# Dissimilarity Measures for Clustering Space Mission Architectures

Cody Kinneer
Institute for Software Research,
Carnegie Mellon University
Pittsburgh, PA
ckinneer@cs.cmu.edu

Sebastian J. I. Herzig
Jet Propulsion Laboratory,
California Institute of Technology
Pasadena, CA
Sebastian.J.Herzig@jpl.nasa.gov

## ABSTRACT

The application of model transformations to the process of design space exploration and multi-objective optimization allows for comprehensive exploration of an architectural trade space. For many applications, such as the design of missions involving multiple spacecraft, the resulting set of Pareto-optimal solution models can be too large to be consumed directly, requiring additional analyses in order to gain meaningful insights. In this paper, we investigate the use of automated clustering techniques for grouping similar solution models, and introduce and study a number of both generic and domain-specific methods for measuring the similarity of the solution models. We report results from applying our approach to the exploration of the design space of a spacecraft-based interferometry array in a lunar orbit. For purposes of evaluation and validation, results from the application to the case study are correlated with the results from a study in which solution models were clustered manually by groups of domain experts. The results show tradeoffs in the granularity and extensibility of applying clustering approaches to spacecraft mission architecture models. Also, what humans consider to be relevant in assessing architectural similarity varies and is often biased by their background and expertise. We conclude that providing the subjects with a range of clustering tools has the potential to strongly enhance the ability to explore the complex design space of multi-spacecraft missions, and gain deep insights into the trade space.

## CCS CONCEPTS

• **Applied computing** → **Aerospace**; • **Computing methodologies** → **Cluster analysis**; Modeling methodologies; Heuristic function construction;

## KEYWORDS

model transformations, clustering, space mission architecture, design space exploration, networked constellations, architecture synthesis



*MODELS '18, October 14–19, 2018, Copenhagen, Denmark*

ACM ISBN 978-1-4503-4949-9/18/10…$15.00
https://doi.org/10.1145/3239372.3239390

**ACM Reference Format:**
Cody Kinneer and Sebastian J. I. Herzig. 2018. Dissimilarity Measures for Clustering Space Mission Architectures. In *ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS '18), October 14–19, 2018, Copenhagen, Denmark.* ACM, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3239372.3239390

## 1 INTRODUCTION

The recent trend in aerospace towards low-mass, low-cost spacecraft such as SmallSats and CubeSats [5] has the potential to make new mission concepts involving multiple spacecraft feasible. While mission concepts involving such swarms of spacecraft are generally considered to have great scientific potential, their design is non-trivial. Performance and acceptable risk must be carefully balanced with cost. The limited performance of small spacecraft may also require distributing tasks such as communication with Earth and science data collection among different, specially designed spacecraft. Answering questions such as "How many spacecraft are in the cluster?", "Will the spacecraft in the constellation have homogeneous or heterogeneous configurations?", and "Who communicates with whom?" can lead to a large numbers of candidate solutions. This paper investigates methods for gaining further insight into complex, high-dimensional design spaces.

Previous work investigated the use of multi-objective evolutionary optimization and model-transformation-based architecture model synthesis to search for a set of Pareto-optimal designs [6]. This optimization-based approach narrows the set of candidate solutions to the most preferred. As a potentially sensible method to gain further insight into the solution space, this paper investigates the use of clustering algorithms [7] to group similar architectures together. The premise is that this will positively support a human by allowing him / her to consider a small set of sample architecture models that are sufficiently different from one another.

Clustering techniques group data such that data instances within groups are more similar than data instances in other groups. Thus, clustering algorithms are highly dependent on a dissimilarity measure, a method for computing the distance or similarity between pairs of individuals to be clustered. Computing a similarity score between mission architectures is a non-trivial task. We tried several approaches for comparing architectures, finding tradeoffs in the granularity and extensibility of applying clustering to mission architectures. To understand how our techniques for comparing architectures compare to human intuition, we also asked human engineers to assess the dissimilarity of several pairs of architectures to provide a point of comparison and allow us to learn how human engineers approach the problem. We learned that what the humans

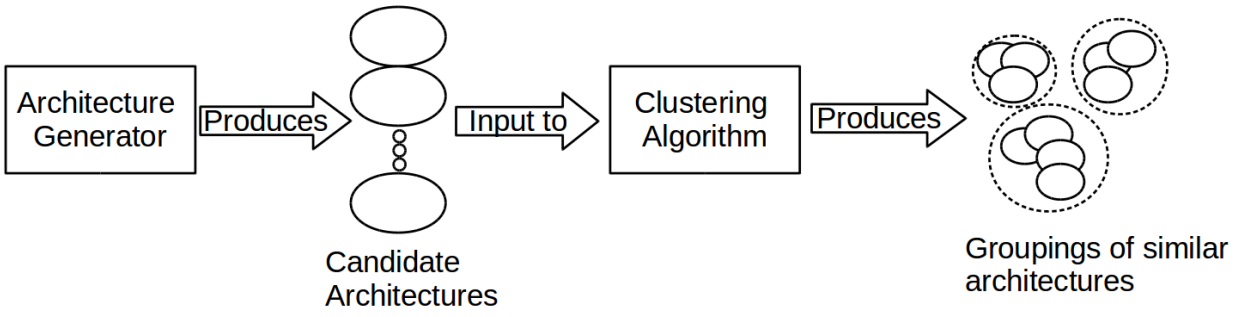


**Figure 1: A high-level overview of the proposed tool for assisting mission design. Prior work [6] focussed on architecture synthesis. Here, we explore extending this tool with a clustering algorithm.**

considered to be relevant in assessing architectural similarity varied, and that providing them with a range of clustering tools has the potential to enhance their ability to explore the complex design space of multiple spacecraft missions.

The rest of the paper is organized as follows, Section 2 provides background, including a description of a case study constellation mission that will serve as a running example. Section 3 describes our approach in applying clustering techniques to mission architectures, including our dissimilarity measures. Section 4 discusses the results of applying the clustering techniques to the case study mission, as well as our comparison of the automated similarity approaches to human intuition. Section 5 provides an overview of related work involving applying clustering techniques to software architecture. Finally, Section 6 summarizes our findings and conclusions.

## 2 BACKGROUND

This section describes the problem domain of space mission design and networked constellation missions. We then describe current efforts to explore the design space of networked constellation missions based on architecture synthesis. To provide a concrete example, we introduce a specific multi-spacecraft mission. Lastly, we provide an overview of machine learning and clustering algorithms that will form the basis for our approach.

### 2.1 Space Mission Design

Space mission design is the process of specifying a space system that pursues some (typically scientific) objective given technical and resource constraints [10]. Multi-objective, optimization-based design allows mission designers to make tradeoffs between the scientific return, cost, and risk. Failing to consider alternatives can result in converging on a suboptimal design.

