Variability-Aware Design of Space Systems: Variability Modelling, Configuration Workflow and Research Directions

Sami Lazreg
sami.lazreg@uni.lu
SnT, University of Luxembourg
Luxembourg, Luxembourg

Vladyslav Bohlachov
vladyslav.bohlachov@uni.lu
SnT, University of Luxembourg
Luxembourg, Luxembourg

Loveneesh Rana
loveneesh.rana@uni.lu
SnT, University of Luxembourg
Luxembourg, Luxembourg

Andreas Hein
andreas.hein@uni.lu
SnT, University of Luxembourg
Luxembourg, Luxembourg

Maxime Cordy
maxime.cordy@uni.lu
SnT, University of Luxembourg
Luxembourg, Luxembourg

ABSTRACT

Designing spacecraft such as satellites is known as an extremely difficult problem. It requires complex decision making at different concerns from mission requirements to system architectures. Such complexity might lead to design system architectures that are not feasible in practice or do not fulfil the requirements. Moreover, the lack of automation to assess high-level designs reduce the benefits of the early design phases by missing more suitable designs and increasing the design time.

In this paper we discuss potential research directions and propose a potential framework that aims to i) drive the engineers in the different design steps ii) capture the feasible system architectures regarding the requirements and constraints, iii) refine and assess selected architectures through simulations. Our framework is model-driven and built on two complementary modules. The first module is a configurator based on attributed features models that support engineers in the design process of system architectures. Second, a simulation engine refines and assess automatically the selected system architectures with respect to the requirements.

CCS CONCEPTS

• **Computing methodologies** → **Model development and analysis**; • **Applied computing** → **Aerospace**.

KEYWORDS

variability modelling, configuration, design alternatives.

ACM Reference Format:
Sami Lazreg, Vladyslav Bohlachov, Loveneesh Rana, Andreas Hein, and Maxime Cordy. 2022. Variability-Aware Design of Space Systems: Variability Modelling, Configuration Workflow and Research Directions. In *Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems (VAMOS '22), February 23–25, 2022, Florence, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3510466.3510472

1 INTRODUCTION

After 203 days in space – and a journey of 472 million kilometers – NASA's Perseverance rover fell towards the dusty surface of Mars. The silver and burnt-orange parachute deployed, the rover slowed, and its radar locked onto the surface. Within a few short minutes, many of the rover's complex components had successfully operated. The achievements of space systems, like those of Perseverance, have made them increasingly popular across numerous industries and among the public. However, the design and development of space systems is complex, error-prone and very costly. Engineers who develop these systems are under extreme pressure to deliver optimal solutions that will not fail – and they have to do so quickly [31].

Design activities are of fundamental importance to build quality space systems. A space system design lifecycle consists of several phases that gradually build up the space system. The very first phase is known as Conceptual Design (CD). It is also the most important phase because crucial decisions are frozen at this phase account for 80% of the total lifecycle cost [12]. These decisions can determine a system to be a success or a failure, be cost-effective or cost-inefficient, be feasible or not.

At the beginning, there exist only the requirements defining the purpose for which the system is needed. The CD phase takes these intangible requirements and develops the concept of a tangible system. This concept is the first blueprint of the system that acts as input for the next design phases where the design is further refined and optimized. This blueprint consists in selecting and "sizing" the system components, setting their parameters. Because CD acts as such an early phase, it has to explore an immense space of design alternatives in order to produce an appropriate baseline concept.

Industrial practices for exploring such a huge design space often rely on engineers' experience and creativity, trials and errors, and different sets of guidelines that lack consistency and automated support. These practices result in an overly long CD process, uncertainty in the produced blueprint, and the ignorance of numerous – and potentially better – design alternatives. Indeed, engineers typically select the system components and their parameters using spreadsheets. They, then, solve differential equations based on the selected parameter values in order to compute analytic information about the produced design and, if need be, iterate over the previous choices to improve it.

This work results from the collaboration between software engineering researchers and space system engineering researchers. Together, we conjecture that variability modelling and analysis methods that have been developed for software systems can substantially improve the space system design process, in particular

VAMOS '22, February 23–25, 2022, Florence, Italy

ACM ISBN 978-1-4503-9604-2/22/02.
https://doi.org/10.1145/3510466.3510472

the CD phase. We foresee the development of engineering tools that guide engineers towards the optimal designs, automate most exploration and analysis steps, and enable the thought sharing of expertise.

As a preliminary endeavour to unite our two fields of research, in this paper, we present our application of variability modelling techniques to a real-world application case of space system design. We consider, more precisely, a communication satellite system [55]. We modelled the set of design choices as a feature model [37], where features/feature attributes represent space system components and their parameters. We, then, use the FeatureIDE tool to enable engineers to specify these design choices in order to configure the design and produce a suitable blueprint that satisfies static design constraints. We have linked FeatureIDE to a prototype tool we developed, which enables the evaluation of the complex parametric equations determining the validity of the design. Our toolchain already overcomes the current state of practice as it benefits from the capability of FeatureIDE to support configuration activities, combined with our domain-specific analysis tool.

Our work on this case has allowed us to identify the major challenges of space system design and to clarify our research vision towards supporting the design process with variability-aware techniques. We retrospectively describe these challenges and present research directions that, we believe, have the potential to disrupt the way space systems are currently designed.

To summarize, our key contributions are:

- We formulate the design of space systems as a configuration problem and we highlight the complexity factors. Most notably: configuration is multi-stage (component selection and parameter setting), it involves Boolean, discrete and continuous parameters, and it is multi-step (multiple iterations in order to refine component selections and their parameters value based on intermediate analysis).
- We develop a case study based on a real-world communication satellite. We identify the different variability dimensions (requirements, technologies, components, etc.) that exist in design process of such systems. We illustrate the complexity of modelling this design activity due to the large number of features, constraints, and differences between engineering guidelines.
- We provide a framework to configure the satellite and generate corresponding parametric models for further analysis. We implement our approach in a tool-chain that may support and automate engineers design activities. Together with our case study, this tool can act as a benchmark for future research on variability modelling and analysis for space systems.
- We describe relevant directions that research can work on in order to solve design space modeling and exploration problems in space engineering. These research directions aim to automate and improve the design process of space systems in general.

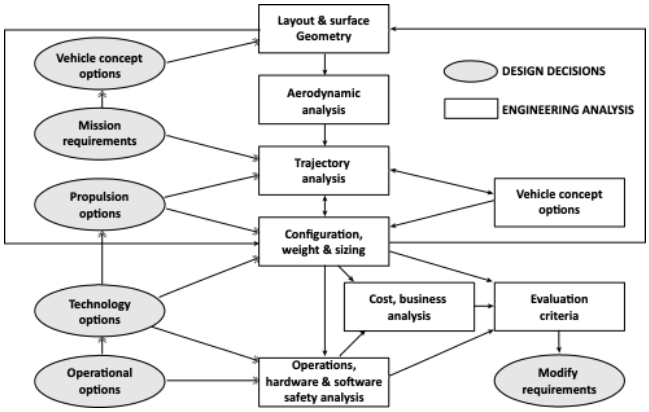

Figure 1: Example of design process[59]

2 SPACE SYSTEM DESIGN: APPROACHES AND CHALLENGES

The CD analysis of complex space systems is typically performed by a multidisciplinary process for designing the system by modelling and integrating its constituting subsystems [45]. The subsystems are modelled through physics-based parametric equation analysis [6] and numerous design synthesis tools have been developed to support the design space exploration and analysis [18, 34, 38, 48, 49].

