10th INTERNATIONAL SYSTEMS & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS CONFERENCE (SECESA 2022)

5 - 7 OCTOBER 2022

VARIABILITY-DRIVEN DESIGN CONFIGURATOR OF SPACE SYSTEMS TO SUPPORT DECISION-MAKERS

Loveneesh Rana[1*], Sami Lazreg[1], Vladyslav Bohlachov[1], Andreas Hein[1], Maxime Cordy[1]

[1] SnT, University of Luxembourg
29 Av. John F. Kennedy, Luxembourg, Luxembourg
**Email: loveneesh.rana@uni.lu*

INTRODUCTION

Designing spacecraft is known as an extremely challenging problem. The very first stage of the design process, aka the conceptual design (CD) phase, is even more abstract due to the lack of information and short turnaround time. It requires multi-criteria decision-making on different concerns, from mission analysis to system architectures. In recent years, spacecraft missions have become increasingly ambitious, requiring spacecraft to be designed for various demanding applications. As a result, spacecraft design has become an increasingly challenging endeavour that requires highly complex and multidisciplinary integration of multiple disciplines in a multi-layer hierarchical system architecture solution.
Additionally, at the CD stage, initial unknowns are significant, while system inter-dependencies are complex. Such complexity might lead to design system architectures that are not feasible in practice or do not fulfil nor optimize functional requirements. Moreover, the lack of efficient automation to model and assess numerous design alternatives reduce the benefits of the early design phases by missing more suitable designs and increasing the design time and cost. Thus, either many design iterations by engineers are required, or a well-proven design with flight heritage is selected and refined. Such approaches can give a promising baseline. However, neither approach is certain to yield an optimal solution.

In this paper, we propose a novel spacecraft conceptual design approach to close the previously mentioned gaps. Our framework focus on abstraction, modularity, and reusability of the different concepts used in the traditional spacecraft design process. The different concepts such as mission requirements, architecture, technology, subsystem, components, etc. are captured as features of the design process. A feature is a conceptual element of a tangible solution (e.g., product, configuration, architecture) that exhibits variability concerns such as optionality, constraints, or dependency. Features are reusable across entirely different missions and systems design applications. With such a feature and variability-oriented technique, we propose to capture the whole spacecraft design process as configuration formulation and design decision-making activities that are implemented through requirements engineering and parametric analysis framework. The configuration process is based on a systematic and iterative approach that allows for rapidly exploring a wide range of design options. Our approach is highly flexible and can be easily adapted to meet the specific needs of each spacecraft mission.
This work results from the collaboration between researchers from space system engineering and software engineering domains. Together, we conjecture that variability modelling and analysis methods that have been developed for software systems can improve the space system design process, in particular 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 approach of variability modelling techniques to an application case of space system design. We consider, more precisely, attitude determination and control satellite sub-system.

Our framework mainly consists of three parts. Each part is associated with a step. Step-1) A model that captures requirements, technologies, and constraints between both. This model also includes the variability of the different elements. The design processes of the different technology components are also defined using this model. Step-2) A configurator that supports engineers in making suitable design decisions such as technology components selections concerning requirements. Step-3) A solver that will analyse and optimize the different design variables and parameters of the elements, such as technology components of the selected architectures.

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 addressing design of a subsystem for a 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 system architecture and generate corresponding parametric and constraint optimization models for further analysis. We implement our approach in a toolchain that may support and automate engineers design activities. Our proposed approach and toolchain could potentially act as a benchmark for future research on variability modelling and design of space systems.

SPACECRAFT DESIGN

The CD analysis of complex space systems is typically performed by a multidisciplinary process for designing the system by modeling and integrating its constituting subsystems. The subsystems are modeled through parametric analysis composed of physics-based equations and integrated into a system design framework [1, 2]. In addition, numerous design synthesis tools and frameworks have been developed to support the design of space exploration and analysis [3,4,5,6]. In essence, all design processes and toolchain consist of elements that can be grouped into three main categories. These categories can be considered as the key aspects of a design process. For our context, we refer to these aspects as features classes of a design process. They are namely, the requirements and constraints, parametric analysis, and design decisions, see Fig. **1**.

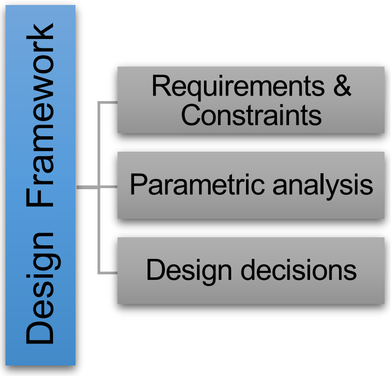


Fig. 1 Three feature-classes as attributes of a design process

Requirements and Constraints: Every design process begins with a set of top-level mission requirements (and constraints) that must be satisfied by the spacecraft. Requirements start and drive the overall design process. Requirements define how the system shall function over its lifetime and what performance is needed to meet the mission objectives. The top-level mission requirements are broken down into system-level requirements from which further subsystem-level requirements are derived. The engineering and management of these requirements is a critical aspect of the design of complex systems. In the traditional waterfall process, requirements are managed sequentially and monolithically, while in agile processes, they are managed incrementally [7]. Requirements management approaches for space systems design have been proposed based on different maturity levels, such as DOORS [8] and ReqIF [9], which support the requirements decomposition and traceability.