Networked constellation missions are space missions that involve more than one spacecraft. Often deemed infeasible due to the high cost of designing and launching multiple spacecraft, recent advances in aerospace towards small, light, and cheap spacecraft such as CubeSats [5] that are built primarily using commercial off-the shelf parts have been perceived as enablers to implement cost-effective multi-asset missions. While networked constellations can

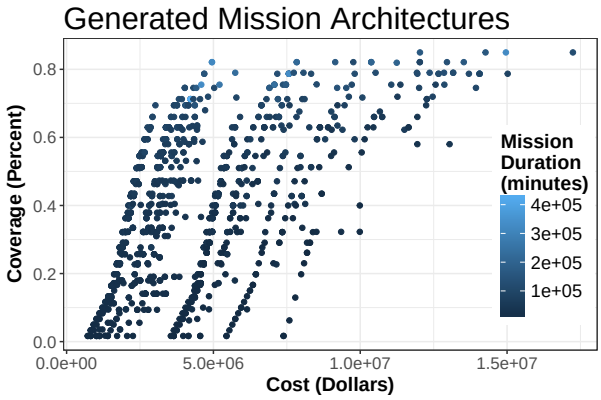


**Figure 2: Pareto frontier of optimal architectures with respect to cost, coverage (proxy for value of performed science), and mission duration (from prior work [6]). Each dot represents one unique architecture in the space of candidate solutions.**

enable novel mission concepts, they also bring new challenges: for instance, increasing the number of spacecraft can increase physical redundancy, but the limited performance of small spacecraft may require distribution of functions (such as communicating with Earth, collecting scientific data, etc.). This greatly increases the space of possible candidate mission architectures. Figure 2 shows the space of identified Pareto-optimal architectures for a multi-spacecraft space-based radio interferometer (from previous work [6]).

### 2.2 Model-Transformation-Based Mission Architecture Synthesis

To address the problem of exploring a very large architectural design space, prior work introduced a method for synthesizing mission architectures for networked constellation missions using a combination of model-transformations and evolutionary algorithms [6].

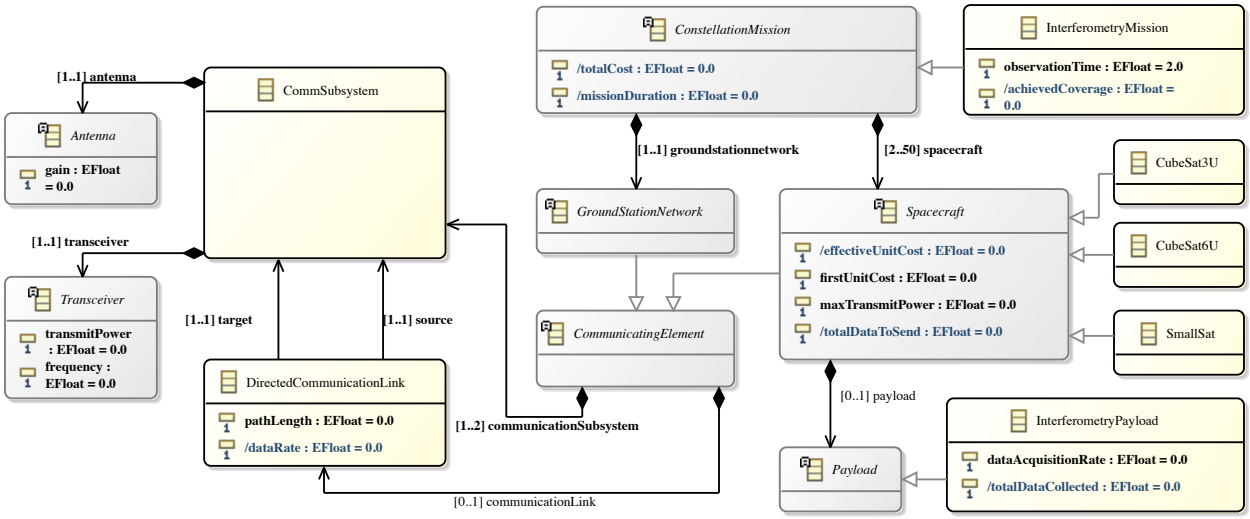


**Figure 3: Meta-model for the case study problem (from prior work [6]).**

The method is intended to support designers by (a) capturing design knowledge formally, (b) support the creative process, and (c) provide insights into major trade-offs as early as possible in the design process. A meta-model describes the space of legal designs. In-place endogenous model-transformation rules [13] are used for the purpose of generating solutions in this design space. Sequences of these rules represent individuals in a population that is evolved using genetic algorithms. The search is guided by user specified design objectives, such as minimizing the cost of the overall mission, maximizing scientific return, and minimizing the time needed to complete the mission. This approach is enabled by a stack of tools in the Eclipse Modeling Framework (EMF) [16] ecosystem, including Ecore for specifying a meta-model (or domain model), Henshin [2] for specifying exploration rules as model transformation rules, and MOMoT [4] which provides a framework for integrating model transformations with search-based optimization algorithms. Cost, scientific return, and other objectives, performance measures and desired traits of candidate solutions are computed using application- or domain-specific analysis models. The output is a set of mission architectures that are Pareto-optimal with respect to the specified design objectives. The set of generated candidate mission architecture can then be analyzed further by mission designers, e.g., for the purpose of gaining insights into major trade-offs and possible technical solutions.

## 2.3 Case Study

To develop and evaluate our approach, we use the motivating case study from previous work [6]. This case study encompasses the design of a multi-spacecraft-based interferometer (an array of antennas simulating a single, larger antenna) that is capable of observing distant radio galaxies at frequencies below 30Mhz.A space-based application is required, since signals below 30Mhz are blocked by the Earth's ionosphere. To gain sufficient distance from Earth, all spacecraft are placed in an orbit around the Moon.

