biological cells	Basic redundancy with differentiation as and when required
Localised hardware failure	Embryonics, POE model	Regenerate healthy logic	Electronics	Biological cells	Could be used in computing as well
Hardware ‘cell’ failure	DNA approach	Regenerate via DNA recovery	Software, electronics, materials	Biological cells	Redundancy and/or rebuilding of information and structure
Dynamic environment upsets	Artificial neural networks	Adapt system behaviour	Software, robotics	Biological neural networks	Not usable when it must be tractable how the system achieves to find a solution
Dynamic environment upsets	Stigmergy	Self-regulation of behaviour	Software, robotics, smart systems	Social insects	Could work in any system that is able to interact with its environment
Data errors	Data-restoring algorithm	Restore correct data	Software, electronics	–	Could be used in any system requiring robust control

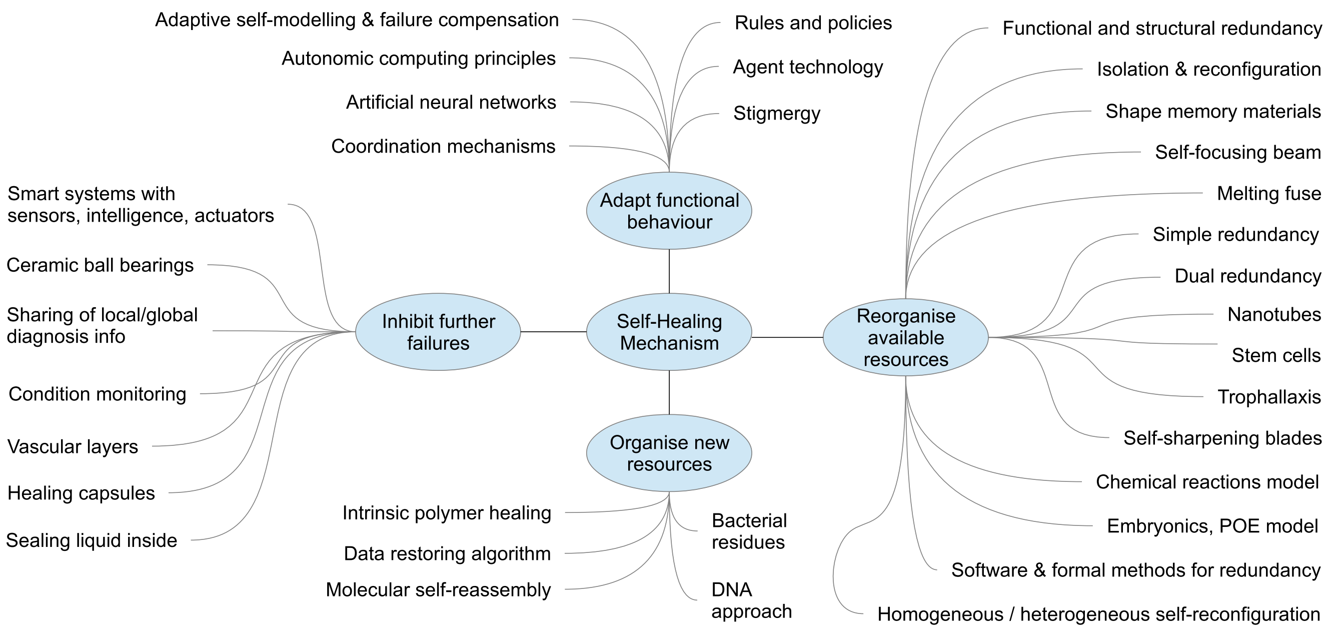

Fig. 3 Categorisation of mechanisms for self-healing and self-repair

technological systems. Such research is indeed being done (see [7] for indications on chemical artificial life, synthetic biology and related fields), but not in the area of self-healing (yet). Researchers wishing to engage in such work definitely need to consider ethical issues. In terms of safety, the work done in self-repair and self-healing does not appear to be critical from the current perspective.

To judge if self-repair and self-healing actually result in the desired advantage, the *performance* of a system facing failures with and without self-healing needs to be compared objectively. Also, its characteristics before and after self-healing need to be analysed. Does the system revert back to its original performance or is it in degraded mode? How often can self-healing be executed and how does each repeated self-healing process influence the system's characteristics?

Furthermore, it remains to be investigated how can self-healing be tested. Does it require the 'natural' (accidental) occurrence of failures, or is it possible and acceptable to work with 'fake' failures that are user-induced or simulated? The answers to these questions may vary from case to case or from application area to application area, and further investigations are definitely necessary. This is a shift between qualitative and quantitative description; how would one measure self-healing, and how many dimensions would it have?

