On the Autonomy Requirements for Space Missions

Emil Vassev
Lero–the Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
emil.vassev@lero.ie

Mike Hinchey
Lero–the Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
mike.hinchey@lero.ie

Abstract — In new space exploration initiatives of NASA and ESA, there is emphasis on both human and robotic exploration. Risk and feasibility are major factors supporting the use of unmanned craft and the use of automation and robotic technologies where possible. In that context, an autonomous system is able to monitor its behavior and eventually modify the same according to changes in the operational environment, thus being considered as self-adaptation. Requirements engineering for autonomous systems, therefore, must address what adaptations are possible and under what constraints, and how those adaptations are realized. Requirements engineering for autonomous systems appears to be a wide open research area with only a limited number of approaches yet considered. In this paper, we present initial results of our research and study on autonomy requirements for space systems.

Keywords — space missions; autonomy requirements; autonomic systems.

I. INTRODUCTION

Over the years, both ESA and NASA missions have been gradually adding autonomy to flight and ground systems to increase the amount of science data returned from missions, perform new science, and reduce mission costs. In the new space exploration initiatives, there is emphasis on both human and robotic exploration. Even when humans are involved in the exploration, human tending of space assets must be evaluated carefully during mission definition and design in terms of benefit, cost, risk, and feasibility. Risk and feasibility are major factors supporting the use of unmanned craft and the use of automation and robotic technologies where possible.

Both NASA and ESA are currently recognizing the autonomic computing (AC) paradigm [1] as a valuable approach to the development of special autonomous components for spacecraft systems (e.g., ExoMars). However, to tackle the AC issues, both organizations are currently conducting a joint project with ESA targeting such an approach that shall help ESA developers properly: 1) express autonomy requirements and 2) verify and validate autonomy requirements. In this paper, we present our first cut of autonomy requirements for space missions. We analyze different classes of space missions, along with mission challenges, to deduce autonomy requirements per class of missions.

The rest of this paper is organized as follows. Section II presents the project objectives. Section III discusses the notions of autonomy and automation. Section IV presents the standard levels of autonomy considered for ESA missions. Section V briefly discusses requirements engineering for autonomous systems. Section VI elaborates on generic autonomy requirements. In Section VII, we discuss different classes of space missions and put the generic requirements in the context of these mission classes. Finally, Section VIII provides brief concluding remarks and a summary of our future goals.

II. PROJECT’S OBJECTIVES AND AREM

The goal of this particular project is to select and present formal methods that will help ESA developers express and understand autonomy requirements to some extent. Those formal methods shall assist in the construction of a new Autonomy Requirements Engineering Model (AREM) suitable for the development of autonomous components for ESA systems. AREM should take into account all the autonomy aspects of a targeted system and emphasize the so-called self-* requirements by taking into consideration the traditional functional and non-functional requirements of spacecraft systems (e.g., safety requirements). However, we can ask the question: Why formal methods? Traditionally, formal methods have had the needed potential for modeling and validating the control behavior of software-intensive systems and they might help in expressing autonomy requirements and modeling autonomic and self-adaptive behavior. It is our understanding that the application of formal methods will help ESA developers unambiguously express autonomy requirements, which are currently expressed in natural language. Hence, we expect that proper

1 The term “autonomy requirements” is used throughout this text as a synonym for all types of specific requirements related to the autonomic and self-adaptive behavior of aerospace systems.

2 Modern spacecraft and autonomous robotics systems are considered to be software-intensive.
formal methods will improve the software development cycle of autonomic features for the ESA’s software-intensive systems in terms of:

1) rigorous and unambiguous specification of autonomy requirements;
2) autonomy requirements traceability, verification and validation;
3) derivation of test cases and automatic test case generation based on requirements specification.

The Autonomy Requirements Engineering (ARE) should be considered as a software engineering process of 1) determining what autonomic features are to be developed for a particular software-intensive system or subsystems; and 2) the software artifacts generated by that process. Note that the outcome of ARE (requirements specifications, models, etc.) is a precursor of design of autonomic features. The ARE process should involve all of the following:

- autonomy requirements elicitation;
- autonomy requirements analysis;
- autonomy requirements representation;
- autonomy requirements communication;
- development of acceptance criteria and procedures for autonomy requirements.