Over the years, numerous design process would drive the analysis of various spacecraft subsystems separately whereas information is passed from one subsystem to other in a sequential manner. Examples of design process are; Raymer (aircraft design) [58]; K.D. Wood (launch vehicle design) [73]; Rowell and Korte (NASA Launch Vehicle Design Process) [59]; Hammond (space system design)[30] and several space-system design tools such as SSSP (space Shuttle Sizing Process) [16]; PrADO-Hy [32]; AVDS [57]; FLOPS [46]; ModelCenter [2] etc. Figure 1 comes from a NASA technical memorandum report by Rowell and Korte [59] and shows a launch vehicle design process.

Modern-day spacecrafts, particularly satellites can be considered as complex cyber-physical systems as compared with the traditional space access systems (launch and reentry vehicles) which tend to be more of physical systems. The primary subsystems of a satellite involve disciplines addressing attitude control, thermal control, communication, command and data handling, electrical power, propulsion, and structural mechanisms. The major subsystems of a launch vehicle address disciplines including propulsion, structure, aerodynamics, thermodynamics, and stability and control [56].

Another major difference between a satellite and a launch vehicle is the operationally dynamic behavior of the two space systems. A launch vehicle consumes fuel at a very high rate such that its total mass changes substantially within the first few minutes of its operation. This aspect plays a crucial role in defining the design process used to design these space systems. About 85 percent of the total mass of a launch vehicle is the propellant while the majority of remaining 15 percent is the structural mass required to carry and burn the propellant [50]. In this regard, the design methods used for a launch vehicle are oriented towards solving mathematical

problems that needs to converge with respect to the total mass. The launch vehicle design process shown in Figure 1 is one such example.

The above examples show that different space systems require different design processes and corresponding tool-sets. Compared to a typical launch vehicle design process, the design method applied for a satellite is not a strict mathematical problem but rather a step-by-step process integrating the configuration of constituent subsystems into a system solution that would satisfy the mission objectives and requirements. The classical book "Space Mission Analysis and Design" by Wertz and Larson [40] propose a well-adopted example of such satellite design process. References [3, 22, 23, 62, 67] are further similar examples of processes to design satellites. All these methodologies provide a formalized process implementing a satellite configuration workflow to satisfy mission objectives and requirements.

The number of design choices at subsystem and component levels creates a vast solution space. A holistic modeling platform should be able to parametrically design and trade critical mission, and subsystem-level configurations and parameters to consistently compare various design solutions and identify optimum baselines. Moreover, it should navigate across multiple interdependent subsystems within the configuration of these particular subsystems. Hence, the configuration process is multi-stage because it involves increasingly detailed configuration activities. At the same time, this process is also multi-step as dependencies between the different subsystems prevent their one-shot configuration. Multiple iterations are needed to refine the subsystem components properly and converge towards a suitable design.

The modeling of design decisions, and requirements is another key differentiating aspect across design processes and tools that may impact the designs' quality. For example, design decisions such as the selection of a specific material or a technology affect the parametric analysis and have a large impact on the final configuration formulation. Variation in these design decisions leads to conducting trade studies and developing solution spaces. This is a critical and challenging aspect that combines requirements engineering and design decision impact upon configuration variability and analysis.

3 APPLICATION CASE: SATELLITE SUBSYSTEMS

In general, a typical satellite has seven major subsystems, namely: Attitude Determination and Control (ADCS), power subsystem, communication, command and data handling, thermal, propulsion and structures and mechanism. Each subsystem has its own sub-requirements derived from top-level system requirements and mission analysis (e.g. orbital conditions and trajectory analysis). Numerous hardware components exist and can be used to compose a subsystem that will meet the requirements. Each component is modeled mathematically by a set of parameters. These parameters can be direct input from engineers or may requires complex parametric equations to be determined. Some parameters (output) of a component will be connected as input for other component possibly from a totally different subsystem. Some parameters represent quality attributes such as mass, power consumption, and manufacturing cost, but are also used to *size* the entire satellite at each iteration [40, 60].

Configuring the satellite from high-level requirements and orbital analysis requires solving a two-step problem. First, engineers have to determine which components seems to be the most suitable. Second, the engineers need to *size* the selected components by determining component parameters values that meet and optimize the given requirements (i.e. space mission description). Different component architectures and parameter values can trade off relevant concerns and qualities differently, while analyzing all those design alternatives and the trade-offs they offer is demanding [26, 36, 74].

This configuration problem is complex because the above two steps and their related components are all interdependent. In application, the design workflow – i.e. the order in which the different subsystems are designed – can change the produced design. Hence, while the configuration of an earlier component can affect the configuration of a later one, the converse is also true. This means that the configuration workflow is inherently iterative: after setting the parameters of the later components, engineers may have to go back to the earlier ones and refine their configurations in order to improve the overall system.

More pernicious instances of this iterative complexity create a chicken-egg paradox in the configuration workflow. At the beginning of the design process, some parameters are unknown and cannot be determined yet as they require specific analysis. However, the same parameters are prerequisite for component sizing (parameterization). For example, sizing the total mass of the satellite requires specifying the thruster. Generally, the heavier the satellite is, the more powerful and heavier the thruster will be. After sizing the thruster, the mass of the satellite will be refined with the computed mass of the required thruster. This example illustrates also that required propellant for the satellite mass and mission duration may also impact the mass. This will require other sizing iterations for all the concerned components. As a result, in order to ensure design convergence, engineers have to identify such workflow dependencies in advance and provide appropriate initial estimations and constraints for some parameters that will be later refined.

To further illustrate all these challenges that variability poses on our application case, we focus on two subsystems: ADCS and Power.

3.1 Attitude Determination and Control Subsystem (ADCS)

The ADCS stabilizes the vehicle and orients it in desired direction despite the external disturbance torques acting on it. This subsystem can be divided into 2 hardware parts: sensors that determine the position and orientation of the spacecraft, and actuators that perform the control function and compensate disturbances.

There are 7-11 various types of sensors for attitude determination (star trackers, sun sensors, magnetometer, gyroscopes, etc). It is a common practice to use combinations of different sensors in order to fulfill mission goals. The most common actuators are: magnetic torquers, reaction wheels, momentum wheels and thrusters. Performance characteristics of each actuator affect and depend on the size of the hardware. The change in size of the hardware, in turn, impacts other variables of the spacecraft. Thus, the feasible

solutions for ADCS can be numerous due to the variety of all possible combinations. The process to find the most optimal solution can be very complex and requires a high-level expertise and a lot of analysis iterations of the whole system.

ADCS requirements are connected to mission needs and other subsystem characteristics. These requirements may change considerably depending on the mission phase or on control modes of the spacecraft. For instance, during the orbit insertion, the major requirement for the ADCS is to dump all the torques that are applied to the spacecraft and deliver it to its final orbit. By contrast, while the spacecraft is on station, the dominating requirement is a pointing accuracy for the payload and communications.

In addition, frequency and need of slew maneuvers must be defined. Such maneuvers might be necessary for re-pointing of the payload to the target of interest. The rate of such maneuvers and the time needed for re-positioning may influence the choice and size of the actuators. Basically, from these maneuvers, engineers can derive subsystem requirements such as the torque required by thruster, reaction wheels, etc. From which power required by ADCS for the entire mission can be computed.