To fulfill the mission objectives, the cluster must collect scientific data and send this data back to Earth. Prior work [6] investigated an approach to automatically generating Pareto-optimal mission architectures for this case study that specify the number of assets to deploy, their capabilities, and configuration to maximize scientific return while minimizing cost and the amount of time it takes to complete the mission. Several simplifying assumptions are made to reduce the complexity of the example. An excerpt of the meta-model used for the case study is shown in Figure 3. Not shown are concrete sub-classes of *Transceiver* and *Antenna*. Only three, discrete types of spacecraft are considered for implementing the

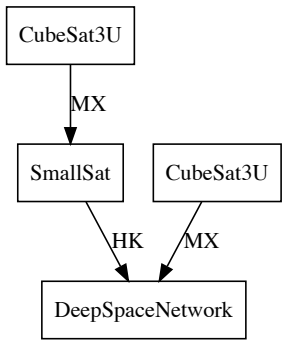


**Figure 4: A mission architecture consisting of a ground station and three spacecraft. The top CubeSat3U communicates to a SmallSat using a medium gain antenna (*M*) over X-band (*X*). The SmallSat communicates using a high gain antenna (*H*) over K-band (*K*). The third spacecraft is a similarly configured CubeSat3U that communicates directly to the ground. All spacecraft carry the scientific payload (not shown).**

mission: 3U and 6U CubeSats (small, light 4kg and 12 kg satellites respectively), and a heavier and more costly, but more capable and performant 100kg Small Satellite. Each asset can have up to two separate communication subsystems, where each communication subsystem consists of an antenna and a transceiver. Assets can be equipped with a high (25 dBi), medium (10 dBi), or low gain (1 dBi) antenna, and an Ultra High Frequency, X-band, or Ka-band transceiver. The choice of communication components affects the speed of data transmission, which in turn affects how long it takes to downlink the collected data to ground. The choice of antennae and transceivers also affects the mission cost and power usage. Each spacecraft can also optionally carry an interferometry payload for observing the target, or not carry a payload and serve as a communication relay only. The science value obtained by the mission is influenced by how many assets with the interferometry payload observe the target, and for how long. Assets can transmit science data collected from the payload directly to the ground station, or to other spacecraft to be relayed to ground. To simplify the problem, the launch system, orbit, and much of the details concerning spacecraft operations are not included in the scope of the specification of the design problem.

Candidate mission architectures are models conforming to the meta-model illustrated in Figure 3. An example candidate mission architecture is shown in Figure 4. Given a candidate mission architecture, the cost, mission duration, and scientific return (measured as coverage) can be calculated based on performance analysis models developed in prior work [6].

## 3 APPROACH

Architecture synthesis has the potential to aid mission designers in exploring large design spaces for architectural alternatives, yet it remains difficult for humans to make sense of the large number of optimal architectures that are generated. To address this problem, we apply clustering algorithms to mission architectures to allow humans to more easily explore the design space.

Mission designers do not gain much additional insight into the design space by inspecting architectures that are very similar to those that they have already seen. Since many automatically generated candidate architectures are similar to one another, this means that human experts must waste time sifting through many variations of similar architectural concepts to make use of the architectural synthesis product.

Clustering offers a way to automatically perform this time expensive process for human mission designers. By placing similar architectures into groups, mission designers can quickly get an overview of the design space by examining representative architectures from each group.


**Figure 5: Pareto space of optimal architectures clustered by feature selection.**

### 3.1 Clustering Algorithm

Clustering is an unsupervised machine learning technique [1]. Given a dataset of unlabeled data instances, a clustering algorithm attempts to find patterns in the data by grouping similar instances together. The goal of clustering is to produce groups, or clusters, where data instances within the cluster are more similar to one another, and less similar to instances outside.

We used the Partitioning Around Medoids [7] (PAM) algorithm. PAM is a partitional clustering algorithm related to the k-means algorithm. PAM distinguishes itself from k-means by taking $k$ data instances to be the centers of clusters, or medoids, rather than using centroids (points that need not be data instances). Data instances are clustered based on the nearest medoid (according to least distance). The PAM algorithm minimizes the sum of pairwise dissimilarity between each data instance and its closest medoid. Using data instances as the centers of clusters makes PAM more robust than k-means. PAM also has the advantage of accepting a dissimilarity matrix as input, allowing arbitrary similarity measures. Selecting data instances as medoids rather than centroids makes the results of clustering easier to evaluate, as the evaluator can examine the medoid architectures. The number of clusters $k$ is determined automatically by the optimum average silhouette width, which is a higher-is-better indicator of how well a point fits into a cluster.

### 3.2 Similarity Measures

To enable clustering, a suitable similarity measure must exist to measure the dissimilarity between objects. The results of clustering are heavily influenced by the similarity measure [18], and determining architectural similarity is a non-trivial problem. Some aspects of the architecture are primarily structural, such as the number of spacecraft of a given class, or the network topology of the mission. Other properties of the architecture describe what the mission does, such as the resulting science value of a mission. We also observed that architectural similarity can be considered in ways that are generalizable to many types of networked constellation missions, for example, all networked constellations have a network topology. Alternatively, considering mission specific concepts like whether an asset is equipped with an interferometry payload can allow for a more fine grained analysis of dissimilarity, at the cost of not being as easily applied to other missions.

With these tradeoffs in mind, we chose three similarity measurement approaches to apply to the case study mission, including feature selection, the EMF Compare tool from the EMF ecosystem, and a graph-edit distance approach. The remainder of this section will explain each similarity measure in detail.

**Table 1: Features used for clustering.**

| Name | Unit | Description |
|---|---|---|
| CubeSat3U | count | CubeSat3Us in the mission. |
| CubeSat6U | count | CubeSat6Us in the mission. |
| SmallSat | count | SmallSats in the mission. |
| Observation Time | hours | Hours spent measuring the target. |
| Mission Duration | hours | Hours before results are returned. |
| Cost | dollars | The total cost of the mission. |
| Coverage | percent | A proxy for scientific value. |

*3.2.1 Feature Selection.* One common technique for clustering complex entities is called feature selection [18], where each entity to be clustered is converted into a numeric vector of features. To compute a dissimilarity matrix from a set of feature vectors, the distance between each pair of features is computed. The sum of these distances then becomes the distance for that pair of individuals. The features can be scaled and normalized so each features contributes equally to the distance value. Optionally, features can be weighted such that some features affect the distance more or less than others.

The feature selection approach has the advantage of being flexible, as it is easy to add and remove features to the process. The extensibility of the approach can also be affected by the choice of features. Some features might be applicable to all networked constellation missions, such as the number of assets present. Other might be specific to certain missions, such as the number of assets equipped with an interferometry payload.

Figure 5 shows an example clustering of the generated mission architectures. Each color denotes a separate group. In this case, the clustering algorithm separated the space into three regions, the green architectures are cheap and complete the mission quickly, but sacrifice scientific return. The blue architectures are also cheap, but tradeoff mission duration in exchange for increased science value. The red architectures are more expensive, in exchange for increased science and fast mission completion.

We identified a number of features for candidate architectures of the case study mission. We selected features based on the characteristics that a human designer would use to evaluate the architectures. Table 1 shows the features. Some features, like the number of assets in a mission, are helpful for evaluating an architecture, but do not directly correspond to the mission's objectives. Other features, like cost, are directly relevant to the missions objectives, but do not tell us much about the mission architecture. It is not immediately apparent whether clustering based one type of feature or a combination thereof would be most helpful to a human evaluator, so we tried clustering using all features, only the structural features (the number of assets of each class), and only the design objectives (cost, coverage, and mission duration).

