Daphne: A Virtual Assistant for Designing Earth Observation Distributed Spacecraft Missions

Antoni Viros i Martin, *Student Member, IEEE*, and Daniel Selva, *Member, IEEE*

***Abstract*—This article describes Daphne, a virtual assistant for designing Earth observation distributed spacecraft missions. It is, to the best of our knowledge, the first virtual assistant for such application. The article provides a thorough description of Daphne, including its question answering system and the main features we have implemented to help system engineers design distributed spacecraft missions. In addition, the article describes a study performed at NASA's Jet Propulsion Laboratory (JPL) to assess the usefulness of Daphne in this use case. The study was conducted with $N = 9$ subjects from JPL, who were asked to work on a mission design task with two versions of Daphne, one that was fully featured implementing the cognitive assistance functions, and one that only had the features one would find in a traditional design space exploration tool. After the task, they filled out a standard user experience survey, completed a test to assess how much they learned about the task, and were asked a number of questions in a semi-structured exit interview. Results of the study suggest that Daphne can help improve performance during system design tasks compared to traditional tools, while keeping the system usable. However, the study also raises some concerns with respect to a potential reduction in human learning due to the use of the cognitive assistant. The article ends with a list of suggestions for future development of virtual assistants for space mission design.**

***Index Terms*—Distributed spacecraft missions, earth observation, machine learning, mixed initiative, virtual assistant.**

I. Introduction

MOTIVATED by the challenges of system architecture in general and architecting distributed satellite missions (DSM) in particular, and inspired by the success of commercial virtual assistants (VA), such as Siri, Google Assistant, Alexa, or Mycroft, we have developed Daphne, the first VA – to the best of our knowledge – to support the high-level design of DSM. This article describes how Daphne can be used to design DSM and includes a quantitative validation study performed at NASA JPL where the test subjects had expertise in mission design.

The contribution of this article is two fold. First, we introduce Daphne as a complete, open-source package for DSM tradespace analysis that includes the standard VA functionality, such as taking natural language inputs from users and aggregating information from different sources to reduce cognitive load, as well as more proactive features, and more traditional tradespace exploration tools, such as a tradespace scatter plots, model inspection and explanation, and data mining capabilities. The second contribution is the result of a quantitative study with human subjects performed at JPL, from which we have obtained data comparing Daphne with the VA capabilities to a version of Daphne without those capabilities, as well as feedback from field experts on how they think Daphne should evolve.

Different features and versions of Daphne have been described both in [1] and [2]. This article will describe the current version of Daphne as it applies to DSM design, including: 1) how the user can interact with Daphne, both in traditional ways and using the VA functionality; 2) how the system processes user requests and sends them to different roles; 3) how all those roles can interact directly with the user through the use of the new proactive functionalities described in [2].

The main motivation for Daphne is that architecting multiplatform Earth observing systems is a challenging task due to a number of factors. First, the number of alternatives in the design space can be extremely large. For example, if we consider the architecture space of a multiplatform system to be defined by any binary relation between a set of N instruments and a set of M orbits, then there are 2^{NM} possible architectures. And that is only looking at instrument-orbit assignment, neglecting other important tradeoffs related to, for example, formation flying or communications architecture. Second, in order to evaluate each of these architectures, many factors need to be considered, such as the attributes that define data quality and quantity (e.g., spatial resolution, revisit time, latency) for many different measurements and data products (e.g., soil moisture, atmospheric temperature), different aspects of the system's lifecycle cost (e.g., payload and bus cost, launch cost, operations cost), as well as some metrics related to schedule and risk (e.g., time to science, system reliability). Third, in the early stages of mission design there might be significant uncertainty on some of the technical parameters, or even ambiguity in some of the system requirements and goals. Moreover, these challenges are likely to become more daunting in the future as we demand missions that are more robust, reliable, and affordable, and which generate better data products using less resources.

In addition, many distributed concepts for DSM, such as constellations, clusters, swarms, trains, fractionated or federated spacecraft, are becoming more feasible and appealing, which has opened up even more the space of alternatives, intensified

Manuscript received April 30, 2019; revised September 18, 2019; accepted October 7, 2019. Date of publication November 13, 2019; date of current version February 12, 2020. This work is part of a large research project funded by NSF under Grant CMMI 1635253. *(Corresponding author: Antoni Viros i Martin.)*

The authors are with the Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: aviros@tamu.edu; dselva@tamu.edu).

Digital Object Identifier 10.1109/JSTARS.2019.2948921

existing challenges, and created new ones in the architecting process [3]. This trend toward more distributed missions in Earth observation is seen in the latest missions flown by NASA, such as the Cyclone Global Navigation Satellite System (CYGNSS) and the Time-Resolved Observations of Precipitation structure and storm Intensity with a Constellation of Smallsats (TROPICS) [4].

Improved tools are necessary to enable these complex constellations, clusters, and trains. NASA's technology roadmap for technology area "TA 11: Modeling, Simulation, Information Technology, and Processing" recognizes the growing need for improved "Analysis Tools for Mission Design." Specifically, the document states that current tools are designed for monolithic missions and only take into account small parts of the system at a time [5]. This limits NASA's ability to consider new trends on mission design [6].

To respond to this need, NASA is developing the Tradespace Analysis Tool for Constellations (TAT-C) [7], [8]. Other relevant tools to design DSMs include VASSAR [9] and DISCO [10].

One shortcoming of most of these tools is that they provide limited cognitive support to the users, who can suffer from information overload when analyzing large, high-dimensional design spaces. For example, as mentioned earlier, a formulation of a DSM architecting space as an assignment problem with 10 candidate instruments and 5 candidate orbits defines over $2^{50} \approx 10^{15}$ valid design alternatives, that have to be assessed against several performance, cost, risk, and schedule metrics. The evaluation of performance in terms of spatial resolution, temporal resolution, accuracy, spectral content, and other attributes is a problem because preferences among these attributes depend on what is being measured and on the specific community, e.g., for soil moisture measurements, climate scientists may prefer accuracy over spatial resolution, whereas the opposite may be true for numerical weather prediction. Thus, the dimensionality of the design space is extremely large, both in terms of design decisions and design attributes, and objectives.

To address this increase in cognitive load, we advocate for the use of VAs, such as Daphne. The objective of VAs is not substantially different from that of most decision support tools, namely to "augment human intellect" as stated by Engelbart [11] in one of the first works in human-computer interaction (HCI). What makes VA different from other decision support systems, even from systems implementing some kind of intelligent agent, is the types and modes of interaction with the user, as well as the use of cognitive architectures. Most VA send and receive information to/from the user by means of natural language, either through a voice or text-based interface. Some of them also can take image inputs, like Clinc [12]. Another important difference with other intelligent agents is the fact that some VA can take the initiative and act on what they think the user wants, by using different cognitive architectures [13], such as belief-desire intention (BDI) [14], logic-based agents [15], reactive agents [10], or layered architectures combining different models. The explicit use of some model of the user's cognitive process is arguably the most distinctive trait of modern VA.

The rest of the article is organized as follows. Section II reviews the literature on intelligent tools for system design and intelligent assistants, both inside and outside the design for aerospace domain. Section III provides an overview of Daphne for DSM design. Section IV describes the NASA JPL study and results, both quantitative and qualitative. Finally, Section V provides a summary of the article and discusses its limitations and directions of future work.

II. Background

A. Artificial Intelligence in System Design

Intelligent tools have been used to support the design of complex systems since the dawn of the computing era. While VAs are rather a new concept, design automation tools have been used for decades. An early example of a design automation tool is the R1 system [16], which was used to design the layout of VAX-11 computers based on a customer order. This system was rule based, and it ensured that the computer could fit in the customer room, and at the same time, have all the needed components. This is one clear example of an intelligent design tool that performs automated design, as the technician only needed to introduce the sets of restrictions and needs to obtain a working design. Other examples of intelligent design tools performing design automation can be found in review papers and special issues by Hayes *et al.* [17] and Goel *et al.* [18], among others.

All tools described in this section are intelligent tools for either design automation or conceptual design, including Daphne. This being said, most of them are not VAs, while Daphne is one. Daphne takes functionalities from design automation tools, conceptual design tools, and VAs to have a feature set that can help aerospace engineers design better missions.

In addition to fully autonomous agents that create designs automatically, there are different types of intelligent tools to support system design, including intelligent computer aided design (CAD) systems [19], [20], knowledge databases [21], [22], design assistants [23]–[26], and design critics [27], [28]. As can be seen, design automation tools include a wide range of behaviors and technologies in artificial intelligence, from brute-force search, to explanation, machine learning, and human-agent interaction.

Another important and relevant aspect of intelligence in design tools is adapting to individual designer differences and preferences. Peng and Gero [29] demonstrated a situated agent that adds on to Matlab's Optimization Toolbox and learns the optimization algorithm that is most appropriate for a given design problem. Another example of tool that adapts to the users are modern integrated development environments (IDE), and the way they adapt is by tuning the code suggestions, i.e., the Autocomplete function, to past user choices. A last example on the topic of self-tuning autocomplete features are all of Google's text editors, including diverse products, such as Gmail, Google Docs, or Messages, which suggest answers to e-mails to the user or the next word to write in a sentence based on past inputs from the user community.

Since Daphne for DSM applications is centered on the first stages of design, sometimes referred to as conceptual design (or system architecture in the complex systems literature), we focus

our literature search on intelligent tools that support these early stages of design. Conceptual design is a relatively unstructured and ambiguous task that emphasizes creativity and tradeoffs. Thus, most tools supporting this task are catered toward helping the designer and enhancing their cognitive abilities (e.g., decision support) instead of replacing them (design automation). Such tools often take the shape of interactive visualization tools, which allow for the analysis of different design alternatives, and which have the capacity to handle hundreds to millions of designs and a handful of objectives. Examples of such tools include [30]–[32]. In these cases the extent of intelligent behavior may be as narrow as providing some basic data mining and/or advanced interaction. Indeed, some of these visualization tools allow the design engineers to compare different representations of the data (e.g., decision space versus objective space, different two-dimensional slices/projections of the objective space, or parallel coordinates plots), highlight architectures with common features, and reduce the objective space search to a much more manageable one by applying various filters [23], [30], [31], [33]. Other tools use the results of unsupervised machine learning algorithms, such as feature extraction, manifold learning, and clustering to help users get a clearer picture of the trade space [25], [34]. To further reduce the cognitive load of system engineers, other tools combine visualization and data mining algorithms that extract patterns, for instance in the form of if-then rules (e.g. "IF there is an atmospheric chemistry instrument in an AM orbit, THEN the architecture is likely to have low science benefit") [26], [35], [36]. The use of logical rules as data structure for these patterns has been used for decades in artificial intelligence, due to evidence that these rules are easy to understand by humans and that they may resemble how human experts solve problems [37].

Despite all these efforts to reduce information overload in complex system design [38], [39], often the amount of data is still so large that it is hard to extract the relevant information for the design task at hand. For example, a data mining algorithm may give a list of features that are common among good mission concepts, but the designer still needs to choose which ones to apply to the final design, and this bears a nonnegligible cognitive load on the engineer. This means there is a need to further support the systems engineer to help him or her direct their attention to specific portions of the dataset or specific aspects of the problem, depending on relevance and other factors, and this is where VAs come into play: They can answer questions from the designer or act on their own to suggest focus points or potential improvements to the current configurations, all while providing easier access to the data through a natural interface.

B. Virtual Assistants

In the beginnings of computer science, the term VA was used to define any system that could "enhance human intellect." As years passed and the AI field matured, different names appeared for different types of AI systems, such as expert systems, rule-based systems, Bayesian networks, or machine learning. Thus, the term VA fell out of use in favor of more specific terminology. Later, with the advent of systems, such as RADAR [40], Cognitive Assistant that Learns and Organizes (CALO) [14], or DARPA's Personalized Assistant that Learns (PAL) [41], the concept came back with a much more limited scope: Now it is only used to describe systems which perform tasks at the request of the user, and interact with him/her through voice, textual, or visual interfaces. Other early examples of this modern architecture include Open Agent Architecture (OAA) [42] and Integrate. Relate. Infer. Share (IRIS) [43].

