System Architecture Optimization: An Example Application to Space Mission Planning

Jasper H. Bussemaker[1] [0000-0002-5421-6419] and Thomas Firchau[2] [0000-0001-5404-2482]

[1] Institute of System Architectures in Aeronautics, DLR, Hamburg, Germany
[2] Institute of Space Systems, DLR, Bremen, Germany

Abstract. Space mission planning involves coupled architecture decisions that non-trivially influence system-level performance metrics such as weight, power usage, and scientific value. We present an exemplary space mission planning problem solved using System Architecture Optimization (SAO): a technique where numerical optimization algorithms are used to explore the architecture design space and find a Pareto front of architectures. The architecture design space is modeled based on functions using the Architecture Design Space Graph (ADSG) implemented in the ADORE editing and optimization tool. Evaluation code is implemented in Python and linked to the design space model using class factories. The design space is explored using NSGA-II, a multi-objective evolutionary algorithm, resulting in a Pareto front trading-off system mass and total experiment duration.

Keywords: System Architecture · MBSE · Missing Planning · Optimization

1 Introduction

Increasing system complexity and more stringent stakeholder needs in the space domain require a shift towards innovative model-based design space exploration approaches in early design phases [14]. Such a shift would allow exploring more potential concepts, at a higher level of detail and subject to less designer bias, earlier in the design phase [15]. In particular, design decisions early in the design process include selecting payload such as scientific instruments, calculating operational budgets such as operational time and data handling, and sizing support systems such as power supply and thermal regulation systems. Space mission design is a non-trivial task due to strong coupling of architecture decisions with respect to system-level performance metrics [11].

In this paper, we demonstrate the application of System Architecture Optimization (SAO) techniques to space mission design. SAO is an emerging field that applies numerical optimization to exploring system architecture design space, thereby enabling exploring a larger number of potential candidates earlier in the design process [5]. We continue with an overview of SAO in Section 2. The space mission planning application case in described in Section 3. Results are presented and discussed in Section 4 and Section 5 concludes the paper.

2 System Architecture Optimization

System Architecture Optimization (SAO) involves the application of numerical optimization algorithms to designing system architectures. The architecture of a system is a central artifact in systems engineering, and specifies what components a system consists of (the elements of *form*), and how they collaborate to fulfill the system *functions* (i.e. what the system performs) [8]. Many potential architectures may exist for a given design problem, making it infeasible to exhaustively consider all architectures [13]. This warrants the application of optimization techniques to selectively search the design space.

Implementing an architecting problem as an optimization problem requires the availability of an *architecture generator* and an *architecture evaluator*. In this work, we use the Architecture Design Space Graph (ADSG) [5] to model the architecture design space and enable architecture generation. The ADSG is a directed graph modeling functions fulfillment choices, function induction by components, component characterization choices, and component connection choices. Nodes represent architecture elements such as functions, components, and component instances, and edges represent derivation relationships. For a comparison to other architecture optimization approaches, the interested reader is referred to [5].

The ADSG is implemented in ADORE, a Python tool developed by the DLR that includes a web-based user interface for graphically modeling the architecture design space [4]. ADORE additionally contains interfaces to various optimization algorithms and interfaces for connecting to architecture evaluation code. *Architecture evaluation* code is problem specific and depending on what tools are useful for a given architecting problem can be implemented using different tools, environments, and/or programming languages. Generated architecture alternatives must be translated to the domain-specific input needed for the evaluation code. ADORE provides several different interfaces for this. For example, if evaluation is performed in Python code, ADORE's data model can be used to instantiate objects based on selected architecture elements. If connection to an external environment is needed, generated architectures can be serialized to JSON or XML.

3 Space Mission Planning Problem Implementation

This section presents the application of SAO to space mission planning. The mission planning problem involves maximizing scientific value provided by a selected mission payload, subject to weight and power budgets. This application case was developed as part of the MBSE-Ops project [3], which had the goal of promoting MBSE adoption within the DLR.