3.2 Power Subsystem

The power subsystem performs four major functions on the spacecraft. It generates, stores, distributes and controls power on the vehicle. Peak power consumption of the spacecraft, orbital profile of the mission and mission duration are ones of the most important requirements for the power subsystem. Electrical power loads for mission operations can vary over the life of the spacecraft and these differences must be anticipated as well.

These requirements connect directly power with other subsystems of the vehicle (including ADCS), as all of them may require a certain amount of power that will affect the total consumption of the spacecraft. Hence, the slightest change of any of the subsystems (e.g. that would result in a heavier spacecraft) will change the demands for power. As an example, having a heavier satellite will require the ADCS to include reaction wheels with a higher radius in order to stabilize and orient the satellite. However, the power consumption and mass of these bigger reaction wheels will increase both the mass and power consumption of the vehicle. This may, in turn, require resizing the solar panel that will again increase the overall mass. Other side effects exist and ultimately require engineers to rethink the whole system design.

The choice of power source has an important role because it will affect the total mass of the spacecraft. Sources of electricity for the spacecraft can be solar panels, radioisotopes, fuel cells, nuclear reactor or solar thermal dynamic. Another important aspect of hardware selection for the power subsystem is power storage. Power stored in the batteries are providing power during penumbra or eclipse periods. The type and material of the battery impact the required size of the batteries and the number of them. In addition, power subsystem storage design includes complexity of the orbital characteristics as it affects orbital period, number and duration of eclipses which spacecraft needs to survive.

4 FEATURE MODELLING

In order to support engineers in the design process of their satellites, we propose a model-based design approach. This approach enables the modeling of requirements and system components and support engineering in the configuration and sizing of satellite components with respect to mission requirements and constraints. To this aim, we captured the configuration space of the different subsystems. Such configuration space must remain consistent to capture only the feasible system architectures given mission and orbit analysis requirements. The inter-dependencies between components have also to be captured to automate the feasibility check of a system architecture.

4.1 Source of Knowledge: SMAD

Space Mission Analysis and Design [41] (SMAD, for short) is the most famous reference to introduce engineers to the field of satellite design. It is highly used in academia but also in industry and governmental organizations. This "bible" of space system design proposes a structured design process for the entire space mission. From "Chap. 1 : The Space Mission Analysis and Design Process", "Chap. 4 : Requirements Definitions" to "Chap. 10 : Spacecraft Design and Sizing" and "Chap. 11 : Spacecraft Subsystems", SMAD proposes a step-by-step design methodology to manage and reduce the complexity of the overall design process. For each subsystem, it proposes a detailed step-by-step methodoloy with guidelines, gives an overview of the major hardware components, and recommends the most suitable components for common requirements.

Figure 2 is an example of table that recommends which hardware to select depending on the requirements. We observe that, depending on the pointing control accuracy requirement (e.g., >5 degrees, between 1 to 5 degrees, etc.) the recommended ADCS actuators (e.g., reaction wheels, thruster, magnetorquer, etc.) vary significantly. The recommendation also varies depending on the spacecraft's attitude determination capabilities that rely on sensors such as sun sensors or, more accurate, star trackers. In the end, depending on the control accuracy requirement (which is one requirement from many) SMAD will recommend, or constrain, the set of valid combinations of sensors and actuators for the ADCS subsystem.

SMAD also recommends hardware selection depending on the kind of maneuvers the spacecraft will have to perform during his mission. For example, "normal on station" maneuvers tend to keep the satellite in orbit. In this case, relatively small external disturbance torques (such as gravity gradient and solar radiation) have to be countered. On the contrary, "orbit insertion" maneuvers will likely require a heavier thruster to insert the satellite in orbit. Similarly, while reaction wheels can be recommended for 0.05 to 0.5deg/s reorientations, higher will require thrusters [40].

4.2 From SMAD to Feature Models

Based on SMAD and our research team experts in satellite engineering, we propose a variability-aware approach that supports satellites design. For this, we have modelled the variability and the configuration space of satellite systems using feature models [37, 61]. In this feature model, features represent the variable subsystem components but also requirements, control modes, etc. that

TABLE 11-8. Effect of Control Accuracy on Sensor Selection and ADCS Design. Accurate pointing requires better, higher cost, sensors, and actuators.

Required Accuracy (3σ)	Effect on Spacecraft	Effect on ADCS
> 5 deg	• Permits major cost savings • Permits gravity-gradient (GG) stabilization	***Without attitude determination*** • No sensors required for GG stabilization • Boom motor, GG damper, and a bias momentum wheel are only required actuators ***With attitude determination*** • Sun sensors & magnetometer adequate for attitude determination at ≥ 2 deg • Higher accuracies may require star trackers or horizon sensors
1 deg to 5 deg	• GG not feasible • Spin stabilization feasible if stiff, inertially fixed attitude is acceptable • Payload needs may require despun platform on spinner • 3-axis stabilization will work	• Sun sensors and horizon sensors may be adequate for sensors, especially a spinner • Accuracy for 3-axis stabilization can be met with RCS deadband control but reaction wheels will save propellant for long missions • Thrusters and damper adequate for spinner actuators • Magnetic torquers (and magnetometer) useful

Figure 2: ADCS Hardware Recommendation given Control Accuracy Requirement (from [41])

will constrain the components selections. Feature model Cross-tree constraints encode two types of information: (a) the hardware constraints and dependencies, and (b) the recommendations found in SMAD. Regarding these recommendations, we kept only the unanimously admitted ones in the field. Indeed, experts may deviate from some recommendations based on their experience and favourite technologies. Moreover, the recommendations can evolve as ingenious technologies are constantly proposed to optimize maneuvers (i.e., specific reaction wheels and gyroscope combination for high and precise reorientations[75]).

We, therefore, adopt an open approach where we allow engineers to deviate from the conservative recommendations of the established book. This is important because any solution that violates the "hard constraints" (the constraints that feature-model-based configuration enforces) is interpreted as unfeasible. These omitted recommendations could be encoded as an optional set of constraints ("soft" constraints) in the feature model, whose violation is permitted. In the same way, other expert recommendations destined to simplify the configuration process for less experienced engineers can be encoded as soft constraints. During a configuration process, these constraints could, e.g., be overruled or be disabled depending on the expertise level of the engineers. We have left this extension for future work.

Figure 3 gives an overview of our feature model. The root feature represents the satellite and is decomposed into multiple subfeatures, i.e. one per subsystem. ADCS, power and communication subsystems have been modelled using 152 features and 42 constraints. To give an idea, we think that the complete feature model (i.e., with command and data handling, thermal, propulsion and structures and mechanism subsystems) may contain around 500 features and 200 cross tree constraints. Hereafter, we illustrate in detail the ADCS subsystem only. It is composed of 72 design decisions encoded as Boolean features and 27 cross-tree constraints. This feature decomposition follows the design workflow presented in the SMAD reference book [40].