More recently, with the disruption of deep learning, and as voice recognition software and natural language processing have made important improvements in performance, commercial VAs have appeared, such as IBM Watson [44], [45], Wolfram Alpha [46], Siri [47], Google Assistant [48], Microsoft Cortana [49], Amazon Alexa [50], or Mycroft [51]. Several VAs have been developed in the research world as well, including Clinc [12], [52], YodaQA [53], or OAQA LiveQA [54]. All these systems share the characteristics that the interaction is done either by natural language or through pictures and that they are "generalist": They try to answer as many queries as possible from the user using a plethora of data sources, as opposed to providing high-quality answers in more specialized domains. For example, generalist VAs help the user with personal organization by reading e-mails, setting appointments, and other mundane tasks, such as playing music. Naturally, generalist VAs are of no use when the task at hand is very specialized. Thus, specialized VAs in the field of design are described next.

C. VAs for System Design

Daphne is conceived to support the user during the system's early design process. There are some other VAs used in design. TAC (The Architect Collaborator) [55], for example, can conduct trade-space exploration of house/building designs by following commands from different users. In [56], a rule-based system is used to provide recommendations on manufacturing technology designs. In [57], an expert system is used to evaluate alternatives on highway building. In this case, the user chooses from some proposed alternatives that are refined in an interactive process where the designer is aided by the assistant, which checks for regulations compliance for all user modifications. Finally, the best choices are evaluated again and the user can choose the most preferred one. A more recent effort, PQE [58], tries to model user curiosity and creativity, and tries to come up with design alternatives the user might not have thought of, with the goal of helping the user think out of the box. Another recent work is [59], where a process for innovative problem solving is proposed based on the human and computer assisting each other in the identification and characterization of the obscure features of a problem. Finally, ESA's Design Engineering Assistant (DEA) [60] introduces a VA for concurrent engineering processes that helps engineers by providing data aggregation and synthesis capabilities of past unstructured documentation in an automated manner. It also includes a user interface (UI) for fast access to this information.

DEA is, by far, the most similar work to the one presented here. However, it is still in the very early stages and it appears to focus on data retrieval from unstructured documents. Daphne

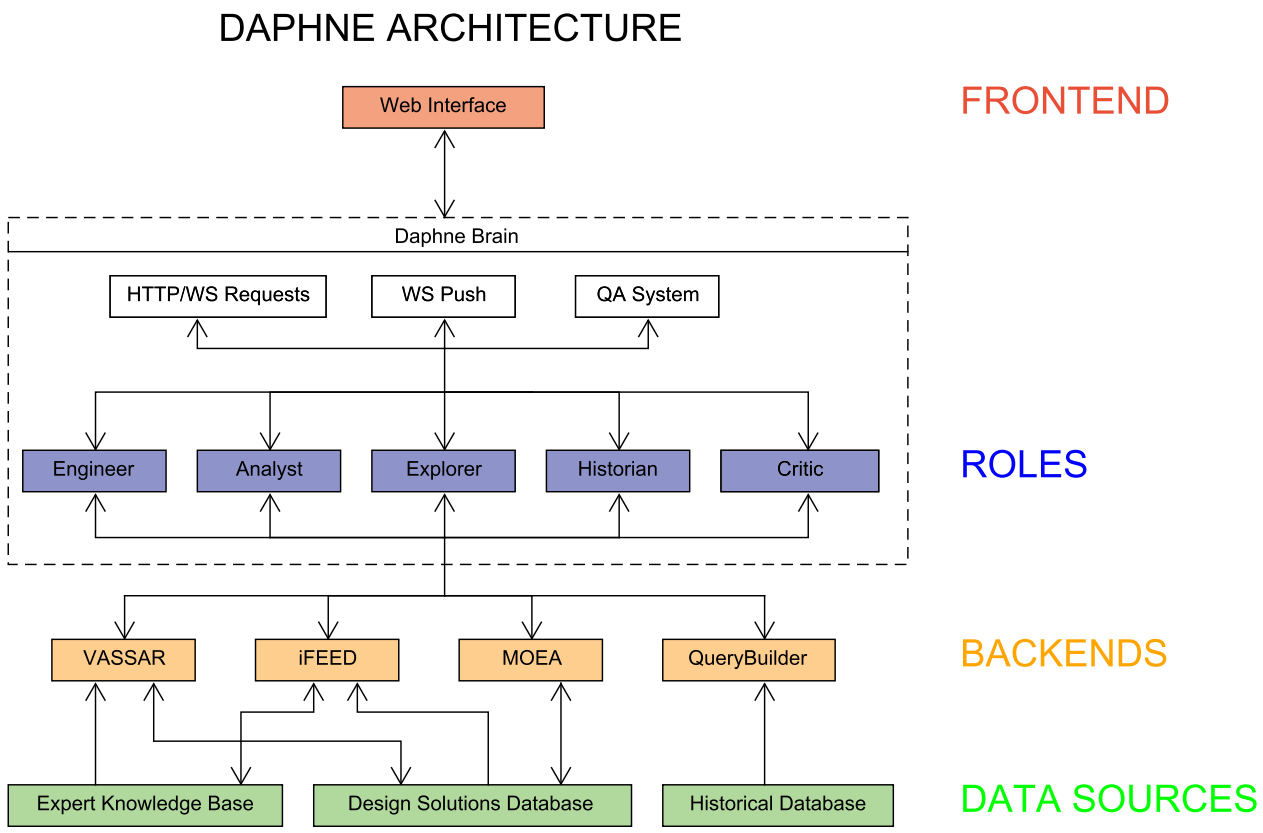

Fig. 1. Overview of Daphne architecture for DSM applications.

currently does not use unstructured documents as a source of information and relies instead on structured databases as well as a knowledge base to encode rules of thumb and procedural knowledge as rules.

III. Daphne: System Architecture Overview

Daphne (Fig. 1) is structured as a microservices system [61]. It consists of: 1) a web front-end; 2) a front-end server (the Daphne Brain) that directs all user requests –which are either based on HTTP or on Websockets and can be classic requests (e.g., from a user interface button) or natural language through the QA system– to the appropriate role; 3) a set of roles, which can be understood as software snippets, that use some of the available microservices—be it backends or data sources—to obtain the result the user is asking for. All backends and data sources can run independently of one another and we are currently in the process of making the whole system run in different machines, making Daphne scale horizontally, as the microservices design allows for. These software components are described in detail in both [1] and [2], but summaries of all of them are provided below for completeness.

A. Data Sources

There are three different data sources from which Daphne can get its information: 1) an Expert Knowledge Database, with expert rules and recommendations about how to design (good) spacecraft in general and DSM in particular; 2) a Design Solutions Database, which contains information from a great variety of possible architectural solutions for the problem at hand – typically obtained from the output of a search or optimization process; 3) a Historical Database, with data from all past and planned civilian earth observation missions.

The Expert Knowledge Database is different from the other two in that it does not contain factual knowledge in the form of tables, but rather procedural knowledge in the form of logical (if-then) rules. Each rule encodes a domain-specific chunk of knowledge about how to architect space missions. These rules are not to be interpreted as hard constraints or absolute truths, but rather heuristics or rules of thumb with different degrees of certainty. One example is that UV/VNIR atmospheric chemistry spectrometers should be flown in afternoon sun-synchronous orbits when possible rather than morning or dawn-dusk orbits, as good illumination conditions are needed for this kind of instrument, pollution peaks in the afternoon, and other synergistic atmospheric chemistry instruments are flown in an afternoon orbit (e.g., NASA Aura mission). There is an extensive description of the rules in the Expert Knowledge Database in [9], [62].

The Design Solutions Database has three main objectives: First, to provide an initial dataset of possible designs for the system to serve as a baseline with which to compare new designs; second, to save the new designs the user or Daphne come up with so they are not lost after the work session is finished; and third, to provide a dataset on which the VASSAR, iFEED, and

MOEA back ends (described later in the article) can run. The database consists of a set of designs. One design is represented as a set of spacecraft, each of which is primarily characterized by a set of instruments and orbit, from which a preliminary design of the spacecraft as well as lifecycle cost and science benefit are computed by the architecture evaluation back end. Additional parameters can be recorded, such as the level of satisfaction of the hundreds of measurement requirements (e.g., revisit time and spatial resolution for various data products) that make up the scientific benefit metric, or the launch vehicle or each spacecraft's mass and power budgets. More information can be found in [1] and [62].

The Historical Database has been obtained from the Committee on Earth Observation Satellites (CEOS) database [63]. This is a joint effort between CEOS and ESA to create a comprehensive database of all the Earth observation missions that have been launched (and those that are planned) since the beginning of the Space Age. It has detailed information on the missions, the instruments, the measurements which can be taken, and the space agencies that have developed them. As of March 2019, the number of missions in the database is 620, with 891 instruments also described. We have performed data mining on the database to be able to answer questions, such as "Which is the most common orbit for measuring <measurement>?" by obtaining information, such as the most common kinds of orbits for each kind of instrument and measurement mentioned on the database.

B. Back Ends

The back ends in Daphne use the data coming from the different data sources to compute the information needed by the different *roles* to form answers to the users' questions. There are four back ends.

1) The VASSAR back end, an architecture evaluator which given a design, returns the value of the objective functions (i.e., lifecycle cost and scientific benefit primarily).
2) the iFEED back end, which runs machine learning algorithms on the database of designs.
3) the MOEA back end, which is in charge of running the background search that fuels the Explorer role.
4) the QueryBuilder back end, which is in charge of translating the questions for the Historical Database into queries that the database can understand.

All of them are briefly described in this section.

The VASSAR back end uses the methodology described in [62], which is a systematic framework that allows automating the architecture evaluation process by combining both objective and subjective information encoded in logical rules. The methodology is applied to the use case of Daphne described in this article, namely architecting DSM, and returns a cost and a science benefit measure when given a design. In order to estimate lifecycle cost, VASSAR performs a preliminary design for each spacecraft in the mission, containing basic mass, power, and volume budgets at the subsystem level. Cost estimating relationships are then used to estimate payload and bus cost from spacecraft design parameters. Launch and operations cost are also estimated. The most cost effective launch strategy is selected for the system based on spacecraft wet mass, dimensions, orbit, and launch pricing. The scientific benefit metric essentially counts how many measurement requirements—out of a database of thousands—are fully or partially satisfied for the selected architecture. Full or partial requirement satisfaction is assessed by comparing the capabilities of the given architecture with target and threshold values for various data products and measurement attributes, such as spatial resolution, revisit time, and accuracy. In addition to the numerical values of lifecycle cost and scientific benefit, VASSAR can also provide explanations for those scores, tracing back their numerical values to specific measurement requirements, instrument capabilities, or model assumptions. For more information on VASSAR, the reader is referred to [62].

The MOEA (from multiobjective evolutionary algorithm) back end is in charge of running the background search feature of the Explorer role. It implements a variant of the genetic algorithm (GA) described in [64] and [65]. Specifically, it uses the knowledge-driven optimization with adaptive operator selection to try and find designs that improve the Pareto frontier of the current data. The data are sent back and forth through message queues, such as RabbitMQ [66], so the system can run on multiple machines and scale horizontally.

The iFEED back end mines the Design Solution Database and extracts features that are consistently more present in "good designs" than in "poor designs," so the roles can use those features to create new or modify existing designs, or just to provide suggestions to the user and help the user gain insight about the problem. What constitutes a "good design" can be defined by the user—by default, a design is good if it is on or close to the science-cost Pareto front. An example of a dominant feature could be that most solutions currently on the Pareto front have instruments A and B together in the same orbit (either a specific orbit, or more generally in any orbit), or never together in the same orbit. Specifically, the data mining back end uses classification rule mining, an extension of association rule mining. Standard association rule mining [67] seeks to extract frequent patterns in the attributes of observations in large datasets, where attributes can be discrete-valued decisions and objective values of solutions. Classification rule mining applies association rules in a classification context, and extracts rules $F \rightarrow G$ where the precedent F is a feature or logical combination of features – e.g., instruments A and B together in any orbit, and the consequent G is a class label, in our case, indicating whether or not the design is "good." The algorithm used to extract the rules is based on genetic programming. Features are encoded as trees, and an evolutionary algorithm with adaptive operator selection is used to search the space of possible features, while simultaneously optimizing specificity (a.k.a. recall), coverage (a.k.a. precision), and rule complexity (number of literals). For more information on the iFEED back end, the reader is referred to [26].