Architecture generation is enabled by modeling the architecture design space as an ADORE model, and using that model to formulate the optimization problem. Fig. 1 shows the system view of the design space model, starting from the "Do Science" top-level function. This function represents the system-level goal and therefore also has the system-level performance parameters associated to it: "System Mass" and "Total Experiment Duration" act as optimization objectives, to be minimized and maximized, respectively. "Valid schedule" is a constraint that ensures the experiment schedule is valid (no experiments are overlapping in time). The top level function is fulfilled by the "Experiment" component, of which the detailed view is shown in Fig. 2. The Experiment can be instantiated between 1 and 5 times, and each instance has two continuous design variables: "Start Time" and "Duration". Instantiating the experiment and choosing values for the two design variables is a good example of decision hierarchy, a common occurrence in SAO problems, where an upstream decision (instantiation) determines whether downstream decisions (start time and duration) are active or not [5].

The experiment needs the "Gather Data" function, which is fulfilled by the "Instrument". The instrument contains no additional specialization, however has two static inputs associated to it: "Mass" and "Power Requirement". Static inputs make assumptions and component properties explicit, by moving their definition from the evaluation code to the system model. The Instrument needs two further functions: "Store Data", fulfilled by the "Onboard Computer", and "Provide Power", fulfilled by the "Battery". The onboard computer also needs data, showing that it is possible for multiple components to need a function. The Battery, see Fig. 3 for the component view, can be instantiated between 1 and 5 times. Each instance has three static inputs: "Mass",

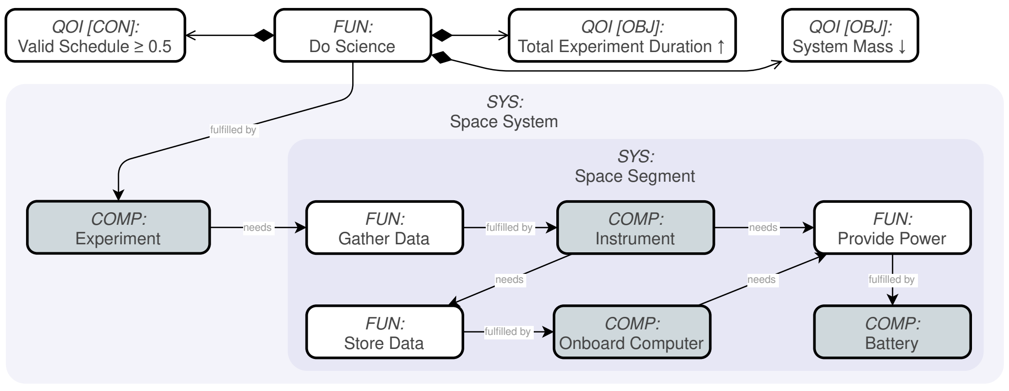

Fig. 1: ADORE design space model showing the system view.

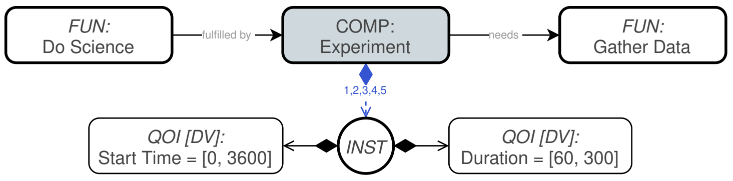

Fig. 2: ADORE design space model showing the "Experiment" component.

"Maximum Power", and "Capacity". On the component-level, the Battery has a "Remaining Energy" constraint that ensures that the battery is not depleted when running experiments.

