Fault Protection for Space Missions

02228 - Fault-Tolerant Systems
Autumn 2013

Cosmin Stefan-Dobrin
Technical University of Denmark
s121038

Rafaela Voiculescu
Technical University of Denmark
s120931

Abstract

Space has been a topic of high interest in the past century and the space exploration missions have aimed to help new discoveries. The spacecrafts used for this purpose have been designed for the unknown, high-risk nature of the missions and, thus, fault protection has been an important necessity. This paper offers a brief analysis of general approaches and specific implementations for fault management architectures used in space exploration missions.

Contents

1 Introduction

For more than 50 years the human kind has been sending ships into the outer space in order to look for answers to centuries-old questions about what is beyond our atmosphere. And we have managed to explore and understand more and more of our Solar System and the space beyond. The successful missions have been the result of the reliable architectural design, implementation and testing done by the engineers working on them.

The extremely high costs, in both money and time, and the high risks of such missions require an extremely high level of reliability that can guarantee a very high chance of success. Thus, fault protection of the systems involved in these missions is of utter importance.

Spacecrafts are extremely complex and their design is usually very specific to the particular mission objective. Furthermore, space missions are meant to have a duration of months or years and sustain operation in extreme environment and under harsh conditions. This, in addition to financial and temporal limitations, make the design of fault protection mechanisms a challenging task for the engineers. When taking into consideration the deep space exploration missions (i.e missions meant to last for multiple years and have the purpose of exploring the Solar System and beyond), an even higher level of reliability is required and various techniques to assure fault prevention, detection, containment and recovery need to be employed to assure mission success.

This paper aims to take a look into the fault protection approaches used in the spacecrafts domain, with a particular focus on deep space explorations missions. A general look at the characteristics of such missions is first taken, followed by a brief introduction to the general approaches followed when designing and implementing fault protection for spacecrafts. Then, a series of previous space missions (both successful and unsuccessful) are analyzed, with a focus on the fault management.

2 Space missions characteristics and requirements

The engineers working on the construction of spacecrafts have to face a multitude of requirements which are specific to this domain and cannot be found in other, more 'traditional' fields. Thus, in order to properly understand the specifics of fault-tolerant approaches used for space-related applications, this chapter will briefly describe these requirements and the characteristics of space missions.

One of the most important characteristics of this domain is that it deals with a considerable amount of uncertainty. While we have a strong understanding of the Earth environment (and even surrounding space), moving farther away from our home planet brings any ship in an environment where unexpected factors can influence the mission success. Thus, the ships must be prepared to respond to conditions for which they might not have been initially designed.

Another important factor is the harshness of the environment and the extreme conditions present in space, which may have dangerous effects. A few examples include[8]:

- extremely low or extremely high temperatures (usually due to incoming solar radiation, reflected solar energy or friction when entering a planetary atmosphere) can affect various instruments or components
- ionized gas or plasma can charge the surface of a spaceship and disrupt the operation of electrical instruments
- meteoroids and orbital debris may cause damage and affect the structural integrity of ships if hit[1]
- ionizing radiation consists of high-energy particles that can 'travel through spacecraft material and deposit kinetic energy'
- harmful/toxic substances in the atmosphere of other planets may also cause corrosion and affect the proper function of components

Spacecrafts which are traveling far away from Earth also face other issues due to operating with limited ground contact (mostly because of large distances to our planet). According to [9], these can include:

- extended periods with no planned contact (1 to 4 weeks)
- planned contact periods may be short (1 to 2 hours)
- ground may not show for planned contacts (5% to 10%)
- large one-way light travel times, thus high communication latency (minutes to hours)
- low down-link and up-link data rates (10 to 40 bps)

For example, light travels 2h and 50 minutes to reach Saturn from Earth so autonomous operation was required for probes such as Voyager when at such distances, as no 'live' commands could be received from Earth if any situation had occurred.

Two other aspects that are characteristic to long-term space missions is that there are no humans on-board them (unmanned) and they need to survive without any maintenance during the entire

[1]The speed with which a meteoroid or debris are hit is very important when it comes to the effects the encounter can have.

mission time, which can last for more than a couple of years[2]. No spare parts are available and no humans can directly intervene to fix any issues that might appear, so the ships have to deal with this using other methods (e.g. redundancy).

Lastly, we can add a couple of other characteristics of spacecrafts:

- power supply is usually limited and the generated energy needs to be divided to all the instruments and components of the ship
- costs need to be kept down as all components are extremely expensive to produce
- space and weight usually need to be restricted and their use optimized

As we have seen, there are a multitude of characteristics of space missions (and long-term space missions in particular) and a series of specific requirements, all of which need to be taken into account when designing spacecrafts. Thus, all of these have a direct influence on the fault protection mechanisms used.

[2]For example, both Voyager 1 and Voyager 2 have been active for more than 36 years.

3 Generic Spaceship Architecture

One of the most important characteristics of this domain is that spacecraft missions have extremely diverse purposes, so the design of each is specific and unique. The spacecrafts that go outside Earth's atmosphere vary from simple ones (e.g. satellites) to very complicated ones (e.g. deep-space missions). However, the principles on which their designs are based are common. Virtually all the spacecrafts have similar requirements regarding aspects such as power, telecommunications, propulsion etc. and share a general structural construction. Thus, in order to aid the understanding of the fault-tolerant approaches used for space missions, a generic architecture for spacecrafts is briefly described below.

Typically, spaceships have an architecture that contains, at minimum, 6 subsystems[18]. These are *propulsion*, *communications*, *power*, *attitude control*, *thermal control* and *command & control*, as seen in Figure 1.

Propulsion This subsystem is in charge of accelerating and stabilizing the ship and controlling its orientation. Multiple methods can be used for propulsion, but the most common ones are gas[3] or electric[4] based.

Communication This subsystem handles the communication between the spaceship and the Earth control center. The spacecraft employs transmitters and receivers to send and receive messages which travel through space as radio waves. The communication going from a spaceship to ground is called down-link and the one going from ground to a spaceship is called up-link. Unlike the other modules, the Communications subsystem interfaces not only with components on the ship, but also with external entities.

Power All the components of a spacecraft require electrical power for powering the various functions. This subsystem is in charge of the generation, storage and distribution of electrical energy in the spaceship. Various generation methods are employed[5], the most common one involving solar cells. The power is passed through a power distribution unit over an electrical bus to other spacecraft components. Batteries are used to provide electrical power during periods when primary power is not available.