The QueryBuilder back end is in charge of translating natural language questions into SQL queries that can be answered by the Historical Database. Upon receiving a question, the QueryBuilder back end classifies the question between different types that have to be answered by different database queries, loads

the parameters that need to be extracted by the natural language processing (NLP) module, the query template, and the response template from a file for each type, extracts the relevant features (the information needed to answer the question contained inside of it), augments the data with context information sent to it from the historian role, crafts the database query, and puts everything together to create a response based on the results of the query that can be in any format demanded by the user, be it plots, text, images, or lists. These steps are detailed in the next paragraphs, as while the rest of the back ends are described thoroughly in previous papers, there is no published work that describes this back end.

1) Text Classification: The raw text input needs to be preprocessed so that the classifier can do its job better. This is done by leveraging the models in the spaCy Python library [68]. The decision to use spaCy instead of using more sophisticated existing solutions, like NLTK or OpenNLP [69], [70], or developing a custom solution in house was taken after observing that the needed functionality existed in all three solutions, but spaCy was the only system that provided a fast enough computation to make the system work in almost real time.

The raw string is processed by lemmatization, part of speech tagging, and stop word classification, which outputs the following:

1) the lemma (the version of the word found on a dictionary) for each word, which is used as the input of the classification model;
2) its classification as a number or not, which is useful for feature extraction;
3) whether the word can be ignored for the rest of the pipeline (referenced as a stop word in the literature);
4) the lower-case version of the word, which is used to normalize the text for the rest of the pipeline.

With all this information obtained, the next step in the QueryBuilder is to perform a classification to determine the type of query to do on the database and the data to be extracted from the question in order to successfully perform the query.

Classification is a recurring problem when dealing with text, so many algorithms are available, including [71] rule-based systems, nave Bayes, k-nearest neighbors, support vector machines, decision trees, logistic regressions and, for the last three to four years, deep neural networks of the convolutional and recurrent varieties.

Focusing on neural networks, the state of the art evolved from using the relatively simple convolutional neural networks [72], to augmenting them [73] with word embeddings, such as word2vec [74] or GloVe [75], to using recurrent neural networks (RNNs) with long short term memory (LSTM) units [76] and attention mechanisms [77]. State-of-the-art algorithms keep using LSTM RNNs, but have moved from considering words as a unit to considering each individual letter in a sentence to perform the classification. Examples of this current trend include [78], which add context dependent word embeddings, and BERT [79], which is a pretrained neural network model from Google that can be understood as a more powerful version of word embeddings.

When deciding what algorithm to use for classification, we strove for a balance between accuracy—as in number of questions correctly classified, training performance—the time the model needs to be properly trained, - and evaluation performance—the time the model needs to come up with a classification for a question–, as well as implementation ease.

As seen in [77], the accuracy of neural networks is usually higher than that of other algorithms before them, especially if trained with enough data, for problems, such as text classification. But training RNNs takes much longer than training CNNs, due to the more sequential nature of RNNs over CNNs, which makes RNN much harder to parallelize. This large difference in training time coupled with a less acute loss of accuracy ended up tipping the balance in favor of a model based on CNN with word embeddings, as in [73]. Its implementation has been relatively fast to train (it takes around 8 min on a Nvidia GTX1080 for our dataset).

The whole classification subsystem is implemented in Tensorflow [80], and the hyperparameter tuning of the model has been done using the results in [81]: The size of the word filters has been set to 3, 4, and 5 words and the number of filters per word size has been set to 100, as these values appear to give consistently good results on different datasets.

Regarding the training data, it is known that machine learning algorithms, and especially neural networks, need large datasets to be able to perform with good accuracy. To solve this problem for the QueryBuilder as well as the main QA system of Daphne, where no datasets existed, we created a question generator, which can add random variations to generate multiple versions of the same question. The system also adds examples with noise (random words that make no sense in the middle of the question) so the algorithm can be robust to spelling mistakes in the speech-to-text translation. A set of 8670 examples has been generated in this way, with different numbers for each question type depending on how many variations of the question are plausible. The input for an NN based on word embeddings is a vector of word vectors, so all the words in the questions are separated into a vector, and each word is turned into a vector of 128 randomly initialized numbers, which are then trained together with the rest of the network, whose weights are also randomly initialized.

This training approach has given good results, as system testers have a hard time finding questions that can fool the neural network into misclassifying a question, while at the same time, keeping the time and effort required to add a new function relatively low, as the only two steps needed are generating a new set of examples with the random generator and retraining the network, which is already programmed to get the number of output classes from the number of files with examples that are provided to it.

The reported accuracy of the neural networks is computed according to (1), with the batch referring to the set of questions being evaluated at the same time, output_{ij} being the sigmoid score being computed by the model for an output class j of a sentence i and expected_{ij} indicating whether sentence i is of class j or not.

$$\text{acc} = \frac{1}{|\text{classes}|\,|\text{batch}|} \sum_{i}^{\text{batch}} \left(\sum_{j}^{\text{classes}} \left| \lfloor \text{output}_{ij} \rceil - \lfloor \text{expected}_{ij} \rceil \right| \right). \tag{1}$$

Both are rounded to obtain the classes computed as most probable as correct scores and the rest as incorrect. Although most inputs only belong to one class, some inputs might be valid for more than one role, so the system has been designed to accept such a case, and this is why output_{ij} has to be a vector instead of an indicator function, and this is also the reason for using a sigmoid score instead of the more common softmax for classification problems. The training and test sets are created by using a random sample with a size of 80% of the dataset as a training set and the other 20% as a evaluation set. The accuracy obtained for the evaluation set of inputs is of 100% for all models we have trained, which include the one that gives the QueryBuilder the type of database query it should use as well as the main one for the QA System and one for each of the roles. All of these are described later when explaining the QA System in Section III-C. Since the number of trainable parameters is greater than the number of examples, there is some risk of overfitting. However, so far, all the models have performed well both in a pilot test we had run in 2017 and the JPL experiment. All 46 questions the test subjects on the pilot asked were classified correctly while out of the 249 questions asked by JPL test subjects, 85% got answered correctly. This decrease is expected as test subjects were less familiar with the syntax the system has been trained for, and their expectation was that the system would understand questions of any type, as per their own feedback.

2) Question Type Definition: There are three pieces of information that define a question type: The parameters that need to be obtained from the question, its database query template, and the response template. All three pieces are defined in a file we call the "question definition file" that is unique to each question type. To make the QA system as easy as possible to expand, adding a new question type should not require anything more than basic database knowledge. This means the person adding it should not have to be an expert on the internal workings of Daphne. Making changes to existing question types and adding new ones does not involve writing new code for Daphne. To make this section easier to follow, we will use the following example question during the whole section: "What is the most common orbit for soil moisture at the surface?". The following list describes the information that needs to be defined for each question type in more detail.

1) *Parameters*: A parameter is a piece of information needed for the query that is contained within the question. What is contained in the question definition file is an array with the names and the types of the parameters, some extra data, and whether parameters are mandatory for the question type. For example, in our question, there is a single parameter for the "measurement" parameter, with name "measurement1" and that is mandatory to obtain an answer.
2) *Query*: The query is built by stitching together different building blocks (or templates). There is an "always" section, which is executed all the time, followed by an array of optional filters and subqueries that depend on some of the optional parameters being present. Then, there is the "end" portion, which is always appended at the end of the query. The last two sections are the "result_type" and the "result_field," which define if the result is a number, a text, or a list, for example, and tell the system which field of all those returned by the database is the needed result. For example, in our example question, the query would be (in pseudocode): "SELECT common_orbit WHERE measurement = measurement1."
3) *Response*: The response is built from a template that can use all the different parameters and the result from the query. It can contain text, audio, graphics, and video. For the example question, this would be the sentence: "The most common orbit for <measurement1> is <common_orbit>."

3) Extraction of Parameters: The next step is to extract the parameters. The algorithm is the following.

1) For each parameter, an *Extract* function is called depending on the type of the parameter. This function can be of three types: a) it can use the named entity recognition of spaCy to obtain the required features from the question, if spaCy already implements the algorithm for that type of feature. For example, numbers are easily recognized by spaCy, so all numerical parameters such as IDs are recognized through it; b) it can be a substring matcher based on the lists of possible values for each parameter type being extracted from the database. Instruments, missions, measurements, and instrument types are all extracted in this way, and the algorithm will be thoroughly described below; c) it can be just a substring search to see if there are substrings in the question that comply with certain conditions. For example, the years are extracted by checking if there is any substring that looks like a year—four digits—and then saving all of them in order in case more than one is needed.

   To extract names of missions, measurements and technologies the process is more complex, and is described in the following steps.
   a) First, a list E with all the possible values (e.g., of measurements) is obtained from the database.
   b) Then, each element in E, $e \in E$, is compared to the whole question Q using Sellers' algorithm [82]. As a result, the elements in E are sorted by maximum similarity to Q. The maximum similarity between e and Q is computed, according to Sellers' algorithm, as the minimum edit distance between e and a substring of Q, $s \subset Q$, of the same length as e, normalized as follows: $\text{max\_similarity}(e, Q) = \max_s(\frac{|e| - \text{edit\_dist}(e,s)}{|e|})$. The list is then cropped to only the elements with a high enough max similarity (the threshold is 0.75).
   c) The list is cropped to only the number of required elements of the same type if it is longer than that.
   d) The list is then reordered by the appearing position of each element in the question and then saved in that order for the rest of the pipeline.
2) Once the extraction is done, the list of features is passed through a processing function that applies certain modifications to each feature depending on its type. For example,

years are converted into specific dates depending on some extra data from the definition file. Most other features remain roughly the same, with some minor adjustments, such as the elimination of spaces at the beginning and at the end of the string or some changes in the capitalization.

3) All the extracted and processed features are sent back to the main pipeline so they can be used in the next steps.

In our example question, we would run the question through this algorithm to extract a parameter of type "measurement," which would trigger the substring matcher based on lists and would yield the value of "soil moisture at the surface" for "measurement1."

4) Adding Contextual Information: After extracting all the parameters from the question, the next step is appending all the contextual information available to the system so more questions can be answered. One example of this is the meaning of "now" or "currently," which may appear in some questions and is not explicitly told by the user to the system. Another example is the meaning of "this design," which changes every time the user clicks on a new design on the Daphne UI. A third one is the last measurement the user has asked about.

In our example question, all of this information is added, but it will not be used. Example questions where some of this information might be used are "What do you think of *this* design?" and "What missions *currently* flying are measuring soil moisture at the surface?"

5) Querying the Database: The second to last step in the pipeline is querying the database to obtain the required information from it. The data needed in this step are all the augmented data from the question along with all the query templates that have already been mentioned. There are a few substeps to it, which are described in the following list.

1) First, the "always" template is run through the Python template engine to obtain the first part of the query, which is going to be run. In our example question, that is the only part that exists, and it is similar to what we described in the question-type definition.
2) Then, for each "optional" template, the condition of activation is checked against all the available data, and if it evaluates to true, then the template is run through the engine and appended to the end of the query. For example, some questions have an optional parameter to filter the results by space agency, or a year range. These filters would be applied here.
3) Finally, the "end" portion of the query is run through the engine and it is appended to the end of the query. This is used when listing missions to order the results by year, for example.

With the query fully constructed, it is run in the database and its results are obtained. In our example, this will return the most common orbit for this kind of measurement, which is a low earth, sun-synchronous, high-altitude (within LEO), no repeat cycle orbit.

6) Answer Building: With the results in hand, the portion of the answer which depends on them is built. Responses can be of different types and can use multiple fields from the query response. Each type has a completely different process to build itself.

For example, if the response is a list, all the query results are appended in a string with separating commas. In the case of a date, the date is written in a human readable form, as the database stores it as a UNIX timestamp. The last implemented case is for orbits, as the orbit-related questions have their answers stored in a format which can encode all the information from the decision tree used for data mining the Historical Database, so a parser was built in order to decode that information into a human readable format.

Finally, the last step in the pipeline is building the answer the user will see in his/her screen. The answer is built by running the answer template corresponding with the question type on the template engine with all the augmented data from the question together with the response from the last step. The final answer for our example question will look like this: "The most common orbit for soil moisture at the surface is a low earth, sun-synchronous, high-altitude, no repeat cycle orbit."

C. Daphne Brain

The Daphne Brain is the key component of the system architecture, as it sends user requests of any kind to the different *roles*, be it voice commands, a click on a button, or just some textual instructions. Its job also includes returning the response from the *roles* to the front end. The latest addition to this module is the ability for the *roles* to send messages directly to the user without waiting for the user first. This is used extensively by the proactive or mixed-initiative functionalities.