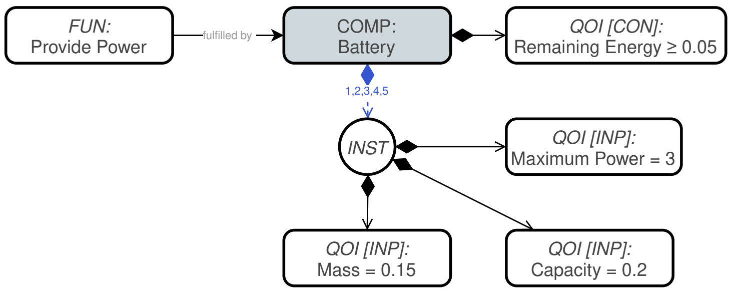

Fig. 3: ADORE design space model showing the "Battery" component.

The optimization problem is a mixed-discrete, constrained, multi-objective problem. It is defined by two discrete (instantiations) and ten continuous design variables (experiment start time and duration). It contains two objectives (System Mass and Total Experiment Duration) and two constraints (Valid Schedule and Remaining Energy).

Fig. 4 shows the data model used as input performance calculation: the SpaceMission object contains the experiment schedule, represented by a list of Operation or Experiment classes, and one Spacecraft class that contains a list of Equipment classes. Several types of equipment are implemented, separated in PowerProvider and PowerConsumer classes. Currently the only power provider is a Battery, and consumers are comprised of OnBoardComputer and Instrument. This object-oriented structure of the data model enables separation of concern and thereby independent development. Additionally, it eases multidisciplinary calculation of

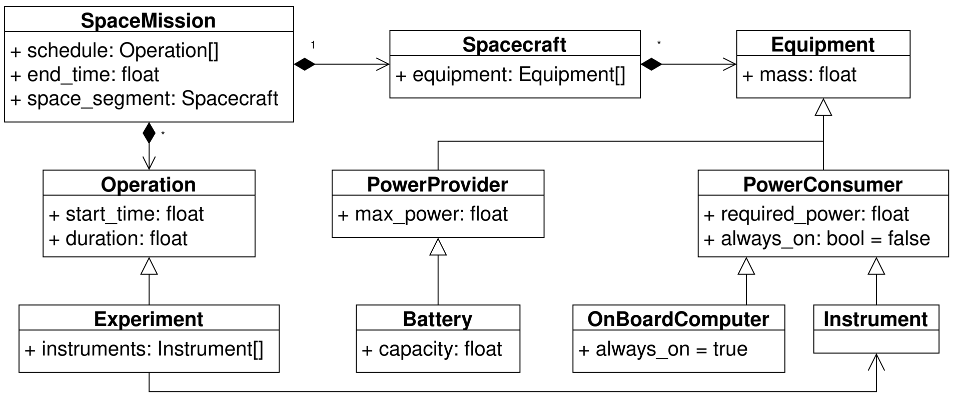

Fig. 4: Evaluation code data model.

performance metrics by using the principle of abstraction. The four performance metrics are calculated by following disciplines:

- The "mass analysis" discipline calculates the System Mass by summing all equipment masses.
- The "payload analysis" disciplines calculates the Total Experiment Duration by summing the duration of all operations of type Experiment.
- The "operations analysis" discipline determines whether the schedule is valid (the Valid Schedule flag) by checking if there are any overlapping operations.
- The "power analysis" discipline calculates the Remaining Energy by simulating power consumption over the mission duration, based on Battery capacities and required power of power consumers. Power consumers are either always on (e.g. the OnBoardComputer) or only consume power when used in an operation.

Fig. 5 shows power analysis output for an example mission containing two experiments.

To enable architecture evaluation using ADORE, the architectures generated by ADORE have to be translated to the data model presented in Fig. 4. This is done using class factories: rules for instantiating Python classes based on architecture element occurrences. Class factories are defined for all objects in the design space, such that the SpaceMission object can be populated.