Parametric Analysis: In addition to the requirements management, a design process also includes some form of parametric analysis capability. The key function of a parametric analysis engine is to design various subsystem disciplines and integrate them into a system framework such that mission objective could be achieved in the optimal manner. This typically involves physics-based parametric equations that model the characteristic features, performance, and behavior of different subsystems and the entire system in a mathematical framework. For example, the characteristic attributes at the system-level could be the overall physical dimensions, total mass etc., while the performance could be its power consumption. Similarly, an example of the behavior attribute could be how the spacecraft attains the final operational orbit. The important point to note here is that the parametric analysis provides a way to measure and check when the requirements are satisfied. The parametric analysis can be implemented in numerous ways. One way is using any programming language or numeric computing environment such as MATLAB, Python, Simulink, or MS Excel. Alternatively, there is also a wide range of discipline-oriented high-fidelity tools and methods available such as GMAT [10] and STK [11] for mission analysis, several CFD tools for aerothermodynamic analysis, CAD software for structural analysis etc. This level of high-fidelity is not suitable at the CD stage as the inputs required for these tools are usually not available at this early design phase. Further, these tools are computationally exhaustive and lead to a high turn-around time which act as a drawback in developing and exploring the design solution space.

Design Decisions: The third critical aspect of a design process is the design decisions that are usually coupled with the first two. 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. For example, a design decision related to the material of the spacecraft's primary structure influences the structural load capacity. Similarly, the decision regarding the solar cell material (Silicon v/s Gallium Arsenide) influences the solar array sizing and configuration that further defines the power production capability of the spacecraft and hence the performance of the electric power subsystem. At the technology level, design decisions such as using an electric versus a chemical propulsion system directly and significantly impact the entire spacecraft size, configuration, and overall design solution. It should also be considered that some design decisions could be driven by the requirements or imposed by constraints from top-level mission objectives. Hence, modeling design decisions and requirements is a key differentiating aspect across design processes and tools that impact design quality. Additionally, how well this modeling is integrated with the parametric analysis also impacts the efficiency of the design framework. 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.

FEATURE-DRIVEN MODELING FRAMEWORK

In this section, we outline our solution that combines and extends spacecraft design and variability engineering [12] to provide a framework that improves and support the design process of spacecraft. The proposed framework is model driven [13] and aims to, i) assist engineers in the different modeling and design steps ii) automatically derives and captures the feasible system architectures concerning requirements and constraints, iii) refines selected architectures through an optimizer, iv) support engineers during trade-off analysis.

The key aspect of our approach is the explicit captures of system architecture variability at both component and parameter levels. This allows us to infer inter-dependencies between subsystems but also commonalities and differences across architectures. Consequently, engineers can select and assess suitable architectures. As mentioned in the previous section, a design process comprises three feature classes: requirements and constraints, parametric analysis, and design decisions. We use this as the fundamental tenet for our approach, where the concept of features drives the design and modelling process. In this feature-driven framework, requirements are integrated with parametric analysis and design decisions in the design process and toolchain from the very beginning.