As a critical piece of software, it is very important for the Brain not to fail, as it can be considered a single point of failure: The system becomes unresponsive under a Brain error (e.g., due to a malformed HTTP request). This is why it is implemented using multiple processes, which are replicated, so functionality is not lost in case one fails.

Requests received by the Brain can be "classic" requests, such as HTTP or Websockets, which are triggered by UI interactions, such as clicking or hovering and are processed through REST and WebSockets end points, or they can be sentences in natural language. This second kind of requests is processed by the QA system, which also lives inside the Brain. All requests can use the whole power of Daphne: Roles have access to all the back ends and data sources. This gives a lot of flexibility when implementing them, as anything can be done inside each request handler: Back ends can be called, data sources can be accessed, and computations can be performed. At the same time, having this flexibility means it is even more important to have the Brain be resilient to failures, as they are bound to happen.

While HTTP and WebSockets requests are handled in a standard way, the QA system, which processes questions, merits additional explanation, as it is a key differentiator between Daphne and other decision support tools for engineering design. It helps in creating a more intuitive interface for Daphne, by giving the users a natural language interface, which is believed to be a more natural HCI (in some circumstances) compared

to the other most common ways of interacting with a decision support tool, such as querying databases, programming charts, or just using a mouse [83], [84].

The QA receives questions in natural language (English) and then processes them to call the appropriate role. The user can input these questions by either voice or text. In case the user uses their voice, we use Google STT service to turn it into text. After this, it generates an answer based on the result from the role, which is human readable and may contain text, audio, and/or images.

The QA system classifies the input questions according to their intent and type from a finite set of intents and question types. To classify the intent of the user, the same CNN model used on the QueryBuilder back end is used. This model, trained with different datasets for each case, is used to classify the input to decide which role will process the sentence. For example, a question such as "Which is the most common orbit for atmospheric lidars?" would first be classified as a question for the Historian role, and then the QA system would send it to that role. When the answer comes back from the role, the QA system is responsible for presenting it in the front end, which can involve showing the visual response sent by the different roles or voicing it out using a text-to-speech service.

D. Roles

Roles, as called by most VA, are computer programs that leverage all the services available (data sources, back ends) to perform the actual useful tasks to the end user. The next few paragraphs describe the roles in Daphne for DSM design.

1) The Engineer, a role in charge of obtaining data from a design.
2) The Analyst, a data mining role that provides all the functionalities of iFEED [26].
3) The Explorer, a role in charge of fostering exploration of the design space.
4) The Historian, the role in charge of answering questions about past missions to foster innovation and creativity.
5) The Critic, which, given a design, can identify potential weaknesses and give back suggestions on how to improve it.

The Engineer role is in charge of evaluating new DSM designs found by the user and return both cost and science performance back to the user, as well as answering questions about the different metrics of a design, such as "Why does this design get this cost/science score?" Both functionalities are fulfilled by calling the VASSAR back end.

The Analyst role supports the systems engineer in the task of mining the dataset of design solutions for features common to good designs. It looks for features that best describe those good designs or, more generally, that best describe designs in a user-defined region of interest, e.g., the Pareto front. It uses the iFEED mining back end for this purpose.

The Explorer role controls the background search being performed by the MOEA back end, and keeps track of the diversity of the designs found by the user. A sequence diagram is shown in Fig. 2. This role begins the background search on startup. The role also receives new designs and presents them to the user. This includes sending notifications to the user when enough new designs have been found. The second task of the explorer is to aid in increasing the diversity of the designs found by the user. This feature is named "Diversifier" in the front end. In our experiments, we found that users of Daphne tend to focus on small areas of the dataset to search for new designs, missing out on opportunities to explore certain design configurations. Thus, we created a feature that recommends new areas of the design space that might be interesting to the user. This subsystem works by partitioning the Pareto frontier in ten smaller sets of the same size and computing the mean crowding distance (a well-known metric for diversity in multiobjective optimization algorithms, such as NSGA-II [85]) in each set. If an area has a high maximum crowding distance compared to the rest, the system recommends that the user explore it.

The Historian role takes questions about past and planned missions, channels them to the QueryBuilder back end, and displays the responses back to the user. For example, the user can ask: "Which missions can measure <measurement>?," "When was mission <mission> launched?," or "Which missions are currently flying <technology>?" The role is designed as a restricted domain question answering system, meaning it can only answer a limited set of question types about a limited set of knowledge. This is appropriate for the use case of answering questions about Earth observing missions, as the set of questions which can be asked is small compared to that of general-purpose VAs. Priority is placed on getting correct answers rather than tackling a lot of question types. This is important because trust is a critical factor in the interaction with any kind of intelligent system [86].

The Critic role receives a system design as an input, and its output is some feedback to the user about the strengths and weaknesses of that design, together with specific suggestions to the user about how to improve it. It includes four different agents based on each of the roles. All of them create different kinds of feedback and suggestions: The rule-driven agent (Engineer role), which uses the expert knowledge base (i.e., expert design rules about designing spacecraft in general and DSM in particular) to obtain recommendations; the legacy-driven agent (Historian role), which uses the Historical Database to identify past or planned missions that are similar to those of the current design; the data-driven agent (Analyst role), which uses the Design Solutions Database to notify the user if a given design shares some of the driving features usually found among good designs so far; and the exploration-driven agent (Explorer role), which attempts to improve the design at hand by searching its close neighborhood. One of the new mixed-initiative features lives inside the Critic: It is named "Live Recommender System" on the front end. This new feature of the Critic shows part of the feedback from the aforementioned agents in real-time while the user of Daphne is modifying an DSM design. This needs to be real-time, so only the feedback that can be gathered fast enough is shown. This includes feedback from the rule-driven agent, which shows some rule-of-thumb feedback; feedback from the

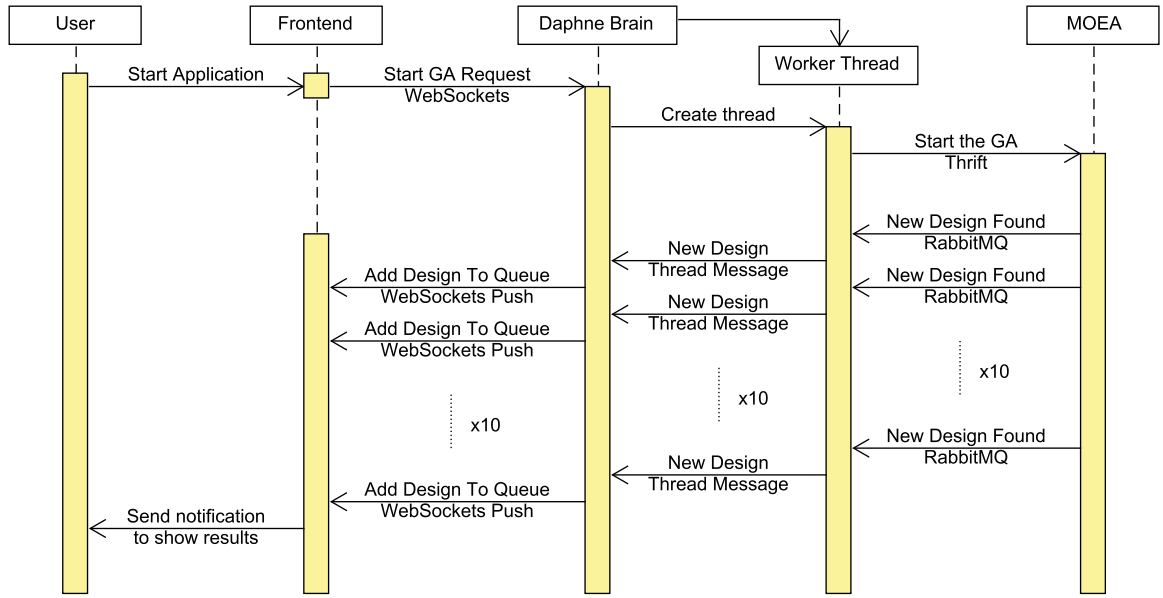

Fig. 2. Background search interaction timeline.

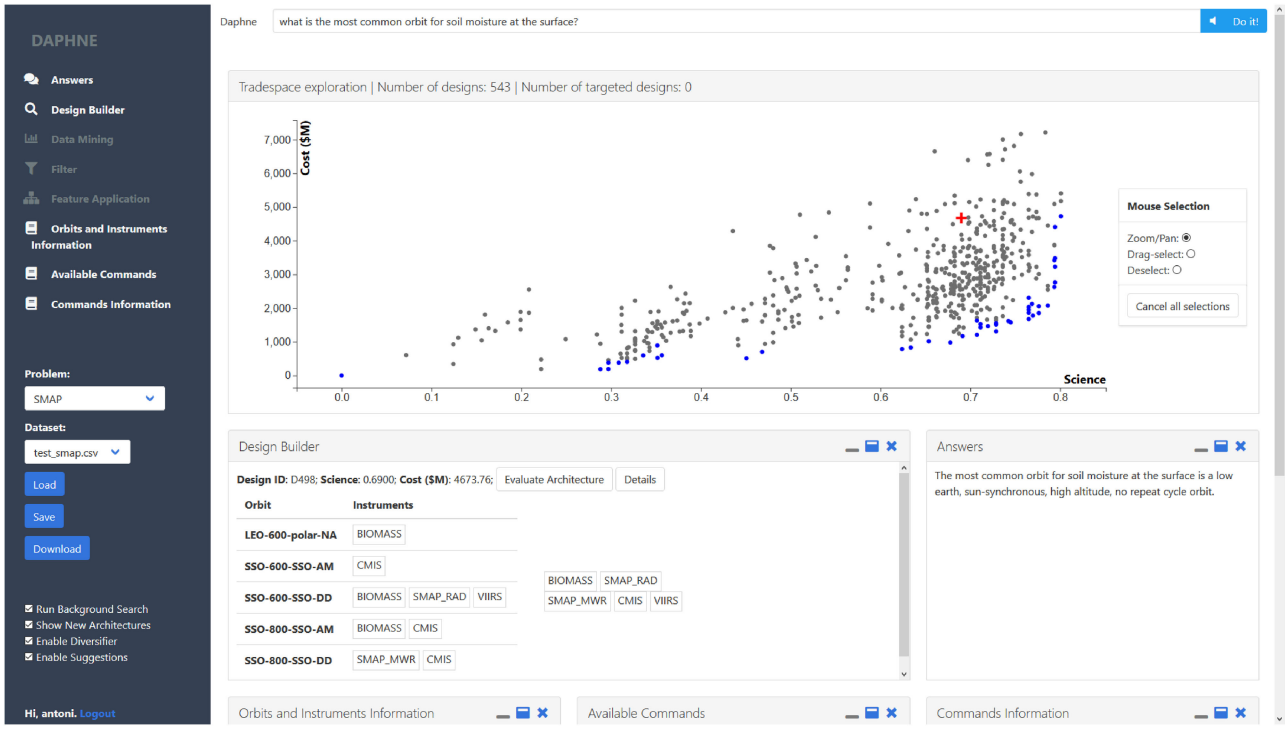

Fig. 3. Daphne's interface.

Historian, which compares the currently modified design to past and current missions; and feedback from the Analyst, which runs the iFEED back end every few seconds to find features that describe the Pareto frontier and looks at the current design to check whether those features are present in the design or not. The reader is directed to [2] for more information on this addition to the system.