Note that the targeted AREM approach shall be a framework incorporating formal methods dedicated to autonomic features of software-intensive systems. The AREM framework shall allow for specification and modeling of autonomy requirements and it shall provide for validation and traceability of specified autonomy requirements. Thus, AREM shall be a requirements engineering approach helping to create reliable software that maximizes the probability of satisfying user expectations. This shall be possible because the framework toolset is going to provide verification mechanisms for automatic reasoning about specified autonomy requirements. A basic validation approach could be consistency checking where autonomy requirements shall be verified by performing exhaustive traversal to check for both syntax and consistency errors and to check whether requirements conform to predefined autonomy correctness properties, defined by ESA engineers. For example, correctness properties can be set to target the requirements feasibility.

Moreover, to handle logical errors (specification flaws) and to be able to assert safety (e.g., freedom from deadlock) and liveness (nice-to-have) properties, AREM can eventually provide for both model-checking and test-case generation mechanisms. Finally, AREM could be supplied with code generation mechanisms to facilitate the implementation of autonomic features.

III. AUTONOMY VS. AUTOMATION

Both autonomy and automation refer to processes that may be executed independently from start to finish without any human intervention. Automated processes simply replace routine manual processes with software/hardware ones that follow a step-by-step sequence that may still include human participation. Autonomous processes, on the other hand, have the more ambitious goal of emulating human processes rather than simply replacing them.

An example of an automated ground data trending program would be one that regularly extracts from the data archive a set list of telemetry parameters, performs a standard statistical analysis of the data, outputs in report form the results of the analysis, and generates appropriate alerts regarding any identified anomalies. So, in contrast to an autonomous process, in this case the ground system performs no independent decision making based on real-time events, and a human participant is required to respond to the outcome of the activity [2].

On the other hand, the more elaborate process of autonomy is displayed by a ground software program that independently identifies when communication with a spacecraft is possible, establishes the link, decides what files to uplink, uplinks those files, accepts downlinked data from the spacecraft, validates the downlinked data, requests retransmission as necessary, instructs the freeing-up of onboard storage as appropriate, and finally archives all validated data. This would be an example of a fully autonomous ground process for uplink/downlink [2].

Complete autonomy may not be desirable or possible for some systems. In such cases, adjustable and mixed autonomy may need to be used [3]. In adjustable autonomy, the level of autonomy of the system (e.g., spacecraft) can vary depending on the circumstances or the needed interaction and control. The autonomy can be adjusted to be either complete, partial, or no autonomy. In these cases the adjustment may be done automatically by the system depending on the situation (e.g., an autonomous spacecraft may ask for help from mission control) or may be requested by the human control. Challenges in adjustable autonomy include knowing when it needs to be adjusted, as well as how much and how to make the transition between levels of autonomy. In mixed autonomy, autonomous agents and people work together to accomplish a goal or perform a task. Often the agents perform the low level details of the task (e.g., analogous to the craft’s preparation for landing) while the human performs the higher-level functions (e.g., analogous to the actual landing).

IV. LEVELS OF AUTONOMY FOR ESA MISSIONS

ESA considers four autonomy levels for the execution of nominal mission operations [4]:

- execution mainly under real-time ground control;
- execution of pre-planned mission operations onboard;
- execution of adaptive mission operations onboard;
- execution of goal-oriented mission operations onboard.

These autonomy levels are summarized in Table 1. As shown in that table, ESA approaches the autonomicity problem very carefully in a stepwise manner. In this approach the highest-possible autonomy is the goal-oriented autonomy (level E4) where goals are determined by human operators and autonomous spacecraft decide what to do to
autonomously achieve the desired goals. Still, this autonomy level hasn’t been achieved yet. The current level of spacecraft autonomy is level E2 and ExoMars is expected to operate at level E3.

Table 1. ESA’s Mission-execution Autonomy Levels [4]

<table>
<thead>
<tr>
<th>Autonomy Level</th>
<th>Description</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>1) Mission execution under ground control; 2) Limited onboard capability for safety issues.</td>
<td>1) Real-time control from ground for nominal operations. 2) Execution of time-tagged commands for safety issues.</td>
</tr>
<tr>
<td>E2</td>
<td>Execution of pre-planned, ground-defined, mission operations onboard.</td>
<td>Capability to store time-based commands in an onboard scheduler.</td>
</tr>
<tr>
<td>E3</td>
<td>Execution of adaptive mission operations onboard.</td>
<td>Event-based autonomous operations. Execution of onboard operations control procedures.</td>
</tr>
<tr>
<td>E4</td>
<td>Execution of goal-oriented mission operations onboard.</td>
<td>Goal-oriented mission re-planning.</td>
</tr>
</tbody>
</table>