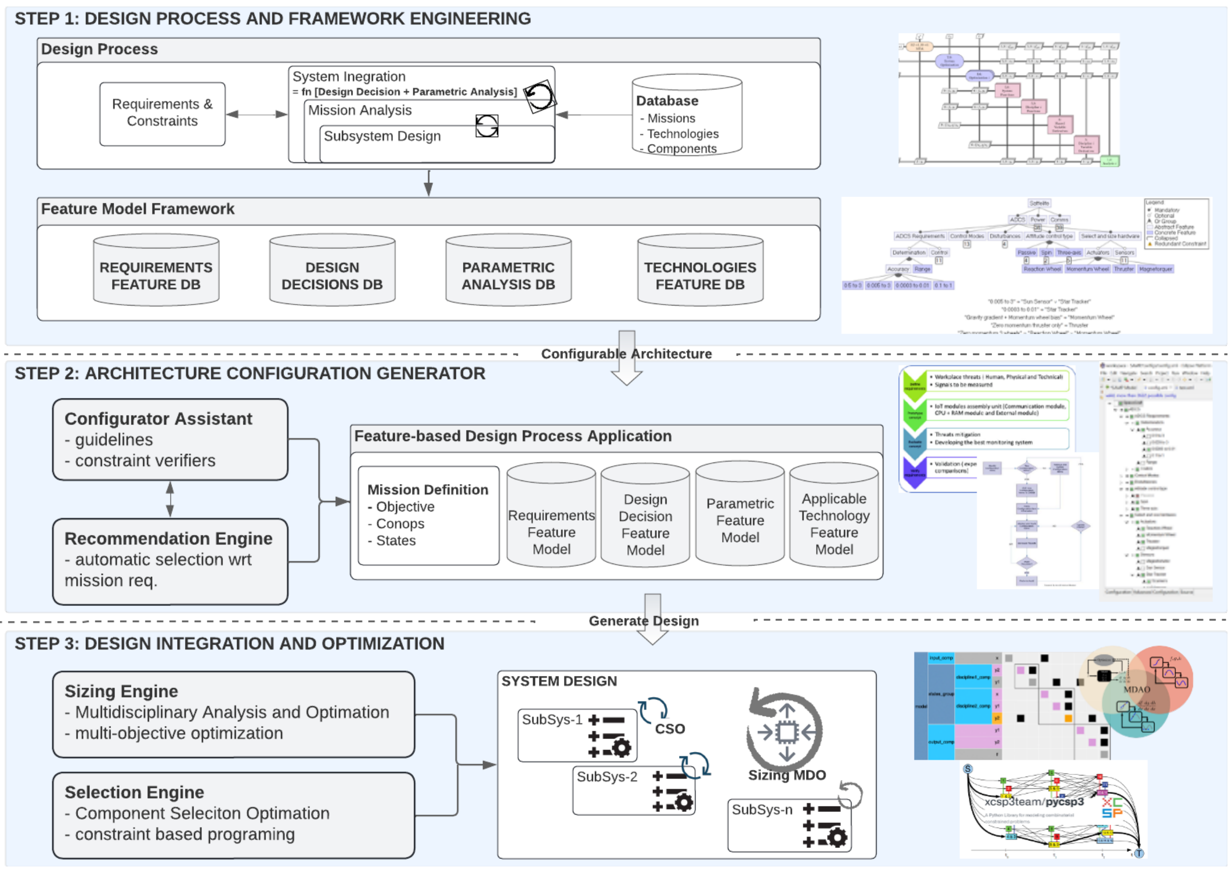

STEP 1: DESIGN PROCESS AND FRAMEWORK ENGINEERING
Design Process
Requirements & Constraints
System Inegration
= fn [Design Decision + Parametric Analysis]
Mission Analysis
Subsystem Design
Database
- Missions
- Technologies
- Components
Feature Model Framework
REQUIREMENTS FEATURE DB
DESIGN DECISIONS DB
PARAMETRIC ANALYSIS DB
TECHNOLOGIES FEATURE DB
Configurable Architecture
STEP 2: ARCHITECTURE CONFIGURATION GENERATOR
Configurator Assistant
- guidelines
- constraint verifiers
Recommendation Engine
- automatic selection wrt mission req.
Feature-based Design Process Application
Mission Definition
- Objective
- Conops
- States
Requirements Feature Model
Design Decision Feature Model
Parametric Feature Model
Applicable Technology Feature Model
Generate Design
STEP 3: DESIGN INTEGRATION AND OPTIMIZATION
Sizing Engine
- Multidisciplinary Analysis and Optimation
- multi-objective optimization
Selection Engine
- Component Seleciton Optimation
- constraint based programing
SYSTEM DESIGN
SubSys-1
CSO
SubSys-2
Sizing MDO
SubSys-n
MDAO
xcsp3team/pycsp3

Fig. 2 Feature-driven design framework

Our novel framework is developed in three steps, as shown in Fig. 2 and described next.

Step-1: The design process and feature framework development

The first step involves setting up the design process in terms of a feature model [14]. Features are all artifacts that do not depend on a particular mission or a design process. In this step, the focus is on developing generic feature-driven databases that capture the three feature classes. In a way, this step focuses on developing feature-based building blocks of a design process. In order to accomplish this, we have to set up a design process that could be decomposed into the three feature-based classes. It's important to note that this first step is independent of any specific design process. Features can capture every decisional aspect of the mission definition and system architecture and dependency

relationships between them. Instead, the main goal is to develop the design framework composed of three feature classes. For the current framework, we are using the generic design process from the well-known Space Mission Analysis and Design (SMAD) textbook [15]. In this regard, the first step of the framework could be thought of as converting the SMAD into various feature models that are generic in nature and are applied in the specific context of designing missions on demand.
The design process begins with mission objectives accompanied by top-level mission requirements, which then drive the requirements at system and subsystem levels. The requirements engineering leads to parametric analysis for mission analysis and all spacecraft subsystems that are integrated into a system-level solution. The parametric analysis applies a well-established set of physics-based equations from SMAD, and here, we are coding parametric analysis for different subsystems (ADC, power, structures, propulsion etc.) in python script modules. We select python for its advantages of being an object-oriented language and large open-source libraries. In addition to the physics-based equations, the parametric analysis framework is also composed of a database of previous missions, systems, technologies, and components. Following this design framework setup, we next model the variability and the configuration space using feature models [16]. Here, we integrate requirements and technologies with variability concerns in a Feature-Oriented Engineering Model. In this feature model, features represent the variable subsystem components but also requirements, mission modes, spacecraft states, etc.

Step-2: Automatic configurator providing combination of all possible architecture solutions
Once the feature model is setup, the next step addresses mission definition and system architecture generation. In the second step, we propose a reactive configurator that will automatically provide all possible system architecture combinations that satisfy the defined top-level requirements and performance constraints [17, 18].
Mission definition includes a definition of mission con-ops, development of mission profile, and identifying spacecraft states based on the mission profile. Further, mission definition also includes top-level mission requirements and constraints from the mission objective. The designer uses all this information to specify system-level design decisions that are valid for the mission requirements. The mission-specific information is then applied to the feature model framework developed in step-1. The feature model uses this mission-specific information and matches it with the technology/ component database to provide all possible combinations of various components where each combination is system architecture. Thus, in this step, the designer is initially involved in setting up the requirements feature database and implementing design decisions. Note that in this step, the design framework is not implementing any parametric feasibility checks or even specifying any one particular solution but rather only produces all possible architecture combinations that could be plausible.