*3.2.2 EMF Compare.* We also applied EMF Compare to measure the difference between architectures. EMF Compare is a tool for the Eclipse Modeling Framework for performing model matching. This tool can compute the number of edits necessary to transform one EMF model into another, which can serve as a similarity measure. This approach is highly generalizable, since it can be applied to any EMF models as long as they correspond to the same meta-model.

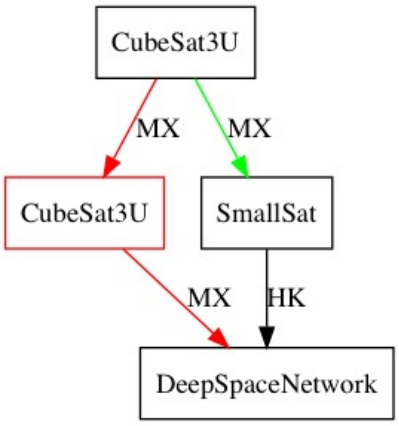


**Figure 6: Computing dissimilarity by graph-edit distance.**

The generality of the technique however comes at the price of low granularity, the tool only counts the number of changes in the UML model, without taking into account domain specific information (such as some elements in the model being more important, or some elements needing custom distance functions).

*3.2.3 Graph-edit Distance.* Another technique for measuring architecture similarity is to represent mission architectures as a labeled graph, and compute the cost of transforming one graph into another. Figure 6 shows how missions and edits are represented in graphs. The ground station and spacecrafts are represented as nodes. Each node is labeled with the asset class (in the case of spacecraft), or with the name of the ground station. Data flow between assets is represented as directed edges, with the direction corresponding to the direction of data flow. The edges are labeled with the antenna and transceiver type used by the assets to communicate.

Computing the edit distance between graphs is a well known NP-complete problem, but there are heuristic algorithms that offer an approximation in a reasonable amount of time. We implemented the heuristic-A*-beamsearch algorithm [14] for computing the difference. This version of the algorithm trades off the guarantee of optimality in exchange for a performance speedup by only considering some $S$ number of promising paths at a time, instead of performing an exhaustive search. In practice, the solutions obtained are often "good enough". Even with a heuristic algorithm, calculating the pairwise graph-edit distance can be expensive. For mission architectures that can be represented as trees, computing the tree-edit distance instead can reduce the amount of time needed to perform the comparisons [15].

The graph-edit distance method is similar to the EMF Compare approach since both compute distance based on the number of edits it takes to transform one model to another. The difference is that representing architectures as a graph allows increased granularity, since the properties of the architecture to consider are specified outright (as opposed to considering everything in the EMF model). Additionally, the graph-edit distance approach allows the semantics of distance to be customized where appropriate, as opposed to EMF Compare which provides its own weightings.

# 4 RESULTS

To evaluate our approach for making the results of architecture synthesis more understandable to human mission designers, we performed clustering using each similarity measure from Section 3 on over 1000 candidate architectures for the space-based interferometry mission from prior work [6]. Section 4.1 discusses the resulting clusterings, and provides box and whisker plots showing summary statistics. We showed the results of clustering to engineers involved in mission design at JPL and received positive feedback.

To evaluate how closely our dissimilarity measures matched with human intuition about comparing architectural similarity, we also asked two groups of engineers at JPL to assess the similarity between pairs of mission architectures, and then compared the output of our similarity measures to the humans' judgement.

## 4.1 Applying Clustering

Box and whisker plots showing summary statistics for each resulting clustering using feature selection are shown in Figures 7, 8, and 9 respectively. In all figures, each color corresponds to a cluster. Subplots show the range of values occurring in each cluster for the number of assets of each class, the time the assets spend observing the target, how long the mission takes to complete, the mission cost, and the coverage proxy for scientific value. A final size subplot show the number of individuals in each cluster.

Figure 7 shows the results of clustering based on using all of the features in Table 1. The clustering algorithm divided the candidate architectures into three groups. Group one has fewer CubSat3Us than the other two groups, but more CubeSat6U and SmallSats. Group one is also much more expensive, gets the highest coverage, and a mission duration that is in between the other two groups. Group two has a number of CubeSat3Us in between groups one and three, fewer CubeSat6Us and SmallSats than group one and about equal to group three. Group two has the lowest mission duration and cost, but also the lowest coverage. The third group has the most CubeSat3Us, and about an equal number of CubeSat6Us and SmallSats as group two. Group three has the highest mission duration, but its cost is only slightly higher than group two's, and its coverage is nearly as good as group three's.

Figure 8 shows the results of clustering based only on the number of assets in each architecture. In this case, the algorithm chose to make four clusters. Group one contains missions that have a mix of all three spacecraft type, group two contains missions with CubeSat3Us and SmallSats, group three consists of missions with CubeSat3Us and CubeSat6Us when there are more CubeSat3Us than CubeSat6Us, and group four also contains architectures with CubeSat3Us and CubeSat6Us, but are more equal in number.

Figure 9 shows the results of clustering based only on the features that capture mission objectives. PAM produced two clusters for these features. The first group has long mission duration, high cost, and high coverage, while the second group has shorter mission duration, lower cost, but also lower coverage.

While clustering based on feature selection obtained good results, the feature selection process necessitates some information loss. To keep more information, we also tried clustering based on the results of the EMF Compare tool, which reports the number of differences between two EMF models. We computed a dissimilarity matrix by using EMF Compare [9] to measure the number of differences between every pair of candidate architectures. We then passed the dissimilarity matrix to the PAM algorithm. Aggregate statistics for the resulting clustering are shown in Figure 10.

**Table 2: Expert Topics**

| Keyword | Group 1 | Group 2 |
|---|---|---|
| relay | 2 | 5 |
| bands | 2 | 3 |
| layers / levels | 2 | 6 |
| SmallSats | 2 | 2 |
| threads | 0 | 2 |

Figure 10 shows the ten clusters produced by using EMF Compare. One noticeable pattern is that groups 4–10 all have a distinct observation time. As the observation time increases, they also have increasing mission duration, cost, and coverage. This is an interesting result, since the algorithm was not given these features as input, showing that clustering based on even a coarse similarity measure can capture interesting patterns.

Figure 11 shows aggregate stats resulting from the graph-edit distance clustering method. Like the EMF Compare approach, an interesting result is that there are clear differences in the objective values between clusters, despite the algorithm not having this information during clustering. This is likely because the objective values are closely connected to the structural features of the architectures.

As a preliminary effort to validate the usefulness of our clustering techniques to mission design, we showed the results of clustering to NASA engineers involved in early mission design, explained the case-study scenario, and showed the result of the clustering algorithms. They indicated that the results were interesting and thought that the results of clustering would be useful in evaluating the architectures. This preliminary evaluation builds confidence that our approach is useful to mission designers.