Attitude Control This subsystem handles the correct orientation and stability of the spacecraft by computing the response to external torques and forces. Its main components are sensors and actuators and it uses specialized algorithms for control.

Thermal Control As temperatures in the surrounding environment may vary by hundreds of degrees, the ships require this subsystem to assure that all the components have a proper temperature. It can be passive, such as insulation, special materials etc., or active, employing electrical heaters to manipulate the temperature of the components.

Command/Control All the aspects related to the spaceship's control are handled by this subsystem. Its main tasks include: receives commands from the communications subsystem; validates, decodes and distributes the commands to the appropriate spacecraft components; receives internal and scientific data from the other spacecraft subsystems and components and packages the data for storage on a data recorder or transmission to

[3] Chemical or pressurized gas is forced at a high speed through a nozzle. This is still the most common method.

[4] Commonly ion thrusters or Hall effect thrusters are used.

[5] For ships near the Sun, solar panels are frequently used to generate electrical power. In more distant locations, for example Jupiter, a Radioisotope Thermoelectric Generator (RTG) might be employed to generate electrical power.

the ground. Also, this subsystem is in charge of system health-monitoring and recovery from failures.

Besides these main components, most of the deep space exploration ships have a **payload of scientific instruments** (usually simply referred to as *Payload*), that are used to gather various types of data from the surrounding environment. A few examples of scientific instruments used are magnetometers, energetic particle detectors, temperature sensors, spectrometers, radiation sensors, high resolution cameras.

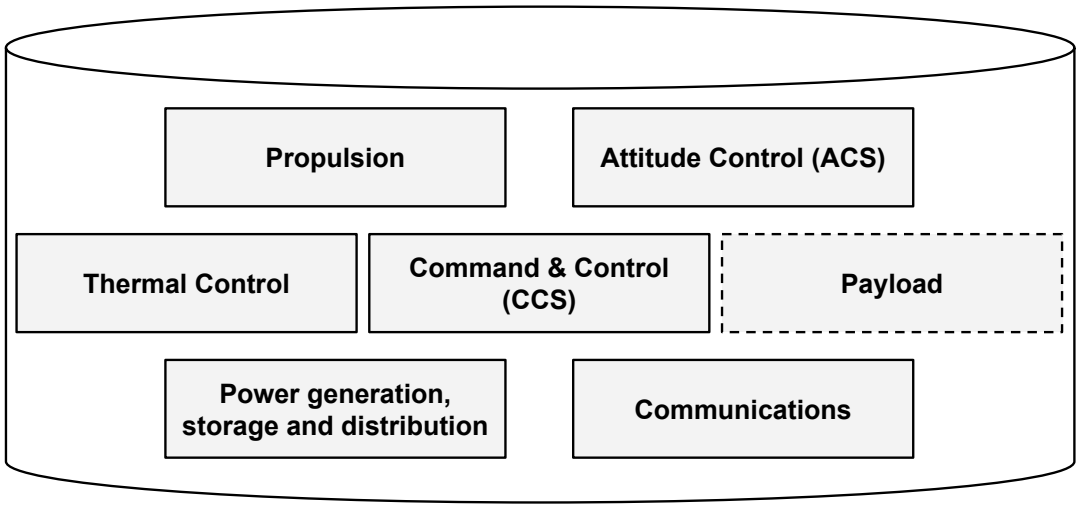

Figure 1: Components of Generic Spacecraft

4 General approach to fault protection in spacecrafts

The characteristics of space exploration missions have always required innovative fault management (FM) strategies[6] in order to achieve mission success. The used FM strategies and implementations vary significantly from mission to mission, but there are a set of general characteristics that influence the design of the FM. Engineers from NASA have been working on a Fault Management Handbook with the purpose of being *'a guidance document to provide guidelines and recommendations for defining, developing, analyzing, evaluating, testing, and operating the Fault Management (FM) element of flight systems'*[14].

According to [14], a space missions FM engineer has the role of analysing mission attributes and characteristics and defining FM requirements, as well as developing hardware and software architectures that provide fault protection and assure the success of missions. This process is illustrated in Figure 2.

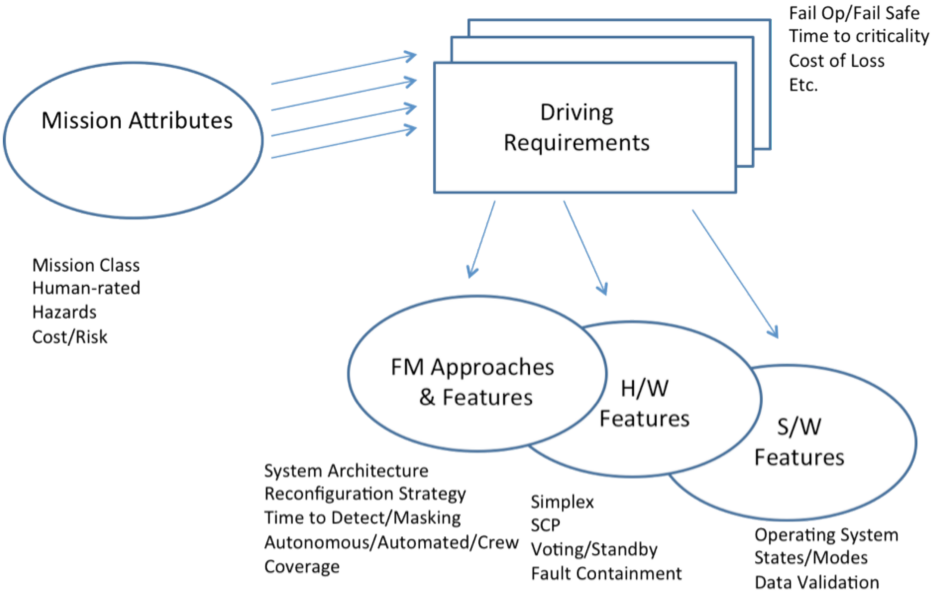

Figure 2: Fault Management design process. *Source: NASA Fault Tolerance Handbook, draft 2, 2012*

The first step during development of the FM for a space mission is a clear identification of mission goals and of functions and resources that are critical and need to be protected[7]. Subsequently, the FM engineers need to perform an analysis of the system architecture to identify the mission characteristics which are definitory for the fault protection functions. Fault management will need to be implemented throughout the system, however they need to take into account restrictions enforced by the mission characteristics. For example, in the case of deep space exploration missions, which have delayed and infrequent contact with the mission command on Earth, or during time critical steps of missions (e.g. planetary atmospheric entry), the **response latency** of fault management is critical. According to the NASA Fault Management Handbook [14], response latency is *'the time from the occurrence of a fault to the correction of the failure condition'* and is *'built from the various mission and system characteristics that impact the execution*

[6]The term *Fault Management* is often used interchangeably with the term *Fault Protection* in NASA documentation.

