From Design Assistants to Design Peers: Turning Daphne into an AI Companion for Mission Designers

Antoni Virós* and Daniel Selva†
Texas A&M University, College Station, Texas, 77843

This paper describes the changes done to Daphne, a virtual assistant for architecting earth observing satellite systems, to turn it from a reactive assistant that only acts when asked by the user into a proactive assistant that can perform actions without not directly linked to a user request, taking its own initiative. Specifically, the paper describes a new way for Daphne to communicate with the user with Websockets that allows for a broader range of interactivity. The new features enabled by this new communication system are: (1) an agent that searches over the space of designs and shows interesting designs to the user; (2) an agent that tries to encourage the user to diversify their search of the tradespace; and (3) a live recommender system that acts when the user is modifying a certain design by suggesting changes that are likely to improve cost and/or performance. The paper also describes changes in the existing sub-systems as well as the interface to accommodate the new systems and interactivity. Finally, the paper has a short discussion on how a common user case scenario would unfold with all these new features.

I. Introduction

Architecting Earth Observing Satellites Systems (EOSS) such as the NASA Earth Observing System or ESA's Copernicus is a challenging task. Some strictly computational factors that explain this difficulty are: 1) The number of possible valid designs can be extremely high –if we consider the architecture space of an EOSS to be defined by any binary relation between a set of N instruments and a set of M orbits (and spacecraft), then there are 2^{NM} possible architectures, which, even for small values of N and M can give too many architectures to evaluate. 2) Each evaluation of a system, where we define evaluation as computing some relevant attributes for it, such as various coverage metrics and cost, can take from seconds to a few hours, depending on the accuracy of the models being used, which makes evaluating all or even more than just a few of all possible architectures unfeasible. 3) The requirements and goals for a system can evolve over time, which would require running the analysis multiple times under various scenarios and requirement sets.

Many organizations have their own set of tools to perform these tasks, although few have publications describing them. Some examples of published papers describing similar tools include TAT-C [1] (in development at the NASA Goddard Space Flight Center), VASSAR [2], iFEED [3, 4], DISCO [5] and Daphne [6], which is the object of this paper.

Daphne was conceived as a virtual assistant –also known in the literature as intelligent personal assistants or cognitive assistants. Other modern examples of virtual assistants include commercial systems such as IBM Watson [7], Siri, Google Assistant, Microsoft Cortana, Amazon Alexa, or Mycroft, as well as research systems such as Lucida [8], YodaQA [9] and OAQA LiveQA [10]. Daphne, in contrast to all these systems, caters only to the systems engineer performing high-level system design or architecting tasks: its job is to ease the cognitive load of the engineer by making relevant information more accessible and providing feedback and recommendations on a specific design when asked.

A recent study conducted by the authors compared two roles of Daphne: a more conventional "assistant" role that only helps by making information retrieval from various data sources easier, and a more innovative approach where the assistant acts more as a "peer", and gives feedback and suggestions on a design when asked by the user. The experiment consisted in having each test subject ($N = 17$ undergraduate students from Cornell University) perform the same task with two different datasets, one with Daphne as an "assistant" and one with Daphne as a "peer". This helps in accounting for the learning effects in the results. The results from that experiment, currently in final stages of preparation for submission, suggest that a human aided by Daphne acting as a "peer" can find better designs than a state-of-the-art genetic algorithm (GA) with the same number of function evaluations. Equivalently, humans assisted by the "peer" can reach the same level of performance in the search (measured by the hypervolume of the Pareto front) as the GA using roughly an order of magnitude fewer evaluations (a mean of 36 evaluations versus 280

*PhD Student, Aerospace Engineering, Student Member AIAA
†Assistant Professor, Aerospace Engineering, Senior Member AIAA