## 4.2 Expert Opinion

To evaluate how closely our dissimilarity measures matched with human intuition about comparing architectural similarity, we performed a study asking two groups of 2–3 engineers at JPL to assess the similarity between pairs of mission architectures. Each group was given 31 pairs of randomly selected mission architectures from the set of generated architectures. The architectures were represented as graphs in the format of Figure 4, with the addition of the mission cost, coverage, and duration included as a caption. Figure 12 shows an example pair of mission architectures shown to the human engineers. For each pair, the engineers were instructed to, as a group, rate the similarity between architectures on a six point Likert scale (completely different, mostly dissimilar, somewhat dissimilar, somewhat similar, mostly similar, practically identical). We were intentionally vague in our instructions about what criteria they should use when assessing similarity to avoid biasing them, asking that they rate the similarity based on their own intuition and experience in designing missions. We had the engineers work in small groups to enable us to listen to their discussions and learn more about how they approach the problem. To recruit the engineers, we sent emails to engineers who work on mission design and

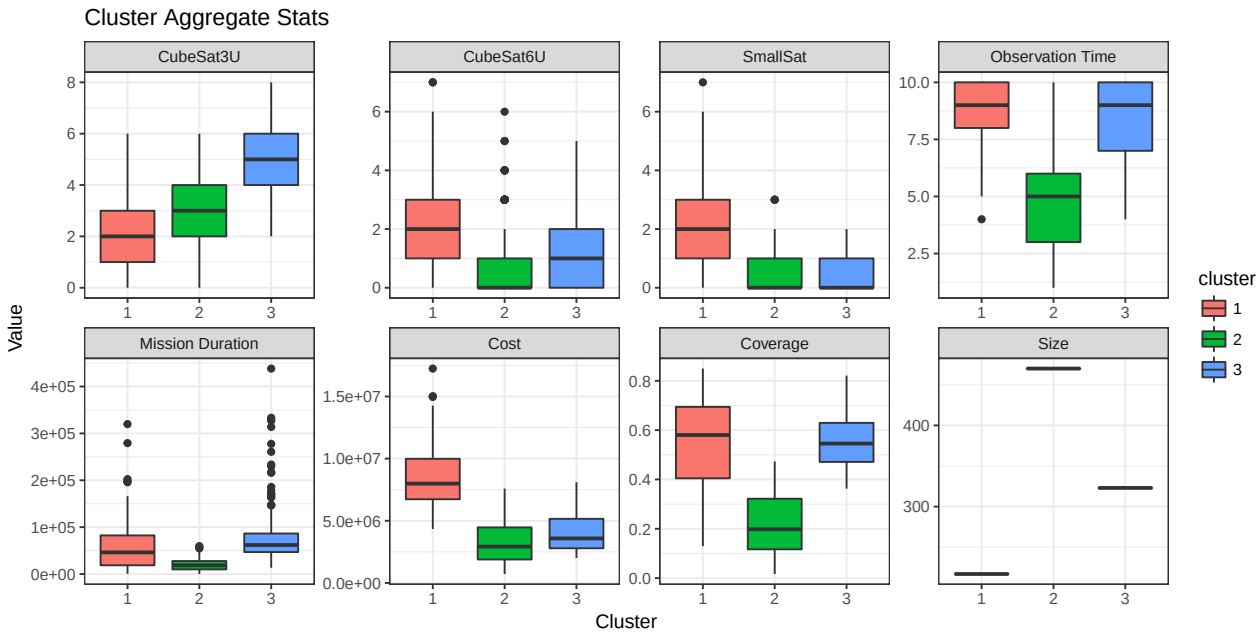


**Figure 7: Clustering by structural and design objectives.**

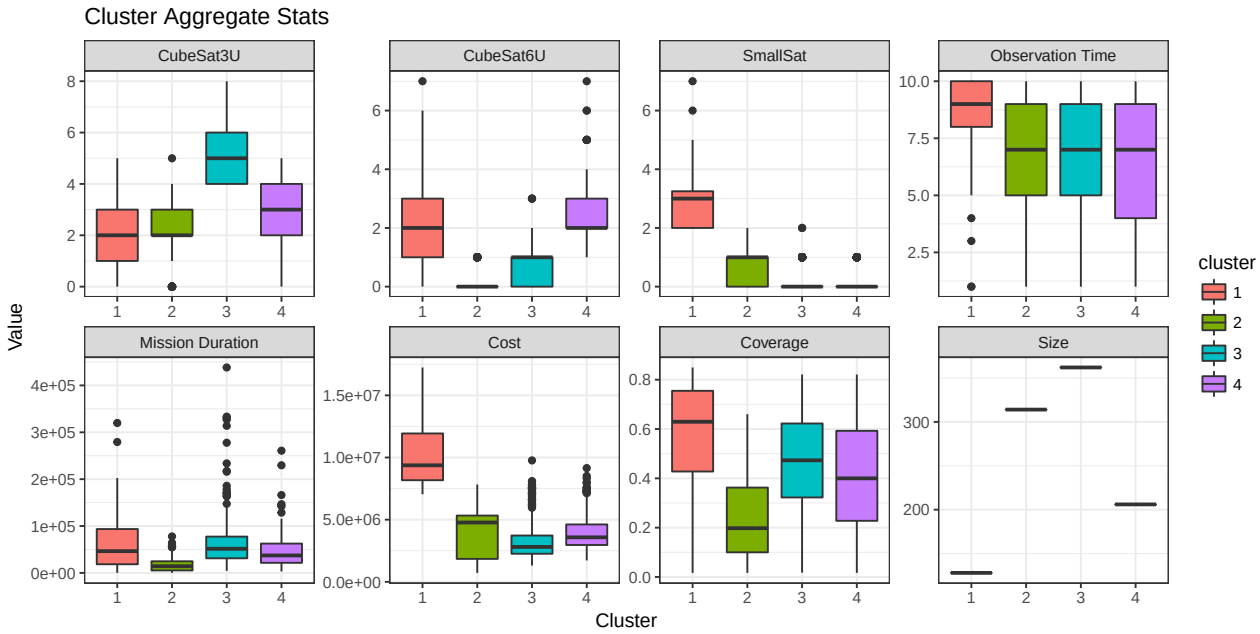


**Figure 8: Clustering by structural features.**

might be potential users of a future clustering tool. The engineers were grouped based on schedule availability. For each group, we explained the case study scenario, how to interpret the architecture diagrams, and instructed them to discuss aloud as much as possible to enable us to follow their thoughts. The first two pairs of architectures were considered to be practice, to allow the engineers to get comfortable with the process and calibrate their sense of similarity, as such they are not included in the analysis.