[7]These include functions, resources, events which are needed to achieve the mission's goals. E.g.: successful launch, scientific payload, collected data etc.

of each of the core FM functions'. To assure system reliability, the FM response needs to either clear or contain the effects of a failure before the failure is propagated to other components or before system issues occur.

Over the years, the technologies used for fault protection have been significantly improved, but the fundamental principals of spacecraft design have suffered little changes. For example, the fault tolerance design at the NASA Jet Propulsion Laboratory is based on the following **fault management principles** (quoted from [9]):

1. Respond only to unacceptable conditions
2. Avoid hair triggers and re-triggering
3. Tolerate false alarms
4. Make parameters commandable
5. Corroborate before severe responses
6. Ensure commandability and long term safety
7. Preserve consumables and critical data
8. Log events and actions

The next step followed by the FM engineers is to design and implement software and hardware solutions to assure the fault protection of a spacecraft. A series of techniques can be used in different situations and at different moments of time (different stages). In the presentation of Fault Management at the Jet Propulsion Lab[9], John Day and Michel Ingham detail a hierarchy of approaches and give examples of a series of strategies that may be used to assure fault protection. The hierarchy can be seen in Figure 3. A first strategy that should be followed is to to avoid faults altogether, through **fault avoidance** techniques. However, for cases when faults do occur, **fault tolerance** needs to be in place, to assure that any faults have minimal effects on the ship and do not cause a mission failure. From this category, for example, **Fault Masking**, **Gradual Degradation** and **Fault Detection, Isolation and Recovery** (FDIR) are some general strategies that can be employed.

The design process is one where, mainly because of costs and time restrictions, usually engineers start from already developed and previously used techniques and technologies and improve them, increasing their safety, reliability and accuracy. For example, the most important strategy used for FM is **Redundancy**, since the first space missions. This is usually implemented in multiple ways[10]:

- Block Redundancy - which ensures that a failure can be solved through the use of parallel elements
- Functional Redundancy - which permits the handling of a failure in various ways
- Cooperative Redundancy - which permits the division of a system function in more elements. This helps by allowing the function to be able to succeed despite the probability of having one of its elements fail
- Cross Strapping - which permits the system to handle multiple failures that occur in the system.

In the following chapter, we will go into the details of some of the FM implementations for former NASA deep space exploration missions. However, before that, a look at the typical execution

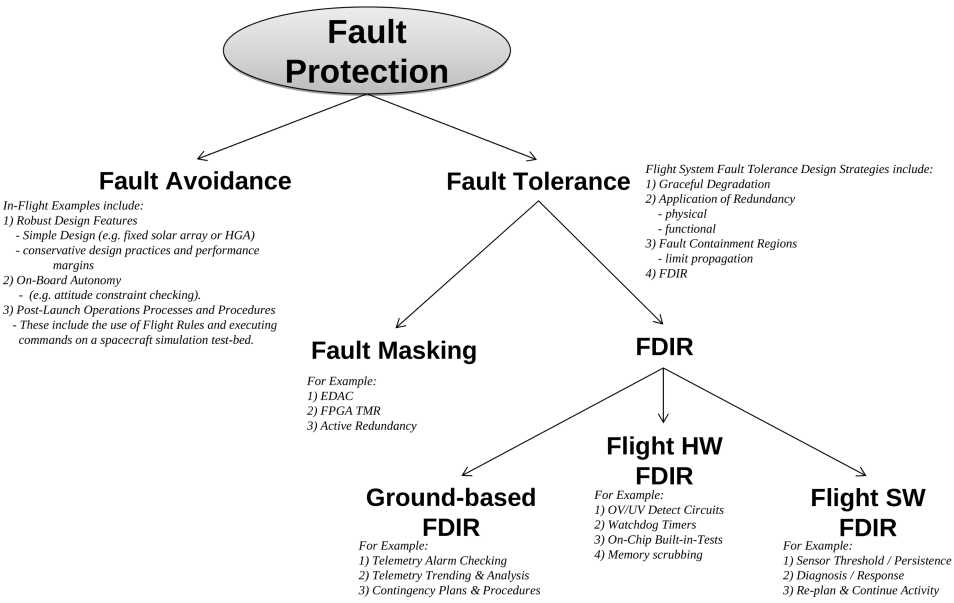

Figure 3: Fault protection general hierarchy. *Source: J. Day and M. Ingham, Fault management at JPL: past, present and future, 2011*

architecture of spacecrafts is beneficial. Figure 4 depicts such a generic architecture, as seen in [9]. As it can be observed, command sequences[8] are being executed by a *nominal sequencing engine*. Simultaneously, fault protection software is run and is always prepared to take over the control and execute fault protection actions, if the behaviours indicate the presence of a fault. Typically, the fault protection is handled by the Command and Control Subsystem.

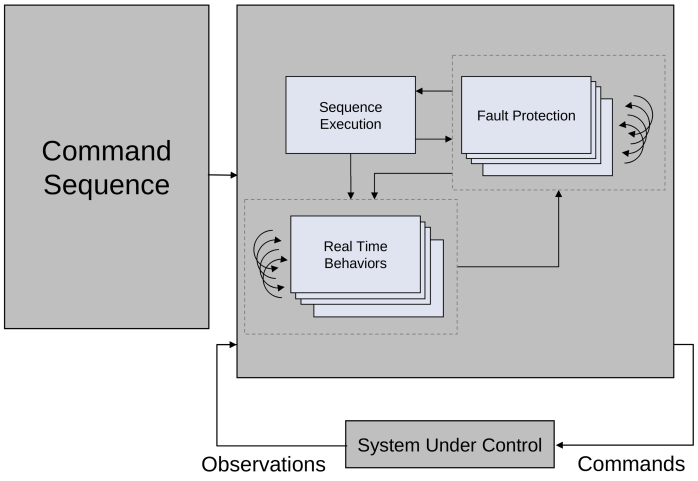

Figure 4: Typical Spacecraft Execution Architecture. *Source: J. Day and M. Ingham, Fault management at JPL: past, present and future, 2011*

The design and implementation of fault management needs to be followed by Verification and Validation, to prove that the system complies with the system requirements and that the response to failures is the intended one. Besides the mission specific requirements, all space missions have to respect some general safety requirements. Mandatory ones include the **NASA**

[8] Low-level commands or macros defining actions that need to be executed for the mission completion.