E. Front End

Daphne has a web front end, shown in Fig. 3, which serves as the main interface of the system. It has the following functionalities: the user can work on a dataset of design solutions, select arbitrary regions of interest, use the design inspector to hover over designs and obtain information about them, evaluate new designs, create and visualize a new iFEED feature, use

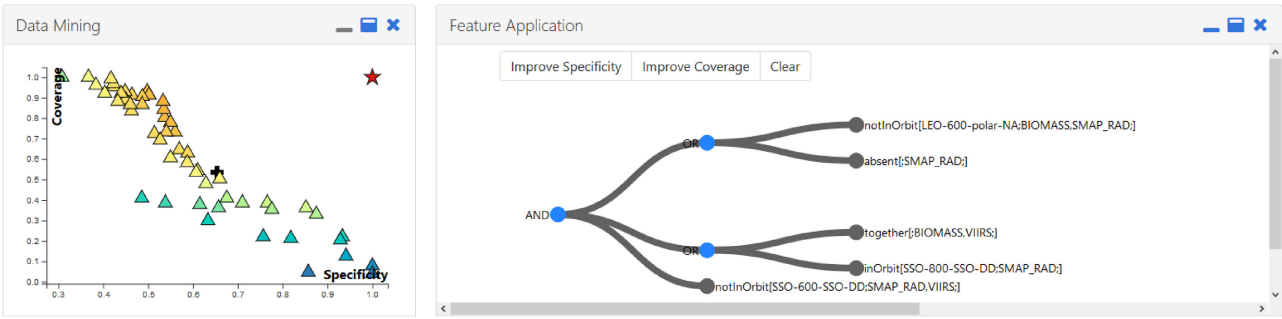

Fig. 4. iFEED Feature space plot and a logical tree representing a feature.

the data mining interface to extract dominant iFEED features, ask questions through the natural language interface, change datasets and problem definitions, read about the different functionalities available, download the working dataset, register and login, and control the mixed-initiative features. The designs on the scatter plot are obtained from the Design Solutions Database. The user can also explore the iFEED feature space—a second scatter plot displaying all the best features found so far in a space of specificity (a.k.a. recall) versus coverage (a.k.a. precision). By hovering over a given feature, the user can graphically see what the feature is in a logical tree, and modify the feature (e.g, change a logical connective from "AND" to "OR", or combine the existing feature with another one from the feature space), by manipulating the tree. Fig. 4 gives an example of this secondary plot and its accompanying tree.

IV. Experiment

To validate whether Daphne is useful to engineers designing Earth observing missions, a human study was conducted at NASA JPL with system engineers as participants. Based on some pilot experiments we conducted, we had some concerns that Daphne's advanced VA functionalities could lead to reduced learning and/or perceived usability. When we refer to learning in this article, we are thinking about the improvement in understanding of the design problem by the system engineers. For example, we want to test their understanding of the main tradeoffs involved, how sensitive the models are to changes in the design, what are the driving features (as in iFEED features) for both cost and science scores, and the mapping between design decisions and mission attributes. Therefore, the ideal outcome going into the experiment was that Daphne would lead to better designs, without sacrificing human learning or usability.

Thus, our hypotheses when performing this experiment are the following. H1) Test subjects using Daphne with all its features will have a better performance when designing DSMs than those using the version with more traditional tradespace exploration features. H2) Users will learn the same amount about the problem when using the fully featured version of Daphne when compared to the more limited one. H3) The usability scores for both versions of Daphne will be similar, according to past experimental results.

The experimental design and the details of how the measures of performance, learning, and usability are taken are provided below.

A. Experimental Design

We recruited $N = 9$ NASA JPL Engineers. Recruitment was done by soliciting existing contacts from the authors' network over e-mail, in person, and through LinkedIn. The demographics are the following: eight subjects were male, one was female; six of them had a Ph.D. degree, two of them an M.S. degree, and one, a B.S. degree; eight subjects are Aerospace Engineers, while one was a Designer. Finally, six of them have direct experience in conceptual satellite design, while three of them did not.

The procedure for the experiment consisted of three steps: first, the subjects completed a 20-min tutorial during which they familiarized themselves with the Daphne system. Then, subjects were asked to solve a mission design task in two conditions (independent variable): 1) using a version of Daphne loaded with all the capabilities described in the previous sections except the Data Mining ones, named Daphne-VA from now on, and 2) using a version of Daphne implementing a baseline tradespace exploration tool only containing the interactive scatter plot, a Design Builder, and the data mining capabilities (scatter plot, filters, and feature trees), named Daphne-DM from now on. Our idea is that Daphne-DM provides an experience to the test subjects more akin to that of other tools in the field, while Daphne-VA provides access to all of the new features we developed that are not present in other tools. This is done to get a clearer comparison between a tool that test subjects are used to versus one where they are not.

The experimental design was within subjects, so each participant solved a different instance of the same task in each condition. The order of conditions and tools was randomly chosen for each subject, in order to minimize learning effects. Each task had different performance models to minimize learning effects. Care was taken so that the two problem instances were of similar difficulty, yet different enough to minimize learning effects. All interactions of users with Daphne (e.g., clicking buttons) were recorded, as well as all questions and answers of the dialogue between the subject and Daphne. At the end of each task, users were asked to do a short test to measure learning and respond

TABLE I
CANDIDATE INSTRUMENTS

Instrument	Description
VIIRS	Visible and Infrared Atmospheric Sounder
CMIS	Conically Scanning Microwave Radiometer (Imaging and Sounding)
SMAP_RAD	L-band Synthetic Aperture Radar
SMAP_MWR	L-band Radiometer
BIOMASS	P-band Synthetic Aperture Radar

TABLE II
CANDIDATE ORBITS

Orbit	Description
LEO-600-polar	LEO with 90deg inclination at 600km altitude
SSO-600-AM	SSO with morning LTAN at 600km altitude
SSO-600-DD	SSO with dawn-dusk LTAN at 600km altitude
SSO-800-AM	SSO with morning LTAN at 800km altitude
SSO-800-DD	SSO with dawn-dusk LTAN at 800km altitude

SSO = Sun-synchronous orbit; LTAN = Local time of the ascending node.

to a usability survey. At the end of both tasks, a semi-structured survey was conducted to gather additional feedback and insights.

B. Task Details

The task is based on an existing problem developed by the authors to design a DSM for monitoring soil moisture [62]. Specifically, subjects were given a set of five candidate instruments (see Table I) and five orbits (see Table II), and they were asked to create DSM designs by combining different sets of instruments and orbits, with no constraints: Each instrument can, in principle, be assigned to any subset of the orbits, including none or all of them. The VASSAR engine described earlier was used to assess the scientific benefit and cost of the architectures [9]. The goal of the task is for users to find the best possible set of designs, in terms of science and cost. Specifically, they were asked to interact with each tool (Daphne-VA and Daphne-DM) and come up with a set of satellite mission designs, which were as close to the "true" science-cost Pareto front as possible within a cost range ranging from $800 M to $4000 M.

C. Dependent Variables

1) Performance: Since the actual Pareto front for this problem is not known, an approximate "true Pareto front" was obtained by running a multiobjective GA [85], with an initial population size of 500 different architectures and a maximum of 20 000 generations. It can be seen, with normalized values for science and cost, in Fig. 5. Distance to the "true" Pareto front was determined through the relative improvement in hypervolume (HV), which is computed through (2), where HV is a well-known metric in multiobjective optimization that computes the volume of a dominated set of designs. The reason for creating such a metric is being able to compare problems that are of similar difficulty, but have slightly different starting, ending, and "optimal" HV measures. HV improvement metric as defined below is a large-is-better metric bounded between 0 (no improvement in HV with respect to the initial population) and 1 (same final HV as the "true" Pareto front)

$$\mathrm{HVI} = \frac{(\mathrm{HV}_{\mathrm{end}} - \mathrm{HV}_{\mathrm{start}})}{(\mathrm{HV}_{\mathrm{optimal}} - \mathrm{HV}_{\mathrm{start}})}. \tag{2}$$

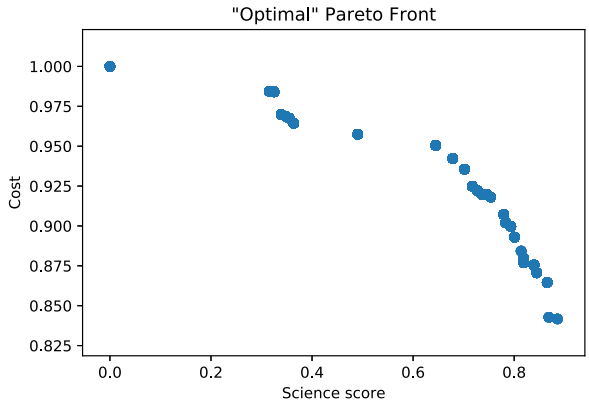

Fig. 5. Plot of the "true" Pareto front found by the GA.

TABLE III
LIST OF INITIAL QUESTIONS ASKED IN THE INTERVIEW

Question
What was your first impression of the interface?
What strategies did you use (if any) to solve the problem with/without the cognitive assistant?
Do you think you had all the tools needed to complete the task? (If not, which ones did you miss?)
What did you like the most about the interface? (list 2-3 things) And the least? (2-3) What could we improve?
How difficult did you find the task? Did you have enough time?
Do you find Daphne useful? For what tasks? (e.g., in a Team-X session)
Which role (Historian, Analyst...) gave the most useful feedback?
What other kinds of questions do you wish Daphne could answer?
Any other feedback?

2) Human Learning: To assess how much users learned about the problem at hand and underlying model by interacting with the tools, they were asked to do a test at the end of each task that consisted of ten questions. Each question asked them to choose among two designs with the same cost, the one they thought had the largest science score as predicted by the model. The use of tests to assess learning is commonplace in the education literature [87], [88]. Specifically, this test is designed to assess the ability of the user to *predict* the outputs of the model. While this is clearly a limited view of human learning, which has multiple dimensions as stated for example in Bloom's taxonomy, prediction is a reasonable choice given the time constraints, as it is a higher-level cognitive task that requires integrating multiple pieces of knowledge learned before [89].

3) Usability: A standard usability survey was conducted after each task, namely the system usability scale (SUS) [90]. The SUS survey consists of ten Likert items listed in the Appendix. The SUS instrument has been validated and has been widely used to assess software usability including for intelligent systems [91]–[93].

D. Exit Interview

An exit interview was conducted after the experiment. The interview was semi structured, consisting of several open-ended questions shown in Table III about their experience with the interface. More in-depth follow-up questions were asked after those, depending on the answers of each subject.

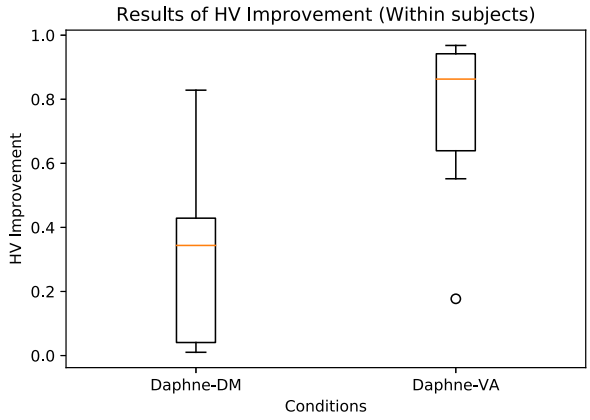

Fig. 6. Results of the within subjects comparison.

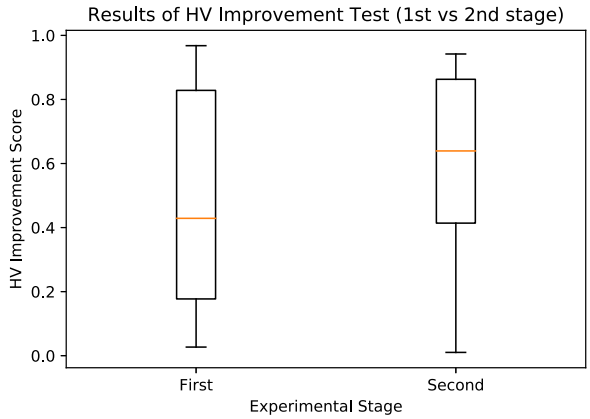

Fig. 7. Results of the learning effects comparison.

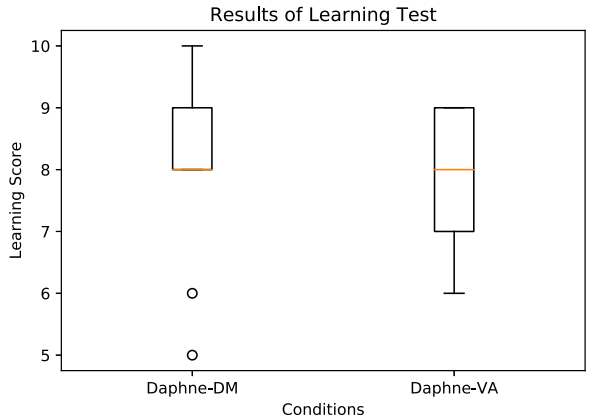

Fig. 8. Results of the learning test.