Table 3 shows Pearson correlations and P values between pairs of dissimilarity metrics and the scores provided by each group of engineers. Correlations are statistically significant at the $P < 0.05$ level, including the correlation between the two human groups,

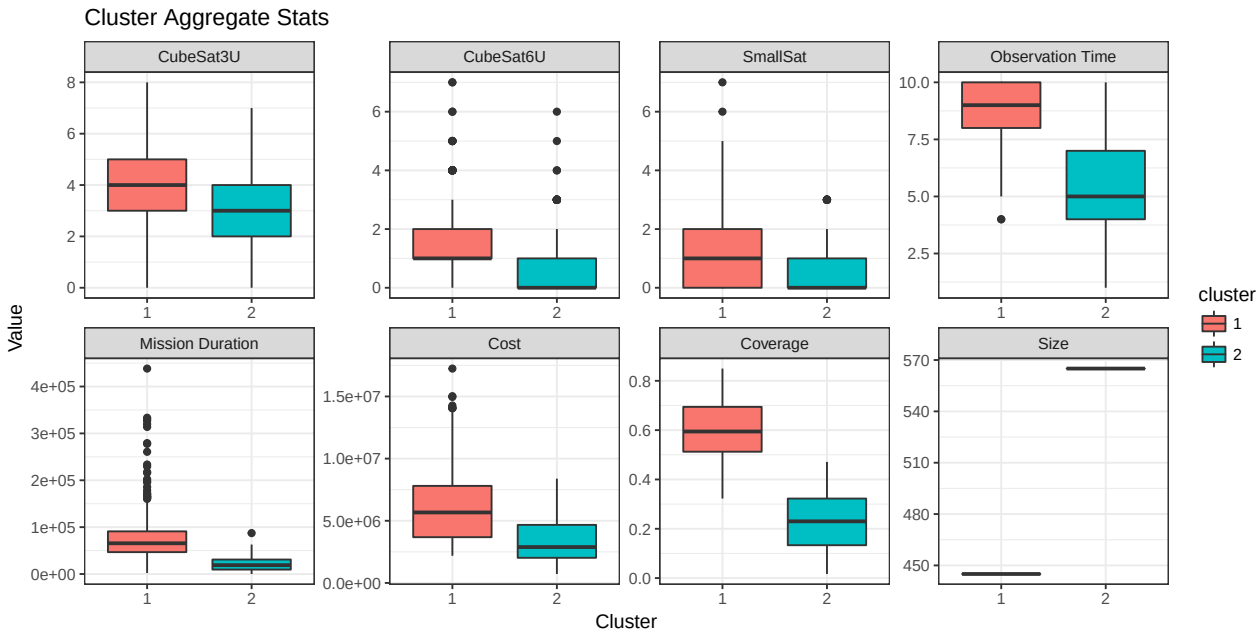


**Figure 9: Clustering by design objectives.**

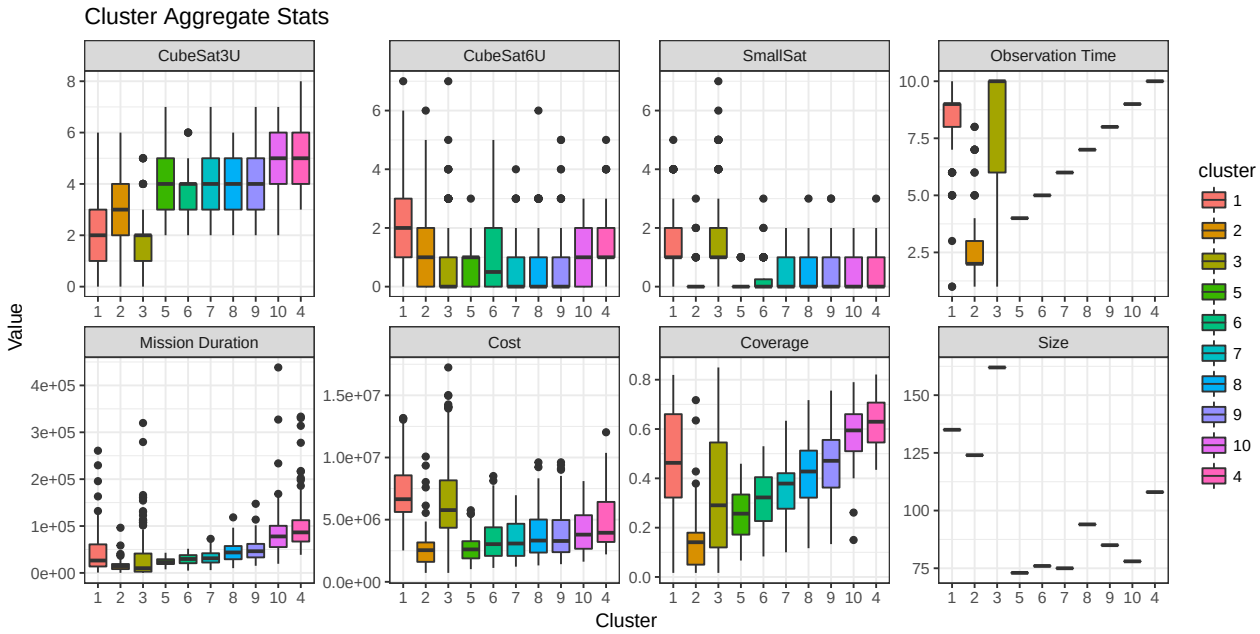


**Figure 10: Clustering by EMFCompare difference count.**

and between group two and using the asset count features. The pairs between each human group and the all features dissimilarity measure were on the threshold of significance with $P$ values of 0.06 and 0.05. One interesting takeaway is the 0.5 correlation between the two groups of engineers, indicating some level of agreement, but also revealing that the two groups also disagreed. The first group did not have a strong correlation to any of the dissimilarity measure that we tried. There were weak correlations to most of the measures, and no correlation to the results of using EMF Compare. The second group showed similar results, but with a stronger correlation with the assets similarity measure of 0.56. This means that group two