V. REQUIREMENTS ENGINEERING FOR AUTONOMOUS SYSTEMS

In general, an autonomous system is able to monitor its behavior and eventually modify the same according to changes in the operational environment, thus being considered as self-adaptation. As such, autonomous systems must continuously monitor changes in its context and react accordingly. But what aspects of the environment should such a system monitor? Clearly, the system cannot monitor everything. And exactly what should the system do if it detects less than optimal conditions in the environment? Presumably, the system still needs to maintain a set of high-level goals that should be satisfied regardless of the environmental conditions, e.g., mission goals of unmanned spacecraft used for space exploration. But non-critical goals could be not that strict [5], thus allowing the system a degree of flexibility during operation. These questions (and others) form the core considerations for building autonomous systems.

Traditionally, requirements engineering is concerned with what a system should do and within which constraints it must do it. Requirements engineering for autonomous systems (and self-adaptive systems), therefore, must address what adaptations are possible and under what constraints, and how those adaptations are realized. In particular, questions to be addressed include: 1) “What aspects of the environment are relevant for adaptation?”; and 2) “Which requirements are allowed to vary or evolve at run-time, and which must always be maintained?”. Requirements engineering for autonomous systems must deal with uncertainty, because the execution environment often is dynamic and the information about future execution environments is incomplete, and therefore the requirements for the behavior of the system may need to change (at run-time) in response to the changing environment.

Requirements engineering for autonomous systems appears to be a wide open research area with only a limited number of approaches yet considered. In this paper, we present a few formal methods that eventually can be successful in capturing autonomy requirements.

VI. GENERIC AUTONOMY REQUIREMENTS

The first step towards development of a new software-intensive system is to determine the system’s requirements, which includes both elicitation and specification (or modeling) of the same. In general, requirements fall into two categories: functional and non-functional. Whereas the former define the system’s functionality the latter emphasize system’s qualities (e.g. performance) and constraints under which a system is required to operate. Like any computer system, autonomic systems (ASs) also need to fulfill specific requirements from these two categories. However, unlike the other systems, the development of an AS is driven by the so called self-management objectives (also could be considered as self-adaptive objectives) and attributes, which introduce special requirements termed self-* requirements [6]. Despite their differences in terms of application domain and functionality, all ASs are capable of self-management and are driven by one or more self-management objectives. Note that this requirement automatically involves 1) self-diagnosis (to analyze a problem situation and to determine a diagnosis), and 2) self-adaptation (to repair the discovered faults). The ability to perform adequate self-diagnosis depends largely on the quality and quantity of its knowledge of its current state, i.e., on the system awareness.

The following is a list of generic autonomy requirements [6] stemming from the self-* requirements.

**Autonomicity (Self-* Requirements).** As the term already suggests, autonomicity is one of the essential characteristics of ASs. Autonomicity aims at freeing human operators from complex tasks, which typically require a lot of decision making without human intervention. Autonomicity, however, is not only intelligent behavior but also an organizational manner. Adaptability is not possible without a certain degree of autonomy. A rule engine obeying a predefined set of conditional statements (e.g., if-then-else) put in an endless loop is the simplest form of autonomicity implementation. In many cases though, such a simple rule-based mechanism may not be sufficient and the rule engine should force feedback learning and learning by observation to refine the decisions concerning the priority of services and their granted objectives and quality of service, respectively.

**Knowledge.** In general, an AS is intended to possess awareness capabilities based on well-structured knowledge.

---

3 The term “autonomic systems” is often used in the scientific literature as a synonym of self-adaptive and autonomous systems.
and algorithms operating over the same. Conceptually, knowledge can be regarded as a large complex aggregation composed of constituent parts representing knowledge of different kind. Every kind of knowledge may be used to derive knowledge models of specific domains of interest. For example, the following kinds of knowledge may be considered [7]:

- **domain knowledge** – refers to the application domain facts, theories, and heuristics;
- **control knowledge** – describes problem-solving strategies, functional models, etc.;
- **explanatory knowledge** – defines rules and explanations of the system's reasoning process, as well as the way they are generated.
- **system knowledge** – describes data contents and structure, pointers to the implementation of useful algorithms needed to process both data and knowledge, etc. System knowledge also may define user models and strategies for communication with users.