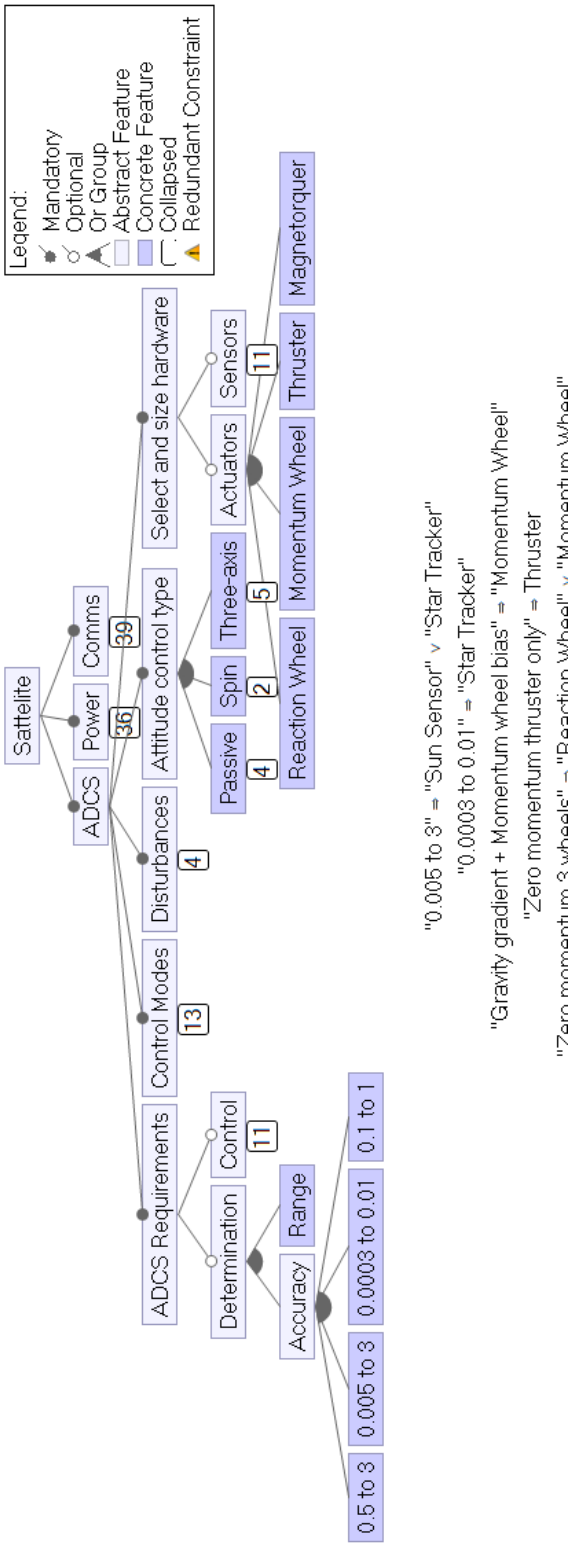

Figure 3: Satellite ADCS Feature Model

The first subfeature of ADCS concerns the definition of this subsystem's requirements. This is essential because these requirements have the largest impact on the constraints, recommendations and selection of particular hardware combinations. As an example, accuracy is requirement on both determination and control capacities, which will – together with other factors – constrain the set of suitable sensors and actuators. For example an accuracy of 0.005 to 3 degrees in attitude determination will demand sensors such as a sun sensor or a star tracker. However, if the accuracy requirement is even more strict (e.g, 0.0003 to 0.01), only a star tracker can deliver it. Similarly, mission duration (e.g., < 1 day, > 1 day, > 1 year) is a key requirement driving the configuration of the power subsystem.

The "Control Modes" feature captures the different types of maneuvers that a satellite can perform during a mission. Selecting a particular set of control modes will constrain the acceptable set of satellite architectures. Disturbances are the external torques the satellite will face during its mission. All disturbances are mandatory, except those related to aerodynamics because these concern super low-orbit satellites only. Attitude control type determines the main control systems (or technologies). These can be seen as a popular combination of hardware for standard purposes.

The last subfeature of ADCS actually concerns the selection of the relevant hardware components. The corresponding subtree is mostly constrained by the previous subfeatures (requirements, control modes, etc.). Hence, in a concrete configuration workflow, engineers would normally set the previous subfeatures to reduce hardware configurations to suitable architectures.

Yet, the advantage of using feature model is that its declarative semantics [61] makes its representation independent of the actual configuration workflow. The left-to-right configuration workflow that we have depicted above (i.e. from requirements to component selection and sizing) is only an alternative. Other configuration workflows that would navigate from components to the requirements these components can meet are also inherently supported. This capability is particularly useful in use cases where engineers have to maximize the set of fulfilled requirements given a limited set of resources within a limited budget.

Through our investigations, we, however, identified limitations in the use of standard feature models. A first limitation is that the vast majority of the hardware components can be selected multiple times to be placed in the vehicle. For example, the attitude control type of ADCS can comprise multiple 3-Axis wheels that can reorient the vehicle in three axes with high precision. In our current modelling, we have specified a predefined number of 3-Axis wheel to configure. We can overcome these limitations through the use of feature cardinalities [19], though at the cost of an increase in the semantic complexity of feature models. A second limitation is that satellite components can have real-valued parameters determined through parametric equation computations. To encode these parameters, we need to use numeric feature attributes which, again, make feature model analysis more complex. Moreover, we need to simulate the effect of these numeric features on the system through dedicated analyses that solve the parametric equations. This makes the configuration workflow involve other artefacts than the feature model.