10.2 Application of the taxonomy to an example

Only a systematic application of the taxonomy introduced in Section 2 will validate it. Adjustments and additions will certainly be necessary at a later stage. For now, the example of capsule-based self-healing materials [3, 119, 120], discussed in Section 5, serves as an illustration:

- *Failure causes:* mechanic, thermic
- *Failure types:* accidental damage, corrosion, fatigue
- *Failure characteristics:* permanent
- *Damage:* mechanical
- *Solution characteristics:* one-off, extrinsic, autonomic, without external power, without intelligence
- *Solution mechanisms:* material self-healing, intrinsic
- *Outcome:* within a few seconds to minutes, permanent, consuming the self-healing capability, restoring full functionality

10.3 Age of reviewed literature

Most articles reviewed in this survey are fairly recent and date from just a few years ago. Only the articles found in the area of mechanics are generally older, being published in the 1980s and 1990s, the oldest relevant patent being from 1785. Reasons for this may be that many modern technologies are not purely mechanical but rather mechatronic and that implementing self-* properties in purely mechanical systems is particularly challenging. The oldest publication about self-healing cited here dates back to 1934, when self-healing tires were invented. Other self-healing materials were mostly developed within the last decade. In electronics and robotics, the oldest cited articles are from 1990 and 1998, respectively, most others being fairly recent. Whereas in MEMS, there are only recent articles.

The survey attempts to focus on hardware repair and restoration of function rather than on the decorruption of the abstract representation of the hardware state, i.e. a root cause fix rather than a patch. Some review sections are longer than those of others, which is due to the availability of literature in the respective areas. Also, the amount of cited work in software and materials had to be limited due to space constraints.

10.4 Robust versus self-*

The difference between robust systems that are able to cope with failing components, and systems that are truly self-healing, is sometimes difficult to draw. An example of such a robust and intrinsically fault-tolerant system is the highly redundant coil actuator [174, 175] which is composed of a high number of individual actuators, connected both in parallel and in series. Failing components are neither removed not repaired, but the system continues to be functional, although in a degraded mode with an increasing number of failing coils. For the system to qualify as a self-healing or self-repairing system, one might expect that the failing coils are either removed from the system or restored to a functional state.

The self-repairing MEMS accelerometer [138] mentioned in Section 7 in this sense also is a border-line case; faulty gauges are not physically removed, but they are electronically disconnected. Strictly speaking, this may not qualify as proper self-repair. In any case, it does not restore the system's ability to self-repair, as the mechanism is one-off. Sometimes the difference in terminology is due to where the focus of the approach lies: avoiding that a system deviates from normalcy or bringing the system back to normalcy after a failure has occurred [16].

10.5 Self-consciousness

While some authors claim that a system needs to be 'cognisant" (i.e. aware) of what normal operation means and detect deviations [16], it seems that this may not be necessary for self-healing systems as we define them. They do not need to be 'intelligent' either, as their mechanisms may be based on chemical or mechanical properties and function automatically.

Some people argue that generally for achieving more than basic self-* capabilities, systems need to have some kind of self-consciousness[14] or self-awareness.[15] This goes into the same direction as people claiming that self-* systems necessarily include sensors to detect the need for action. There are, however, counterexamples: many self-assembly systems function due to intrinsic geometric or chemical properties, without the need for actively sensing and acting, or at least without silicon-based computing being involved.

[14]http://www.conscious-robots.com

[15]http://www.aware-project.eu

10.6 Boundaries

The distinction between normal and faulty behaviour is often fuzzy, as it may depend on circumstances and users. Also, system performance often degrades over time; without anything having drastically changed, a 'healthy' system may suddenly be considered as 'faulty'. It may therefore be necessary to gauge the degree of a malfunction and to establish criteria for whether a system needs repair or not. Sometimes, a global problem may then require local corrective actions [16].

The definition of system boundaries are crucial to determine if a system may be classified as self-* or not. While the term 'self-repair', as it is used by the *Fixit* community[16], explicitly includes the user as part of the system (fix the problem yourself instead of sending the product back to the producer), the self-* engineering community clearly intends the opposite, meaning that neither the user nor the producer is needed for the system to repair itself. The same system boundary issue applies to many self-organising systems: if a controlling agent is considered, as included in the system, it qualifies as being self-organising; otherwise, it is controlled externally and therefore not self-*.