Moreover, being considered as essential system and environment information, knowledge may be classified as 1) **internal knowledge** - knowledge about the system itself; and 2) **external knowledge** - knowledge about the system environment. Another knowledge classification could consider a priori knowledge (knowledge initially given to a system) and experience knowledge (knowledge gained from analysis of tasks performed during the lifetime of a system). Therefore, it depends on the problem domain what kinds of knowledge may be considered and what knowledge models may be derived from those kinds. For example, we may consider knowledge specific to:

- internal component structure and behavior;
- system-level structure and behavior;
- environment structure and behavior;
- different situations where an AS component or the system itself might end up in;
- components’ and system’s capabilities of communication and integration with other systems.

**Awareness.** One of the success factors for an AS is to employ its knowledge and become aware system. Such a system is able to sense and analyze its components and the environment where it operates. A primary task is to determine the state of each component and the status of the service-level objectives. Thus, an aware system should be able to notice a change and understand the implications of that change. Therefore, both self-monitoring and monitoring of the environment are key issues in awareness. Moreover, an aware system should be able to apply both pattern analysis and pattern recognition to determine normal and abnormal states. Conceptually, awareness is a product of knowledge representation, knowledge processing and monitoring. We can consider two awareness types in ASs:

- **self-awareness** – a system (or a system’s component) has detailed knowledge about its own entities, current states, capacity and capabilities, physical connections and ownership relations with other systems in its environment;
- **context-awareness** – a system (or a system’s component) knows how to negotiate, communicate and interact with its environment and how to anticipate environmental states, situations and changes.

**Monitoring.** Since monitoring is often regarded as a prerequisite for awareness, it constitutes a subset of awareness. For ASs, monitoring is the process of obtaining knowledge through a collection of sensors instrumented within the AS in question. Note that monitoring is not responsible for diagnostic reasoning or adaptation tasks. One of the main challenges of monitoring is to determine which information is most crucial for analysis of the system's behavior, and when. The notion of monitoring is closely related to the notion of awareness because it is a matter of awareness, which information indicates a situation in which a certain adaptation is necessary.

**Adaptability.** The core concept behind adaptability is the general ability of an AS to change its observable behavior and structure. This requirement is amplified by self-adaptation (or automatic adaptation). Self-adaptation helps an AS decide on-the-fly about an eventual adaptation on its own. Adaptation may result to changes in some functionality, algorithms or system parameters as well as the system’s structure or any other aspect of the system. Note that self-adaptation requires a model of the system’s environment. Adaptability is conceptualized as a concept to achieve change. It is in sharp contrast to creating new builds. A key research gap in this area is how to measure “adaptability”.

**Dynamicity.** Dynamicity shows the system’s ability to change at runtime. Whereas adaptability refers to the conceptual change of certain system aspects (this does not necessarily imply the change of components or services), dynamicity is about the technical ability to remove, add or exchange services and components. Dynamicity may also include a system’s ability to exchange certain (defective or obsolete) components without changing the observable behavior. Conceptually, dynamicity deals with concerns like preserving states during functionality change, starting, stopping and restarting system functions, etc.

**Robustness.** Robustness is a requirement that is claimed for almost every system. ASs should benefit from robustness since this may facilitate the design of system parts that deal with self-healing and self-protecting. In addition, the system architecture could ease the appliance of measures in cases of errors and attacks. Beside a special focus on error avoidance, several requirements aiming at correcting errors should also be forced. Robustness could be often achieved by decoupling and asynchronous communication, e.g., between interacting AS components. Error avoidance, error prevention, and fault tolerance are approved techniques in software engineering, which shall help us in preventing from error propagation when designing ASs.

**Resilience.** Adaptability might be considered as a quality attribute that is a prerequisite for resilience and system agility [8]. Resilience is a system quality attribute important
to the Aerospace Industry. Closely related to safety, resilience enables aerospace systems to bounce back from unanticipated disruptions as well as to equip aging systems with the ability to respond to changing operational requirements.

**Mobility.** Mobility enfolds all parts of the system: from mobility of code on the lowest granularity level via mobility of services or components up to mobility of devices or even mobility of the overall system. Mobility enables dynamic discovery and usage of new resources, recovery of crucial functionalities, etc. For example, ASs may rely on mobility of code to transfer some functionality relevant for security updates or other self-management issues.

**VII. AUTONOMY REQUIREMENTS FOR SPACE MISSIONS**

In general, space missions can be classified into two main groups: Earth-orbiting missions and interplanetary missions [9]. In this section, we present the classes of space missions and elaborate on the autonomy requirements for these classes.

**A. Earth-Orbiting Missions**

The Earth-Orbiting Missions is a class of missions that represents artificial satellites placed into Earth orbit and used for a large number of purposes. Different orbits give satellites different vantage points for viewing Earth, i.e., different Earth orbits give satellites varying perspectives, each valuable for different reasons. Some satellites hover over a single spot, providing a constant view of one face of the Earth, while others circle the planet. Figure 1 depicts common Earth satellite orbits [10].