General Safety Program Requirements (NPR-8715.3) and the **Eastern and Western Range Safety Requirements** (EWR 127.1)[9][10]. In addition, a multitude of other documents include recommendations for requirements related to fault protection in the case of spacecrafts. Some examples include: Rules for the Design, Development, Verification, and Operation of Flight Systems (GSFC-STD-1000E), NASA Systems Engineering Processes and Requirements (NPR 7123.1A), Risk Classification for NASA Payloads (NPR 8705.4) etc.

[9]EWR-127.1 represents the specification of safety requirements for the launch site. It is being replaced with the **Air Force Space Command Manual Range Safety User Requirements** (AFSCPCMAN 91-710).

5 Fault Tolerance in Specific Space Missions

5.1 Voyager Missions

Voyager 1 and 2 [5][6] are 2 NASA missions created in order to explore the outer Solar System [10] The missions were possible due to a rare planetary alignment. Voyager 2 targeted Jupiter, Saturn, Uranus and Neptune, while Voyager 1 targeted Jupiter and Saturn. After visiting these planets, they both continued on to chart the far edges of our Solar System.

There are quite a few works which take a closer look at the Voyager missions. These missions have brought humanity an unimaginable boost of knowledge about our Solar System and even beyond. R. L. Heacock presents, in one of his papers [12], various engineering aspects of the two spacecrafts.

The two spacecrafts have been launched into space on 20th of August, respectively 5th of September in 1977. Their trajectories were controlled through the use of the gravity field and orbital velocity of the planets. This was the key factor which allowed the encounter with multiple planets. The way in which the planets where aligned at that time has allowed for each spacecraft to pass by more planets and it has also helped by reducing the time needed to reach these planets (e.g. from 30 years normally for a flight to Neptune, the travel time was reduced to only 12 years).

Before the mission started, hundreds of computer simulations for the trajectories were carried out. Also, to assure fault protection, research was made in order to design a self test and repair (STAR) system for the spacecrafts, as well as an on-board computer access telemetry system (CATS). The latter allowed STAR to monitor the on-board performances or failures of the spaceship and, in case it would be needed, to take measures for remedying the problem.

The spacecrafts had special needs that defined parts of their appearance. For example, Voyager needed a large diameter antenna since the communication needed to be possible despite the very long distance to Earth. It also had an external surface which needed to be adequate for the temperatures and radiation it was facing. Internally generated heat was enclosed with the help of special blankets which were insulating. Also, most of the subsystems had replacement heaters.

The attitude control system (ACS) was used in order to keep the antenna in the right position needed for communication. The ACS was also in charge of keeping or modifying the spacecraft's trajectory. A block diagram of the ACS can be seen in Figure 5.

The spacecrafts kept track of their position by using the Sun and a star (Canopus). The sensors that keep track of both the Sun and Canopus are block redundant in order to ensure that the spacecrafts will be able to compute their position and that complete failure in doing so would be unlikely. There is, additionally, a periodic calibration of the Sun sensor. This allows keeping the accuracy level needed in order for the spacecraft to position the scan platform so that it accurately points the remote sensing instruments. In case the sensor that keeps track of the Sun position is not working and the solar panels cannot be pointed in the right direction, the spacecraft has an inbuilt solar independent power system given by the radio-isotope thermoelectric generators. This ensures that even if the spacecraft cannot use the Sun as a power supply, it can still maintain functionality.

[10] http://science.nasa.gov/planetary-science/focus-areas/outer-solar-system/

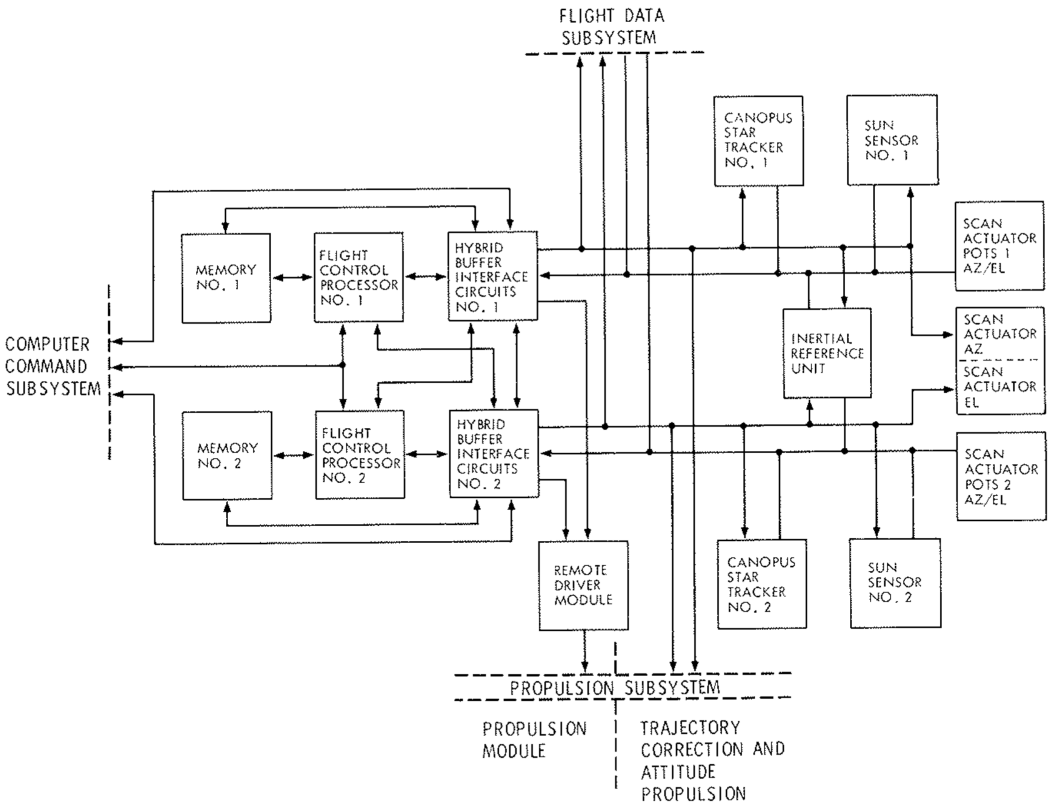

Figure 5: The block diagram of Attitude Control Subsystem on Voyager. *Source: R.L. Heacock, The Voyager Spacecraft, 1980*

Great amount of testing was also done in order to ensure that, in case of an accident at launch time, the radioactive material which existed on board the spacecraft would be contained and would not represent any danger. This is possible because the shells that contain the plutonium spheres were wrapped in an impregnated graphite yarn which provided a good energy absorption in the case of an accident.