11 Conclusion and outlook

As in the case of self-organising systems [32], self-healing systems [16] also may have to be introduced gradually. The human reluctance to change and the fear of technology going out of control are factors which must be considered carefully. Design for maintenance or repair has recently attracted increasing interest, as companies are moving from the idea of selling a product (and leaving it to the customer to care for it) towards a business model where the manufacturer also provides life cycle support. Nevertheless, research in the 1980s [176] suggested already that estimated repair time should be considered at design time.

Four decades ago, it was claimed that failures in a system are mostly mutually independent [177]; it would, however, be worth to investigate this again today, with system components being in close interactions with each other. It seems likely that failures would be found to be correlated with each other, that is, a failure in one component leads to another failure in a related component.

[16]http://www.ifixit.com/Manifesto

A paradigm shift is taking place, from the current 'test and repair' regime towards building systems possessing self-* capabilities to take care of themselves during the functional lifetime of the component or system, leading to exciting possibilities for future product design and manufacture. Although it may still take several years until first self-repairing products will reach the broad consumer market and probably decades until advanced self-healing is included in everyday products, we predict an increasingly strong economic impact. Self-repair will both reduce the maintenance and repair interventions required for a product to function and remain in good health, as well as lengthen the products overall lifetime. This is another important step towards a more sustainable manufacturing industry and a responsible use of the Earth's resources.

As self-healing and self-repairing systems will be far more robust and fault-tolerant than those based on common technology, their use may also be very interesting for the military/defence sector. Especially unmanned exploration systems such as drones and robots would benefit from the capability to recover from damage.

This article identifies strong activities in software, materials and self-reconfigurable robotics, which have been growing over the last decade. In mechanics, electronics, mechatronics and MEMS, however, self-healing and self-repair have only recently started to appear. Furthermore, this article suggests a taxonomy for self-* strategies that establish connections among failure, solution, outcome and area of application. We are currently investigating the typical failures occurring in mechatronic and electronic systems. Our next steps will be to decide which concrete mechanisms for self-healing and self-repairing mechatronics we will investigate. Besides the fundamental research on usable mechanisms, we aim at quickly building simple prototypes for proof-of-concept.

Predicting opportunities for a quick application of these concepts is challenging. As with any disruptive technology, one would consider the overlap of extreme situations, the most demanding environment protected by the most robust mechanism available and which is most easily integrated. Software and electronics in avionics are safety critical engineering systems in space applications. The move in these applications is towards finer grains of redundancy, where one does not replace the whole processor as in TMR but distributes the redundancy within the existing controllers. Reconfigurable, locally controlled, cellular architectures seem most promising where the need for software tools to map and simulate existing algorithms on new architectures is pressing. However, progress is such that true fly by wire systems without mechanical backup relying solely on replicated control to make the system safe will soon be with us. Having reviewed the existing work in self-healing systems, the ultimate goal of this article is to motivate more researchers to join forces to make advances in the very exciting field of self-healing technologies.

Acknowledgments The authors would like to thank Dr. Christian Knipprath for explaining self-healing composite materials.