A common challenge in designing Earth-orbiting missions is the orbital perturbations. There are a variety of effects that will cause orbital perturbations during the lifetime of a satellite [11]:

- **third body perturbations**: dominated by the gravitational forces of the Sun and the Moon;
- **perturbations due to a non-spherical Earth**;
- **atmospheric drag**: principal non-gravitational force acting on a satellite and it only affects satellites in low-Earth orbit. Drag acts in the direction of opposite of the satellite’s velocity resulting in a removal of energy from the orbit. This loss of energy results in the size of the orbit decreasing, which then leads to a further increase in drag;
- **solar radiation**: solar radiation pressure is an effect that is strongest on satellites with large area to mass ratios. It results in periodic variations in all orbital elements.

![Figure 1. Common Earth Orbits][10]

1) **Polar Low Earth Orbit (LEO)/Remote-Sensing Satellite Missions.** These missions involve satellites that fly low orbit and use different earth-observation instruments that gathered information about the Earth (land, water, ice and atmosphere) using a variety of measurement principles.
The choice of orbit for a Low Earth Orbit (LEO) remote sensing spacecraft is governed by mission objectives and payload operational requirements. A LEO orbit is below an altitude of approximately 2000 km (1200 mi). Spacecraft in LEO encounter atmospheric drag in the form of gases in the thermosphere (approximately 80–500 km up) or exosphere (approximately 500 km and up), depending on orbit height. LEO is an orbit around Earth between the atmosphere and below the inner Van Allen radiation belt. The altitude is usually not less than 300 km because that would be impractical due to the larger atmospheric drag [9].

Equatorial low Earth orbits (ELEO) are a subset of LEO. These orbits, with low inclination to the Equator, allow rapid revisit times and have the lowest $\Delta V$ (a measure of the amount of "effort" that is needed to change from one trajectory to another by making an orbital maneuver) requirement of any orbit. Orbits with a high inclination angle are usually called polar orbits. Higher orbits include Medium Earth Orbit (MEO), sometimes called intermediate circular orbit (ICO), and further above, Geostationary Orbit (GEO).

A common challenge to this class of missions is to determine the right orbit altitude. The orbit altitude is principally established by a trade-off between instrument resolution and the fuel required to maintain the orbit in the presence of aerodynamic drag. Orbits higher than low orbit can lead to early failure of electronic components due to intense radiation and charge accumulation.

The following is a list of deducted autonomy requirements for this class of missions.

- **self-* requirements (autonomicity):**
  - *self-orbit* (autonomously acquire the target orbit; adapt to orbit perturbations);
  - *self-protection* (autonomously detect the presence of radiation and move to escape);
  - *self-scheduling* (based on operational goals and knowledge of the system and its environment, autonomously determine what task to perform next);
  - *self-reparation* (implies operations re-planning based on performance degradation or failures);

- **knowledge:** mission objectives, payload operational requirements, instruments onboard together with their characteristics (e.g., instruments resolution), the Van Allen radiation belt, ground stations, communication links, data transmission format, orbit planes, eclipse period, spacecraft altitude, communication mechanisms onboard, Earth gravity;

- **awareness:** orbit awareness, radiation awareness, altitude awareness, position awareness, instrument awareness, neighboring satellites, sensitive to thermal stimuli, Earth gravitational force, data-transfer awareness, ground station visibility awareness, Earth rotation awareness, speed awareness, communication awareness, altitude awareness, air resistance awareness;

- **monitoring:** electronic components, surrounding environment (e.g., radiation level), atmospheric drags, ground station, altitude and orbit;

- **adaptability:** adaptable mission parameters, adapt to loss of energy, adapt to high radiation, adapt to weak satellite-ground station communication link, adapt to low energy;

- **dynamicity:** dynamic communication links;

- **robustness:** robust to temperature changes, robust to orbital perturbations, robust to communication losses;

- **resilience:** loss of energy is recoverable, resilient to radiation;

- **mobility:** information goes in and out, changing position within the orbit plane.

Figure 2. Satellite Constellation Missions [12]
2) Satellite Constellation Missions. These missions are presented by multi-satellite systems where a group of satellites called “constellation” work together. Such a constellation can be considered to be a number of satellites with coordinated ground coverage, operating together under shared control, synchronized so that they overlap well in coverage and complement rather than interfere with other satellites’ important coverage [9].