Since in the later stages of the mission, the spacecraft needed to wait for almost three hours for a response from the ground base, it was obvious from the beginning that it would need to have at least some operations done autonomously. However, control parameters and even control laws could be re-programmed in flight in case it is necessary. This has allowed solving several in-flight problems that have occurred throughout the years [13] simply by sending software patches.

The computer and control subsystem fault correction and detection routines incorporate internal error and power levels checks, received command and transmitted signal loss, spacecrafts maneuver faults, attitude control subsystem failures and many others. For example, the command loss routine verifies if any commands have been given to the spaceship. A timer is used in order to make checks to see if commands have reached the spaceship components or not. If a command has been given and received, then the timer is reset. In case the command is either not received or was not given when expected, the command loss routine stars, meaning that a series of re-configurations are initiated.

A description of the error detection mechanism for the attitude control subsystem is given in [18]. It consists of:

- failure of receiving a 'I am healthy' report every 2 seconds by the command control subsystem
- loss of references to Canopus or the Sun
- failure of re-writing memory every 10 hours
- taking more time to adjust trajectory than expected
- gyro failure
- power supply failure
- parity error on commands received from the command and control subsystem
- unresponsiveness to commands
- incorrect sequence of commands

A high redundancy of subsystems and hardware was used when designing the two spacecrafts. This has proven to be a good safety measure since, for example, the receiver for Voyager 2 has malfunctioned and almost could have caused the failure of the mission. However, the problem has been identified fast and the spacecraft has been functioning on the back-up receiver since 1978. The attitude control subsystem is made of redundant computers (one of them is a standby which is ready to take the other's place if needed). The command and control subsystem also has redundancy incorporated in it. However, in this case, the redundant computer monitors the functional computer.

Voyager 1 and 2 have achieved their goals and have surpassed them. They have made unexpected discoveries (e.g. at least nine active volcanoes on Io[11] [12]) and are currently the most distant human made objects. Voyager 1 has already left our solar system in 2012[12] and is currently traveling through interstellar space.

5.2 Cassini-Huygens

The Cassini-Huygens [1] mission was a NASA-ESA joint mission aimed to study Saturn, its moons and its rings. The two names that form the name of the spacecraft are present because Cassini represents the NASA-supplied spacecraft while Huygens represents the lander designed by the European Space Agency. Cassini was the first spacecraft to actually orbit Saturn [1].

An interesting description of the fault tolerant design of both the Cassini spacecraft and the Huygens probe is given in D.P Siewiorek and P. Narasimhan's article [18]. The Cassini orbiter was also developed with possibility of self-recovery in mind, since it would take awhile before any answer could be transmitted back from Earth.

The Huygens probe [11] has landed on Titan on 14th January 2005. It's purpose was to make a variety of measurements. It was formed out of various subsystems. Among them we can find the ones that are in charge of overseeing the mechanical and thermal components, probe-relay data, command and data management and electrical power. The states of each of its subsystems are influenced by the mode in which the probe is at a particular moment of the mission. The modes it can enter are: cruise, coast, entry and decent. Huygens' mission aimed to last for the 2.5 hours it took for it to descend in the atmosphere of Titan.

[11] Io is one of the Jupiter moons
[12] http://www.space.com/22729-voyager-1-spacecraft-interstellar-space.html

The Huygens' fault tolerant design was focused on autonomy. This was needed because the probe could not have been commanded after its detach from the spacecraft, due to distance from Earth. The probe had a completely redundant electrical architecture (most specifically it used block redundancy) and the software architecture also used redundancy. It also had two identical command and data management units, a triply redundant mission timer unit as well as a triply redundant central acceleration sensor unit. Huygens also had double redundant accelerometers and double redundant proximity sensors.

The 2 identical command and data management units have been developed with only small differences that aimed to avoid common-mode failures. They both execute their own software as to be able to run on their own, if needed, in order to keep the mission afloat. Also, a delay was introduced so that, in case there would be a problem generated by the temporarily loss of the telemetry link, a fault could be avoided since the two replicas had a delay between their telemetry of 6 seconds. Both of the two units do checks and consider themselves invalid in the case they discover that they have a two bit error in the same word in the memory or an under-voltage value of the power line.

In order to ensure that the experimental results and collected data were safe, Huygens probe was not the only sending them to Earth. They were, in fact, also kept as a copy in the Cassini orbiter so that, in any case, they could be transmitted to the ground team on Earth.

Cassini's fault protection mechanisms [16] aimed to avoid the possibility of having a single point of failure somewhere in the systems which formed the orbiter. Other goals the fault protection mechanisms had where to make possible the system recovery and reconfiguration from multiple faults, in case these faults would take place in regions which were independent.

The approach the Cassini system fault protection used was that it was driven by the priority of the faults identified and it could work on the recovery of only one fault at the time. More faults could have the same priority assigned to them, but they would be served in a first come first served manner. The way in which priority was assigned to faults is that the ones which would take the most to be solved would be assigned a lower priority while the most time-critical ones would have the highest priority [18].

The primary mission of the Cassini orbiter ended on 30 June 2008. It did achieve all the original goals and then moved on to take on a new mission - to observe and gather data about the seasonal changes brought by the Sun's changing angle on Saturn. It has also continued to study the rings and moons of Saturn, sending back to Earth priceless information and knowledge about the seventh planet from the Sun [1].

5.3 Mars Climate Orbiter

Mars Climate Orbiter (MCO)[2] was part of the NASA Mars Surveyor program which included the Mars Global Surveyor as well as Mars Polar Lander. It was launched in 1998 with the main objective of studying the climate of Mars. It aimed to monitor atmospheric dust and water vapor as well as taking pictures of the planet's surface in order to gain insight on the climatic changes. The hope was to gather evidences that would support the theory of existing water underneath the surface of the planet.