References

1. Blaiszik B, Kramer S, Olugebefola S, Moore J, Sottos N, White S (2010) Annu Rev Mater Res 40(1):179
2. Kirkby E, Rule J, Michaud V, Sottos N, White S, Manson J (2008) Adv Funct Mater 18(15):2253
3. Pelletier M, Brown R, Shukla A, Bose A (2011) Self-healing concrete with a microencapsulated healing agent. Technical report. University of Rhode Island, Kingston, RI
4. Psaier H, Dustdar S (2011) Computing 91(1):43
5. Frei R, Di Marzo Serugendo G (2011) Int J Bio-Inspired Comput 3(2):123
6. Frei R, Di Marzo Serugendo G (2011) Int J Bio-Inspired Comput 3(4):199
7. Frei R, Di Marzo Serugendo G (2012) Cent Eur J Eng 2(2):164
8. Von Neumann J, Burks A (1966) Theory of self-replicating automata. University of Illinois Press, London
9. Gross R, Dorigo M (2008) Proc IEEE 96(9):1490
10. Thoma Y, Upegui A, Perez-Uribe A, Sanchez E (2007) In: International conference on architecture of computing systems (ARCS). VDE Verlag, pp 105–112
11. Pan Z, Reggia J (2010) Artif Life 16(1):39
12. Hutton T (2010) Artif Life 16(2):99
13. Kabamba P, Owens P, Ulsoy A (2011) Robotica 29(Special Issue 01):123
14. Stauffer A, Mange D, Tempesti G, Teuscher C (2001) In: International conference on evolvable systems: from biology to hardware (ICES), pp 112–127
15. Frei R, Whitacre J (2012) Degeneracy and networked buffering: principles for supporting emergent evolvability in agile manufacturing systems. J Nat Comput–Special Issue Eng Emergence 11(3):417–430. Springer, Netherlands
16. Ghosh D, Sharman R, Raghav Rao H, Upadhyaya S (2007) Decis Support Syst 42:2164
17. Avizienis A, Gilley G, Mathur F, Rennels D, Rohr J, Rubin D (1971) IEEE Trans Comput C-20(11):1312
18. Ehrig H, Ermel C, Runge O, Bucchiarone A, Pelliccione P (2010) In: Rosenblum D, Taentzer G (eds) Fundamental approaches to software engineering. LNCS Springer, Berlin, pp 139–153
19. Trumler W, Bagci F, Petzold J, Ungerer T (2005) Adv Eng Inform 19(3):243
20. Puviani M, Di Marzo Serugendo G, Frei R, Cabri G (2011) ACM Trans Auton Adapt Syst (TAAS) 7(3):33:1
21. Viroli M, Pianini D, Montagna S, Stevenson G (2012) In: Ossowski S, Lecca P, Hung C, Hong J (eds) 27th annual ACM symposium on applied computing (SAC). ACM, Riva del Garda
22. Loucks D (2011) IEEE Trans Ind Appl 47(2):688
23. Islam A, Alam M (2008) Appl Phys Lett 92(17):173504
24. Wieman S, Didkovsky L, Judge D, McMullin D (2011) AGU Fall Meeting Abstracts, p B1948
25. Pierce W (1965) Failure-tolerant computer design. Academic Press, New York
26. Bongard J, Zykov V, Lipson H (2006) Science 314:1118
27. Di Marzo Serugendo G, Fitzgerald J, Romanovsky A, Guelfi N (2007) In: ACM symposium on applied computing (SAC). ACM, Seoul, pp 566–573