for the GA). Humans with the "assistant" Daphne did not perform as well. The reduction in number of function evaluations becomes more important when each function evaluation takes longer to compute: if an evaluation takes 1 second, a GA can perform 1200 of them in 20 minutes, while humans need some time to think each step, making the reduction in function evaluations futile. On the other hand, there exists a threshold value for the time to execute one function evaluation beyond which the human-agent collaborative approach also outperforms the GA in terms of elapsed time, thanks to the fact that humans need an order of magnitude fewer evaluations than the GA to obtain similar results. In this study, humans evaluated around 2 architectures/minute, so an estimated value for this threshold in our experiment would be 4s. This number comes from computing an "equivalent" number of evaluations between humans and GA $280/36 = 7.78 GA\ eval/human\ eval$ and then computing the time per evaluation for the GA from the time per evaluation of a human $30s/1\ human\ eval \cdot 1\ human\ eval/7.78\ GA\ eval = 3.86s$ Other interesting results from the same experiment include that usability scores (measured using the System Usability Scale (SUS) scale [11]) were similar for the assistant and the peer roles, and that learning effects are important in the experiment, as the results improved drastically when doing the same design task for a second time even if the data was completely different. Finally, it is interesting to note that errors by the "peer" role (i.e., advice provided by the virtual assistant that turns out to lead to worse designs) are penalized much more heavily in terms of user trust compared to errors by the "assistant", and this effect can be seen both in a trust questionnaire and the qualitative feedback given by the test subjects. This suggests that users "expect more" of a "peer" role than they do of an "assistant" role.

The results from this experiment have motivated us to keep researching the peer role of Daphne, with the vision of enabling a truly mixed-initiative approach [12] to design space exploration. This paper describes some of our next steps in that direction, namely how to make Daphne take initiative and act or interject in the dialogue without the user asking Daphne to do so.

Implementing an effective true mixed initiative is challenging, as the agent can easily be perceived as incompetent or annoying, thus decreasing productivity and leading the user to abandon the system. This fear can be seen in most modern assistants, as they never take the initiative over the user; they will only activate when the user calls for them. Agents typically have a wake-up phrase, and they are expected to do nothing unless they hear that phrase. In [13], the authors perform some interviews and conclude that users have much higher expectations of virtual assistants than what they can actually do. Thus, they propose having the virtual assistants reveal their true capabilities and conveying the limitations of the system when communicating with the user. Meurisch et al. [14] come to a similar conclusion, and the proposed solution is to make virtual assistants less reactive, by what they call "anticipatory mobile computing", which can be summarized as having the virtual assistant consider the user goals and act proactively on them. This is a concept that has been investigated in Human Robot Interaction (HRI) for more than a decade, and is just now coming to the field of virtual assistants. [15, 16] They justify taking steps in this direction by saying that anticipating user goals and having models for how the assistant actions change the user behavior can help reduce the gap in expectations. At the same time, they recognize this is still in the very early stages of development, with such important parts as goal detection, pro-activity, and tackling users' fear of loss of control and mistrust still being underdeveloped. Their first steps in developing these kind of assistants can be seen in [17]. Bernard [18], in a broad review of work on virtual assistants, also states that there is a gap between user expectations and virtual assistant capabilities, and that the assistant must be able to reason about user cognition to perform better. Bernard also provides some principles for designing proactive virtual assistants: the assistant should make its intent clear to the user at all times, it should be transparent, and act according to human understandable rules. The paper also makes a point of having a shared context between human and machine to optimize communication.

Other studies have pointed out that more transparency is not always better. In [19], the authors perform an experiment to check how trust in a virtual assistant changes depending on how intelligible its actions are to the user when compared to how sure the assistant is on its results, and whether the assistant is right or wrong. The main result is that intelligibility is only good for a virtual assistant when it is sure of what it is doing. Explaining to the user why it took a bad decision seems to erode the user's trust even more than just being wrong, although the authors mention this could be reversed if the user were allowed to debug the problem.

One early example of an agent that tried to deviate from a purely reactive paradigm and failed is the Clippy assistant from Microsoft Word, which many users disliked, prompting Microsoft to discontinue it. Whitworth [20] argues that software needs to be "polite", and cites Microsoft Clippy as an example of a piece of software that failed at what he perceives as the four requirements for achieving politeness: it did not respect user choice, it did not explain why it suggested some actions, it did not usually offer useful choices, and it was unable to remember past choices from the users. On the other hand, modern examples of virtual assistants that most users like and which take a proactive role are the auto-complete function of Integrated Development Environments (IDEs) or Gmail completing sentences

(or even writing full responses) for the user. Most IDEs use the formal definition of how a programming language is written together with static analysis tools to parse the user's code, and this procedure ends up yielding a list of possible completions for the name the user is typing. Some IDEs even order this list based on previous choices from the user. Armentano [21] describes what makes this kind of agents work: most users only want to be interrupted by a virtual assistant if that interruption is very closely related to the task they are doing, so the assistant must become good at guessing what the user wants to do. Armentano also proposes an approach to this that involves both introducing domain-specific knowledge in the assistant and taking into account user preferences.