MCO was also designed to serve as a communication relay for the Mars Polar Lander, however after the Lander's mission, it was supposed to conduct its own main mission independently.

After its nine months journey to Mars, MCO was lost because it missed its planned orbit altitude and fell into the Martian atmosphere where it was destroyed.

The problem identified in the NASA Mishap Investigation Board Phase I Report [7] was that the MCO failed to use metric units in the coding of a ground software file used in trajectory models. Other causes that contributed to the failure of the mission included system engineering, training and organizational factors.

In order to understand better the circumstances that lead to the failure of this mission, we need to take a closer look at the Attitude Control System and the fault protection architecture of the MCO [10].

The MCO's Attitude Control System contained a Sun sensor, an inertial measurement unit, a star camera and reaction wheels. The MCO was controllable in roll, pitch and yaw so that it could adjust its altitude, perform trajectory maneuvers and achieve Mars Orbit Insertion. It was designed so that it could use the gravity force of Mars in order to adjust to the needed velocity which could allow it to keep the desired orbit and altitude. The design included rocket engine modules which allowed this as well as serving as elements that dissipated any angular momentum which could be accumulated in the reaction wheels. The Spacecraft Performance Analysis Software (SPAS) was used to calculate the angular momentum desaturation, however it was not using the correct units of measure (error in the Newton to lb-f conversion). This has lead to MCO estimating badly its altitude and orbit and thus, its failure to maintain orbit.

The fault protection capabilities were unsatisfying at a number of levels in this mission. First of all, there was an error in the human management process. The calculation of the automatic momentum desaturation involved a human constant, because people where involved in loading the information to servers, also in retrieving it from servers and up-linking it to the MCO. The teams dealing with the information where not specifically the ones who introduced the error, however they could have observed the problem and could have found a solution to solve it. This proves the existence of a fault in the human-system interaction. This could have been avoided by developing a better system that would allow people to assess the correctness of the output data based on the input they are giving. It could be a software solution that would act either as verifier (just calculate and report the expected results) or as a checker that would identify the inconsistency and report it.

The critical problem in the MCO case was the lack of an adequate converter between the Newtons and lb-f. The software used for the MCO was partly carried over from another mission (The Mars Global Surveyor). The previous mission used a converter, however it was not imported for the MCO mission by mistake. The fault could have been avoided if there would have been thorough testing after the importing of the already existing software. If the re-used functions would have been inspected with attention and tests would have been developed in order to check the correct functionality in the new context, the fault could have been discovered and solved.

The expectation to have a software system with no bugs or problems is quite unrealistic. However this should not affect the success of missions. The MCO was already calculating the angular momentum desaturation and afterwards down-linked it to the ground base. At that point the Spacecraft Performance Analysis Software was used in order to calculate the angular momentum desaturation value without making the correct conversion between Newtons and lb-f. However, the result would be provided to the team in charge and only afterwards the plan for the future actions would be up-linked to the spacecraft. In this case, another solution for fault avoidance would have been the limitation or elimination of the need of the spacecraft to rely on external

systems and human intervention. Basically, if the MCO would have been designed to be able to calculate its own position and only then cross check it with the available data provided by the ground team, it would have been possible to at least notice and report the problem.

In order to have a good failure tolerance, it was decided that for future spacecrafts it would be advised to use redundant and independent components (both software and hardware). In the case of performing the same calculations with the use of various software modules it could be possible to identify the differences in the values. In the case of multiple such components, a voting could be implemented in order to rule out the modules that provide wrong results.

Since this case is mainly a failure caused by the inadequate re-use of software, it can be stated that such errors can be avoided through the use of independent evaluation and validation of the re-used software. The use of regression testing [13] as well as comparing results to expected outputs can prove helpful.

5.4 Mars Reconnaissance Orbiter

The Mars Reconnaissance Orbiter (MRO)[4] was launched on the 12th of August, 2005 and it was one of the first spacecrafts to enter Mars' orbit.

Its journey to Mars lasted seven months and it took six more months afterwards in order for it to reach its science orbit. The MRO was created with the purpose of tracking changes of the water and dust in Mars' atmosphere. It was also supposed to look for more evidence of ancient seas and hot springs and to study the climate changes based on Mars' surface minerals. It also serves as a data relay station for other missions.

The objectives of the MRO where complex and the probability of failure was high because of this. There were multiple things that could go wrong with the mission, considering the complexity of its overall goals and functionality expectations [12].

First of all, the mission was supposed to last about 5.4 years. The system on the spacecraft was supposed to have an up-time as close as possible to 100% during this mission. This is hard to achieve because all possible failures need to be prevented or handled without losing functionality or precious data.

An analysis of the fault protection architecture, the effectiveness of this architecture as well as a categorization of the fault protection capabilities of the MRO are described in detail in [10].

In order to assure that no big failures could occur and that, despite its complexity, the system would work accordingly in order to achieve the mission's goals, a semi-autonomous fault protection software (SPIDER - SPacecraft Imbedded Distributed Error Response) has been developed for the MRO. It has been developed using the C-language with the possibility of re-use for any other future missions. SPIDER can support redundancy and cross strapping tasks, requirements which the MRO is supposed to meet.

Despite it's advanced capabilities and the fact that it acts as a first responder to errors and handles most of them, SPIDER was not designed to function on its own. It still needed help from the ground in order for the system to be set back to a normal state for the cases when it entered a safe mode. Also, ground operations where needed in case the system did not recognize some possibly threatening conditions that could appear. The main advantage of the SPIDER is

[13] http://en.wikipedia.org/wiki/Regression_testing

that it can actually prevent the system from entering massive failure without the need to wait for the ground crew to notice and react to the problems. A brief diagram depicting the decision process of SPIDER can be seen in Fig. 6.