28. Di Marzo Serugendo G, Fitzgerald J, Romanovsky A (2010) In: Symposium on applied computing (SAC). Sion, Switzerland, pp 457–461
29. Laprie J (2008) In: IEEE/IFIP international conference on dependable systems and networks. DSN 2008-Fast Abstracts
30. Avizienis A, Laprie J, Randell B, Landwehr C (2004) IEEE Trans Dependable Secure Comput 1(1):11
31. Sole R, Ferrer-Cancho R, Montoya J, Valverde S (2002) Complexity 8(1):20
32. Frei R (2010) Self-organisation in evolvable assembly systems. Ph.D. thesis. Department of Electrical Engineering, Faculty of Science and Technology, Universidade Nova de Lisboa, Portugal
33. Lin P, MacArthur A, Leaney J (2005) In: Australian software engineering conference, pp 88–97
34. Boettner P, Gupta M, Wu Y, Allen A (2009) In: 47th annual southeast regional conference. ACM-SE 47, pp 35:1–35:6
35. Laddaga R (1997) Self-adaptive software. Technical report 98–12. DARPA BAA
36. Seltzer M, Small C (1997) In: 6th workshop on hot topics in operating systems, pp 124–129
37. Goldberg S (1997) Mind Mach 7:515
38. Ibanez J, Anabitarte D, Azpeitia I, Barrera O, Barrutieta A, Blanco H, Echarte F (1995) In: Moran F, Moreno A, Merelo J, Chacon P (eds) Advances in artificial life, LNCS, vol 929. Springer, Berlin, pp 564–576
39. Sterritt R, Bustard D (2003) In: IEEE international conference on engineering of computer-based systems. Huntsville, p 247
40. Treuer R, Agarwal V (1993) IEEE Des Test Comput 10(2):24
41. Stoy K, Christensen D, Brandt D (2010) Self-reconfigurable robots: an introduction. MIT Press, Cambridge
42. Di Marzo Serugendo G, Fitzgerald J, Romanovsky A, Guelfi N (2006) Dependable self-organising software architectures—an approach for self-managing systems. Technical report, BBKCS-05-06. School of Computer Science and Information Systems, Birkbeck College, London
43. Di Marzo Serugendo G, Gleizes MP, Karageorgos A (2006) Informatica 30:45
44. Ganek A, Corbi T (2003) IBM Syst 42(1):5
45. Waldrop M (2003) Autonomic computing: the technology of self-management. White paper, IBM
46. Kaiser G, Parekh J, Gross P, Valetto G (2003) In: Autonomic computing workshop. Seattle, pp 22–30
47. Agarwal M, Bhat V, Liu H, Matossian V, Putty V, Schmidt C, Zhang G, Zhen L, Parashar M, Khargharia B, Hariri S (2003) In: Int. workshop on active middleware services. Los Alamitos, p 48
48. Chen L, Agrawal G (2004) In: International conference on autonomic computing. New York, pp 292–293
49. Horn P (2001) Autonomic computing: IBM perspective on the state of information technology. Technical report, IBM, T. J. Watson Labs, NY; presented at AGENDA 2001, Scottsdale
50. Hinchey M, Sterritt R (2006) Computer 39(2):107
51. E.S. Agency (2012) Strategic readiness level—the ESA science technology development route. In: European space agency. http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=37710
52. Kephart J, Chess D (2003) IEEE Comput 36(1):41
53. Di Marzo Serugendo G, Gleizes MP, Karageorgos A (eds) (2011) Self-organising software-from natural to artificial adaptation. Natural Computing Series. Springer, Berlin
54. Arora H, Mishra B, Raghu T (2006) IEEE Trans Syst Man Cybern Part A Syst Hum 36(3):487
55. Sterritt R, Bantz D (2006) IEEE Trans Syst Man Cybern Part C Appl Rev 36(3):304
56. Mueller-Schloer C, Sick B (2008) In: Wuertz R (ed) Organic computing, understanding complex systems. Springer, Berlin, pp 81–104
57. Amin M (2001) IEEE Comput Appl Power 14(1):20
58. Marik V, McFarlane D (2004) IEEE Intell Syst 22:1542–1672
59. Nichols J, Demirkan H, Goul M (2006) IEEE Trans Syst Man Cybern Part C Appl Rev 36(3):353
60. Cuesta C, Perez-Sotelo J, Ossowski S (2011) ERCIM News 2011(85):35
61. George S, Evans D, Davidson L (2002) In: Workshop on self-healing systems (WOSS). ACM , New York, pp 102–104
62. Pazat JL, Priol T, Tedeschi C (2011) ERCIM News 2011(85): 34
63. Zambonelli F, Castelli G, Ferrari L, Mamei M, Rosi A, Di Marzo G, Risoldi M, Tchao AE, Dobson G, Stevenson S, Ye J, Nardini E, Omicini A, Montagna S, Viroli M, Ferscha A, Maschek S, Wally B (2011) In: European future technologies conference and exhibition, vol 7. Procedia Computer Science, pp 197–199
64. Low K, Leow W, Ang Jr M (2006) IEEE Trans Syst Man Cybern Part C Appl Rev 36(3):315
65. Bonabeau E, Dorigo M, Théraulaz G (1999) Swarm intelligence. Oxford University Press, New York
66. IBM (2005) An architectural blueprint for autonomic computing. Technical report. June, IBM Corporation, Hawthorne
67. De Wolf T, Holvoet T (2007) In: Parashar M, Hariri S (eds) Autonomic computing: concepts, infrastructure, and applications. CRC Press, Boca Raton, pp 101–120
68. Kephart J, Das R (2007) IEEE Intern Comput 11(1):40
69. Kephart J, Walsh W (2004) In: Proceedings in the 5th IEEE international workshop on policies for distributed systems and networks (POLICY). New York, pp 3–12
70. Di Marzo Serugendo G (2009) In: International symposium on stabilization, safety, and security of distributed, systems(SSS), LNCS, vol 5873. Springer, Berlin, LNCS, pp 254–268
71. De Castro L (2006) Fundamentals of natural computing. Chapman & Hall, New York
72. Babaoglu O, Canright G, Deutsch A, Caro G, Ducatelle F, Gambardella L, Ganguly N, Jelasity M, Montemanni R, Montresor A, Urnes T (2006) ACM Trans Auton Adapt Syst 1(1):26
73. De Wolf T, Holvoet T (2007) In: Brueckner S, Hassas S, Jelasity M, Yamins D (eds) Engineering self-organising systems, LNCS, vol 4335. Springer, Berlin, pp 28–49
74. Fernandez-Marquez JL, Di Marzo Serugendo G, Montagna S, Viroli M, Arcos JL (2013) Description and composition of bio-inspired design patterns: a complete overview. J Nat Comput 12(1):43–67. Springer, Netherlands
75. Fernandez-Marquez J, Di Marzo Serurendo G, Montagna S (2011) In: International ICST conference on bio-inspired models of network, information, and computing systems (Bionetics). York
76. Perez-Sotelo J, Cuesta C, Ossowski S (2011) In: International workshop on variability, adaptation and dynamism in software systems and services (VADER 2011)— On the move federated conferences. Crete
77. Wuertz R (2008) In: Würtz R (ed) Organic computing: understanding complex systems. Springer, Berlin, pp 1–6
78. Wuertz R (ed) (2008) Organic computing. Understanding complex systems. Springer, Berlin
79. Mueller-Schloer C (2004) In: 2nd IEEE/ACM/IFIP international conference on hardware/software co-design and system synthesis CODES+ISSS. ACM, New York, pp 2–5
80. Schoeler T, Mueller-Schloer C (2005) In: International conference on architecture of computing systems (ARCS). Innsbruck, pp 139–153
81. Banâtre JP, Fradet P, Le Métayer D (2000) In: Workshop on multiset processing, LNCS, vol 2235. Springer, pp 17–44