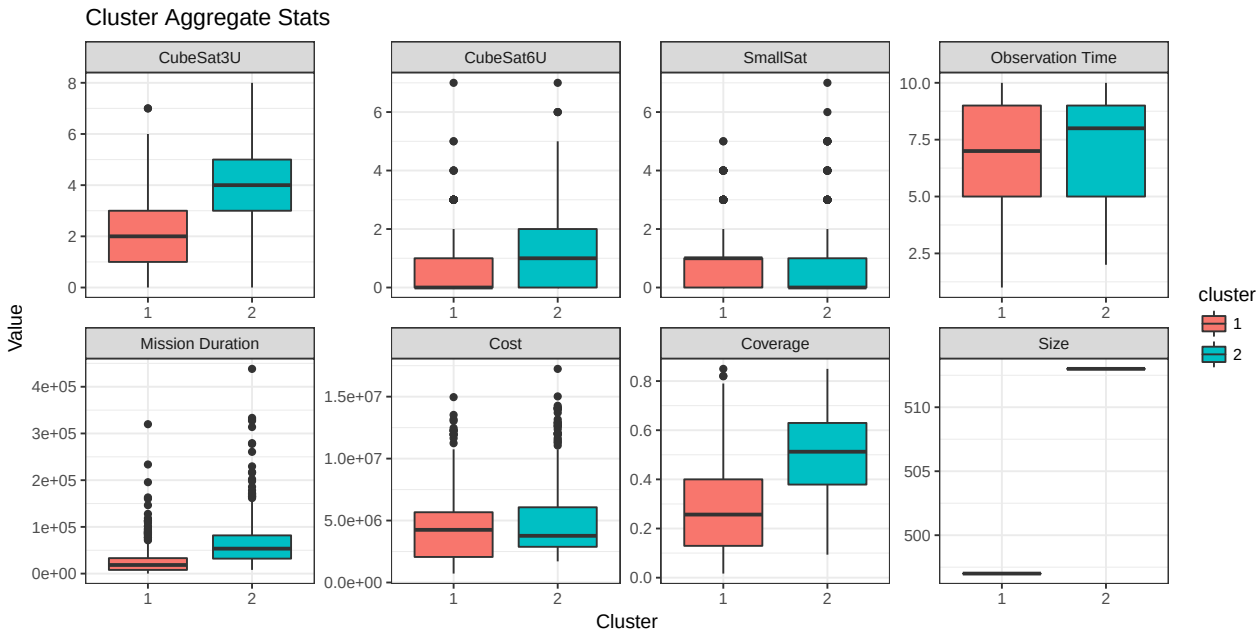


**Figure 11: Clustering by graph-edit distance.**

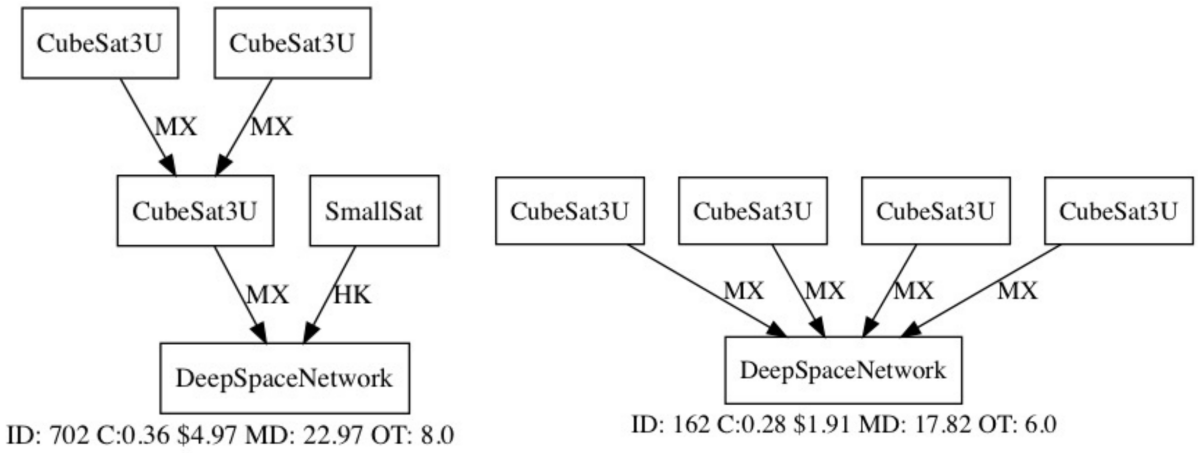


**Figure 12: An example pair of mission architectures shown to human experts. C refers to the missions coverage, the dollar figure is the mission's cost in millions of dollars, MD refers to mission duration in days, and OT is observation time in hours.**

agreed with the results of the asset count similarity measure slightly more than they agreed with the responses of the other group.

While watching the groups deliberate, we noticed that they mostly ignored the mission's objective values (cost, coverage, and time) except occasionally when they decided that the missions were already mostly similar, so the low correlation between the groups and the objectives similarity measure is not surprising. Table 2 shows some commonly occurring topics in the discussion. We noticed that the groups would first take notice of the number of distinct spacecraft types on each architecture. In particular, the presence or absence of a SmallSat indicated different missions classes. The groups also frequently discussed the layout of communication links in the mission, and they took notice of the number of incoming and outgoing connections between each asset, considering missions with more levels of relays more complex. There was a big difference between an asset that had one outgoing link, and an asset that had one outgoing link and one incoming link, even if the assets were of the same class. The number of communication links seemed to matter less after the first type, for example, an asset with one incoming and one outgoing link is more similar to an asset with two incoming and one outgoing links than with an asset with no outgoing links. The participants also talked about how their assessments would

**Table 3: Pearson correlations (below the diagonal) and P values (above the diagonal) between dissimilarity metrics.**

| | Group 1 | Group 2 | Features (All) | Features (Assets) | Features (Objectives) | Graph-Edit Distance | EMF Compare |
|---|---|---|---|---|---|---|---|
| Group 1 | 1 | 0.01 | 0.06 | 0.19 | 0.12 | 0.16 | 0.88 |
| Group 2 | 0.501 | 1 | 0.05 | 0.00 | 0.26 | 0.28 | 0.54 |
| Features (All) | 0.364 | 0.386 | 1 | 0.02 | 0.00 | 0.01 | 0.00 |
| Features (Assets) | 0.263 | 0.560 | 0.436 | 1 | 0.08 | 0.14 | 0.46 |
| Features (Objectives) | 0.304 | 0.223 | 0.869 | 0.341 | 1 | 0.03 | 0.03 |
| Graph-Edit Distance | 0.276 | 0.217 | 0.464 | 0.289 | 0.429 | 1 | 0.00 |
| EMF Compare | 0.029 | 0.123 | 0.536 | 0.147 | 0.424 | 0.789 | 1 |

change based on what task they needed to perform, suggesting that a scientist might only care about looking at the objective values, while when working on the mission communications the number of incoming and outgoing links is the most important.

We expected after the emphasis on the communication links that the graph-edit distance approach would have a stronger correlation to the humans' intuition, but this turned out not to be true. We suspect that this is because our graph-edit distance approach considers all edits to be of equal weight, while the humans placed much greater emphasis on certain kinds of changes. For example, an asset changing from a sender of data to a relay for other assets is a large change, while an asset that is already a relay receiving data from another asset is a minor change.

