doi.org/10.35321/term27-08

Terminology as the Basis for Building Engineering Feature-based Models

RICARDO EITO BRUN
Universidad Carlos III de Madrid

ABSTRACT

Satellite operations require the combined use of different tools to support engineering activities and to control the spacecraft. This communication is managed by the *Monitoring and Control System* (MCS) that receives telemetry data from the spacecraft and releases telecommands to keep the satellite's attitude and flight path. These complex systems are developed as open platforms that can be extended and customised to support mission-specific requirements and objectives. As a general rule, it can be stated that these software applications are good candidates for implementing variability mechanisms in a structured, planned way and that their functionality is a good candidate to analyse the feasibility of applying feature-based modelling techniques. This paper describes the use of terminology analysis to build a feature model to support requirements analysis for this type of software-based systems.

KEYWORDS: Technical terminology, Feature-based models, Aerospace engineering, Terminology extraction

ANOTACIJA

Palydovų operacijoms reikalingi įvairūs įrankiai, skirti užtikrinti sklandų inžinerinį darbą ir valdyti erdvėlaivį. Šį ryšį valdo *Stebėjimo ir kontrolės sistema* (SKS), kuri gauna telemetrinius duomenis iš erdvėlaivio ir duoda telekomandas, kad palaikytų palydovo padėtį ir skriejimo trajektoriją. Šios sudėtingos sistemos yra sukurtos kaip atviros platformos, kurias galima išplėsti ir pritaikyti pagal konkrečios misijos reikalavimus ir tikslus. Galime teigti, kad paprastai šios taikomosios programos gali būti naudojamos norint struktūruotai ir planuotai įdiegti variantiškumo mechanizmus, o jų funkcionalumas leidžia analizuoti požymių modeliavimo metodikų pritaikymo galimybes. Šiame straipsnyje aprašomas terminologinės analizės panaudojimas požymių modeliui, palaikančiam reikalavimų analizę šio tipo programinėms sistemoms, sukurti.

ESMINIAI ŽODŽIAI: techninė terminologija, požymių modeliai, aviacijos inžinerija, terminų atpažinimas

1. INTRODUCTION

NASA glossary defines satellites as: "*a free-flying object that orbits the Earth, another planet, or the sun.*[1]" Satellites are a type of spacecraft that travels in a regular, clearly defined orbit around the centre of gravity of another celestial body (Garner 1996: 4). Since the launch of the first artificial satellites – *Sputnik 1* on October 4th, 1957 and *Explorer 1* on January 31st, 1958, a large number of satellite missions have been launched with different purposes: astronomical exploration, provision of communication and navigation services, earth observation, reconnaissance, and scientific missions. Satellites are complex aerospace systems made up of ground- and space-based elements: the spacecraft must be operated by a controlling element on Earth and remain in contact with it. Today, scientific progress and services for users depend on satellites and satellite constellations. Relevant examples include the *Hubble Space Telescope* (HST), navigation systems like GPS (*Global Positioning System*), GLONASS and the European Galileo, UARS (*Upper Atmosphere Research Satellite*) or GOES (*Geostationary Operational Environment Satellite*). The total number of satellites launched since 1957 – according to the US SSN[2] catalogue – Is close to 18200; UNOOSA's *2017 Index of Objects Launched into Outer Space* reports 4635 satellites currently orbiting the planet, with an increment of 357 satellites (8.95%) concerning the previous year[3]. The purpose of this article is to demonstrate the need of applying terminology management and extraction to support the development of feature-based models to organize the concepts that describe the functions of the software applications used for satellite monitoring and control.

Satellites are classified by purpose and type of orbit. The purpose refers to the services the satellite is intended to provide. Regarding the orbit, a distinction is made between *Low Earth Orbit* (LEO), *Medium Earth Orbit* (MEO), and *Geostationary Orbit* (GEO). LEO satellites – used for science and Earth observation – follow an elliptical orbit; as their visibility from ground stations is limited, data are stored on-board and sent to the ground

[1] See https://www.grc.nasa.gov/www/k-12/TRC/laefs/laefs_s.html#satellite [accessed 2020-08-01].

[2] The *US Space Surveillance Network* is in charge of the detection, tracking, cataloguing and identification of artificial objects orbiting the earth. The catalogue is available at: https://www.space-track.org/#/ssr [accessed 2020-08-01].