Step-3: Selects the most optimum combination or optimize the sizing parameters.
Then the next step provides optimization methods to select the most suitable system architecture from all plausible ones. This requires reasoning on system architecture's different functional and non-functional facets. Depending on the architecture's different subsystems, we may need to reason on both architecture parameters' optimization (sizing) and architecture component optimization (selection).
For this, we propose integrating constraint programming [19] and multi-disciplinary optimization (MDO) [20] capabilities. Our integration uses these paradigms in a complimentary way to select the most optimum combination and/or optimize the sizing parameters. The key point of our optimization approach is to drive the optimization process using the level of detail of the design decisions made during the definition and configuration step (step 2).
For example, a high level of design decisions at the component level will first invoke the MDO capabilities to design the different component parameters using a traditional parametric process. Then the sized components can be manufactured in a tailored fashion, or the "nearest" existing components in the database could be recommended to the designers. On the other hand, we can formulate our architecture selection problem as a constraint optimization problem using our constraint programming capabilities. Thus, we can exhaustively explore the different component alternatives to optimize the component selection. Both constraint optimization problem and MDO problem formulation are automatically generated through using the configurator module.

APPLICATION CASE-STUDY
In this section we demonstrate the application of our proposed framework. We apply the solution outlined in the previous section towards design of a spacecraft subsystem first as a proof-of-concept. The presented application is only intended to act as a demonstration for implementing the solution in a toolchain. The aim here is to get clarity and experience of the development process. This application is rather a simplistic version of the ultimate vision for this framework which will address the entire system. In this regard, the implementation workflow remains the same while the application complexity will increase multifold due to various subsystems inter-dependencies and larger design decisions space.
For this paper, we selected the Attitude Determination and Control Subsystem (ADCS) which is a crucial subsystem of any spacecraft. In the following subsections, we first provide an overview of the generic ADCS design process which corresponds to the first step of our proposed framework. We next present a feature-model corresponding to the design process. These two constitute the first step of the framework. The following subsection implements the second step of our framework explaining ACDS feature-based configurator identifying a valid ADCS solution. The last subsection

addresses the third and final step of our framework and explains optimal architecture selection/sizing through an optimization application.

ADCS Design Process

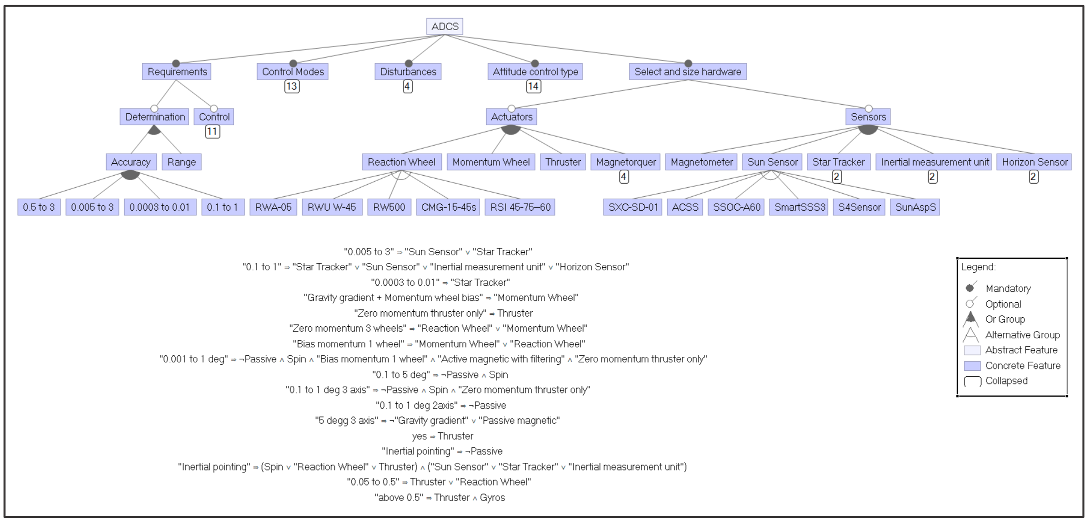
The ADCS stabilizes the vehicle and orients it in desired direction in consideration with the external disturbance torques acting on it. This subsystem can be divided into 2 hardware component categories: sensors that determine the position and orientation of the spacecraft, and actuators that perform the control function and compensate disturbances. Thus, the goal of the ADCS design process is to provide with a combination of sensors and actuators. The design process of a spacecraft's ADCS is divided into six major steps, as follows.

1. Define System-level requirements: Payload and spacecraft pointing accuracy, stability accuracy, slew rate etc
2. Define control modes: Orbit Insertion, Acquisition, Nominal operation, Slew, Safe, Special
3. Quantify disturbances: Gravity gradient, magnetic, aerodynamics, solar pressure, internal disturbances, and powered flight effects on control
4. Select Attitude control type: Passive control, spin control, or 3-axis control
5. Select and size ADCS hardware: Actuators (reaction wheels, thrusters or magnetic torquers), Sensors (Earth, Sun, inertial or other sensors)

The process begins with defining system requirements in first step which corelates to top-level mission requirements. Second step is selection of the control modes which depends on the mission definition and con-ops. In this step, we define how s/c control is implemented in various mission phases. Next, primary disturbances or the external torques the satellite will face during its lifetime are calculated. These are taken from SMAD and calculate four major type of disturbance torques arising from gravity-gradient, magnetic field, aerodynamics, solar pressure. Based on these disturbances and system requirements, the next step then proceeds to either size or select a combination of different actuators and sensors that can fulfil the mission requirements. It can be deduced from above description that this 5-step process for even a subsystem is a combination of three feature attributes of requirements (in step-1), design decisions (in steps 2, 4, and 5), and parametric analysis (in steps 3 and 5).

ADCS Feature Framework Development

We now present an overview of the development of our feature model implemented in the FeatureIDE [21] as shown in FIG. 3. The root feature represents the ADCS subsystem composed of 72 design decisions encoded as boolean features and 27 crosstree constraints. This feature decomposition follows the design workflow presented in the previous subsection.