In this section the architecture design space model, data model, and analysis disciplines are presented sequentially, however it has to be noted that all these aspects are strongly dependent

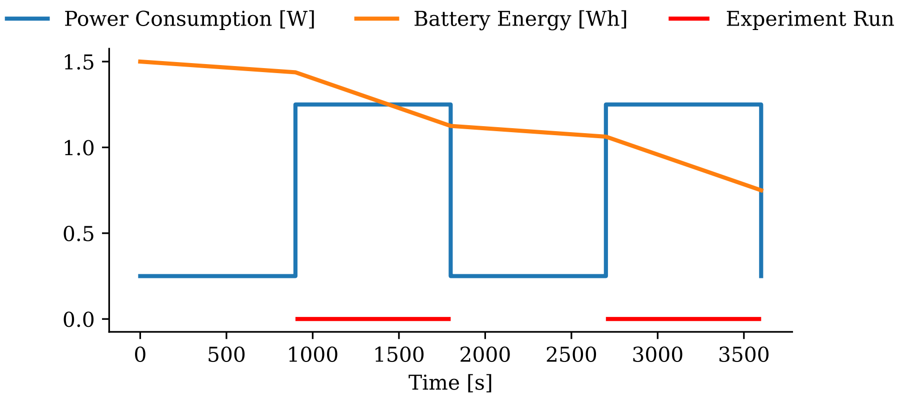

Fig. 5: Power analysis example for a mission architecture containing two experiments.

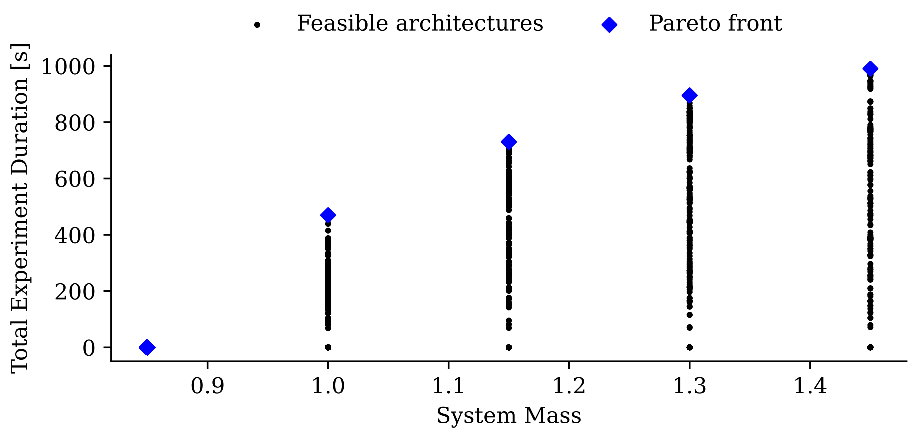

Fig. 6: Architecture optimization results, showing feasible architectures and architectures in the Pareto front.

on each other. The evaluation code is developed based on the elements and architectural choices modeled in the architecture design space, however the design space can only include such elements and choices as to what is feasible to evaluate within a given project context. The data model enables coupling of analysis disciplines and coupling of the architecture generator to the evaluation code, and is usually developed over time by application in various projects [1].

4 Results

The architecture optimization problem defined in Section 3 is solved using NSGA-II [9], a multi-objective evolutionary algorithm that is well suited for solving mixed-discrete problems. The NSGA-II implementation in pymoo [2] is used, accessed through SBArchOpt [6]. SBArchOpt adds several interfaces on top of pymoo to aid solving architecture optimization problems. A design problem defined in ADORE can be directly exposed as an SBArchOpt problem, so no additional integration is needed from the user perspective. NSGA-II is executed with a population size of 100 for 20 generations.

Fig. 6 presents results of the architecture optimization. A Pareto front can clearly be observed, trading-off between system mass and experiment duration. The five (vertical) clusters are formed by the selection of the number of batteries, as this is the only architecture choice influencing mass in the application case. As expected, including more batteries increases mass, however also increases energy availability and therefore allows more and/or longer experiments to be executed.