[3] *United Nations Office for Outer Space Affairs*. See https://www.pixalytics.com/sats-orbiting-earth-2017/

stations when the aircraft becomes visible. GEO satellites are used for telecommunication and meteorological purposes; telecommunication satellites receive radio frequency (RF) signals from the Earth, amplify them, shift their frequency, and transmit them back to the Earth. They orbit 36000 km above the Equator (in the same plane), and their rotation is synchronous with the rotation of the Earth, staying above the same point at the Equator all the time)[4]. Similar to satellites, deep space scientific missions also need similar monitoring and control for telecommand, and telemetry reception.

Satellite missions require dedicated staff to complete real-time, continuous monitoring of the status and position of the satellite. Mission control is the set of tasks executed after launch by operation engineers with the help of software applications, and involves the exchange of data between the ground and space segments for monitoring and control the status of the satellite's onboard subsystems. In scientific missions, it is also necessary to receive the information from the satellite payload and deliver it to the end-users (scientific community). Typical functions executed during mission control include the reception and analysis of telemetry, telecommanding, and tracking. Telemetry is the data transmitted from the satellite to the Earth informing of the status and conditions of the satellite and its subsystems. Telecommands are the orders transmitted from the ground to the spacecraft to configure and operate it. (Uhlig, Sellmaier, and Schmidhuber 2015: 232). Tracking and station keeping, also known as *ranging*, consists of the monitoring and determination of the flight path and position using RF techniques and signals.

From a software engineering perspective, satellite control requires different applications and tools for flight dynamics, mission planning, telemetry and telecommanding, network control and routing, as well as interfaces between them. This complexity has led to the provision of different solutions by space agencies. One of the most relevant milestones in the development of MCS software was the decision of the ESA *Ground Systems Engineering* department to develop and license to the European industry a set of software applications distributed under the name MICONYS®

[4] GEO satellites are distributed in a limited area in space. Orbit slots are assigned by the *International Telecommunication Union* (ITU) to avoid collisions, a risk related to the space debris topic that is receiving greater attention today.

(*Mission Control System*). MICONYS and its SCOS-2000® component are probably the best-known examples of this ESA's policy for software development and innovation and technology transfer (Kaufeler, Jones and Karl 2001). SCOS-2000 supports telecommanding, telemetry reception, display, and archival. SCOS-2000 was the result of the experience acquired by ESA in the development and operation of previous similar systems: MSSS, SCOS-1, and SCOS-2. Today, ESA is developing the new MCS system, aimed to replace SCOS-2000 in the future. Its name is *European Ground Systems – Common Core* (EGS-CC). Similar to SCOS, the project has the purpose of developing a common infrastructure to support the monitoring and control of space missions in the pre- and post-launch phases, using modern technologies and service-oriented architectures (Pecchioli et al. 2012). The development of EGS-CC is not only under ESA responsibility: European national space agencies (CNES, *UK Space Agency,* and DLR) and industrial companies (*AIRBUS Defence and Space*, *Thales Alenia Space* and *OHB Systems*) are part of the project.

Besides ESA projects, there are other initiatives aimed to develop a generic MCS software system. NASA has completed similar projects, most of them in-house developments. The *Goddard Space Flight Centre* developed two systems, ITOS and ASIST, for managing missions like WMAP (*Wilkinson Microware Anisotropy Probe*), IMAGE (*Imager for Magnetopause-to-Aurora Global Exploration*), EO-1 (*Earth Observing 1*), ST-5 (*Space Technology 5*), SDO (*Solar Dynamics Observatory*) or LRO (*Lunar Reconnaissance Orbiter*) (Pfarr et al. 2007).

In all these cases, we are dealing with a complex system that needs to implement different functions that can be activated or not depending on the satellite and the mission's characteristics. Due to that, these software-intensive systems are good candidates to be developed using product-lines and feature-based modelling techniques, which rely on a clear and well-organised organization of concepts to understand the domain and system needed capabilities.