82. Di Napoli C, Giordano M, Németh Z, Tonellotto N (2010) In: International workshop on self-organizing architectures (SOAR). ACM, New York, pp 43–50
83. Maniadakis M, Tani J (2009) Adapt Behav—Animal, Animats, Softw Agents, Robot, Adapt Syst 17(1):58
84. Dai Y, Xiang Y, Li Y, Xing G, Zhang L (2011) IEEE Trans Reliab 60(2):369
85. Mikic-Rakic M, Mehta N, Medvidovic N (2002) In: Workshop on self-healing systems (WOSS). ACM, New York, pp 49–54
86. Dobson S, Denazis S, Fernandez A, Gaiti D, Gelenbe E, Massacci F, Nixon P, Saffre F, Schmidt N, Zambonelli F (2006). ACM Trans Auton Adapt Syst 1(2):223
87. Nakano T (2011) IEEE Trans Syst Man Cybern Part C 41(5):630
88. Nelson V (1990) Fault-tolerant computing: fundamental concepts. IEEE Computer Society Press, Washington
89. Patterson D, Brown A, Broadwell P, Candea G, Chen M, Cutler J, Enriquez P, Fox A, Kiciman E, Merzbacher M, Oppenheimer D, Sastry N, Tetzlaff W, Traupman J, Treuhaft N (2002) Recovery-oriented computing (ROC): motivation, definition, techniques, and case studies. Technical report, UCB//CSD-02-1175. Computer Science Division, UC Berkeley
90. Dijkstra E (1974) Commun ACM 17(11):643
91. Linger R, Mead N, Lipson H (1998) In: International conference on requirements engineering (ICRE). Colorado Springs, pp 6–10
92. Ghosh S (2006) Distributed systems: an algorithmic approach. Chapman & Hall, Boca Raton
93. Jones D, McWilliam R, Purvis A (2010) In: Lazinica A (ed) New advanced technologies. InTech, Rijeka, pp 373–394
94. Jones D, McWilliam R, Purvis A (2010). In: Lazinica A (ed) New advanced technologies. InTech, Rijeka, pp 161–176
95. Samie M, Dragffy G, Pipe T (2009) In: Genetic and evolutionary computation conference (GECCO). Montreal, Quebec, pp 2143–2148
96. Caruso M, Schelkopf S, Jackson A, Landry A, Braun P, Moore J (2009) J Mater Chem 19:6093
97. Kelion L (2011) BBC news technology. December 22
98. Blaiszik B, Kramer S, Grady M, McIlroy D, Moore J, Sottos N, White S (2011) Adv Mater October
99. Geppert L (2004). IEEE Spectrum.
100. Rizzolo R, Foote T, Crafts J, Grosch D, Leung T, Lund D, Mechtly B, Robbins B, Slegel T, Tremblay M, Wiedemeier G (2007) IBM J Res Dev 51(1.2):65
101. Szasz C, Chindris V (2010) In: IEEE international conference on automation quality and testing robotics (AQTR), vol 2, pp 1–6
102. Shen V, Shen F (2002) IEEE Trans Syst Man Cybern Syst Hum 32(1):149
103. Kabuka M, Harjadi S, Younis A (1990) IEEE Trans Syst Man Cybern 20(2)
104. Biteus J, Frisk E, Nyberg M (2011) IEEE Trans Syst Man Cybern Syst Hum 41(6):1262
105. Cheatham J, Emmert J, Baumgart S (2006) ACM Trans Des Autom Electron Syst 11(2):501
106. Habinc S (2002) Functional triple modular redundancy (FTMR) VHDL design methodology for redundancy in combinatorial and sequential logic design and assessment report. In: Gaisler research. http://www.gaisler.com/doc/fpga_003_01-0-2.pdf
107. Kawanaka M, Tsunoyama M, Naito S (1994) Syst Comput Japan 25(6):1
108. Mange D, Stauffer A, Tempesti G (1998) In: Wirsing M, Banatre J, Hoelzl M, Rauschmayer A (eds) Evolvable systems: from biology to hardware, LNCS, vol 1478. Springer, Berlin, pp 185–195