For a constellation to operate, it might be necessary to use more than a single ground station, especially when the space segment consists of a large number of satellites (see Figure 2). Inter-satellite links (ISL) are bidirectional communication links between satellites in LEO or MEO orbits. Some common challenges related to the design and implementation of such missions are:

- Distributed system in space and a distributed system on ground, combined in a distributed space mission. One of the major issues is that the topology of the distributed space mission changes over time, which places stringent requirements on communication. The topology change is on the one side caused from the orbit dynamics, on the other side might be manually controlled to switch to a desired formation or constellation.
- Due to the movement of the satellites on their orbits, the communication links between ground stations and satellites change frequently.
- Data flow in the satellite network has to be coordinated.
- The invariance of the constellation geometry when subject to orbital perturbations (there could be a large number of possible constellation configurations that may satisfy a particular mission requirement).

The following is a list of deducted autonomy requirements for this class of missions (also add the requirements of Polar LEO/Remote-Sensing Satellite Missions described in Section VII.A.1):

- **self-* requirements (autonomicity):**
  - self-formation (autonomously determine the right satellite configuration and perform it);
  - self-reparation (broken communication links must be restored autonomously);
  - self-scheduling (autonomously determine which satellites operate when; payload);
  - self-coordination (autonomously coordinate operations where several spacecraft may coordinate their operations on achieving a common goal: 1) Earth observation of a specific region performed by several spacecraft: at different times or with different instruments for sensor-fusion purposes; 2) coordination of experiments);
  - self-organization (to distribute data in a space network a high degree of self-organization is required, i.e., routing capabilities due to changing topology);
  - self-geometry (adapt the constellation geometry to orbit perturbations);
  - knowledge: constellation satellites (or neighboring satellites), inter-satellite communication links, group payload, constellation orbit planes, constellation geometry (e.g., Walker Delta pattern constellation), total number of satellites;
  - awareness: formation awareness, satellites synchronization awareness;
  - monitoring: constellation configuration;
  - adaptability: adapt to new formations, adapt to weak inter-satellite communication links, adapt constellation geometry to orbital perturbations;
  - dynamicity: dynamic formation (dynamic topology), dynamic inter-satellites communication links (change affects communication);
  - robustness: robust to inter-satellite communication losses, robust to a single satellite loss;
  - resilience: resilient satellite formations;
  - mobility: inter-constellation mobility (information and satellites), moving satellites within an orbit plane, moving satellites from one orbit plane to another.

3) Geostationary Earth Orbit (GEO) Missions. This class of missions involves satellites orbiting at Geostationary Earth Orbit (GEO) usually for providing global communications [9]. Satellites in such an orbit have an orbital period equal to one sidereal day (the Earth's rotational period or 23h 56 min). The 24-hour geostationary orbit clearly offers unique advantages, providing almost complete global coverage from merely three satellites, and with no need for the ground antenna to switch between the satellites. Several transfer orbit revolutions occur before injection of the satellite into near-circular, near-GEO orbit. A common challenge is related to the fact that a geostationary orbit can only be achieved at an altitude very close to 35,786 km (22,236 mi), and directly above the Equator. The following is a list of deducted autonomy requirements for this class of missions (also add the requirements of Polar LEO/Remote-Sensing Satellite Missions described in Section VII.A.1):

- **self-* requirements (autonomicity):**
  - self-GEO-keeping (use thrusters to autonomously maintain the geostationary orbit – position, altitude and speed, by adapting to perturbations such as the solar wind, radiation pressure, variations in the Earth's gravitational field, and the gravitational effect of the Moon and Sun);
  - knowledge: GEO coordinates, perturbation factors, GEO altitude, solar wind, Moon's gravitational field, Sun’s gravitational field;
- **awareness**: orbit perturbation awareness, solar wind awareness, radiation pressure awareness, Moon’s gravitational effect awareness, Sun’s gravitational effect awareness.
- **monitoring**: GEO position, other GEO satellites, Moon position, Sun position.
- **adaptable**: adapt to communication latency (geostationary orbits are far enough away from Earth that communication latency becomes significant — about a quarter of a second), adapt to perturbations (such as the solar wind, radiation pressure, variations in the Earth's gravitational field, and the gravitational effect of the Moon and Sun).
- **dynamicity**: dynamic GEO positioning and altitude.
- **robustness**: robust to communication latency;
- **resilience**: resilient GEO positioning;
- **mobility**: moving the satellite in the GEO plane.