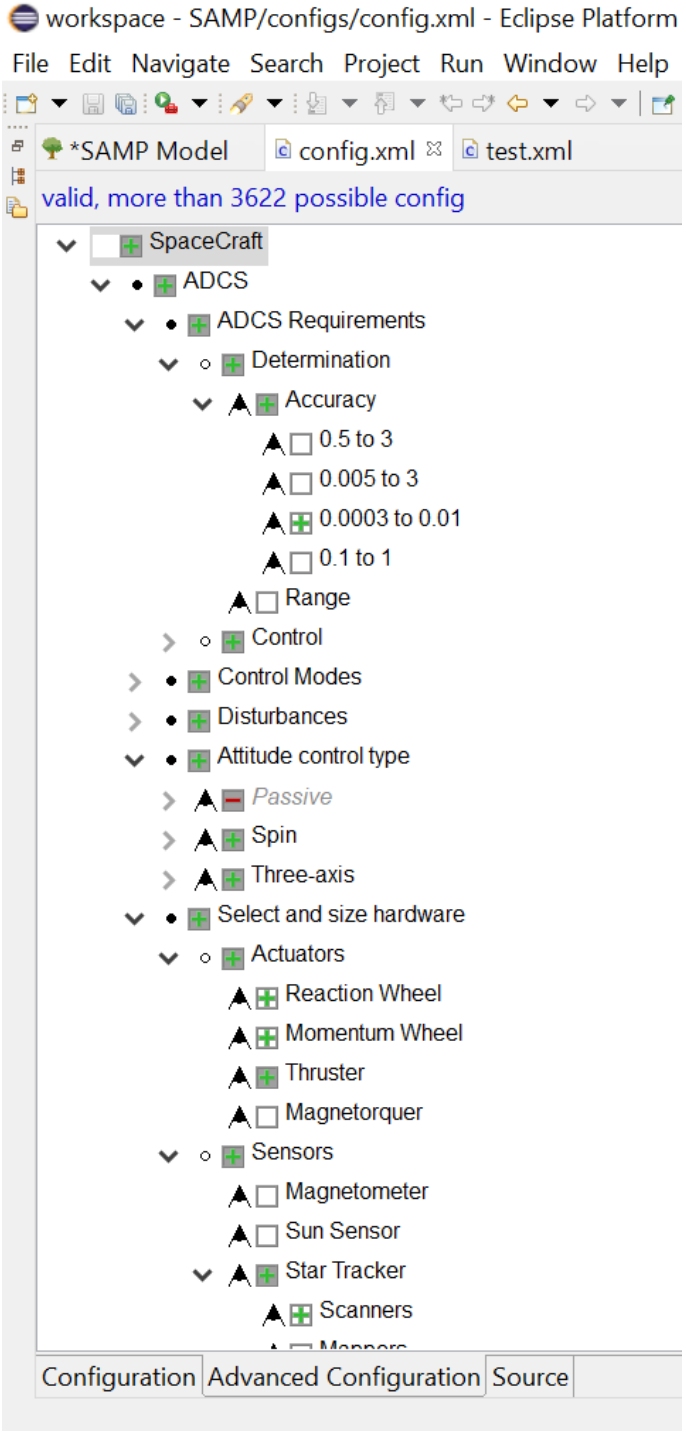

Figure 4: Feature IDE Configuration Interface

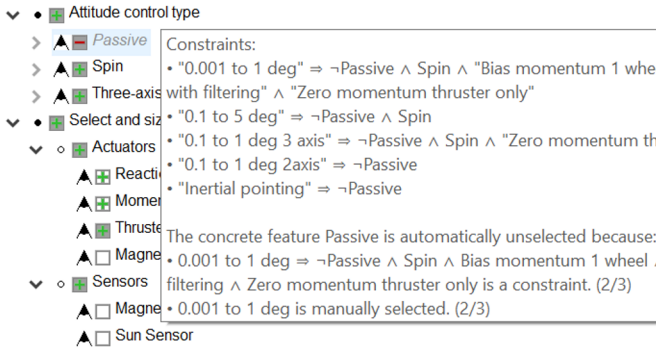

Figure 5: Constraints Implications

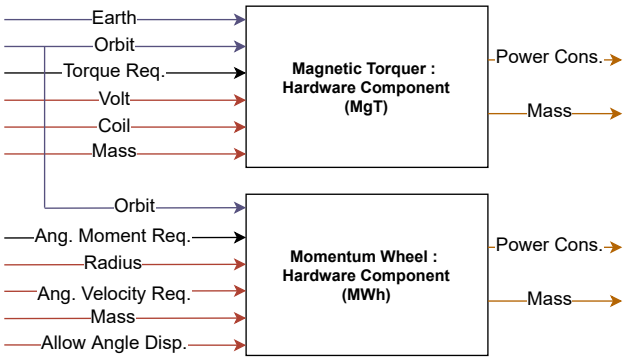

Figure 6: Parametric Simulation

```
if features['Magnetorquer'] ==  True:
    mt = a.magnetic_torquer(inp.Alt,TotalRSS, inp.tfs,inp.Mo
    adcs_mass = adcs_mass + mt.MoT * inp.NumMT
    adcs_power = adcs_power + mt.I * mt.U * inp.NumMT

if features['Momentum Wheel'] ==  True:
    mw = a.momentum_wheel(TotalRSS, OP.OP, inp.mwr,inp.mwav,
    adcs_mass = adcs_mass + mw.MWM * inp.NumMW
    adcs_power = adcs_power + mw.MWP * inp.NumMW
```

Figure 7: Sizing Framework

5 CONFIGURATION WORKFLOW IN FEATURE-IDE AND BEYOND

We have operationalized our feature model within the FeatureIDE [70]. Thanks to this tool, we enable engineers to configure satellites architecture based on mission requirements.

Figure 4 shows a valid configuration that we made based on the expertise of our researchers in space system design. This view shows again that the feature-oriented configuration process allows engineers to configure the different subsystems in an arbitrary order. The interactive activation of the constraints enables the identification and understanding of the interdependencies between the design choices.

Once the space engineers have reached a configuration of the subsystem components that is valid – according to the encoded feature model constraints – some analysis based on complex parametric equations must be conducted to size the selected components in order to make sure that the configured design is feasible in practice.

The parametric equations to solve depend on the selected hardware components that form the satellite. Hence, two different satellite architectures will require solving two different sets of equations. For example, in Figure 6 we illustrate that four combinations of satellite subsystem architectures (i.e., {}, {MgT}, {MWh}, {MgT, MWh}) will lead to different sets of computations. If none of the components depicted will be selected, no mass and power consumption will be added or refined to the overall vehicle (the linear and angular torque and will have to be handled through other components). If both components have been selected in the configuration, the angular and linear torque requirements will determine the internal continuous parameters of the components.

In order to complete our configuration tool based on FeatureIDE with the capability to solve the necessary parametric equations, we have implemented a software tool in Python. Our tool takes as input a partial space design configuration and solves the related parametric equations in order to determine the missing internal parameter values of the subsystem components. As an illustration, Figure 7 shows an excerpt of our implementation. We see that, depending on the enabled features, different computations will be invoked to solve specific equations.

Our tool also checks that the produced configuration satisfies the expected requirements, in addition to the constraints that FeatureIDE previously checked. If it is not the case, the tool will refine the parameter values (through the solving of the parametric equations) in order to repair the design configuration. If no parameter values satisfy the requirements, engineers might have to refine the configuration further (i.e., select other components).

Accordingly, while the feature model (implemented in FeatureIDE) guides engineers in the configuration workflow and enables a fast pruning of the infeasible designs – based on the established SMAD – our analysis framework performs fine-grained computations to make sure that the components parameters can ultimately lead to a viable architecture.

6 RESEARCH DIRECTIONS

Our investigations have revealed the complexity of the space system design process and the toolchain we have set up is but a stepping stone onto which research can build disruptive solutions. Ultimately, the holy grail that research should seek is the automation of the full design process. We elaborate hereafter on promising research directions that, we believe, can have a significant impact on space system engineering practices.

6.1 From Heterogeneous Guidelines to Configuration Knowledge

A general bottleneck in configurator development is the elicitation of the configuration knowledge [17], i.e. the feature model and the set of rules that define the validity of a configuration. In the space domain, design validity is determined by a generic set of physical rules and by heterogeneous (and sometimes, conflicting) set of established guidelines. In our case study, we manually analyzed such guidelines and derived the validity constraints for our satellite subsystems. Though there might be some generality

in the constraints we extracted, developing a feature model and constraints for another space (sub)system would require manual work and engineering expertise. We believe that variability-aware CD tools for space systems can only reach practitioners if these tools minimize the effort involved in acquiring and operationalizing configuration knowledge.

The problem of synthesizing configuration knowledge from natural language documents falls into the larger area of requirements engineering. Space system engineering can thus benefit from research in this field. There exist automated approaches to extract domain models from natural-language requirements [4, 5]. Other approaches synthesize feature models of software product lines from textual descriptions [21, 24, 72]. An important discrepancy between the working assumptions of these methods and the reality of space system engineering lies in the multiple and potentially conflicting sources to extract the configuration knowledge from. This is because the heterogeneous guidelines often provide more recommendations than constraints, and these recommendations can depend on the particular experts that produced the guidelines. In other words, some of the extracted constraints should be seen as "soft constraints" and not "hard constraints". In this context, a synthesis tool that is able to reconcile conflicting sets of soft constraints would bring an added value to space engineers.