5 Conclusions

This work has demonstrated System Architecture Optimization (SAO) in the space domain using a exemplary space mission planning problem. The goal was to maximize science time while minimizing system mass. These two conflicting objectives resulted in a Pareto front of optimal architectures after executing the optimization. The architecture design space was modeled using ADORE, a tool for function-based architecture design space modeling and optimization problem execution. Architecture evaluation was implemented using a custom data structure consisting of

Python classes, and four disciplines calculating four relevant performance metrics. Class factories were used to instantiate classes based on architecture element selection. This structure allows independent development of the architecture design space model, data model, and disciplinary analysis code.

In future studies, SAO may be applied to more realistic space mission planning problems, involving more engineering disciplines and higher fidelity analysis. It can be expected that the tool landscape will become more heterogeneous if more disciplines are involved, potentially warranting a collaborative multidisciplinary optimization approach that is able to integrate distributed analysis tools into a single analysis workflow [7]. The architecture design space can be expanded to represent different types of experiments, different types of instruments, and more realistic integration among on-board subsystems. SAO should be integrated in established MBSE processes, interfacing with standard formats such as SysML [12], Arcadia [16] or OPM [10].

References

1. Alder, M., Moerland, E., Jepsen, J., Nagel, B.: Recent advances in establishing a common language for aircraft design with CPACS. In: Aerospace Europe Conference (2020)
2. Blank, J., Deb, K.: Pymoo: Multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020). https://doi.org/10.1109/access.2020.2990567
3. Boggero, L., Chojnacki, A., Bussemaker, J., Bartels, J., Quantius, D., Nagel, B.: The MBSE competence at the German Aerospace Center. INCOSE International Symposium **33**(1), 910–924 (Jul 2023). https://doi.org/10.1002/iis2.13061
4. Bussemaker, J.H., Boggero, L., Ciampa, P.D.: From system architecting to system design and optimization: A link between MBSE and MDAO. In: 32nd Annual INCOSE International Symposium. Detroit, MI, USA (Jun 2022). https://doi.org/10.1002/iis2.12935
5. Bussemaker, J.H., Boggero, L., Nagel, B.: System architecture design space exploration: Integration with computational environments and efficient optimization. In: AIAA AVIATION 2024 FORUM. Las Vegas, NV, USA (Jul 2024)
6. Bussemaker, J.H.: SBArchOpt: Surrogate-based architecture optimization. Journal of Open Source Software **8**(89), 5564 (sep 2023). https://doi.org/10.21105/joss.05564
7. Ciampa, P.D., Nagel, B.: AGILE paradigm: The next generation of collaborative MDO for the development of aeronautical systems. Progress in Aerospace Sciences **119** (Nov 2020). https://doi.org/10.1016/j.paerosci.2020.100643
8. Crawley, E., Cameron, B., Selva, D.: System architecture: strategy and product development for complex systems. Pearson Education, England (2015). https://doi.org/10.1007/978-1-4020-4399-4
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
10. Dori, D.: Model-based systems engineering with OPM and SysML. Springer New York (2016). https://doi.org/10.1007/978-1-4939-3295-5
11. Fortescue, P., Swinerd, G., Stark, J.: Spacecraft systems engineering. John Wiley & Sons (2011)
12. Friendenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language, vol. 38. Elsevier (2015). https://doi.org/10.1016/C2013-0-14457-1
13. Haberfellner, R., de Weck, O., Fricke, E., Vössner, S.: Systems Engineering. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-13431-0
14. MB4SE: MBSE best practices. Tech. Rep. MB4SE-TN-002, ESA (2022)
15. McDermott, T., Folds, D., Hallo, L.: Addressing cognitive bias in systems engineering teams. In: 30th Annual INCOSE International Symposium. Virtual Event (Jul 2020). https://doi.org/10.1002/j.2334-5837.2020.00721.x
16. Voirin, J. (ed.): Model-based system and architecture engineering with the arcadia method. ISTE Press, London (2018)