2. FEATURE-BASED MODELLING AS AN ORGANIZATION OF CONCEPTS

Feature-based modelling is one of the techniques applied in software product-line engineering, a discipline for building reusable software programs that became popular at the end of the nineties. SPLE intends to

build reusable components and artefacts that can be later combined to build products suited to the needs of a specific client and context. Experiences in SPLE are widely documented in the professional and academic literature. Capilla (2013) included several case studies where SPLE was applied with success: *Boeing*'s operational, mission-critical flight programs for avionics and cockpit functions, *Bosch*'s engine-control software for gasoline systems, *Hewlett Packard*'s printer software, *Toshiba*'s power generation, and transmission equipment and *General Motors*' control software for powertrains. The benefits of software product lines (SPL), according to Apel et al. (2013: 9), include the tailoring of software products to the specific needs of the clients, reduced cost – as a set of reusable assets can be combined in different ways to generate new products –, improved quality and time-to-market.

Features and feature-based modelling are relevant concepts in the development of SPL. ISO/IEC/IEEE 24765:2010, *Systems and software engineering — Vocabulary*, offers a general definition of *features*, taken from IEEE Std 829:2008 *IEEE Standard for Software and System Test Documentation*: "*A distinguishing characteristic of a system item. NOTE: includes both functional and non-functional attributes such as performance and reusability*." Kang and Lee (2013, 28) define features as "*abstract concepts effectively supporting communication among diverse stakeholders of a product line, and therefore, it is natural and intuitive for people to express commonality and variability of product lines in terms of features.*"

Features serve different purposes in the product ideation and development process. They are means to communicate the product characteristics and support the identification of requirements; they are also the concepts that guide design and implementation decisions.

The development of a product-line comprises two complementary life cycles:

- Domain engineering, and
- Application engineering.

IEEE 1517-2010 standard defines domain engineering as: "*Life cycle consisting of a set of processes for specifying and managing the commonality and variability of a product line.* Domain engineering analyses the domain of a product line and develops a set of reusable artefacts. These artefacts include software requirements, design elements, test cases and

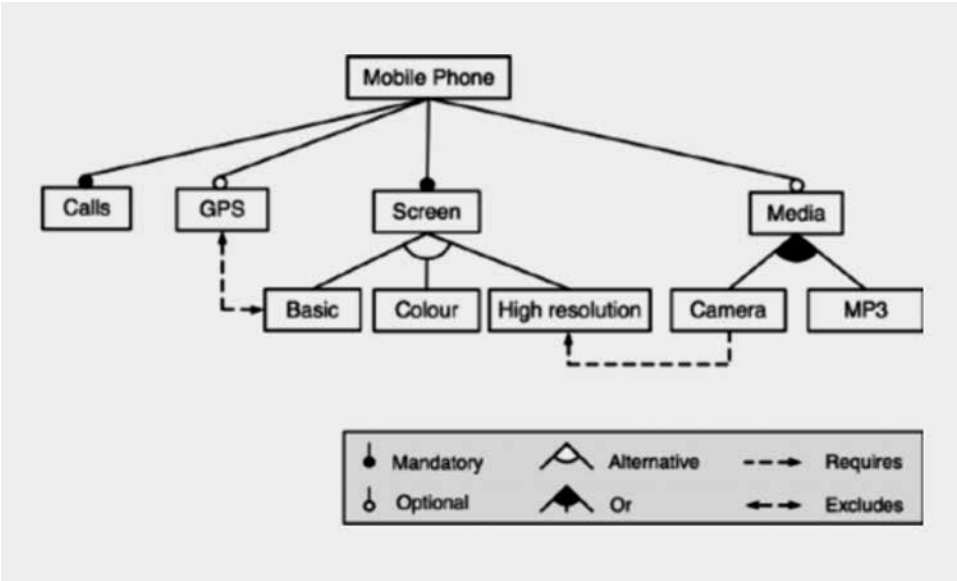
procedures, user documentation, etc. Domain analysis can be seen as a sort of requirements engineering for the whole product line, including the identification of anticipated variability. The main artefact generated by domain analysis is the *feature model* that will specify and describe the products within the line.