6.2 From Requirements to Partial Configuration

During CD, the space mission requirements essentially drive the configuration process. Through the analysis of the mission description, engineers determine the first key decisions they should make in order to guarantee that the design will be able to fulfill the mission. While these early design steps impact the whole design process, they still rely intensively on engineers' (biased) experience and ingenuity.

In order to reduce both the early burden on engineers' shoulders and their inherent bias, the development of automated tools that can translate mission requirements into a set of appropriate design choices can yield significant benefits. We believe that the recent progress of natural language processing research and the development of industry-strength, large-scale language models enable the realization of such design tools. By making these tools a reality, engineers will be able to automatically produce, from natural-language mission requirements, a partial configuration of their space system design, based on established practices and data collected from past design experiences.

Automatic requirements to architecture methods rely on model transformation algorithms. Such algorithms try to transform requirements to hardware architecture. However, the allocation of requirements to hardware architectures could be highly problematic and yet have an immense impact on the system feasibility [44]. This is also observed in embedded system design in a similar problem called the "application mapping" problem (or "hardware/software partitioning" problem) [68]. Similarly, one may lack to capture, model and explore "all" possible allocations of requirements to resources in order to recommend the most promising allocations to the engineers. One reason is that determining possible allocations might be hard to automate [64], especially if it requires human ingenuity.

6.3 Behavioural Analysis to Support Post-CD Design Stages

Though CD enables a significant pruning of the design space, the number of design alternatives to consider (and of parameters to set) can remain significant. At later design stages, static analysis based on parametric equations is not sufficient to properly evaluate design correctness and quality. More detailed and costly analyses that approximate the behaviour of the space systems are needed.

In the broader field of cyber-physical system engineering, computational methods rely on formal verification [63, 66] and testing [8, 47, 76] to check whether the behaviour of the system design complies with intended requirements. Such methods are attractive because they enable reasoning over the dynamic interplay of the requirements and the system design behaviour. They typically start from a behavioural model of the system behaviour (e.g. hybrid automata [20, 52] or Simulink [1]) and, then, check whether the model complies with behavioural requirements expressed in some logic (temporal [54, 71] or hybrid [7, 13, 33, 47, 53]). These methods require engineers to have already decided on a system design and ignore the numerous possible design alternatives that exist.

Search heuristics that use computational methods as backbone evaluation methods are a promising avenue in this direction [27, 65]. Typically, these methods apply optimization and search heuristics to select promising design candidates based on their static characteristics. Then, they invoke a black-box analysis procedure to evaluate the behaviour of a given candidate. Though these methods nicely combine fast pruning of the design space with behavioural analysis, they do not build a global view on all designs' behaviour, which narrows their practical applications to a limited set of design alternatives.

The adjacent field of variability-aware verification software has given rise to model-checking methods that can efficiently verify a large set of variants of a given system [9–11, 14, 15, 25, 28, 29, 39, 42, 43, 51, 69] – a problem analogous to the complexity introduced by numerous design alternatives in space systems. These methods factorize the verification of behaviours common to multiple system variants. One can draw a parallel between variability-intensive software and design alternatives by considering the latter as variants of the same system that share many common behaviours. This enables the reuse of the aforementioned methods to verify multiple designs at reduced computation costs. However, while effective, these methods were developed for software systems that do not interact with the physical world and exhibit a purely discrete state space. Therefore, methods that handle variability cannot check the non-linear computations between discrete-state computational elements and continuous-state physical entities.

Existing research is insufficient to tackle the challenge of evaluating numerous space system design alternatives, because it can efficiently handle either numerous "simple software systems" or one complex system, but *not* numerous complex systems. The respective limitations of these research areas motivate future work to join their strengths and enable the efficient evaluation of numerous design alternatives.

6.4 Concurrent System Design Configuration Workflow

In traditional design methodologies – which we have followed in this paper – the design process is executed in a sequential manner. Here, information flows from one design discipline to the other one step at a time, passing the design from one disciplinary team to the next. The design goes through several iterations until the requirements from all design disciplines are satisfied. Although this category is most widely applied, it has drawbacks that favour a certain level of separation among the design disciplines.

A more modern, flexible approach is concurrent design. In this approach, the design disciplines work in parallel and the information flows faster between the disciplines, which enables a much more efficient design process. However, concurrent design induces an increased complexity in the design configuration workflows. Indeed, because they work in parallel, the different teams can progress at different paces and this increases the likelihood that parameters changed by a given team impact the parameters set earlier by another team.

Concurrent configuration workflows, their challenges, and their solutions have been the object of intense research in the past decade (e.g. [35]). However, these research solutions have not been applied to the specific context of space system engineering methods and processes. An interesting research direction is, therefore, to bridge the gap between configuration workflow research and the practical needs of space system engineers. We believe that our contribution, in particular our application case, feature model and supporting tool, can act as a first benchmark to evaluate the new solutions that research will produce.

7 CONCLUSION

Space system engineering is a promising avenue for variability research. The huge size of the design space – combined with the inherent complexity of the design configuration processes, workflows and analyses – are all rooms to exploit the tangible solutions that research has developed to manage variability in software systems.

In this paper, we have described the challenges of space system design as it exists today. We have demonstrated that notorious variability modelling methods and tools – i.e. feature models and their implementation within FeatureIDE – can be exploited to support the daily design work of space system engineers. We have also depicted research directions that, we believe, can be game changers for space system research and engineering.

Our application of variability modelling methods, together with the configuration analysis software that we specifically developed for our satellite application case, are available to benchmark future research solutions. We hope that our endeavour will drive more research at the intersection of variability modelling and space system engineering, which will ultimately yield significant advances cross-cutting these two fields.

ACKNOWLEDGMENTS

Maxime Cordy and Sami Lazreg are supported by FNR Luxembourg (grant C19/IS/13566661/BEEHIVE/Cordy).