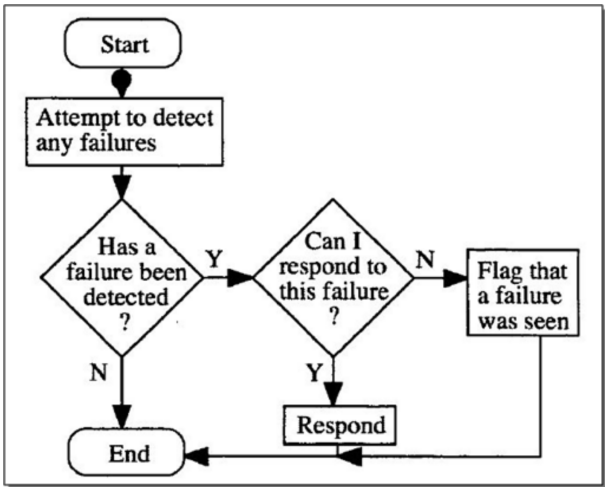

Figure 6: The generic decision process of the SPIDER. *Source: E. Scale, The evolution of a SPIDER fault protection, incremental development, and the Mars Reconnaissance Orbiter mission, 2003*

The SPIDER ensures that the fault responses are general. For example, if the MRO finds itself in a position in which a fault is detected, then it automatically switches to a safe mode and interrupts all the uncritical equipment and functionality. The system can be turned back to normal functionality from the ground level by the team in charge after following a defined protocol.

The SPIDER ensures that the MRO is not affected by fails at any given time, with the presumption that only one system can generate failure at a given time. It basically tries to ensure that the MRO is completely single-fault tolerant and this leads to the robustness and high reliability level of the architecture. As it was mentioned before, SPIDER was created around the idea of keeping things redundant and around the cross strapping concept. Among the redundancy techniques it uses we can identify: block redundancy, functional redundancy, cooperative redundancy and cross strapping.

The architecture of the SPIDER is hierarchical. It has three software levels: the **component level** fault protection which is used for communication with the MRO hardware, the **performance level** fault protection which keeps track of the performance of each subsystem and the **system level** fault protection which tries to keep failures from happening. Depending on the type of failure, it can be handled by the befitting logic. In order to avoid starvation, the higher levels in the hierarchy are allowed to call on to the lower levels in order to assign them specific tasks, but they cannot take priority of the tasks which are already executing in these levels.

The SPIDER has been thoroughly tested before being integrated on the MRO, but a final proof of its capability was the fact that, even though the MRO has encountered quite a few failures during the mission (e.g. memory corruption, down-link connectivity failure, etc.), it was capable to recover and to achieve the mission's goals.

5.5 Mars Exploration Rovers

NASA has launched, in 2003, the twin rovers, Opportunity and Spirit [3], with the purpose of gaining more insight about past water activity on the red planet and if there where, at some point, conditions that could favor life on our neighbor planet. The planned mission was supposed to last for about 3 months from the moment the rovers would land. However, they have exceeded any expectations both as far as their durability is concerned as well as the discoveries they have made.

Spirit has collected evidences that, in the past, Mars was much better than we see it now. Also, it collected information about the winds on the red planet. The rover has been silent as of March 2010 and the ground team has ceased to try to contact it since May 2011 when its mission was closed and considered complete.

Opportunity has found evidence that Mars could have been capable of sustaining microbial life. During its mission it traveled for over 20 km (as of March 2010) in its search for a better understanding of the planet. The rover is still active (as of 5th November 2013).

An overview of why the fault protection worked so well for these rovers is provided in [10].

The rovers needed to enter into a sleep mode during the days in order for them to be able to recharge their batteries. In this state their CPU would be powered off, but the hardware still needed to maintain the safe thermal and power states [14]).

Upon waking from the sleep mode, the rovers were expected to start the communication without any help from ground operation teams on Earth. As a fault protection measure, if during the rover's initialization a server error occurred, the reset of the system would be postponed for a pre-defined amount of time. A software health function was also used in order to check for any lingering or suspended tasks. For the communication system, redundancy was used such so that, in case the high gain antenna would not function properly, then the low gain antenna could perform its tasks (even if this one had lower data rates it could still be enough so that the mission wouldn't be a failure).

The fault protection architecture of the rovers can be divided into local and system-level fault protection, each of them dealing with various kinds of problems[15]. A diagram of the local and system level fault protection can be seen in Figure 7.

There are more general capabilities of the fault protection architecture used in the case of the rovers. First of all, the ground teams can monitor the data which the rovers send back (or fail to send back) and can intervene based on the type of errors they observe.

Since the rovers have an algorithm that controls automatically the wake up and shutdown procedures, it means that a low energy fault can be avoided. The algorithm only allowed the rovers to work when they had enough energy to do this, thus abnormal terminations or incapability to save data before entering sleep mode could be avoided. The rovers were also tolerant to failure because, for example, in case they did not stock up enough energy in order to function properly, then the system would not allow them to wake up from the sleep mode and by doing this it avoided the failure of letting them work without appropriate power.

A plus to the fact that the rovers needed to enter sleep mode everyday and basically shutdown

[14]The temperatures on Mars can be quite extreme - from negative 107 °C up to over positive 30 °C - http://en.wikipedia.org/wiki/Climate_of_Mars