## 5 RELATED WORK

In the systems engineering and spacecraft system engineering communities, such work has not yet been performed. There are several examples in the literature of applying clustering techniques to software architectures. Architecture reconstruction is one problem that clustering has been applied to. The goal of architecture reconstruction is to solve the common issue that the architecture for a software system is not available, either because it has not been documented or the software has evolved to the point that the architectural documentation is no longer correct. Architecture reconstruction seeks to discover the architecture given the software's source code. Mancoridis et al. [11] developed a technique to group software modules together based on on a module dependency graph. They cluster modules hierarchically to obtain a high-level subsystem structure. Maqbool and Babri provide a survey of hierarchical clustering approaches for software architecture recovery [12]. These works are different from our approach because they seek to reconstruct an architecture where none exists, while we are interested in understanding existing architectures. Additionally, the hierarchical reconstruction work typically clusters components within a single architecture, while we cluster at the level of entire architectures.

Software architectures can also be reconstructed or verified using clustering on artifacts other than source code. Kim and Chang [8] identify software components by clustering use cases based on computing a matrix of function dependency between use cases, derived from UML models. Yu et al. [19] used hierarchical clustering to verify software designs according to a matrix computed based on the number of interactions between components.

Other work has clustered software artifacts based on software metrics in order to lower the cost of testing [20], grouped virtual machine instances by their resource utilization to reduce management burden [3], and investigated distance metrics for software system decompositions [17].

## 6 CONCLUSION

Model-transformation-based synthesis of spacecraft mission architectures is a promising enabler for a more comprehensive exploration of architectural variants and trades. However, due to the typically large number of Pareto-optimal solutions (stemming from the combinatorial complexity), analysis of the solution space is non-trivial. In the paper, we introduce an approach which uses Partitioning Around Medoids (PAM) clustering to gain more insight into shared qualities among the candidate solutions. Several means for measuring the similarity of architectural models were studied, including domain- and application-agnostic measures such as graph-edit distance, and problem-specific measures such as the number of spacecraft in each solution. We found that choosing sensible similarity measurements for mission architectures is a nontrivial problem, with trade-offs in technique generality, granularity, and choosing between semantic and syntactic properties. However, we also discovered that even coarse-grained similarity measures can result in clusters that provide potentially valuable insights. Results from the clustering were shown to several domain experts who were enthusiastic about the potential for the technique to provide insights during design. We also compared expert judgment to our automated similarity measures, and found that what humans consider to be relevant in assessing architectural similarity varied strongly, showing the usefulness of providing them with a range of clustering tools to enhance their ability to explore the complex design space of multiple spacecraft missions. Our preliminary evaluation indicated that the technique could be helpful for designing networked constellation missions. Future work should include evaluating more complex measures that look for specific architectural patterns. The evaluation of the technique should also be expanded upon by involving a larger group of domain experts.

## ACKNOWLEDGEMENTS

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration, and was sponsored by the JPL Summer Internship Program (JPL SIP). The cost information contained in this document is of a budgetary and planning nature and is intended for informational purposes only. It does not constitute a commitment on the part of JPL and/or Caltech.

## REFERENCES

[1] Ethem Alpaydin. 2014. *Introduction to machine learning*. MIT press.
[2] Thorsten Arendt, Enrico Biermann, Stefan Jurack, Christian Krause, and Gabriele Taentzer. 2010. *Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations*. Springer Berlin Heidelberg, Berlin, Heidelberg, 121–135.
[3] C. Canali and R. Lancellotti. 2012. Automated clustering of VMs for scalable cloud monitoring and management. In *20th International Conference on Software, Telecommunications and Computer Networks*. 1–5.
[4] Martin Fleck, Javier Troya, and Manuel Wimmer. 2015. Marrying search-based optimization and model transformation technology. *Proc. of NasBASE* (2015).
[5] Hank Heidt, Jordi Puig-Suari, Augustus Moore, Shinichi Nakasuka, and Robert Twiggs. 2000. CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation. (2000).
[6] Sebastian J. I. Herzig, Sanda Mandutianu, Hongman Kim, Sonia Hernandez, and Travis Imken. 2017. Model-Transformation-Based Computational Design Synthesis for Mission Architecture Optimization. In *In Proc. of the International IEEE Aerospace Conference*. IEEE.
[7] Leonard Kaufman and Peter J Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons.
[8] Soo Dong Kim and Soo Ho Chang. 2004. A systematic method to identify software components. In *11th Asia-Pacific Software Engineering Conference*. 538–545.
[9] Dimitrios S Kolovos, Davide Di Ruscio, Alfonso Pierantonio, and Richard F Paige. 2009. Different models for model matching: An analysis of approaches to support model differencing. In *Comparison and Versioning of Software Models, 2009. CVSM'09. ICSE Workshop on*. IEEE, 1–6.
[10] Wiley J Larson and James Richard Wertz. 1992. *Space mission analysis and design*. Technical Report. Microcosm, Inc., Torrance, CA (US).
[11] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner. 1998. Using automatic clustering to produce high-level system organizations of source code. In *In Proc. of 6th International Workshop on Program Comprehension*. 45–52.
[12] O. Maqbool and H. Babri. 2007. Hierarchical Clustering for Software Architecture Recovery. *IEEE Transactions on Software Engineering* 33, 11 (Nov 2007), 759–780.
[13] Tom Mens and Pieter Van Gorp. 2006. A Taxonomy of Model Transformation. *Electron. Notes Theor. Comput. Sci.* 152 (March 2006), 125–142.
[14] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. 2006. Fast suboptimal algorithms for the computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*. Springer, 163–172.
[15] Mateusz Pawlik and Nikolaus Augsten. 2011. RTED: a robust algorithm for the tree edit distance. *In Proc. of the VLDB Endowment* 5, 4 (2011), 334–345.
[16] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. 2008. *EMF: Eclipse Modeling Framework*. Pearson Education.
[17] V. Tzerpos and R. C. Holt. 1999. MoJo: a distance metric for software clusterings. In *Sixth Working Conference on Reverse Engineering*. 187–193.
[18] Rui Xu and Donald Wunsch. 2005. Survey of clustering algorithms. *IEEE Transactions on neural networks* 16, 3 (2005), 645–678.
[19] Liguo Yu and Srini Ramaswamy. 2007. Verifying Design Modularity, Hierarchy, and Interaction Locality Using Data Clustering Techniques. In *In Proc. of the 45th Annual Southeast Regional Conference (ACM-SE 45)*. ACM, New York, NY, USA, 419–424.
[20] S. Zhong, T. M. Khoshgoftaar, and N. Seliya. 2004. Analyzing software measurement data with clustering techniques. *IEEE Intelligent Systems* 19, 2 (Mar 2004), 20–27.