Feature modelling is a diagramming technique that was introduced in the nineties with the FODA (*Feature-Oriented Domain Analysis*) methodology. FODA provided primitives for representing structural relations (composition, generalization, and specialization), optionality, alternativeness, and mutual dependencies. It was later reviewed by different authors (Kang and Lee 2013: 30–31). Feature diagrams are the visual representation of feature models, where features are represented as boxes in a hierarchical tree. Each node has an attached label with the name of the feature. The hierarchical arrangement of the features creates parent-child relationships. If a child feature is selected, its parent must also be selected. Diagrams can make a distinction between mandatory and optional features, and identify the combinations of features that are valid. In particular, feature diagrams can represent:

- Abstract features, which are used to organise the features in the tree but are not bound to implementation artefacts. They are represented with grey boxes.
- Concrete features, which correspond to implementation artefacts and are represented with white boxes.
- Mandatory features, which have a filled bullet on top of the upper border of their box.
- Optional features, which have a non-filled bullet on top of the upper border of their box.
- The need of selecting just one of the child features of a specific parent (exclusive OR, XOR, or *one-out-of-many*). It is represented with an empty arc at the lower border of the parent feature's box.
- The possibility of selecting more than one child features of a specific parent (OR or *some-out-of-many*). It is represented with a filled arc at the lower border of the parent feature's box.
- Dependencies between features, represented by arrows with textual annotations.

The diagram below shows a typical feature diagram with the conventions described above:

Fig. 1. Feature diagramming example (Source: Gargantini 2015)



A feature model should include information additional to the diagram. Apel et al. (2013: 27) indicate the possibility of adding these data:

- "*Description of a feature and its corresponding set of requirements.*
- *Relationship to other features, especially hierarchy, order, and grouping.*
- *External dependencies, such as required hardware resources.*
- *Interested stakeholders.*
- *Estimated or measured cost of realizing a feature.*
- *Etc.*"

The development of a feature model to represent the functional characteristics of software applications for satellite control must start from a clear understanding and modelling of the concepts that make up the domain. To this end, activities related to terminology and terminography, the identification of terms, concepts, and their relationships provide the basis to build the target model.

3. WORK METHODOLOGY

The proposed feature-model has been completed following these steps:

- Identification and review of professional and academic literature published in this area. This involves searching for information about the approach followed in aerospace projects led by entities like ESA (*European Space Agency*) or NASA (*National Aeronautics and Space Administration*).
- Development of a glossary based on the analysed literature, applying terminology management techniques to record information about terms, relationships between them, definitions identified in the documents, and the context where the terms are used.
- Creation of a feature-based model that represents the functions offered by satellite monitoring and control software applications, using the glossary as a basis.

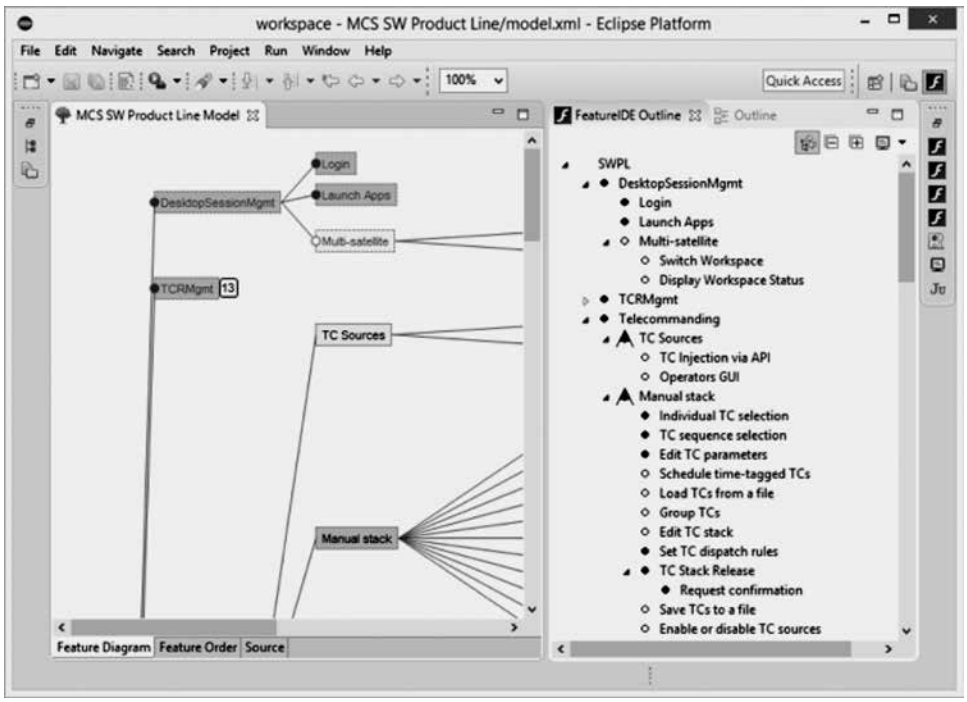
The inputs used to identify terms included a subset of technical documents that describe the selected product: operation manuals, white papers, and training materials. The feature model is a hierarchical tree of features marked as mandatory or optional, with dependencies between them. In the case of the software application under analysis, besides the identification of optional and mandatory features, additional product line variability requirements were identified. In particular, some of the functions supported by the software under analysis require overwriting or customizing existing code. The identification of these variability cases was made with the support of experts who develop their activity in the development and customization of this type of software application. Personal interviews were made to collect that information.