REFERENCES

[1] [n. d.]. Simulink - simulation and model-based design. https://nl.mathworks.com/products/simulink.html
[2] 2020. ModelCenter Integrate: Model Based Engineering Software. https://www.phoenix-int.com/product/modelcenter-integrate/
[3] Christina Aas, B.T.C. Zandbergen, Rob Hamann, and Eberhard Gill. 2009. Development of a System Level Tool for Conceptual Design of Small Satellites. (01 2009).
[4] Vander Alves, Christa Schwanninger, Luciano Barbosa, Awais Rashid, Peter Sawyer, Paul Rayson, Christoph Pohl, and Andreas Rummler. 2008. An Exploratory Study of Information Retrieval Techniques in Domain Analysis. In *SPLC '08*. IEEE Computer Society, Washington, DC, USA, 67–76.
[5] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. 2016. Extracting Domain Models from Natural-language Requirements: Approach and Industrial Evaluation. In *MODELS '16* (Saint-malo, France). ACM, New York, NY, USA, 250–260.
[6] Salima Berrezzoug, Abdelmadjid Boudjemai, and Fethi Tarik Bendimerad. 2019. Interactive design and multidisciplinary optimization of geostationary communication satellite. *International Journal on Interactive Design and Manufacturing (IJIDeM)* 13, 4 (2019), 1519–1540.
[7] Davide Bresolin. 2013. HyLTL: a temporal logic for model checking hybrid systems. *Electronic Proceedings in Theoretical Computer Science* 124 (Aug 2013), 73–84. https://doi.org/10.4204/eptcs.124.8
[8] Lionel C. Briand, Shiva Nejati, Mehrdad Sabetzadeh, and Domenico Bianculli. 2016. Testing the untestable: model testing of complex software-intensive systems. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*, Laura K. Dillon, Willem Visser, and Laurie A. Williams (Eds.). ACM, 789–792. https://doi.org/10.1145/2889160.2889212
[9] Glenn Bruns and Patrice Godefroid. 2004. Model Checking with Multi-valued Logics. In *ICALP '04*. 281–293.
[10] Marsha Chechik, Benet Devereux, Steve M. Easterbrook, and Arie Gurfinkel. 2003. Multi-valued symbolic model-checking. *ACM Trans. Softw. Eng. Methodol.* 12, 4 (2003), 371–408.
[11] Marsha Chechik, Benet Devereux, and Arie Gurfinkel. 2001. Model-Checking Infinite State-Space Systems with Fine-Grained Abstractions Using SPIN. In *SPIN '01*. 16–36.
[12] B Chudoba and W Heinze. 2010. Evolution of generic flight vehicle design synthesis. *Aeronautical Journal* 114, 1159 (2010), 549–567.
[13] Alessandro Cimatti, Marco Roveri, and Stefano Tonetta. 2015. HRELTL: A temporal logic for hybrid systems. *Information and Computation* 245 (2015), 54–71. https://doi.org/10.1016/j.ic.2015.06.006
[14] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-François Raskin. 2013. Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and their Application to LTL Model Checking. *Transactions on Software Engineering* (2013), 1069–1089.
[15] Guillermina Cledou, José Proença, and Luis Soares Barbosa. 2017. Composing Families of Timed Automata. In *Fundamentals of Software Engineering*, Mehdi Dastani and Marjan Sirjani (Eds.). Springer International Publishing, Cham, 51–66.
[16] CONVAIR. 1970. Space Shuttle Synthesis Program (SSSP) Final Report No. GDC-DBB70-002.
[17] Maxime Cordy and Patrick Heymans. 2018. Engineering Configurators for the Retail Industry: Experience Report and Challenges Ahead. In *ACM SAC '18*. 2050–2057.
[18] Steven Cornford, Robert Shishko, Stephen Wall, Bjorn Cole, Steven Jenkins, Nic Rouquette, Greg Dubos, Tyler Ryan, Pezhman Zarifian, and Bryce Durham. 2012. Evaluating a fractionated spacecraft system: A business case tool for darpa's f6 program. In *2012 IEEE Aerospace Conference*. IEEE, 1–20.
[19] Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. 2005. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice* 10, 1 (2005), 7–29.
[20] Alexandre David, Dehui Du, Kim G. Larsen, Axel Legay, Marius Mikucionis, Danny Bøgsted Poulsen, and Sean Sedwards. 2012. Statistical Model Checking for Stochastic Hybrid Systems. In *Proceedings First International Workshop on Hybrid Systems and Biology, HSB 2012, Newcastle Upon Tyne, UK, 3rd September 2012 (EPTCS, Vol. 92)*, Ezio Bartocci and Luca Bortolussi (Eds.). 122–136. https://doi.org/10.4204/EPTCS.92.9
[21] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans. 2013. Feature Model Extraction from Large Collections of Informal Product Descriptions. In *ESEC/FSE '13* (Saint Petersburg, Russia). ACM, 290–300.
[22] Davide Di Domizio and Paolo Gaudenzi. 2008. A Model for Preliminary Design Procedures of Satellite Systems. *Concurrent Engineering* 16, 2 (2008), 149–159. https://doi.org/10.1177/1063293x08092488
[23] William Edmonson, Jules Chenou, Natasha Neogi, and Heber Herencia-Zapana. 2014. Small satellite systems design methodology: A formal and agile design process. In *2014 IEEE International Systems Conference Proceedings*. 518–524. https://doi.org/10.1109/SysCon.2014.6819305

[24] Alessio Ferrari, Giorgio O. Spagnolo, and Felice Dell'Orletta. 2013. Mining Commonalities and Variabilities from Natural Language Documents. In *SPLC '13* (Tokyo, Japan). ACM, New York, NY, USA, 116–120.

[25] Dario Fischbein, Sebastian Uchitel, and Victor Braberman. 2006. A foundation for behavioural conformance in software product line architectures. In *ROSATEA'06, ISSTA 2006 workshop*. ACM Press, 39–48.

[26] Justin S Gray, Tristan A Hearn, Kenneth T Moore, John Hwang, Joaquim RRA Martins, and Andrew Ning. 2014. Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in OpenMDAO. In *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2042.

[27] Matthias Gries. 2004. Methods for evaluating and covering the design space during early design development. *Integration, the VLSI journal* 38, 2 (2004), 131–183.

[28] Alexander Gruler, Martin Leucker, and Kathrin Scheidemann. 2008. Modeling and Model Checking Software Product Lines. In *FMOODS'08* (Oslo, Norway). Springer, 113–131.

[29] Arie Gurfinkel and Marsha Chechik. 2003. Multi-Valued Model Checking via Classical Model Checking. In *CONCUR*. 263–277.

[30] Walter Edward Hammond. 2001. *Design Methodologies for Space Transportation Systems*. Vol. 1. AIAA.

[31] Keith Hartley. 2014. *The political economy of aerospace industries: a key driver of growth and international competitiveness?* Edward Elgar Publishing.

[32] Wolfgang Heinze. 1994. *Ein Beitrag Zur Quantitativen Analyse Der Technischen Und Wirtschaftlichen Auslegungsgrenzen Verschiedener Flugzeugkonzepte Fur Den Transport Grosser Nutzlasten*. Ph. D. Dissertation.

[33] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. 1993. Towards Refining Temporal Specifications into Hybrid Systems. In *Hybrid Systems*. Springer-Verlag, Berlin, Heidelberg, 60–76.

[34] Sebastian JI Herzig, Sanda Mandutianu, Hongman Kim, Sonia Hernandez, and Travis Imken. 2017. Model-transformation-based computational design synthesis for mission architecture optimization. In *2017 IEEE Aerospace Conference*. IEEE, 1–15.

[35] Arnaud Hubaux. 2012. *Feature-based Configuration: Collaborative, Dependable, and Controlled*. Ph. D. Dissertation. Catholic University of Louvain, Louvain-la-Neuve, Belgium. http://hdl.handle.net/2078.2/105420

[36] John T Hwang, Dae Young Lee, James W Cutler, and Joaquim RRA Martins. 2014. Large-scale multidisciplinary optimization of a small satellite's design and operation. *Journal of Spacecraft and Rockets* 51, 5 (2014), 1648–1663.

[37] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report CMU/SEI-90-TR-21.

[38] Aleksandr A Kerzhner, Michel D Ingham, Mohammed O Khan, Jaime Ramirez, Javier De Luis, Jeremy Hollman, Steven Arestie, and David Sternberg. 2013. Architecting cellularized space systems using model-based design exploration. In *AIAA SPACE 2013 Conference and Exposition*. 5371.

[39] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wasowski. 2007. Modal I/O Automata for Interface and Product Line Theories. In *ESOP*. 64–79.