We tried to follow those principles to design Daphne's proactive functionalities, which are described in this paper.

The main new feature added to Daphne is running a Genetic Algorithm (GA) in the background, in what we have called the "Explorer" role. Every time Daphne finds some points that improve the Pareto front, it asks the user if he/she wants them added to the current designs dataset. The algorithm being used is the Multi-Objective GA with adaptive operator selection described in [22].

In the same "Explorer" role, another new functionality – which we named "Diversifier" – attempts to improve user performance during the search for new and better designs. According to Gershman's model [23], the user can be doing two kinds of search: local search (or hill-climbing) and global exploration. These two modes map directly to exploitation and exploration from the well-known trade-off in global optimization [24]. Gershman [23] confirms experimentally that a mixed approach has better outcomes than just doing either exploration or exploitation. It also concludes that humans use this mixed approach, and one of the key drivers in choosing to do more exploration versus exploitation is the perceived uncertainty of the rewards. On the last two human experiments performed on Daphne, we have noticed users tend to focus on small areas for optimization, iterating over a similar design for many evaluations, thus losing the benefits of exploration. This is probably partially due to the fact that without deep knowledge of the problem at hand (the subjects were students), it is hard to know how a change will affect objective variables, and given the limited time in the experiments, students tended to focus on exploitation. While some of this behavior may disappear with longer design tasks and/or expert users (which we will test in an upcoming experiment), we hypothesize that a feature that tries to foster diversity during the search task will lead to better results. To implement this feature, Daphne looks at how similar the designs explored by the user are when compared to one another using crowding distance, a well-known diversity metric used in multi-objective optimization algorithms such as NSGA-II [25]. Then, Daphne suggests to the user to explore areas where crowding distance is large.

In addition to the new Explorer and Diversifier roles, a live recommender system has been developed. This system works by giving real-time recommendations to the user when he/she is editing a design. These suggestions can be gathered from multiple roles, including the Analyst, the Historian, the Engineer and the Aggregator. Each role can be thought of as a small program that performs tasks and answers questions that are closely related. For example, the Historian answers questions about past missions, and the Engineer is responsible for evaluating architectures. All these roles will be described in later sections.

The rest of the paper will be organized as follows: Section II includes a short summary of the current architecture of Daphne, the changes that have been done to add the proactive functionalities, and a detailed description on how these new functionalities are implemented; Section III describes a use case scenario of Daphne in the context of architecting a satellite constellation to measure Soil Moisture; finally, Section IV discusses some avenues for future work including potential new features and experiments to validate the usefulness of these new capabilities for Daphne.

II. Updated System Overview

This section describes the changes done to Daphne to implement the proactive functionalities. First of all, a short summary of the current architecture of Daphne is given. After that, the new features are given a thorough description, including the changes done to the Daphne Brain, all the different roles, the new Explorer Role, and the live recommender system.

A. Base System Overview

As described in Figure 1 Daphne is structured as a micro-services system [26]: it has a web front-end, 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*; and a set of roles, which can be understood as software snippets, that use some of the available micro-services –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, in the future, will be able to run in different machines, making Daphne scale horizontally, as the