The tool selected to create the feature model and diagrams is *FeatureIDE*[5]. This is an open-source tool based on Java and Eclipse, developed by staff at the *Otto-von-Guericke-Universität Magdeburg*. With *FeatureIDE*, it is possible to create a feature model using a graphical editor, mark features as mandatory, optional, or abstract, and build the hierarchical tree. Once the feature model is built, you can create different *configurations*: selections of features that will be used to generate the target software applica-

[5] See https://FeatureIDE.github.io/. Last checked: 01-03-2018.

tion source code. Unfortunately, the tool does not provide capabilities to manage terms or terminology units, which makes necessary the use of complementary tools.

Fig. 2. FeatureIDE (created by the author)



4. PRESENTATION OF RESULTS AND DISCUSSIONS

This section summarizes one section of the feature model for Mission Control Systems (MCS) software applications. The analysis of the concepts extracted from the documents led to an organization of the system functionalities into these areas:

- **Desktop and Session Management**, which provides the functions to log in, start a session, and launch the different applications. In the case of managing multiple satellites, the user will be able of switching between satellites' workspaces.
- **Telemetry chain**, which includes **Telemetry processing**, **Alarm Manager**, and **Telemetry display**.

- **Telecommands chain,** which includes **Telecommands uplink**, **Manual stack**, **Scheduled stack**, and **Telecommands history**.
- **TT&C/TCR Bridge**, which provides visibility and control on the interface with the ground station for Telemetry, Command, Ranging (TCR) and Antenna Pointing data flows.
- **Events Log**, which records events identified during the system operations and lets operators browse and search the entire history of events.
- **On-Board Memory Management**, which provides facilities for managing (downloading, editing, and uplinking) the spacecraft's memory image.

The analysis of the documentation and terms lead to the creation of feature models for these functions of the system. The particular case of Telemetry management is summarized as an example. It can be observed how definitions based on existing literature and technical documentation are provided for the identified features:

The term *telemetry chain* refers to the different processes involved in the processing of telemetry: reception from the TT&C, packetization, archive, and distribution to consumer applications:

- **Telemetry packetization**. MCS takes the telemetry source packets based on the telemetry packet standard, adds metadata (e.g. the telemetry stream quality), and converts them into one or more MCS telemetry internal packets. The criteria for the generation of packets from the incoming data may change depending on the satellite, mission, and BBE protocols.
- **Packets quality check**. MCS checks continuously the quality of the telemetry using checks performed either by the TT&C or by MCS itself. Telemetry quality metadata are appended to each internal packet. Depending on the result of those checks, the packet can be classified as suitable for processing (GOOD) or not suitable for processing (BAD). Telemetry drops shall also be identified during the process.

  A typical quality check is the *Frame sequence check* that assesses whether the frames are received in sequence or not.

Events are raised in case these checks are not satisfied. The operator can select the behaviour to apply when BAD telemetry frames are identified (process or discard them).