[40] W J Larson and J R Wertz. 1992. Space Mission Analysis and Design. (1 1992).

[41] Wiley J Larson and James Richard Wertz. 1992. *Space mission analysis and design*. Technical Report. Torrance, CA (United States); Microcosm, Inc.

[42] Kim Lauenroth, Simon Toehning, and Klaus Pohl. 2009. Model Checking of Domain Artifacts in Product Line Engineering. In *IEEE/ACM ASE*. 269–280.

[43] Lars Luthmann, Andreas Stephan, Johannes Bürdek, and Malte Lochau. 2017. Modeling and Testing Product Lines with Unbounded Parametric Real-Time Constraints. In *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A* (Sevilla, Spain) *(SPLC '17)*. ACM, New York, NY, USA, 104–113. https://doi.org/10.1145/3106195.3106204

[44] Andreas Makoto Hein Marija Jankovic. 2021. Architecting Engineering Systems. (10 2021).

[45] Joaquim RRA Martins and Andrew B Lambe. 2013. Multidisciplinary design optimization: a survey of architectures. *AIAA journal* 51, 9 (2013), 2049–2075.

[46] LA McCullers. 1984. FLOPS: Flight Optimization System. *Proceedings of Recent Experiences in Multidisciplinary Analysis and Optimization, Hampton, Virginia* (1984).

[47] Claudio Menghi, Enrico Viganò, Domenico Bianculli, and Lionel C. Briand. 2020. Trace-Checking CPS Properties: Bridging the Cyber-Physical Gap. arXiv:2009.12250 [cs.SE]

[48] Robert C Moeller, Chester Borden, Thomas Spilker, William Smythe, and Robert Lock. 2011. Space missions trade space generation and assessment using the JPL Rapid Mission Architecture (RMA) team approach. In *2011 Aerospace Conference*. IEEE, 1–11.

[49] Mohsen Mosleh, Kia Dalili, and Babak Heydari. 2016. Distributed or monolithic? a computational architecture decision framework. *IEEE Systems journal* 12, 1 (2016), 125–136.

[50] NASA. [n. d.]. *The Tyranny of the Rocket Equation*. Accessed: 2021-11-21.

[51] Rafael Olaechea, Joanne Atlee, Axel Legay, and Uli Fahrenberg. 2018. Trace Checking for Dynamic Software Product Lines. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems* (Gothenburg, Sweden) *(SEAMS '18)*. ACM, 69–75.

[52] Angela Pappagallo, Annalisa Massini, and Enrico Tronci. 2020. Monte Carlo Based Statistical Model Checking of Cyber-Physical Systems: A Review. *Information* 11, 12 (2020). https://doi.org/10.3390/info11120588

[53] André Platzer. 2008. Differential Dynamic Logic for Hybrid Systems. *Journal of Automated Reasoning* 41 (08 2008), 143–189. https://doi.org/10.1007/s10817-008-9103-8

[54] A. Pnueli. 1977. The Temporal Logic of Programs. In *FOCS'77*. 46–57.

[55] Timothy Pratt and Jeremy E Allnutt. 2019. *Satellite communications*. John Wiley & Sons.

[56] L. Rana. 2017. *Space Access Systems Design: Synthesis methodology development for conceptual design of future space access systems*. Ph.D. Dissertation. The University of Texas at Arlington.

[57] L. Rana and B. Chudoba. 2020. Demonstration of a prototype design synthesis capability for space access vehicle design. *The Aeronautical Journal* 124, 1281 (2020), 1761–1788. https://doi.org/10.1017/aer.2020.55

[58] Daniel P Raymer. 2006. *Aircraft Design: A Conceptual Approach*. AIAA (American Institute of Aeronautics & Ast.

[59] Lawrence F Rowell and John J Korte. 2003. Launch vehicle design and optimization methods and priority for the advanced engineering environment. (2003).

[60] Joseph H Saleh, Daniel E Hastings, and Dava J Newman. 2002. Spacecraft design lifetime. *Journal of Spacecraft and Rockets* 39, 2 (2002), 244–257.

[61] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. 2006. Feature Diagrams: A Survey and A Formal Semantics. In *RE'06*. 139–148.

[62] Holger Schumann, Axel Berres, Olaf Maibaum, and Alexander Röhnsch. 2008. DLR'S VIRTUAL SATELLITE APPROACH. In *10th International Workshop on Simulation on European Space Programmes (SESP 2008)*.

[63] Marco Sgroi, Luciano Lavagno, and Alberto Sangiovanni-Vincentelli. 2000. Formal models for embedded system design. *IEEE Design & Test of Computers* 17, 2 (2000), 14–27.

[64] Joseph Sifakis. 2015. System design automation: Challenges and limitations. *Proc. IEEE* 103, 11 (2015), 2093–2103.

[65] Amit Kumar Singh, Piotr Dziurzanski, Hashan Roshantha Mendis, and Leandro Soares Indrusiak. 2017. A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–40.

[66] Marjan Sirjani, Edward Lee, and Ehsan Khamespanah. 2020. Verification of Cyberphysical Systems. *Mathematics* 8 (07 2020), 1068. https://doi.org/10.3390/math8071068

[67] H. Stoewer, R. Hartmann, and Lutz Richter. 2000. An Advanced Methodology for the Design Process of a Satellite. https://doi.org/10.1002/J.2334-5837.2000.TB00418.X

[68] Jürgen Teich. 2012. Hardware/software codesign: The past, the present, and predicting the future. *Proc. IEEE* 100, Special Centennial Issue (2012), 1411–1430.

[69] Maurice H. ter Beek, Alessandro Fantechi, Stefania Gnesi, and Franco Mazzanti. 2016. Modelling and analysing variability in product families: Model checking of modal transition systems with variability constraints. *Journal of Logical and Algebraic Methods in Programming* 85, 2 (2016), 287 – 315.

[70] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. 2014. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Comput. Surv.* 47, 1 (2014), 6:1–6:45.

[71] Moshe Y. Vardi and Pierre Wolper. 1986. An automata-theoretic approach to automatic program verification. In *LICS'86*. IEEE CS, 332–344.

[72] Nathan Weston, Ruzanna Chitchyan, and Awais Rashid. 2009. A Framework for Constructing Semantically Composable Feature Models from Natural Language Requirements. In *SPLC '09* (San Francisco, California, USA). Carnegie Mellon University, 211–220.

[73] Karl Dawson Wood. 1964. *Aerospace Vehicle Design - Volume II: Spacecraft Design*. Vol. 2. Johnson Publishing Company.

[74] Wenrui Wu, Hai Huang, Shenyan Chen, and Beibei Wu. 2013. Satellite multidisciplinary design optimization with a high-fidelity model. *Journal of Spacecraft and Rockets* 50, 2 (2013), 463–466.

[75] Yun-Hua Wu, Mo-Hong Zheng, Chao-yong Li, Meng-Jie He, Zhi-Ming Chen, and Bing Hua. 2020. Hybrid actuator optimal angular momentum management with analytical solution for spacecraft attitude agile maneuvering mission. *Aerospace Science and Technology* 97 (2020), 105597.

[76] X. Zhou, X. Gou, T. Huang, and S. Yang. 2018. Review on Testing of Cyber Physical Systems: Methods and Testbeds. *IEEE Access* 6 (2018), 52179–52194. https://doi.org/10.1109/ACCESS.2018.2869834