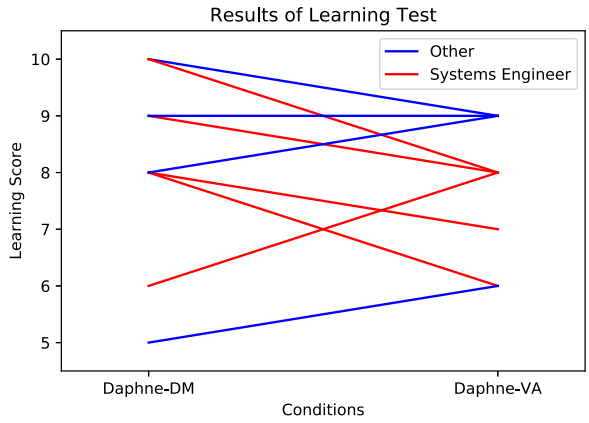

Fig. 9. Results of the learning test for each user.

E. Results

1) H1. Daphne Helps Engineers Find Better DSM Designs: To test Hypothesis H1, we performed a within-subjects comparison. The results are shown in Fig. 6. The paired t-test for the null hypothesis of equal performance between the two Daphne versions has a p-value of 0.00433. (Normality tests were performed). Thus, we observe a significant improvement in performance in the Daphne-VA condition. Specifically, test subjects using Daphne-DM have a mean HV Improvement score of 0.317, while those that used Daphne-VA have a mean of 0.766. The difference of scores is 0.449, or almost half the distance between the initial dataset and the "optimal" Pareto front that was found by running a GA for 5 h 30.

To ensure this result is not due to learning effects between stages, we checked for significant differences in the performance of the first stage versus the second one. The results are shown in Fig. 7. In this case, the paired t-test with null hypothesis of equal performance has a p-value of 0.378, which is not significant. Thus, there are no significant learning effects.

2) H2. The Cognitive Assistant Does Not Hinder Human Learning: Our next hypothesis is that learning will not be affected when using Daphne-VA. The results of the learning survey are summarized in Fig. 8. In this case, the paired t-test with null hypothesis of equal number of questions gives a p-value of 0.710. Thus, there is no significant difference in learning in either condition, which provides support for H2. We note that in both versions, the test mean results are well over two standard deviations compared to randomly answering the questions.

We also compared test scores between systems engineers (five subjects) versus participants with other positions (four subjects). The results are shown in Fig. 9. It can be seen that there are no clear trends.

3) H3. The Cognitive Assistant Does Not Reduce Usability: The final hypothesis for our experiment is that the usability scores will not be significantly different between versions of the tool, and we are testing this through the SUS score. The results of this survey are shown in Fig. 10. The statistical test yields a p-value of 0.411, so the results are not significant, thus providing support for H3. However, we do see a trend of Daphne-VA being a little better than the traditional tradespace exploration tool.

4) Exit Interview: A summary of the feedback gathered from the interviews is presented in the following paragraphs. To begin with, when asked about the general feeling about the interface,

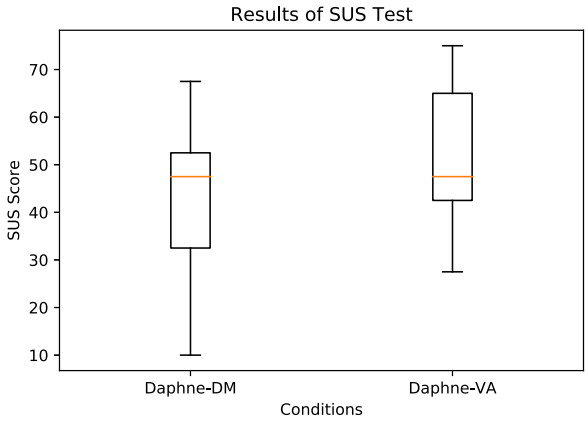

Fig. 10. Results of the SUS scale.

the general opinion was that it was a well-designed interface, clean and easy to understand.

The strategies being used to solve the task with Daphne were mixed, with no clear patterns. When users used Daphne-DM, six out nine found the tree-based representation of the features hard to understand, so they relied mostly on the scatter plot, the filters, and the design builder to explore the dataset, while the other three relied a lot on the Data Mining feature to solve the problem. Strategies for Daphne-VA, meanwhile, revolved around the GA, mentioning how the GA can be a double-edged sword, as it helps find better designs, but, at the same time, hinders the learning of the dataset structure. Most of them used mostly the Critic (which responds to "What do you think of this design?"), ignoring many of the other abilities of Daphne. Some users still resorted to just basic random exploration, as they thought it was faster than waiting for the Critic's feedback.

Difficulty wise, five out of nine of the test subjects thought the task was easy enough, while six out of nine said more time would have made the experience better. Asked on what they would improve from the experience, we got a lot of different opinions. Some recurring ones are that the tutorial for the experiment would have been much better if done live or in video format, the models for cost and performance should justify themselves better, and that some of the Critic feedback could be made better by generalizing some answers to instrument types and taking into account what has already been said.

As far as interface improvements and limitations go, test subjects pointed out the following.

1) The way to ask questions should be revamped by means of an autocomplete feature.
2) The feature trees should be shown in the design builder window.
3) Some way to remove unimportant points from the dataset would be helpful.
4) The choice of colors in the scatter plot could be better and more contrasting.
5) The relationship between different functionalities should be more transparent.
6) The changes in different parts of the interface should be more telegraphed by means of animation or changes in color or font, so it is easier to notice something changed when they scroll to it.

When asked what changes would be needed for Daphne to be useful in the context of JPL, the main answer was to have easy traceability of the science and cost models. While the current models are already traceable, test subjects found it hard to make sense of most of the internal information, and they mentioned that having a clear understanding of this information is vital to their job as decisions need to be justified to humans. The ways to do so can range from graphical representations improving on the already existing Details panels, to summary statistics, new plots, or a way to discover what differences make an architecture better than any other one. They also mentioned the ability to change the problem configuration, models, and assumptions easily online, as requirements can and will change fast in an architectural study environment.

Finally, when asked about new questions for Daphne to answer, there was a wide range of recommendations. In relation to the Historian, test subjects were interested in learning why (and not just whether) something has or has not been done before. Questions for the Data Mining role include asking what instruments go best together, and knowing which are the most similar architectures in the dataset to the current one. Finally, users were interested in asking questions about sets of point designs, and not only a single architecture.

F. Discussion

The results from the study support our three hypotheses. First, Daphne-VA does significantly improve the quality of the generated designs: The mean score improvement, when compared to Daphne-DM, is around half of the maximum improvement a computer found by working on the problem for 5 h and 30 min. Following that Daphne-VA does improve the quality of the designs generated, we were interested in understanding how the improvement comes about. To shed some light into this question, we plotted the evolution of the HV Improvement metric for each user in terms of both the number of function evaluations (NFE) and the elapsed time of the experiment. NFE measures how many times the objective function (i.e., the function that computes the scores for an architecture) is called throughout the optimization. The number of NFE required to achieve convergence (e.g., to achieve a certain target value for HV) is a common metric to compare the performance of different optimization algorithms, since unlike run time, it is independent of the computing hardware used and the computational expense of the objective function itself. The results are provided in Fig. 11(a) and (b).

If we compare the results between Daphne-DM and Daphne-VA time wise, we can see different trends. Regarding Daphne-VA, the early advantage in performance is due to the designs found by the Background Search algorithm. The great jump in performance seen in some of the users of the VA around the 5-min mark is mainly due to them using the responses of the

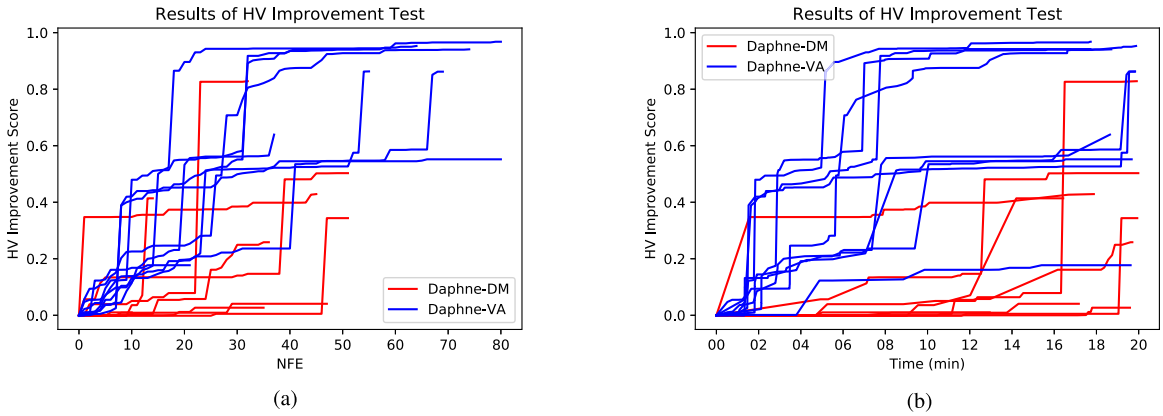

Fig. 11. HV Improvement for each user. (a) HV Improvement as a function of NFE. (b) HV Improvement as a function of time.

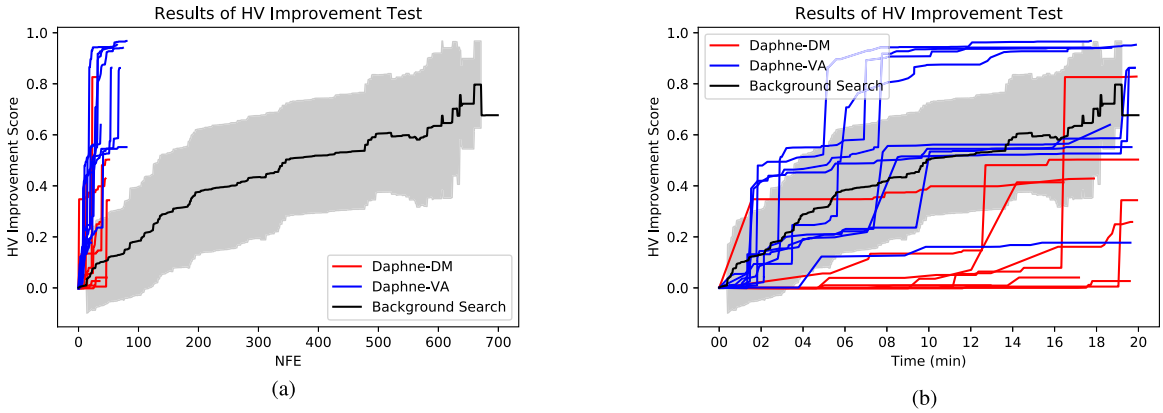

Fig. 12. HV Improvement for each user compared to GA. (a) Comparison as a function of NFE. (b) Comparison as a function of time.

Critic question ("What do you think of this design?") to improve further on the designs found by Daphne. The rest of the users of the VA relied mostly on the Background Search to get results, and did not ask as many questions to the system as the first group, resulting in lower task performances. As for the Daphne-DM results, the late performance improvements for most users match with their perceived confusion with the tree-based feature representation in the Data Mining, which resulted in many users not evaluating many new architectures for the first half of the experiment or just resorting to random exploration, with mixed results.

One potential criticism of the comparison in Fig. 12 is that while the human with Daphne-VA obtains better search performance than the human with Daphne-DM, it is unclear whether it outperforms Daphne by itself. To address this question, we plotted the user results against a GA running for the same time on the same problems to compare the user performance with Daphne to that of a GA running for the same amount of time (during which it can evaluate many more architectures than the human with Daphne). These results can be seen in Fig. 12(a) and (b). The human-Daphne combination systematically obtains better performance than Daphne (the GA) alone when the comparison is on a per NFE basis. When an isotime comparison is done, we observe that the test subjects using Daphne-VA can still do better than the GA about half of the times, even though that the GA evaluates many more architectures during that time. The differences in performance within the Daphne-VA group can, as already mentioned, be explained by some users only relying mostly on the GA to obtain the results, while some others used the full extent of the Daphne features, such as the responses from the VA, and specially the Critic responses. It is important to note that the isotime comparison depends on how long it takes to evaluate an architecture, and the longer it takes to evaluate, the more relevant the per-NFE comparison becomes.

There are several limitations of this paper. First, we believe the way we are testing for learning is not ideal. Being based on ten binary questions, the test does not provide much power. In addition, we have observed significant variance in this and other experiments using similar tests [94]. Another limitation is that we are only testing Daphne with all its features versus a very limited version. So we do not know which roles are the ones contributing the most to the improvements we are seeing, other than from anecdotal evidence based on the interview feedback, and thus we do not have quantitative guidance on where we

should focus our efforts next if we want to maximize the return on investment on the system.