- **Time quality checks**. These checks verify whether the time reported by the TT&C for each telemetry stream and the MCS local times are not too far. The difference between the current MCS time and the time tag in the telemetry frame (set by the ground station upon the reception of the frame), is within a configurable time window. Event messages are raised if these checks fail. Time quality checks require the time-stamping of the telemetry packets.
- **Telemetry decommutation**, which extracts telemetry parameter samples from the packets[6]. It involves parameter extraction, bit decoding, calibration, out of range checking, validity checking, and parameter dating.
  - o **Telemetry parameter extraction** locates and extracts the raw values of telemetry parameters from the packets. Raw values are usually signed, unsigned integers, or floating-point numbers. This process is trivial for fixed telemetry as all the required information about the parameter location is available at the satellite MIB, but is more complex with programmable telemetry.
  - o **Bit decoding** obtains the raw value from the value encoded in the telemetry packet. In a typical situation, the raw value matches the one encoded in the packet, but there are cases in which bit- (such as bit reversal or two's complement) or byte-based transformation is needed.
  - o **Derived or synthetic parameters**. They are not acquired from the spacecraft but algorithmically derived at the ground using acquired telemetry parameters as operands. They are calculated with formulas written in scripting languages. They are calculated before calibration, validity checks, and parameter dating.

    A special type of derived parameter is *hardcoded derived parameters* (HCDP), written in low-level programming languages

[6] The parameters may be synchronous (downlinked in telemetry as soon as sampled on-board) or asynchronous (not downlinked in telemetry as soon as sampled on-board, but stored in on-board memory and downlinked afterwards). Parameters are usually sampled and encoded once or several times per format, with the exception of sub commutated parameters, which are not sampled and encoded in all formats and super commutated parameters, which are sampled and encoded multiple times.

like C++, Fortran, or C. Their modification requires the recompilation of source code.

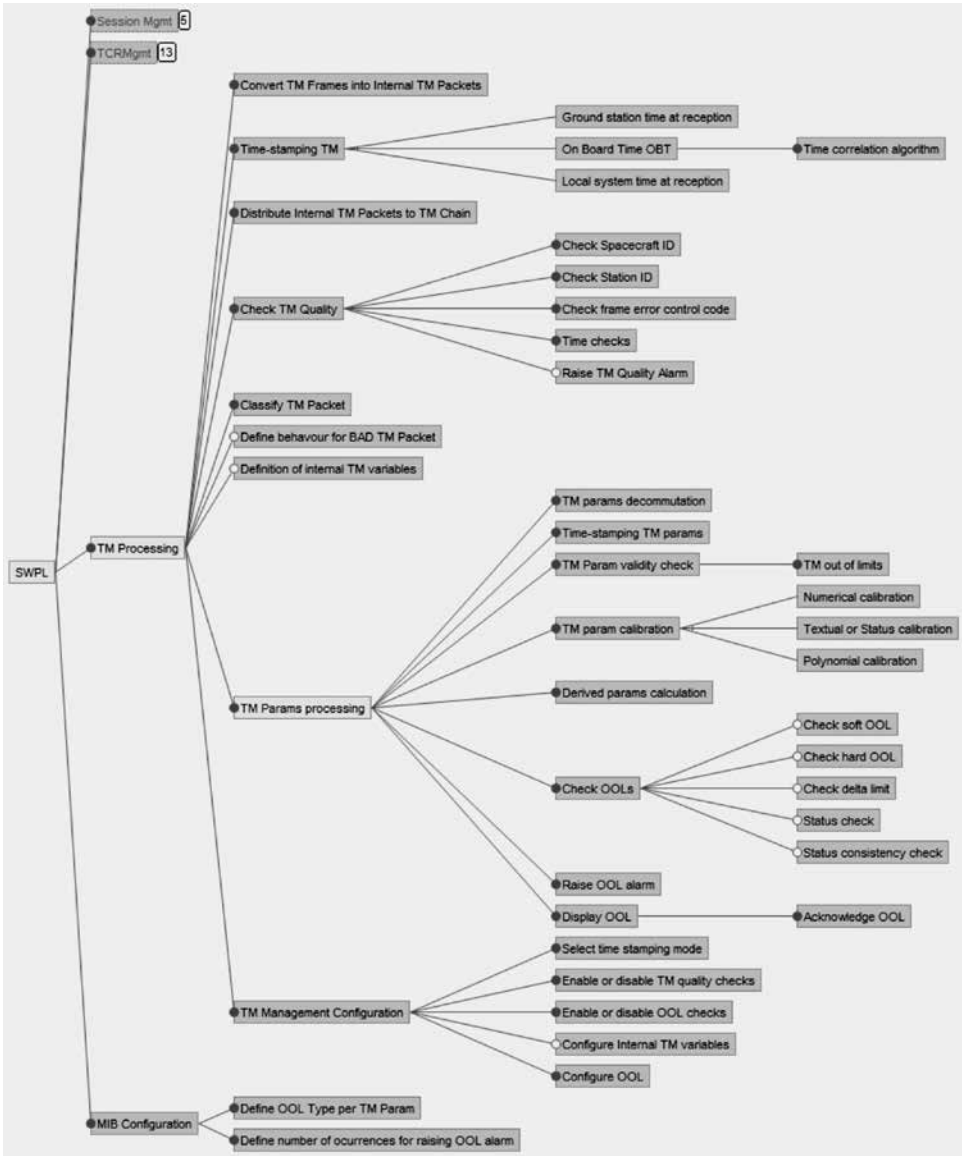
- **Telemetry parameter dating** gives a date to the parameter samples, using the time of the packet that contains them and adding an offset as specified in the satellite MIB.
- **Out of range (OOR) control** analyses whether the raw value of a telemetry parameter is outside of the defined calibration range. Out of range telemetry parameters are stated invalid with category *OOR*. OOR is used to assess the validity of the telemetry parameter, and let know whether its value is useful or meaningless.