4) **Highly Elliptic Orbit Missions**. In this class of missions, spacecraft in **elliptic orbits** move more rapidly at perigee than at apogee. This offers the prospect of a pass of increased duration over a ground station if the apogee is situated above it. Two mission subclasses are derived [9]: Space-born Observatories and Communication Spacecraft.

**Space-born Observatories.** Spacecraft are used in observatory mode, which means the spacecraft instruments are operated as if they were located in a room adjacent to the astronomer’s workstation. To achieve extended periods of time, the payload can be pointed to desired astrophysical targets whilst uninterrupted contact with a ground station is maintained. In general, there will be an interruption of observational time while the spacecraft passes through the perigee region. Common challenges are related 1) to how to optimize the spacecraft’s orbit period with respect to the ground station coverage; and 2) the radiation environment, which may preclude the operation of certain types of payload. The following is a list of deduced autonomy requirements for this class of missions (also add the requirements of Polar LEO/Remote-Sensing Satellite Missions described in Section VII.A.1):

- **self-* requirements (autonomicity):**
  - **self-optimization** (autonomously maintain the optimum spacecraft’s orbit period with respect to the ground station coverage and keep up with it);
  - **self-protection** (autonomously detect high radiation and cover sensitive instruments);
  - **self-repair** (autonomously detect problems in instruments and repair; broken links must be restored autonomously);
  - **self-command** (autonomously evaluate the effect of executing remote commands before perform those to guarantee that the spacecraft will not fall in a dangerous situation due to a command execution);

- **self-scheduling** (autonomously determine which instruments operate when);
- **self-coordination** (autonomously coordinate data flow gathered by different instruments);
- **self-tuning** (autonomously tune the instruments onboard);
- **knowledge**: instruments onboard, inter-instrument communication links, objects/phenomena to observe, Moon’s gravitational field, Sun’s gravitational field;
- **awareness**: operation awareness, instruments synchronization awareness, Moon’s gravitational force and Sun’s gravitational force awareness;
- **monitoring**: instruments operation, Moon position, Sun position;
- **adaptable**: adapt to new tasks, adapt to instrument losses, adapt to instrument performance degradation;
- **dynamicity**: dynamic instrument configuration and tuning;
- **robustness**: robust to inter-instrument communication losses, robust to a single instrument loss;
- **resilience**: resilient instruments: 1) implies possible mitigations for the performance degradation; 2) autonomous recalibration to maintain the measurement data quality;
- **mobility**: inter-instrument mobility of information, moving observatory within an orbit plane, moving observatory from one plane to another.

**Communication Spacecraft.** In this mission subclass, orbiting communication spacecraft fly highly elliptic orbits and are used to transfer data on Earth. With regard to the elliptic orbit, there are two possible orbits [9]:

- **Molniya Orbit**: Highly elliptic with a 12-hour period where the spacecraft moves relatively slowly in the apogee region. For a 24-h regional service, at least three spacecraft are needed.
- **Tundra Orbit**: Elliptical orbit with a period one sidereal day (23h 56 min). It can provide 24h coverage with a minimum of only two spacecraft. The orbital parameters can be chosen so that the spacecraft does not traverse the radiation belts.

Common challenges to these subclasses of missions are:

- **orbit perturbations**: third-body forces may perturb the perigee height, causing atmosphere reentry;
- **passage through the Van Allen radiation belts**: accelerated degradation of power and electronic systems;
- **variation in satellite range and range-rate**: may have a number of impacts upon the communication payload design:
  - variation in time propagation;
  - frequency variation due to Doppler effect;
  - variation in received signal power;
  - change of ground coverage pattern during each orbit.
The following is a list of deducted autonomy requirements for this class of missions (also add the requirements of Polar LEO/Remote-Sensing Satellite Missions described in Section VII.A.1):

- **self-* requirements (autonomicity):**
  - self-protection (autonomously detect when the spacecraft is passing through the Van Allen radiation belts to cover the electronic systems and minimize the power usage);
  - self-optimization (autonomously optimize the communication payload by taking into consideration the impact caused by: variation in time propagation, frequency variation due to Doppler effect, variation in received signal power, and change of ground coverage pattern during each orbit);
  - self-reparation (autonomously detect communication system problems and repair);
  - self-scheduling (autonomously determine when to emit transmissions);

- **knowledge:** Van Allen radiation belts, Doppler effect, ground coverage pattern, Moon gravity, Sun gravity, Molniya Orbit/Tundra Orbit;

- **awareness:** signal power awareness, Moon’s gravitational force awareness, Sun’s gravitational force awareness;

- **monitoring:** Van Allen radiation belts, Moon position, Sun position;

- **adaptability:** adapt to changes in the ground coverage pattern, adapt to changes in time propagation, adapt to changes in communication frequency;

- **dynamicity:** dynamic communication frequency, dynamic ground coverage pattern, avoid radiation belts;

- **robustness:** robust to radiation;

- **resilience:** resilient communication payload;

- **mobility:** moving satellite within the orbit plane.