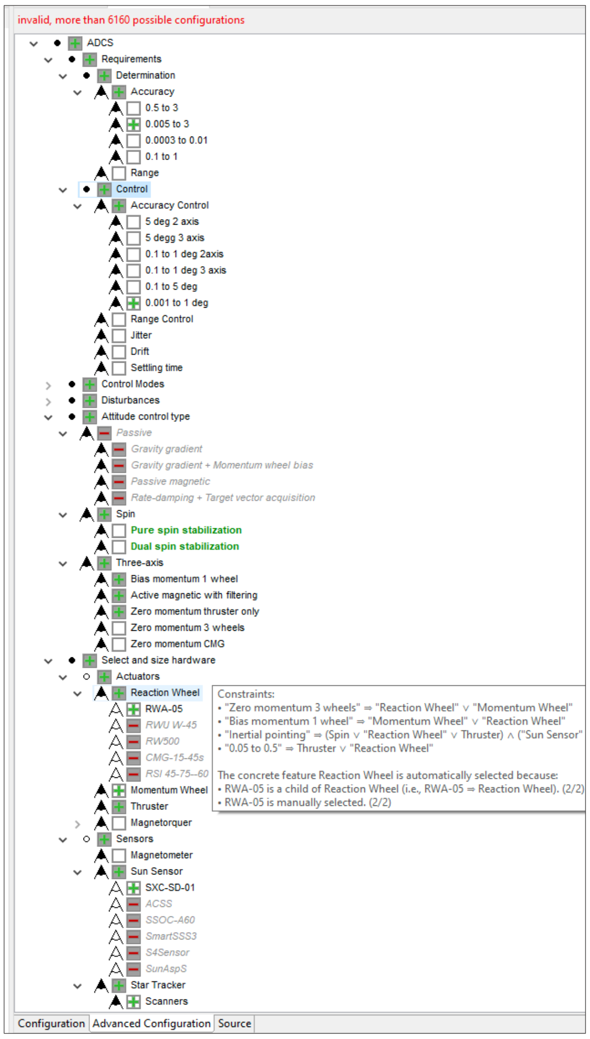


"0.005 to 3" ⇒ "Sun Sensor" ∨ "Star Tracker"
"0.1 to 1" ⇒ "Star Tracker" ∨ "Sun Sensor" ∨ "Inertial measurement unit" ∨ "Horizon Sensor"
"0.0003 to 0.01" ⇒ "Star Tracker"
"Gravity gradient + Momentum wheel bias" ⇒ "Momentum Wheel"
"Zero momentum thruster only" ⇒ Thruster
"Zero momentum 3 wheels" ⇒ "Reaction Wheel" ∨ "Momentum Wheel"
"Bias momentum 1 wheel" ⇒ "Momentum Wheel" ∨ "Reaction Wheel"
"0.001 to 1 deg" ⇒ ¬Passive ∧ Spin ∧ "Bias momentum 1 wheel" ∧ "Active magnetic with filtering" ∧ "Zero momentum thruster only"
"0.1 to 5 deg" ⇒ ¬Passive ∧ Spin
"0.1 to 1 deg 3 axis" ⇒ ¬Passive ∧ Spin ∧ "Zero momentum thruster only"
"0.1 to 1 deg 2axis" ⇒ ¬Passive
"5 degg 3 axis" ⇒ ¬"Gravity gradient" ∨ "Passive magnetic"
yes ⇒ Thruster
"Inertial pointing" ⇒ ¬Passive
"Inertial pointing" ⇒ (Spin ∨ "Reaction Wheel" ∨ Thruster) ∧ ("Sun Sensor" ∨ "Star Tracker" ∨ "Inertial measurement unit")
"0.05 to 0.5" ⇒ Thruster ∨ "Reaction Wheel"
"above 0.5" ⇒ Thruster ∧ Gyros

Fig. 3 Feature model developed for ADCS application.

The first subfeature of ACDS concerns the definition of this subsystem's requirements, corresponding to the first design step of the ACDS design process. This is essential because these requirements have the largest impact on the desing decisions, recommendations, and selection of articular hardware combinations. 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 corresponds to the second step in the ACDS design process and demonstrates capturing the design decision feature. Selecting a particular set of control modes will constrain the acceptable set of satellite architectures. The next feature attribute, 'Disturbances', captures the parametric equations analysis feature class used to compute external disturbances torques. Attitude control type determines the main control systems concept (and technologies). These can be seen as a popular combination of hardware for standard purposes. The last subfeature of ACDS "select and size hardware" addresses 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.