In addition, out of range parameters must be displayed and acknowledged by the operators. Other conditions to declare a parameter as invalid include: the parameter is expired, or – in the case of derived parameters -, they were calculated using invalid parameters. Whatever the case, the telemetry parameter invalidity metadata shall detail the reason for this status.

- **Calibration of telemetry parameters** transforms their raw values into a human-readable form (engineering or calibrated values with the units that correspond to the physical magnitude). Calibration can be made using different methods:
  - **Numerical or linear discrete calibration**: linear interpolation is used from a calibration curve made up of (X, Y) pairs, where X stands for the raw value and Y for the engineering value.
  - **Textual, digital, or status calibration**: it associates text strings to parameters' raw value ranges. Textual values can be obtained from discrete values or a range of them. A simple example is the ON / OFF status of the on-board units, which could be obtained from a one-bit parameter where 0 means OFF and 1 means ON.
  - **Polynomial Calibration**: it uses an up to fifth-degree polynomial to translate raw values into engineering values.

Without going into more details on the extracted features (just a subset are described to show how their identification requires sound terminology work), the next diagram presents the main features identified for this functional area:

Fig. 3. TM Processing – Features Fig 1: FeatureIDE (created by the author)



5. CONCLUSIONS

This paper describes the development of a feature model for satellite control software applications, using as inputs technical documents (operation manuals, white papers, and training materials) that describe the functionalities of the type of system under study. The first step is the

identification and extraction of terms through the inspection of the documents, identifying definitions for them through the context where they are applied. This collection of terms is further analysed to identify features, that were arranged in the feature model and diagram attending to functional criteria, within twelve main categories: session management, ground station connection management, telemetry processing, telemetry data display, telecommands processing, mission archive, etc.

For each functional area, the description of the main concepts and the characteristics that must be supported by a software system were summarized and represented in feature diagrams created with the *FeatureIDE* tool. The definitions and the context of use were also the basis to establish the relationship between the features and for the identification of variability opportunities. Three main reasons for variability were identified:

- Mission specific configuration parameters usually managed with public configuration files.
- Selection of optional features, depending on the type of the mission: e.g., position-based telecommands that are characteristics of LEO satellites for Earth observation.
- Customization of the mission-specific services that extend the basic services and functions defined in telemetry and telecommands standards like PUS.

The work demonstrates the need of applying terminology-related techniques to collect, analyse, and document the meaning of terms, and to extract relationships that can be later used for different purposes. In the case of feature-based engineering models, terms constitute the basis to describe the functions of the software and systems unambiguously and to identify the characteristics of complex systems. Consistency in the use of terms is also needed to ensure that the different agents involved in subsequent engineering tasks (design, implementation, testing, etc.) share a common understanding of the system features, avoiding errors caused by a poor definition of the domain area the system is intended to support.

REFERENCES

Apel et al. 2013: *Feature-Oriented Software Product Lines: Concepts and Implementation*, Berlin: Springer.

Capilla Rafael 2013: Variability Scope. – *System and Software Variability Management: Concepts, Tools, and Experiences*, Berlin: Springer, 43–56.

CCSDS 520.0-G-3 (*Mission operations services concept*), December, 2010.

Clements, Paul C. and Northrop Linda M. 2002: *Software Product Lines: Practices and Patterns*, Boston: Addison-Wesley Professional.