B. Interplanetary Missions

Interplanetary missions involve more than one planet or planet satellite and usually general trajectory information needs to be developed and understood for each mission [9]. Interplanetary trajectories are influenced by perturbations caused by the gravitational influence of the Sun and planetary bodies within the solar system. Software tools are used to compute a number of trajectories. Figure 3 presents possible trajectories for Mars missions’ opportunities.

A common challenge in the design and implementation of interplanetary missions is trajectory optimization. There are three possible sub-classes of interplanetary missions with regard to their operation mode.

1) **Small Object Missions** – “To Orbit” and “To Land” Missions. The objective of such missions is to investigate the properties of minor bodies in the solar system, mainly asteroids, comets and the satellites of the major planets [9]. Particular interest lies in the hypothesis that these small objects might help us understand the genesis and evolution of the solar system.

2) **Missions using Low-Thrust Trajectories.** Such missions use spacecraft for orbit control activities in GEO (Geostationary Earth Orbit), drag compensation in LEO (Low Earth Orbit), Lunar orbit missions and missions to comets and asteroids [9]. These missions often have a complex mission profile utilizing ion propulsion in combination with multiple gravity-assist manoeuvres.
3) **Planetary Atmospheric Entry and Aeromaneuvering Missions.** Such missions require entering the atmosphere of a planet to take probes or land [9]. Principle effect of an atmosphere on a satellite trajectory is to reduce the energy of the orbit. These missions include some degree of aeromaneuvering, with a “mass penalty” – additional propellant mass is required to protect the vehicle from the dynamic pressure and thermal effects of aeromaneuvering.

The following is a list of generic autonomy requirements for interplanetary missions:

- **self-* requirements (autonomicity):**
  - **self-trajectory** (autonomously acquire the most optimal trajectory; adapt to trajectory perturbations);
  - **self-protection** (autonomously detect the presence of radiation);
  - **self-scheduling** (autonomously determine what task to perform next - equipment onboard should support the tasks execution);
  - **self-reparation** (broken communication links must be restored autonomously; when malfunctioning, component should be fixed autonomously where possible);

- **knowledge:** mission objectives, payload operational requirements, instruments onboard together with their characteristics (e.g., instruments resolution), Van Allen radiation belt, ground stations, communication links, data transmission format, eclipse period, altitude, communication mechanisms onboard, Earth gravity, Moon gravity, Sun gravity, solar system, target planet characteristics;

- **awareness:** trajectory awareness, radiation awareness, air resistance awareness, instrument awareness, sensitive to thermal stimuli, gravitational forces awareness, data-transfer awareness, speed awareness, communication awareness;

- **monitoring:** electronic components onboard, surrounding environment (e.g., radiation level, space objects), planned operations (status, progress, feasibility, etc.);

- **adaptability:** adaptable mission parameters, possibility for re-planning (adaptation) of operations, adapt to loss of energy, adapt to high radiation, adapt to weak a satellite-ground station communication link, adapt to low energy;

- **dynamicy:** dynamic communication links;

- **robustness:** robust to temperature changes, robust to trajectory perturbations, robust to communication losses;

- **resilience:** loss of energy is recoverable, resilient to radiation;

- **mobility:** information mobility, changing trajectory.

**VIII. Conclusion and Future Work**

Contemporary software-intensive systems, such as modern spacecraft and unmanned exploration platforms (e.g., ExoMars) generally exhibit a number of autonomic features resulting in complex behavior and complex interactions with the operational environment, often leading to a need of self-adaptation. To properly develop such systems, it is very important to properly handle the autonomy requirements. In this paper, we have presented the classes of space missions and our first cut of generic autonomy requirements for these classes of missions.

Future work is mainly concerned with further development of the Autonomy Requirements Engineering Model involving formal methods for autonomy requirements specification, verification and validation.

**Acknowledgement**

This work was supported by ESTEC ESA (contract No. 4000106016) and by Science Foundation Ireland grant 03/CE2/I303_1 to Lero—the Irish Software Engineering Research Centre at University of Limerick, Ireland.

**References**