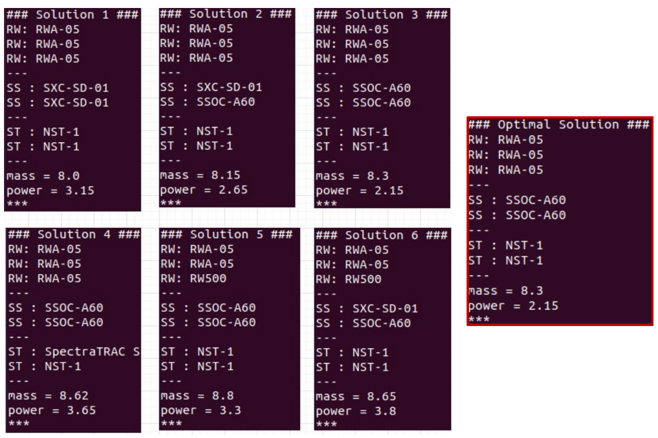


```
invalid, more than 6160 possible configurations
ADCS
  Requirements
    Determination
      Accuracy
        0.5 to 3
        0.005 to 3
        0.0003 to 0.01
        0.1 to 1
      Range
    Control
      Accuracy Control
        5 deg 2 axis
        5 degg 3 axis
        0.1 to 1 deg 2axis
        0.1 to 1 deg 3 axis
        0.1 to 5 deg
        0.001 to 1 deg
      Range Control
      Jitter
      Drift
      Settling time
  Control Modes
  Disturbances
  Attitude control type
    Passive
      Gravity gradient
      Gravity gradient + Momentum wheel bias
      Passive magnetic
      Rate-damping + Target vector acquisition
    Spin
      Pure spin stabilization
      Dual spin stabilization
    Three-axis
      Bias momentum 1 wheel
      Active magnetic with filtering
      Zero momentum thruster only
      Zero momentum 3 wheels
      Zero momentum CMG
  Select and size hardware
    Actuators
      Reaction Wheel
        RWA-05
        RWU W-45
        RW500
        CMG-15-45s
        RSI 45-75--60
      Momentum Wheel
      Thruster
      Magnetorquer
    Sensors
      Magnetometer
      Sun Sensor
        SXC-SD-01
        ACSS
        SSOC-A60
        SmartSSS3
        S4Sensor
        SunAspS
      Star Tracker
        Scanners

Constraints:
• "Zero momentum 3 wheels" ⇒ "Reaction Wheel" ∨ "Momentum Wheel"
• "Bias momentum 1 wheel" ⇒ "Momentum Wheel" ∨ "Reaction Wheel"
• "Inertial pointing" ⇒ (Spin ∨ "Reaction Wheel" ∨ Thruster) ∧ ("Sun Sensor"
• "0.05 to 0.5" ⇒ Thruster ∨ "Reaction Wheel"

The concrete feature Reaction Wheel is automatically selected because:
• RWA-05 is a child of Reaction Wheel (i.e., RWA-05 ⇒ Reaction Wheel). (2/2)
• RWA-05 is manually selected. (2/2)
```

Fig. 4 FeatureIDE Configurator implementation showing selected feature attributes for ADCS application.

ADCS FeatureIDE Configurator

Once the feature model for ADCS is developed, we now apply this model by identifying few selective attributes that represent an ADCS configuration.
Fig. 4 shows a partial valid configuration that describes an ADCS solution. The configuration process usually starts by defining the sub-system requirements such as accuracy in determination and control. These first design decisions represented by features selection will drive the possible actuators and sensors component space thanks to constraints between features. These constraints can be hard constraints that capture an impossibility in practice or more general design guidelines. As can be seen in the figure, the configurator implements specific requirements (determination and control accuracy levels), decisions (selection of control type and hardware components such as reaction wheels, sensors et al.) and parametric analysis (calculation of disturbance torques). These selections are made specific to the mission and spacecraft being designed. The component selection is also done out of a database of various components based on mission requirements and design decisions. For this application, we have developed a representative database of various off-the-shelf actuators and sensors.
With this setup, we enable engineers to configure subsystem and system architecture based on mission requirements and design decisions attributed to the specific configuration, in this case, for the ADCS solution. This view shows again that the feature-oriented configuration process allows engineers to configure the different subsystems in an arbitrary order. Furthermore, the interactive activation of the constraints enables the identification and understanding of the interdependencies between the design choices.



```
### Solution 1 ###
RW: RWA-05
RW: RWA-05
RW: RWA-05
---
SS : SXC-SD-01
SS : SXC-SD-01
---
ST : NST-1
ST : NST-1
---
mass = 8.0
power = 3.15
***
```

```
### Solution 2 ###
RW: RWA-05
RW: RWA-05
RW: RWA-05
---
SS : SXC-SD-01
SS : SSOC-A60
---
ST : NST-1
ST : NST-1
---
mass = 8.15
power = 2.65
***
```

```
### Solution 3 ###
RW: RWA-05
RW: RWA-05
RW: RWA-05
---
SS : SSOC-A60
SS : SSOC-A60
---
ST : NST-1
ST : NST-1
---
mass = 8.3
power = 2.15
***
```

```
### Solution 4 ###
RW: RWA-05
RW: RWA-05
RW: RWA-05
---
SS : SSOC-A60
SS : SSOC-A60
---
ST : SpectraTRAC S
ST : NST-1
---
mass = 8.62
power = 3.65
***
```

```
### Solution 5 ###
RW: RWA-05
RW: RWA-05
RW: RW500
---
SS : SSOC-A60
SS : SSOC-A60
---
ST : NST-1
ST : NST-1
---
mass = 8.8
power = 3.3
***
```

```
### Solution 6 ###
RW: RWA-05
RW: RWA-05
RW: RW500
---
SS : SXC-SD-01
SS : SSOC-A60
---
ST : NST-1
ST : NST-1
---
mass = 8.65
power = 3.8
***
```

```
### Optimal Solution ###
RW: RWA-05
RW: RWA-05
RW: RWA-05
---
SS : SSOC-A60
SS : SSOC-A60
---
ST : NST-1
ST : NST-1
---
mass = 8.3
power = 2.15
***
```

Fig. 5 Constraint optimization and optimal component architecture selection using PyCSP3

ADCS Optimal selection and sizing

Once the design engineers have reached the desired level of detail in the architecture configuration that is valid -- according to the encoded feature model constraints -- further analysis must be conducted to size and/or select components in order to make sure that at least one architecture meet the functional and non-functional requirements in practice. For this aim, we propose two complementary ways to reason about the architecture derived from the features configuration.