Gargantini Angelo 2015: Generating Tests for Detecting Faults in Feature Models. – *2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 – Proceedings.*

Garner John T. 1996: *Satellite Control: a Comprehensive Approach,* Chichester: John Wiley and sons, xv.
Jones, M; Ssrensen E. M. and Wolff T. 1993: "A review of mission planning systems". – *JPL SpaceOps 1992: Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations* (SEE N94-23832 06-66), 219–224. Available at: https://ntrs.nasa.gov/search.jsp?R=19940019393
Kang Kyo-Chul and Lee H. 2013: Variability Modelling. – *System and Software Variability Management: Concepts, Tools, and Experiences*, eds. R. Capilla, J. Bosch and K. C. Kang, Berlin: Springer, 25–42.
Kaufeler J.-F., Jones M., Karl H.-U. 2001: Promoting ESA Software as a European Product: The SCOS-2000 Example. – *ESA Bulletin* 108, 72–77.
Pecchioli Mauro et al. 2012: Objectives and Concepts of the European Ground Systems Common Core (EGS-CC). – *SESP 2012: Simulation and EGSE facilities for Space Programs*, ESTEC Noordwijk, 25–27 September. Available at: http://www.egscc.esa.int/downloads/20120925_SESP_2012_egscc_objectives_and_concepts.pdf.
Pfarr Barbara et al. 2007: Proven and Robust Ground Support Systems – GSFC Success and Lessons Learned. (2008). – *IEEE Aerospace Conference*, paper # 1219, version 6. Available at: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080045437.pdf
Uhlig Thomas; Sellmaier Florian and Schmidhuber Michael 2015: *Spacecraft Operations*, Wien: Springer.

TERMINOLOGIJA KAIP INŽINERINIŲ POŽYMIŲ MODELIŲ KŪRIMO PAGRINDAS

Santrauka

Palydovai yra erdvėlaiviai, kurie skrieja aplink kito dangaus kūno gravitacijos centrą pastovia, aiškiai apibrėžta orbita (Garner 1996: 4). Nuo pirmųjų dirbtinių palydovų paleidimo – „Sputnik 1" 1957 m. spalio 4 d. ir „Explorer 1" 1958 m. sausio 31 d. – buvo paleista daug palydovų misijų, kuriomis siekta skirtingų tikslų: astronominiai tyrimai, komunikacinių ir navigacinių paslaugų teikimas, Žemės stebėjimas, žvalgyba ir mokslinės misijos. Palydovai turi būti valdomi Žemėje esančiu valdymo elementu ir palaikyti su juo ryšį. Šį ryšį valdo *Stebėjimo ir kontrolės sistema* (SKS), kuri gauna telemetrinius duomenis iš erdvėlaivio ir duoda telekomandas, kad palaikytų palydovo padėtį ir skriejimo trajektoriją. Galime teigti, kad šios taikomosios programos gali būti naudojamos norint struktūruotai ir planuotai įdiegti variantiškumo mechanizmus, o jų funkcionalumas leidžia analizuoti požymių modeliavimo metodikų pritaikymo galimybes.

Požymiai yra naudojami norint perduoti produkto charakteristikas ir palaikyti reikalavimų nustatymą. Jie taip pat naudojami priimant projektavimo ir įgyvendinimo sprendimus. Straipsnyje aprašomas požymių modelio sukūrimas remiantis šios srities profesinės ir akademinės literatūros apžvalga, žodynėlio, nurodančio ryšius tarp terminų, sudarymu ir požymių modelio, apimančio taikomųjų palydovų stebėjimo ir kontrolės programų funkcijas, sukūrimu. Įvesties duomenys, skirti identifikuoti terminus, apėmė techninių dokumentų, aprašančių pasirinktą produktą, pogrupį: valdymo vadovus, baltąsias knygas ir mokomąją medžiagą. Požymius nurodančios sąvokos ir terminai yra išdėstyti hierarchinės struktūros principu, o tarpusavio sąsajoms naudojamas *FeatureIDE* įrankis.

Gauta 2020-09-07

Ricardo Eito Brun
Universidad Carlos III de Madrid
Calle Madrid, 126, 28903 Getafe, Madrid, Spain
E-mail reito@bib.uc3m.es