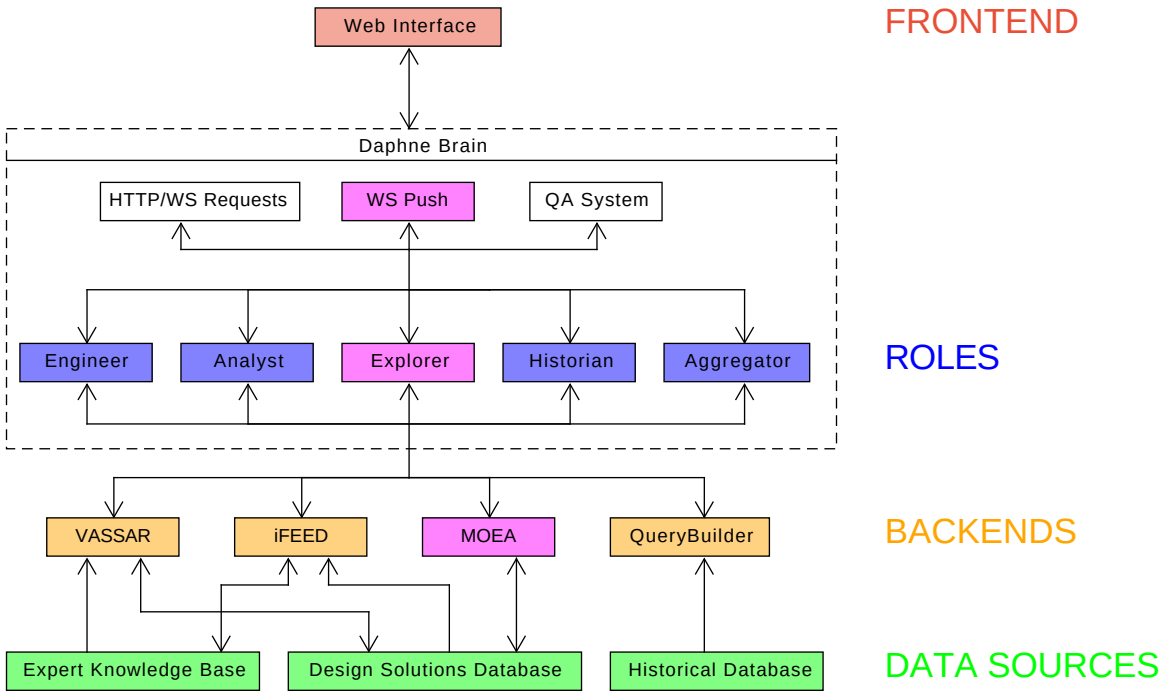

Fig. 1 Daphne architecture - Changes to make it proactive are marked in magenta

micro-services design allows for. Most of the architecture parts not in magenta in Figure 1 (albeit with different names) are described in detail in [6], but short summaries of all of them are provided below for completeness.

The three data sources are: 1) The Expert Knowledge Database, which contains mostly prescriptive domain-specific knowledge about designing Earth observing satellites, in the form of if-then rules. An example of such rules is "IF there is a visual and near infrared instrument, THEN do not put in a dawn-dusk orbit". For more information, the reader is referred to [27]. 2) the Design Solutions Database, which contains the current dataset of architectures and populates the main scatter plot of Daphne. It is also used by different backends such as iFEED, VASSAR or MOEA. 3) The Historical database, which is an SQL database with information about past and planned Earth Observation missions, obtained from the Committee on Earth Observation Satellites (CEOS) online database. This is used to answer questions about past missions, such as which instruments are used to measure specific parameters, or which orbits are most common for certain types of instruments.

At the moment, there are four backends in Daphne. They are used to perform operations on the data sources, and these processed results are then sent back to the *roles*. The VASSAR backend uses the methodology described in [2] to estimate lifecycle cost and various performance-related metrics (e.g., average revisit time, ground spatial resolution) for a given design. These metrics are then used to perform the tradespace exploration. The iFEED backend mines the Design Solutions database to obtain features that are consistently present in desirable designs. These features are then sent to different *roles* so they can be used in different ways, such as suggesting improvements to the user. The QueryBuilder backend takes a natural language question that has been asked by a user and converts it into an SQL query through a custom template system that is able to extract features from the natural language utterance. The same backend is tasked with sending the information back to the QA System so it can process it into the format demanded by the user, be it plots, text, images or lists. Finally, the MOEA (from Multi Objective Evolutionary Algorithm) backend is the responsible for running the design space search background process that is the basis of the Explorer role.

The Daphne Brain is the key component that ties everything together: it forwards requests of any kind from the user to the different *roles*. To do so, it implements two different listeners: one for HTTP requests and one for WebSockets. Requests to the Daphne Brain can be either traditional web requests or natural language commands and queries that go through the QA system. A new addition to it is the WebSocket Push functionality, which is the main enabler for all the proactive actions that Daphne takes: it allows the Brain to send messages to the frontend without the frontend

requesting them. The Brain is failure-proofed, as it can restart itself on failure and it is run in two parallel processes to significantly decrease the probability of it failing. The QA system works by classifying the incoming requests in different types or intents with a Neural Network and then piping that request through a series of transformations and processing steps to end up running some kind of computation or action whose results are shown to the user.