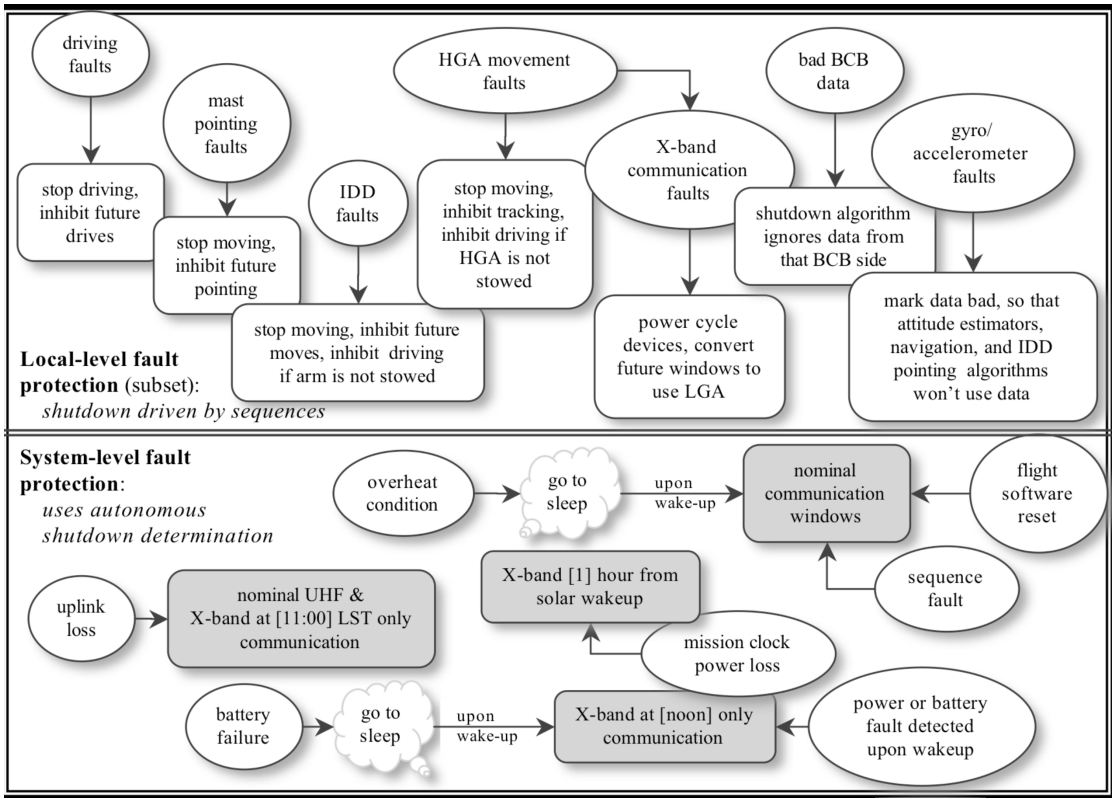

Figure 7: Rovers surface fault protection overview. *Source: D. Clark, C. Brennan and M. Jaunich, Survey of Nasa Mars Exploration Missions - Software Fault Protection Architecture, 2010*

for a period of time was software rejuvenation[15]. Memory leaks could be removed through this process.

Aside from this power-related fault protection, they benefited from the presence of a navigation algorithm that looked for possible hazards on the rovers' paths and tried to avoid them by choosing between possible identified routes.

Even though testing was developed previous to mission launch, complete fault avoidance could not be guaranteed. For example, the ground team could identify that after 17 sols (Martian days) one of the rovers was rebooting over and over. They managed to identify the problem as being the fact that, when a file needed to be deleted from RAM, even though the link to it was lost, the size of the table of contents of the RAM was not decreased so the system managed to consume all the RAM space available after some time. The problem was solved from the ground when the team instructed the software to not use the software in memory and use another one instead. This prevented the failure of the mission.

The rovers have been an example of a mission that exceeded by far its expectations. Despite the original mission duration of 90 days, one of the rovers is still active and providing scientiests with data from Mars.

[15] As a way of helping against software aging - http://en.wikipedia.org/wiki/Software_aging

6 Conclusions

This paper aimed to take a closer look into the fault-tolerance techniques used in space missions. The need to explore our Universe as far as we can reach will persist as long as we humans will exist. This means that more and more space missions will be conducted and, in order for them to be successful, we need to keep in mind what went good in previous missions, what we should use again, but also what were our mistakes and how to avoid them in the future.

We have presented some of the conditions which make space missions special as well as some general architectures and fault-tolerance techniques which can be applied to most of them. But, since missions are in general very different because of multiple reasons like goals, duration, target distance etc., they mostly need to face different or even unknown problems and as such, solutions need to be always adapted for each mission. We have, therefore taken a closer look at some specific mission and what they have used as far as fault tolerance goes.

We have seen that even the smallest mistakes can lead to a mission failure, but also that even though software will always contain small bugs, a good and healthy fault tolerant design can ensure the success of a mission, despite some problems which might occur from time to time.

As a last thought, when it comes to space missions, every detail, no matter how small, needs to be taken into consideration, all theories need to be tested thoroughly and every possibility needs to be taken into account. However, the work this involves will always be richly rewarded each time we will find a new piece in the puzzle that our Universe represents.

References

[1] National Aeronautics and Space Administration. Cassini-huygens. http://solarsystem.nasa.gov/missions/profile.cfm?Sort=Alpha&Letter=C&Alias=Cassini-Huygens.

[2] National Aeronautics and Space Administration. Mars climate orbiter mission. http://solarsystem.nasa.gov/missions/profile.cfm?Sort=Alpha&Alias=Mars%20Climate%20Orbiter&Letter=M&Display=ReadMore.

[3] National Aeronautics and Space Administration. Mars exploration rovers. http://solarsystem.nasa.gov/missions/profile.cfm?Sort=Alpha&Letter=M&Alias=Mars%20Exploration%20Rovers.

[4] National Aeronautics and Space Administration. Mars reconnaisance orbiter mission. http://solarsystem.nasa.gov/missions/profile.cfm?Sort=Alpha&Alias=Mars%20Reconnaissance%20Orbiter&Letter=M&Display=ReadMore.

[5] National Aeronautics and Space Administration. Voyager 1. http://solarsystem.nasa.gov/missions/profile.cfm?Sort=Alpha&Letter=V&Alias=Voyager%201.

[6] National Aeronautics and Space Administration. Voyager 2. http://solarsystem.nasa.gov/missions/profile.cfm?Sort=Alpha&Letter=V&Alias=Voyager%202.

[7] National Aeronautics and Space Administration. Mars climate mishap investigation board phase I report. ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf, 1999.

[8] K. L. Bedingfield, R. D. Leach, and M. B. Alexander. *Spacecraft system failures and anomalies attributed to the natural space environment.* National Aeronautics and Space Administration, Marshall Space Flight Center, 1996.

[9] J. Day and M. Ingham. Fault management at JPL : past, present and future, 2011.

[10] C. B. Dennis Clark and M. Jaunich. Survey of nasa mars exploration missions - software fault protection architecture, 2010.

[11] H. Hassan and J. Jones. The huygens probe. *ESA Bulletin*, (92):33–43, 1997.

[12] R. L. Heacock. The voyager spacecraft. *Proceedings of the Institution of Mechanical Engineers*, 194(28):211–224, 1980.

[13] E. Litty. Attitude control fault protection-the voyager experience. *American Astronautical Society, Paper No. AAS 80-018*, 1980.

[14] National Aeronautics and Space Administration. Fault management handbook - NASA-HDBK-1002. Draft 2 - 2012.

[15] T. Neilson. Mars exploration rovers surface fault protection. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 14–19. IEEE, 2005.

[16] E. Ong and N. Leveson. *Fault protection in a component-based spacecraft architecture.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2003.

[17] E. Scale. The evolution of a spider fault protection, incremental development, and the mars reconnaissance orbiter mission. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 5, pages 5_2493–5_2499. IEEE, 2003.

[18] D. P. Siewiorek and P. Narasimhan. Fault-tolerant architectures for space and avionics applications. *NASA Ames Research*, 2005.