109. Mange D, Sipper M, Stauffer A, Tempesti G (2000) Proc IEEE 88(4):516
110. Thoma Y, Tempesti G, Sanchez E (2004) Biosystems 76 (1–3):191
111. Rivers J, Gupta M, Shin J, Kudva P, Bose P (2011) IEEE Trans Comput Aided Des Integr Circ Syst 30(7):945
112. Reick K, Sanda P, Swaney S, Kellington J, Mack M, Floyd M, Henderson D (2008) Micro IEEE 28(2):30
113. Henderson D, Mitchell J, Ahrens G (2012) POWER7 System RAS; key aspects of power systems reliability, availability, and serviceability. http://www-03.ibm.com/systems/power/hardware/whitepapers/ras7.html
114. IBM Corporation (1999) IBM chipkill memory. http://www.ece.umd.edu/courses/enee759h.S2003/references/chipkill_white_paper.pdf
115. Quach N (2000) IEEE Micro 20(5):61
116. Agny R, DeLano E, Kumar M, Nachimuthu M, Shiveley R (2010) The intel itanium processor 9300 series. http://download.intel.com/products/processor/itanium/323247.pdf
117. Patel A, Prakash K (2010) Fault tolerant features of modern processors a case study. Technical report. University of Wisconsin-Madison
118. Windsor H (1934) Pop Mech. December 872
119. Cho S, White S, Braun P (2009) Adv Mater 21(6):645
120. Hager M, Greil P, Leyens C, van der Zwaag S, Schubert U (2010) Adv Mater 22(47):5424
121. Norris C, Bond I, Trask R (2011) Compos Sci Technol 71(6):847
122. Williams H, Trask R, Bond I (2008) Compos Sci Technol 68(15–16):3171
123. Pang J, Bond I (2005) Compos A Appl Sci Manuf 36(2):183
124. Gabelli A, Kahlman L (1999). Evol Bus Technol 3
125. Jonkers H, Thijssen A, Muyzer G, Copuroglu O, Schlangen E (2010) Ecol Eng 36(2):230
126. Rasmussen S, Bedau M, Chen L, Deamer D, Krakauer D, Packard N, Stadler P (2008) Protocells, bridging nonliving and living matter. MIT Press, Cambridge
127. Floud R, Johnson P (eds) (2004) The Cambridge economic history of modern britain, vol 1, Industrialisation. Cambridge University Press, Cambridge, pp 17001860
128. Zhuk Y (2007) Int J Microstruct Mater Prop 2(1):90
129. Otsuka K, Wayman C (1999) Shape memory materials. Cambridge University Press, Cambridge
130. Murata S, Yoshida E, Kurokawa H, Tomita K, Kokaji S (2001) Auton Robot 10:7
131. Henderson M (2009) The Times. March 13
132. Knipprath C, McCombe G, Trask R, Bond I (2011) In: 3rd International conference on self-healing materials. Bath
133. Wool R (2008) Soft Matter 4:400
134. Metzner M (2009) Fraunhofer Res News 8:2
135. Adee S (2008) IEEE Spectrum November
136. Hanczyc M (2011) Architect Des 81(2):26
137. Tee B, Wang C, Allen R, Bao Z (2012) Nat Nanotechnol online first
138. Xiong X, Wu Y, Jone W (2005) In: Proceedings of the 20th IEEE international symposium on defect and fault tolerance in VLSI systems, pp 21–32
139. Hsiao E, Bradley L, Kim S (2011) Tribol Lett 41:33
140. Hao D (1981) Trans Chinese Soc Agric Mach 1
141. Emert R, Piria I, Toth A (1987) Agrotehnicar 33(1):8
142. Boncheva M, Bruzewicz D, Whitesides G (2003) Pure Appl Chem 75(5):621
143. Phili D, Stoddart J (1996) Appl Chem Int Ed 35(11):1154
144. Zykov V, Mytilinaios E, Desnoyer M, Lipson H (2007) IEEE Trans Robot 23(2):308
145. Gross R, Bonani M, Mondada F, Dorigo M (2006) IEEE Trans Robot 22(6):1115