Component Selection Optimization

First, we propose a constraint-based programming approach where we select the most optimal combination of hardware component architectures, as shown in Fig. 5. For the current paper, we used PyCSP3 [22], a constraint satisfaction and optimization solver to list the architectural components selections that satisfy performance constraints such as minimum torque required for actuators or accuracy required for sensors. For the ADCS featureIDE configuration developed in the previous step, we first filter out individually the components that first satisfy specified performance constraints. One example for this implementation is the actuators category where in the featureIDE configurator, we selected reaction wheels as the desired category of actuators. We then go to our database of reaction wheels where we have a list of six existing reaction wheels. We filter this list by applying the constraint of minimum torque required from a reaction wheel. In current implementation, we found that only two reaction wheels met our torque constraints. Similarly, we perform constraint-based filtering for other components i.e. sensors. For the current demonstration, we only limited to sun-sensors and star-trackers as our desired sensors. Once this filtering is done, we then combine all filtered components in specific ADCS architectures. This combination is still guided by the high-level architecture design performed in the featureIDE configurator in the previous step. As can be seen in Fig. 5, we notionally show only six ADCS solutions, but in reality, our solver found more than 3000 solutions by combining all filtered out reaction wheels, sun-sensors and star-trackers options. Our PyCSP3 optimizer then selects the most optimal architectural components selection by adding a cost function over mass and power consumption to minimize by the solver.

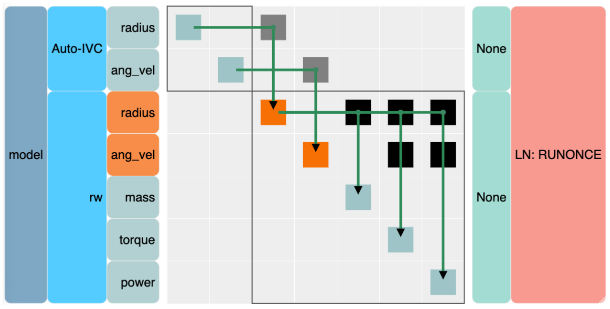
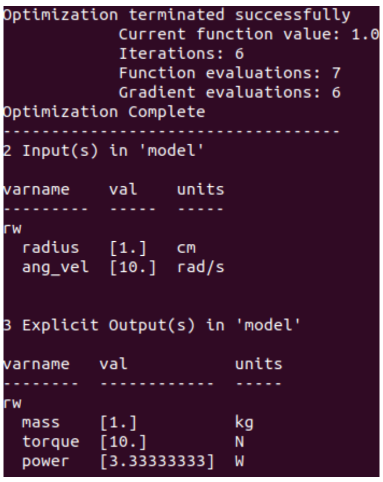


Fig. 6 Parametric sizing optimization example for a reaction wheel in OpenMDAO

Multi-Disciplinary Analysis and Optimization (MDAO)

A second optimization direction is to actually size the components by considering the performance constraints and applying the physical sizing equations to optimize on one or multiple cost functions. The sized components will be used to select the most suitable components in the database. The main assumption is that selection of the nearest/closest components regarding the optimally sized ones will provide the best architecture.

For this we reuse the parametric process and mathematical models previously created in step 1. However, this approach can also drive the selection of the most suitable components in the database. In current case of ADCS subsystem, the sizing equations address actuators such as reaction wheels and thrusters. Fig. 6 illustrates the sizing process of a reaction wheel in OpenMDAO [23]. This process shows the dependency and computation between input and output parameters. The two input parameters that also act as design variables (radius and angular velocity) represent the property of the reaction wheel. These inputs are used to compute the outputs such as mass, torque and power consumption that represent performances of the reaction wheel. Such computations are generally nonlinear. Thus, we used sequential least squares programming optimization algorithms to converge to design variable values that optimize the reaction wheel mass while satisfying a minimum torque requirement.



```
Optimization terminated successfully
            Current function value: 1.0
            Iterations: 6
            Function evaluations: 7
            Gradient evaluations: 6
Optimization Complete
-----------------------------------
2 Input(s) in 'model'

varname    val    units
--------   -----  -----
rw
  radius   [1.]   cm
  ang_vel  [10.]  rad/s


3 Explicit Output(s) in 'model'

varname   val           units
-------   ------------  -----
rw
  mass    [1.]          kg
  torque  [10.]         N
  power   [3.33333333]  W
```

Fig. 7 Reaction wheel sizing optimization results

In this example (see Fig 7), as the mass only depends on the radius, the optimizer converges into the minimal radius value with an angular velocity value that produces the minimum torque requirement. The tool lists the input and output optimized parameter values and some info about the algorithm metrics. Here only 6 gradient evaluations have been sufficient to converge to an optimal reaction wheel parameter values.

For this paper, we do not size the reaction wheels as we select commercially available reaction wheels that can fulfill torque and angular momentum requirements. The example shown here is a representational example to show that MDAO is another optimization method which can be more valuable for subsystems such as propulsion or power where components (propellant tanks or solar arrays) are parametrically sized to mission requirements. MDAO is also a more suitable optimization approach for sizing the entire system as previously described in step-3 of our framework.

CONCLUSION

The goal of this paper has been to propose a novel research direction that stemmed out of collaboration between researchers from space systems engineering and software engineering domains. We have adopted tools and expertise from the two domains towards innovating new ways of approaching the spacecraft design problem and to develop a prototype solution concept.