Given all the results from the experiment, we have gathered a set of recommendations for future development of VAs for engineering design.

1) We have observed how the VA could reduce human learning and understanding of the problem at hand. Future VAs should take this into account to ensure users of the VA can increase task performance without sacrificing learning, which could be a proxy for future task performance.
2) Mixed-initiative features should be explored by turning the VA into a "Virtual Peer," meaning the VA should be able to listen to conversations and chip in with meaningful contributions, apart from the features we already described in this article.
3) The vocabulary used by the VA should be similar to that of the target population, and this involves specializing both a) the speech-to-text (STT) system to understand specialized vocabulary, and b) the answers generated by the system.
4) Natural language interactions should preferably be used to answer questions that are harder to represent or query by traditional means, such as suggestions for improvement. Traditional representations for things, such as performance and cost models, seem to work better for users.
5) Simple is better, at least as far as the VA feedback goes. When answers from Daphne get too long or are too difficult to interpret (such as some feature trees), even professional users will get lost and stop trying to understand them.

While some of this could be alleviated with proper training, to maximize performance, feedback from the system should be as simple as possible.

We have also gathered a set of recommendations for performing experiments in the field of VAs for design.

1) The most important recommendation is that researchers and developers should strive to test the system with expert users, as we have found that their feedback, methods, and usage patterns differ greatly to those of an undergrad student population. Results of pilot experiments with a student population, even if those students are of the same field, are not indicative of the actual usage patterns of the system in the real world. Feedback, both in the form of interviews and scales, such as SUS is much more strict and unforgiving in the professional community.
2) New methods should be studied to measure human learning in design space exploration (we found no significant quantitative differences in our study, whereas many users mentioned in the interview that they learned less with Daphne-VA).
3) Special care needs to be put in designing tasks and experiments to avoid learning effects. Creating different problems of similar difficulty is a difficult task, but one that is essential is to ensure validity. In pilot experiments, we had both problems that were too similar, which led to significant learning effects, and problems that were too different, so the results could not be compared at all. Proper experimental design and the use of relative metrics, such as HV Improvement, help reduce some of those differences, but care should still be put in the actual problem design.

Finally, given all the feedback we obtained from JPL, we believe Daphne and other VAs for design have an opportunity to drastically improve early mission design. The fact that performance clearly improves when combining everything together is a strong signal that such a system can help improve the performance of professionals exploring new mission designs.

V. Conclusion

This article described the architecture of, and an experiment to validate, a VA to architect Earth Observation DSM. Most VAs so far have been developed for commercial general usage. While there have been a few instances of VAs developed for aerospace and design tasks, Daphne is, to the best of our knowledge, the first VA to specifically support the task of architecting DSM. The system is completely open source and publicly available (https://github.com/seakers/daphne_brain).

The aim of Daphne is to help system engineers reduce their cognitive load when exploring large tradespaces for DSMs by providing them with easier and timely access to relevant information. Daphne's different roles support this task in different ways, and accept inputs both in natural language and more classical interactions.

The article also described a validation study where nine JPL engineers worked on a DSM tradespace exploration task with two versions of Daphne (within-subjects experiment): One with all its features, and another one that looked more like traditional tradespace exploration tools. We measured the performance in the task as the hypervolume of the designs generated, and measured human learning and usability at the end of each task using a test and the standard usability survey, respectively. The results provided support for our hypotheses that Daphne with all its capabilities can help engineers find better and more diverse DSM designs without sacrificing learning or usability. Moreover, we showed that the user-Daphne collaboration outperformed both the human with a traditional tool and Daphne alone, even (for some subjects) when the comparison is done on an isotime basis.

In terms of future work on Daphne, we have identified several limitations that need to be addressed, both in the system and the testing that we have performed to date. From the feedback we gathered during the interviews post experiment, most people asked for better ways to foster innovation using Daphne. One common suggestion is the ability to redefine aspects of the problem formulation during the tradespace search exercise, without having to change code or text files and rerun Daphne (e.g., to add a new instrument online). We also want to continue exploring the mixed-initiative functionalities of Daphne. Several subjects shared our vision of Daphne as a team mate during mission architectural and concept studies; one who listens to the conversation in real time and chip in with suggestions as appropriate.

Another major point of improvement concerns human learning. In addition to improving task performance, we would like Daphne to not only hinder learning, but increase it. As we

TABLE IV
LIST OF QUESTIONS IN THE SUS (ANSWERS RANGE FROM 1 - STRONGLY DISAGREE TO 5 - STRONGLY AGREE)

Question
Q1 – I think that I would like to use Daphne frequently.
Q2 – I found Daphne unnecessarily complex.
Q3 – I thought Daphne was easy to use.
Q4 – I think that I would need the support of an expert to be able to use Daphne.
Q5 – I found the various functions in Daphne were well integrated.
Q6 – I thought there was too much inconsistency in Daphne.
Q7 – I would imagine that most people would learn to use Daphne very quickly.
Q8 – I found Daphne very cumbersome to use.
Q9 – I felt very confident using Daphne.
Q10 – I needed to learn a lot of things before I could get going with Daphne.

develop new strategies to do that, we will need to develop and try new measures of learning that take into account a broader set of learning dimensions. Finally, we plan to conduct additional experiments with a similar population to test our new learning instruments as well as determine the effects of different skills on performance and learning.

APPENDIX
ADMINISTERED SUS QUESTIONNAIRE

Table IV contains a list of the questions administered to the test subjects as part of the SUS.

REFERENCES

[1] H. Bang, A. Viros, A. Prat, and D. Selva, "Daphne : An intelligent assistant for architecting earth observing satellite systems," in *Proc. AIAA SciTech*, 2018, pp. 1–14.

[2] A. V. Martin and D. Selva, "From design assistants to design peers: Turning daphne into an AI companion for mission designers," in *Proc. AIAA Scitech Forum*, 2019, p. 0402.

[3] D. Selva, A. Golkar, O. Korobova, I. L. i Cruz, P. Collopy, and O. L. de Weck, "Distributed and federated satellite systems: What is needed to move forward?" *J. Aerosp. Inf. Syst.*, vol. 14, no. 8, pp. 412–438, 2017.

[4] P. Garcia Buzzi, D. Selva, N. Hitomi, and W. J. Blackwell, "A survey and assessment of the capabilities of Cubesats for Earth observation," *Acta Astronautica*, vol. 74, pp. 50–68, 2019.

[5] National Aeronautics and Space Administration, "NASA Technology Roadmaps: Introduction, Crosscutting Technologies, and Index," National Aeronautics and Space Administration, Washington, D.C., USA, Jul. 2015. [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/2015_nasa_technology_roadmaps_ta_0_introduction_crosscutting_index_final_0.pdf

[6] National Aeronautics and Space Administration, "NASA Technology Roadmaps TA 11: Modeling, Simulation, Information Technology, and Processing," National Aeronautics and Space Administration, Washington, D.C., USA, May 2015. [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/2015_nasa_technology_roadmaps_ta_11_modeling_simulation_final.pdf

[7] S. Nag, S. P. Hughes, and J. Le Moigne, "Streamlining the design tradespace for earth imaging constellations," in *Proc. AIAA Space*, 2016, pp. 1–17. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2016-5561

[8] J. Le Moigne *et al.*, "Tradespace analysis tool for designing constellations (tat-c)," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2017, pp. 1181–1184.

[9] D. Selva, "Knowledge-intensive global optimization of Earth observing system architectures: A climate-centric case study," in *Proc. SPIE Remote Sens.*, vol. 9241, 2014, pp. 1–22.

[10] R. E. Thompson, J. M. Colombi, J. Black, and B. J. Ayres, "Disaggregated space system concept optimization: Model-based conceptual design methods," *Syst. Eng.*, vol. 18, no. 6, pp. 549–567, 2015.

[11] D. C. Engelbart, "Augmenting human intellect: a conceptual framework," Stanford Research Institute, Washington D.C., Tech. Rep. AFOSR-3233, 1962.

[12] Clinc AI, "Clinc." (2019). [Online]. Available: https://clinc.com/. Accessed on: Nov. 5, 2019.

[13] G. Weiss, *Multiagent Systems*, 2nd ed. Cambridge, MA, USA: MIT Press, 2013. [Online]. Available: https://mitpress.mit.edu/books/multiagent-systems-0

[14] K. Myers, P. Berry, J. Blythe, K. Conley, and M. Gervasio, "An intelligent personal assistant for task and time management," *AI Mag.*, vol. 28, no. 2, pp. 47–62, 2007. [Online]. Available: http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/2039

[15] J. Grant, S. Kraus, and D. Perlis, "A logic-based model of intention formation and action for multi-agent subcontracting," *Artif. Intell.*, vol. 163, no. 2, pp. 163–201, 2005.

[16] J. McDermott, "R1: A rule-based configurer of computer systems," *Artif. Intell.*, vol. 19, no. 1, pp. 39–88, 1982.

[17] C. C. Hayes, A. K. Goel, I. Y. Tumer, A. M. Agogino, and W. C. Regli, "Intelligent support for product design: Looking backward, looking forward," *J. Comput. Inf. Sci. Eng.*, vol. 11, pp. 1–9, 2011. [Online]. Available: http://computingengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1402355

[18] A. K. Goel, S. Vattam, B. Wiltgen, and M. Helms, "Cognitive, collaborative, conceptual and creative - Four characteristics of the next generation of knowledge-based CAD systems: A study in biologically inspired design," *Comput. Aided Des.*, vol. 44, no. 10, pp. 879–900, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.cad.2011.03.010

[19] W. C. Regli, S. Szykman, and R. D. Sriam, "The role of knowledge in next-generation product development systems," *J. Comput. Inf. Sci. Eng.*, vol. 1, pp. 3–11, 2001.

[20] W. Song, A. Keane, J. Rees, A. Bhaskar, and S. Bagnall, "Turbine blade fir-tree root design optimisation using intelligent cad and finite element analysis," *Comput. Structures*, vol. 80, no. 24, pp. 1853–1867, 2002.

[21] K. L. Wood, R. B. Stone, D. Mcadams, J. Hirtz, and S. Szykman, "A functional basis for engineering design: Reconciling and evolving previous efforts," *Res. Eng. Des.*, vol. 13, pp. 65–82, 2002.

[22] Y. Kitamura, M. Kashiwase, M. Fuse, and R. Mizoguchi, "Deployment of an ontological framework of functional design knowledge," *Adv. Eng. Inform.*, vol. 18, no. 2, pp. 115–127, 2004.

[23] G. Stump, S. Lego, M. Yukish, T. W. Simpson, and J. A. Donndelinger, "Visual Steering Commands for Trade Space Exploration: User-Guided Sampling With Example," *J. Comput. Inf. Sci. Eng.*, vol. 9, no. 4, pp. 1–10, 2009.

[24] S. Watanabe, Y. Chiba, and M. Kanazaki, "A proposal on analysis support system based on association rule analysis for non-dominated solutions," in *Proc.IEEE Congr. Evol. Computation*, 2014, pp. 880–887.

[25] X. Yan, M. Qiao, T. Simpson, J. Li, and X. Zhang, "Work-centered visual analytics to support multidisciplinary design analysis and optimization," in *Proc. 12th AIAA Aviation Technol., Integr., Operations Conf. 14th AIAA/ISSM*, 2012, pp. 1–12.

[26] H. Bang and D. Selva, "iFEED: Interactive feature extraction for engineering design," in *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2016, pp. 1–11.

[27] M. C. Fu, C. C. Hayes, and E. W. East, "Sedar: Expert critiquing system for flat and low-slope roof design and review," *J. Comput. Civil Eng.*, vol. 11, no. 1, pp. 60–68, 1997.

[28] S. A. Guerlain *et al.*, "Interactive critiquing as a form of decision support: An empirical evaluation," *Human Factors*, vol. 41, no. 1, pp. 72–89, 1999.

[29] W. Peng and J. Gero, "Computer-aided design tools that adapt," *Comput.-Aided Architectural Des. Futures*, 2007, pp. 417–430. [Online]. Available: http://link.springer.com/content/pdf/10.1007/978-1-4020-6528-6_31.pdf

[30] J. Eddy and K. E. Lewis, "Visualization of multidimensional design and optimization using cloud visualization," *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2002, pp. 899–908.