146. Kubo M, Melhuish C (2004) In: Towards autonomous robotic systems (TAROS). Colchester, pp 77–84
147. Erbas M, Winfield A (2011) In: European conference on artificial life (ECAL). Paris, pp 218–225
148. McKeegan N (2008) http://www.gizmag.com. May 1, p 9261
149. Park W, Albright D, Eddleston C, Won W, Lee K, Chirikjian G (2004) In: RoboSphere. NASA Ames Research Center, Moffett Field
150. Ju A (2006) Cornell chronicle online. October 20
151. Stirb L, Marian Z, Oltean M (2010) Studia Univ. Babes-Bolyai. Informatica LV(1):41
152. Gilpin K, Knaian A, Rus D In: Robotics and automation (ICRA), 2010 IEEE international conference on (2010), pp 2485–2492
153. Lin C, Chen C (2007) IEEE Trans Syst Man Cybern B 37(1): 110
154. Li Y, Jiang Z (2008) In: Misra K (ed) Handbook of performability engineering. Springer, London, pp 953–966
155. Huang Y, Vasan A, Doraiswami R, Osterman M, Pecht M (2012) IEEE Trans Device Mater Reliab 12(2):482
156. Xiong X, Wu Y, Jone W (2006) In: IEEE international symposium on defect and fault tolerance in VLSI Systems (DFT), pp 236–244
157. Xiong X, Wu Y, Jone W (2008) In: IEEE international symposium on defect and fault tolerance in VLSI systems (DFT), pp 314–322
158. Butler-Purry K, Sarma N (2004) IEEE Trans Power Syst 19(2):754
159. Broky J, Siviloglou G, Dogariu A, Christodoulides D (2008) Opt Exp 16(17):12880
160. Dume B (2010) http://www.PhysicsWorld.com. September 8, pp 43690
161. Ham M, Choi J, Boghossian A, Jeng E, Graff R, Heller D, Chang A, Mattis A, Bayburt T, Grinkova Y, Zeiger A, Van Vliet K, Hobbie E, Sligar S, Wraight C, Strano M (2010) Nat Chem 2(11):929
162. Nouyan S, Gross R, Bonani M, Mondada F, Dorigo M (2008) Teamwork in self-organised robot colonies. Technical report. TR/IRIDIA/2008-005. Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle. Université Libre de Bruxelles, Belgium
163. Dorigo M et al (2011) Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. Technical report IRIDIA
164. Ducatelle F, Di Caro G, Gambardella L (2010) In: Genetic and evolutionary computation conference (GECCO). Portland, Oregon
165. Hardesty L (2012) MIT news. April 2
166. Kahn JM, Katz RH, Pister KSJ (1999) In: 5th annual ACM/IEEE International conference on mobile computing an networking (MobiCom). ACM, New York, pp 271278
167. Kimura S, Takahashi M, Okuyama T, Tsuchiya S, Suzuki Y (1998) IEEE Trans Syst Man Cybern Syst Hum 28(4):521
168. Iyengar S, Prasad L (1995) IEEE Trans Syst Man Cybern 25(4):643
169. Jia S, Lee J, Fleischer J, Siviloglou G, Christodoulides D (2010) Phys Rev Lett 253904:104
170. Frei R, Barata J (2010) Int J Bio-inspired Comput 2(3/4):258
171. McCluskey E (1986) Logic design principles: with emphasis on testable semicustom circuits. Prentice Hall, Upper Saddle River
172. McCluskey E (1985) IEEE Des Test Comput 2(2):21
173. Benso A, Chiusano S, Di Natale G, Prinetto P (2002) IEEE Trans Reliab 51(1):123
174. Davies J, Steffen T, Dixon R, Goodall R, Zolotas A, Pearson J (2008) In: International federation of automatic control (IFAC) world congress. Seoul, pp 3228–3233
175. Davies J, Tsunashima H, Goodall R, Dixon R, Steffen T (2009) In: IFAC symposium on fault detection, supervision and safety of technical processes (SafeProcess). Barcelona, pp 1228–1233
176. Wohl J (1982) IEEE Trans Syst Man Cybern 12(3):241
177. Neuman C, Bonhomme N (1974) IEEE Trans Syst Man Cybern 4(1):58