In this paper, we have presented a prototype design approach that is feature-driven and address variability concerns such as optionality, constraints, or interdisciplinary dependencies in a design process. We modelled the set of design choices, requirements, and parametric analysis aspects of a design process as a feature model. We have applied this logic to develop and implement a toolchain to demonstrate our proposed methodology.

We, then, use the FeatureIDE tool to enable engineers to specify these design choices to configure the design architecture 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 in variability management as it benefits from the capability of FeatureIDE to support configuration activities, combined with our domain-specific analysis tool.

The application in the current paper is limited to a spacecraft subsystem design and demonstrate a working example of our proposed solution and toolchain. We are currently working on expanding our framework for other spacecraft subsystems (power, structure, propulsion et al.) and eventually apply our toolset to comprehensively explore design solution space for the entire spacecraft and system-of-system applications. 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.

REFERENCES

[1] L. Rana and B. Chudoba, “Demonstration of a prototype design synthesis capability for space access vehicle design”, The Aeronautical Journal 124, 1281, 1761–1788, 2020.

[2] Salima Berrezzoug, Abdelmadjid Boudjemai, and Fethi Tarik Bendimerad, “Interactive design and multidisciplinary optimization of geostationary communication satellite”. International Journal on Interactive Design and Manufacturing, (IJIDeM) 13, 1519–1540, 2019.

[3] D. Domizio and P. Gaudenzi, “A Model for Preliminary Design Procedures of Satellite Systems. Concurrent Engineering: Research and Applications”, SAGE Publications, 16 (2), pp.149-159, 2008.

[4] M. Mosleh, K. Dalili and B. Heydari, "Distributed or Monolithic? A Computational Architecture Decision Framework," in IEEE Systems Journal, vol. 12, no. 1, pp. 125-136, March 2018.

[5] F. Wang, M. Schrock, and C. S. Borden, “Trade Space Specification Tool (TSST) for Rapid Mission Architecture (Version 1.2),” NASA Tech Briefs, p. 35, 2013.

[6] S. J. I. Herzig, S. Mandutianu, H. Kim, S. Hernandez and T. Imken, "Model-transformation-based computational design synthesis for mission architecture optimization," 2017 IEEE Aerospace Conference, 2017.

[7] Kazman, R., Bass, L., Klein, M. et al. “A Basis for Analyzing Software Architecture Analysis Methods”, Software Qual J 13, 329–355, 2005.

[8] IBM, “IBM Engineering Requirements Management DOORS”, url: https://www.ibm.com/docs/en/ermd

[9] S. Mazzini, J. Favaro, R. S. Ulrich Lang, H-P de Koning, “Improving Requirements Engineering within the European Space Industry”, Embedded Real Time Software and Systems (ERTS2012), Toulouse, France, Feb 2012.

[10] NASA, “General Mission Analysis Tool (GMAT) v.R2016a”, url: https://software.nasa.gov/software/GSC-17177-1

[11] AGI, “Ansys Systems Tool Kit (STK)”, url: https://www.agi.com/products/stk

[12] K. Pohl, G. Böckle, & F. Van Der Linden, “Software product line engineering” (Vol. 10, pp. 3-540), Heidelberg: Springer, 2005.

[13] T. Basten, et al. (2010), “Model-Driven Design-Space Exploration for Embedded Systems: The Octopus Toolset. In: Margaria, T., Steffen, B. (eds) Leveraging Applications of Formal Methods, Verification, and Validation”, ISoLA 2010, Lecture Notes in Computer Science, vol 6415. Springer, Berlin, Heidelberg, 2010.

[14] D. Batory, “Feature models, grammars, and propositional formulas”, In International Conference on Software Product Lines (pp. 7-20), Springer, Berlin, Heidelberg, 2005.

[15] W. J. Larson, J. R. Wertz, and B. D'Souza, “SMAD III: Space Mission Analysis and Design, 3rd Edition”, El Segundo, CA, USA, Microcosm Press, 2005.

[16] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, & A. S. Peterson, “Feature-oriented domain analysis (FODA) feasibility study”, Software Engineering Inst., Carnegie-Mellon Univ Pittsburgh PA, USA, 2009.

[17] K. Czarnecki, S. Helsen, & U. Eisenecker, “Staged configuration using feature models”, In International conference on software product lines (pp. 266-283), Springer, Berlin, Heidelberg, 2004.

[18] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, & G. Saake, “SPL Conqueror: Toward optimization of non-functional properties in software product lines”, Software Quality Journal, 20(3), 487-517, 2012.

[19] F. Rossi, P. Van Beek, & T. Walsh, “Handbook of constraint programming”, Elsevier, 2006

[20] I. Kroo, S. Altus, R. Braun, P. Gage, & I. Sobieski, “Multidisciplinary optimization methods for aircraft preliminary design”, in 5th symposium on multidisciplinary analysis and optimization (p. 4325), Sept, 2005.

[21] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, & T. Leich, “FeatureIDE: An extensible framework for feature-oriented software development”, Science of Computer Programming, 79, 70-85, 2004.

[22] C. Lecoutre, & N. Szczepanski, “PYCSP3: modeling combinatorial constrained problems in python”, arXiv preprint arXiv:2009.00326, 2009.

[23] J. S. Gray, J. T. Hwang, J. R. Martins, K. T. Moore, & B. A. Naylor, “OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. Structural and Multidisciplinary Optimization", 59(4), 1075-1104, 2019.