[31] G. M. Stump, M. Yukish, T. W. Simpson, and E. N. Harris, "Design space visualization and its application to a design by shopping paradigm," in *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2003, pp. 795–804. [Online]. Available: http://dx.doi.org/10.1115/DETC2003/DAC-48785

[32] P. W. Chiu and C. L. Bloebaum, "Hyper-Radial Visualization (HRV) method with range-based preferences for multi-objective decision making," *Structural Multidisciplinary Optim.*, vol. 40, no. 1–6, pp. 97–115, 2010.

[33] P. Chiu and C. Bloebaum, "Visual steering for design generation in multi-objective optimization problems," in *Proc. 47th AIAA Aerosp. Sci. Meeting*, 2009, pp. 1–14. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/6.2009-1167

[34] N. Knerr and D. Selva, "Cityplot: Visualization of high-dimensional design spaces with multiple criteria," *J. Mech. Des.*, vol. 138, no. 9, pp. 1–53, 2016.

[35] S. Watanabe, Y. Chiba, and M. Kanazaki, "A proposal on analysis support system based on association rule analysis for non-dominated solutions," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 880–887.

[36] G. Cervone, P. Franzese, and A. P. Keesee, "Algorithm quasi-optimal (AQ) learning," *Wiley Interdisciplinary Rev.: Comput. Statist.*, vol. 2, no. 2, pp. 218–236, 2010.

[37] A. Newell, *Human Problem Solving*. Upper Saddle River, NJ, USA: Prentice-Hall, 1972.

[38] N. W. Hirschi and D. D. Frey, "Cognition and complexity: An experiment on the effect of coupling in parameter design," *Res. Eng. Des.*, vol. 13, pp. 123–131, 2002.

[39] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, *Mastering The Information Age Solving Problems with Visual Analytics*. Geneve, Switzerland: Eurographics Assoc., Jan. 2010.

[40] M. Freed *et al.*, "RADAR : A personal assistant that learns to reduce email overload," in *Proc. 23rd AAAI Conf. Artif. Intell.*, 2008, pp. 1287–1293.

[41] DARPA, "PAL - The PAL framework." (2011). [Online]. Available: https://pal.sri.com/. Accessed on: Nov. 5, 2019.

[42] A. Cheyer and D. Martin, "The Open Agent Architecture," *Auton. Agents Multi-Agent Syst.*, vol. 4, no. 1–2, pp. 143–148, 2001. [Online]. Available: https://doi.org/10.1023/A:1010091302035

[43] A. Cheyer, J. Park, and R. Giuli, "IRIS: Integrate. Relate. Infer. Share," in *Proc. Int. Conf. Semantic Desktop Workshop*, Aachen, Germany, 2005, pp. 59–73. [Online]. Available: http://dl.acm.org/citation.cfm?id=2890006.2890011

[44] D. A. Ferrucci, "Introduction to this is watson," *IBM J. Res. Develop.*, vol. 56, no. 3.4, pp. 1–15, 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6177724/

[45] IBM, "IBM Watson products and services." (2019). [Online]. Available: https://www.ibm.com/watson/products-services/. Accessed on: Nov. 5, 2019.

[46] Wolphram Alpha LLC, "Wolfram|Alpha: Computational Intelligence." (2019). [Online]. Available: http://www.wolframalpha.com/. Accessed on: Nov. 5, 2019.

[47] Apple, "Siri - Apple." (2019). [Online]. Available: https://www.apple.com/siri/. Accessed on: Nov. 5, 2019.

[48] Google, "Google Assistant, your own personal Google." (2019). [Online]. Available: https://assistant.google.com/. Accessed on: Nov. 5, 2019.

[49] Microsoft, "Personal Digital Assistant - Cortana Home Assistant - Microsoft." (2019). [Online]. Available: https://www.microsoft.com/en-us/cortana. Accessed on: Nov. 5, 2019.

[50] Amazon, "Amazon Alexa - Build for Amazon Echo Devices." (2019). [Online]. Available: https://developer.amazon.com/alexa. Accessed on: Nov. 5, 2019.

[51] Mycroft AI Inc., "Mycroft - Open Source Voice Assistant - Mycroft." (2019). [Online]. Available: https://mycroft.ai/. Accessed on: Nov. 5, 2019.

[52] J. Hauswald *et al.*, "Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers," in *Proc. 20th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2015, pp. 223–238. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2694344.2694347

[53] P. Baudiš, "YodaQA : A modular question answering system pipeline," in *Proc. 19th Int. Student Conf. Elect. Eng.*, 2015, pp. 1156–1165. [Online]. Available: http://ailao.eu/yodaqa/yodaqa-poster2015.pdf

[54] D. Wang and E. Nyberg, "CMU OAQA at TREC 2016 LiveQA : An attentional neural encoder-decoder approach for answer ranking," in *Proc. Text REtrieval Conf.*, 2016, pp. 1–6.

[55] K. Koile, *An Intelligent Assistant for Conceptual Design*. Dordrecht, The Netherlands: Springer, 2004, pp. 3–22.

[56] P. Floss and J. Talavage, "A knowledge-based design assistant for intelligent manufacturing systems," *J. Manuf. Syst.*, vol. 9, no. 2, pp. 87–102, 1990.

[57] L. Mandow and J. L. Perez-De-La-Cruz, "Sindi: An intelligent assistant for highway design," *Expert Syst. With Appl.*, vol. 27, no. 4, pp. 635–644, 2004.

[58] K. Grace, M. L. Maher, D. Wilson, and N. Najjar, "Personalised specific curiosity for computational design systems," in *Des. Comput. Cognition*, 2017, pp. 593–610. [Online]. Available: http://link.springer.com/10.1007/978-94-017-9112-0

[59] T. McCaffrey and L. Spector, "An approach to human–machine collaboration in innovation," *Artif. Intell. Eng. Des., Anal. Manuf.*, 2017, pp. 1–15. [Online]. Available: https://www.cambridge.org/core/product/identifier/S0890060416000524/type/journal_article

[60] A. Berquand *et al.*, "Towards an artificial intelligence based design engineering assistant for the early design of space missions," in *Proc. 69th Int. Astronaut. Congr.*, 2018.

[61] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, CA, USA: O'Reilly Media, 2015.

[62] D. Selva, B. G. Cameron, and E. F. Crawley, "Rule-Based system architecting of earth observing systems: Earth science decadal survey," *J. Spacecraft Rockets*, vol. 51, no. 5, pp. 1505–1521, 2014. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.A32656

[63] CEOS, "The CEOS database: Mission, instruments and measurements." (2017). [Online]. Available: http://database.eohandbook.com/. Accessed on: Nov. 5, 2019.

[64] N. Hitomi and D. Selva, "Incorporating expert knowledge into evolutionary algorithms with operators and constraints to design satellite systems," *Appl. Soft Comput.*, vol. 66, pp. 330–345, 2018.

[65] N. Hitomi, H. Bang, and D. Selva, "Adaptive knowledge-driven optimization for architecting a distributed satellite system," *J. Aerosp. Inf. Syst.*, vol. 15, no. 8, pp. 485–500, 2018.

[66] Pivotal Software Inc., "Messaging that just works - RabbitMQ," (2019). [Online]. Available: https://www.rabbitmq.com/

[67] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, 1993. [Online]. Available: http://portal.acm.org/citation.cfm?doid=170036.170072

[68] Explosion AI, "spaCy -industrial-strength natural language processing in Python." (2019). [Online]. Available: https://spacy.io/. Accessed on: Nov. 5, 2019.

[69] NLTK Project, "Natural Language Toolkit." (2019). [Online]. Available: http://www.nltk.org/. Accessed on: Nov. 5, 2019.

[70] Apache Software Foundation, "Apache OpenNLP." (2019). [Online]. Available: https://opennlp.apache.org/. Accessed on: Nov. 5, 2019.

[71] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *J. Advances Inf. Technol.*, vol. 1, no. 1, pp. 4–20, 2010.

[72] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, *arXiv:1404.2188*.

[73] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Aug. 2014, pp. 1746–1751.

[74] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[75] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[76] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015, *arXiv:1503.00075*.

[77] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2016, pp. 1480–1489. [Online]. Available: http://aclweb.org/anthology/N16-1174

[78] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. 27th Int. Conf. Comput. Linguistics*, 2018, pp. 1638–1649.

[79] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[80] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.

[81] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," 2015, *arXiv:1510.03820*. [Online]. Available: http://arxiv.org/abs/1510.03820

[82] P. H. Sellers, "The theory and computation of evolutionary distances: Pattern recognition," *J. Algorithms*, vol. 1, no. 4, pp. 359–373, 1980. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0196677480900164

[83] G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a natural language interface to complex data," *ACM Trans. Database Syst.*, vol. 3, no. 2, pp. 105–147, 1978.
[84] G. G. Hendrix, "Natural-language interface," *Comput. Linguistics*, vol. 8, no. 2, pp. 56–61, 1982.
[85] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[86] R. R. Hoffman, M. Johnson, J. M. Bradshaw, and A. Underbrink, "Trust in automation," *IEEE Intell. Syst.*, vol. 28, no. 1, pp. 84–88, Jan. 2013.
[87] N. E. Gronlund, *Assessment of Student Achievement*. London, England, UK: Pearson, 1998.
[88] C. A. Palomba and T. W. Banta, *Assessment Essentials: Planning, Implementing, and Improving Assessment in Higher Education. (Higher and Adult Education Series).* Hoboken, NJ, USA: Wiley, 1999.
[89] L. W. Anderson *et al.*, "A taxonomy for learning, teaching, and assessing: A revision of blooms taxonomy of educational objectives, abridged edition," White Plains, NY, USA: Longman, 2001.
[90] J. Brooke, "Sus - a quick and dirty usability scale," *Usability Eval. Industry*, vol. 189, no. 194, pp. 4–7, 1996.
[91] D. Ghosh, P. S. Foong, S. Zhang, and S. Zhao, "Assessing the utility of the system usability scale for evaluating voice-based user interfaces," in *Proc. Sixth Int. Symp. Chin. CHI*, 2018, pp. 11–15.
[92] G. Cordasco *et al.*, "Assessing voice user interfaces: The vassist system prototype," in *Proc. 5th IEEE Conf. Cogn. Infocommunications*, 2014, pp. 91–96.
[93] X. Rong, A. Fourney, R. N. Brewer, M. R. Morris, and P. N. Bennett, "Managing uncertainty in time expressions for virtual assistants," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2017, pp. 568–579.
[94] H. Bang, Y. L. Z. Shi, S.-Y. Yoon, G. Hoffman, and D. Selva, "Exploring the feature space to aid learning in design space exploration," in *Design Computing and Cognition*. Cham, Switzerland: Springer, 2018.

Antoni Viros i Martin (Student Member, IEEE) received the B.Sc. degree in aerospace engineering and the B.Sc. degree in computer science from the Universitat Politecnica de Catalunya, Barcelona, Spain, in 2017. He is currently working toward the Ph.D. degree in Virtual Assistants for early mission formulation at the Department of Aerospace Engineering at Texas A&M University, College Station, TX, USA.

Prior to starting Ph.D., he worked as a Researcher with Cornell University, Ithaca, NY, USA, on the same fields for a year and a half. His research interests include virtual assistants and artificial intelligence applications in the aerospace field, and in particular the design of distributed missions for earth observation.

Daniel Selva (Member, IEEE) received the Diplome d'Ingenieur degree from École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, which is equivalent to the M.Eng. degree in aerospace engineering in the US system, in 2004, the Ingeniero Superior de Telecomunicaciones degree, equivalent to the M.Eng. degree in electrical engineering, from Universitat Politecnica de Catalunya, Barcelona, Spain, in 2004, and the Ph.D. degree in aerospace engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2012.

Before doing his Ph.D., he worked for four years in Kourou (French Guiana) as an Avionics Specialist within the Ariane 5 Launch team. He is currently an Assistant Professor of Aerospace Engineering with Texas A&M University, College Station, TX, USA, where he directs the Systems Engineering, Architecture, and Knowledge (SEAK) Lab. His research interests include the application of knowledge engineering, global optimization, and machine learning techniques to systems engineering and architecture, with a strong focus on space systems.

Dr. Selva is a member of the AIAA Intelligent Systems Technical Committee, and the European Space Agency's Advisory Committee for Earth Observation.