As for the roles, there are five of them: 1) The Engineer, whose job is twofold: it is responsible for handling all evaluations of new architectures by the user, as well as answering questions about the results it shows for these architectures. It can give explanations on why the different objective metrics have their respective values, e.g.: "Why does this design get this cost/science score?". To do that, it queries the VASSAR backend. 2) The Analyst, which helps the user understand the dataset better by providing information such as common features among desirable designs, or, in general, among any given subset of designs (e.g., a cluster of designs obtaining similar performance). It does so by querying the iFEED backend. 3) The Explorer, which is in charge of controlling the background search by the MOEA backend as well as keeping track of the diversity of the designs created by the user. 4) The Historian, which maps natural language queries to a set of database queries for the Historical Database that are preprogrammed into it, and then answers them in the form of plots, text or lists. 5) The Aggregator, which receives a design as input, and gives out suggestions on improvements to it as an output. These suggestions come from different roles and backends – hence the name of Aggregator. These suggestions look at the design in different ways: from an expert perspective (using the Engineer outputs), from an analyst perspective (using the Analyst role), from a standpoint of the search process (using the Explorer role), and from a historical perspective (using the Historian outputs).

Finally, the interface is a web application that allows the user to show or hide all the features available in Daphne, be it asking questions to the roles or simply using Daphne as a more traditional trade-space analysis tool.

B. Daphne Upgrades

To implement the proactive features in Daphne, a few additions have been made to the core architecture, as shown in Figure 1. To make it easier for the reader, all new parts have been colored in magenta. This subsection will go over these changes in detail.

1. Upgrades to the Daphne Brain

The most important change lies in the Daphne Brain. To be able to send messages to the user without the user requesting them first, we have used an interesting feature of WebSockets: their ability to make communication between a web server and client bidirectional. Before WebSockets came into existence, the only way to interact with a web page was through HTTP Requests, in which the client asks for something to the server and the server answers to that request with a response. With the advent of WebSockets, this bidirectional channel of communication used other computing fields was made available to the World Wide Web. Most of the time, this is only used to make what is known as long-polling requests: the user asks for something which takes a lot of computation time, and the server answers as soon as it gets the information without blocking the application. In our case, though, we use this bidirectionality to have the Brain send messages to the interface without any request from the user. We decided to name this process in Daphne "WebSockets Push", making an analogy to the Push Requests used by most smartphones. Most smartphones have a socket connection permanently open between them and a server run by either the phone manufacturer, the OS creator, or both. Through this open channel, apps can send messages (seen as notifications on the phone) at any point, without the user having to go into the app and refreshing it or asking for updates. The concept in Daphne is similar - hence the name.

2. Changes to Roles

Table 1 shows a summary of all roles in Daphne, describing what their basic function is, and their capabilities in the reactive and proactive versions of the role.

The Explorer Role, as already described, performs two tasks: background search on the design space using the MOEA backend and tracking of the diversity of the solutions generated by the user.

To perform the background search, the Explorer starts a Genetic Algorithm in the background as soon as the user logs into the application. The GA used for this functionality is the one described in [22]. At each iteration of the algorithm, the offspring designs are compared against the current non-dominated set and those that improve the current hyper-volume metric are sent to a queue of new designs, which the user can choose to see or not in the interface. By default, these new designs are hidden from the user to avoid unwanted distractions. When the queue grows larger than a set number (in our case, 10 designs), a notification is shown to the user to ask him/her if they want to see all the new

Name	Function	Reactive	Proactive
Engineer	Evaluate new architectures Answer questions about architecture performance and cost	Users can evaluate architectures and ask questions about their scores	Suggestions based on rules of thumb in the Expert Knowledge Base are introduced in the Live recommender system
Analyst	Feature Extraction	User clicks and selects regions to data mine	Tells user about desirable features present or absent in the currently selected architecture. These desirable features are updated periodically from the set on non-dominated architectures
Explorer	Search the design space Foster diversity in the search	N/A	User can control whether or not to see the results from both functionalities, but they keep running in the background even if the user disables the outputs
Historian	Query the Historical Database	User asks questions	Suggestions based on past missions are introduced in the Live recommender system
Aggregator	Puts together a list of potential improvements to a design	User asks how to improve the current design	The Live recommender system is the implementation of the Aggregator role for proactive Daphne

Table 1 Summary table with all roles, showing differences between reactive and proactive versions

designs recently found by the background search. If the user agrees, the designs in the queue are added to the scatter plot, and every time a new design is found, it will also be shown in real time. The user can decide to stop the background search or stop seeing new designs at any time. Figure 2 shows a sequence diagram describing this interaction in more detail.

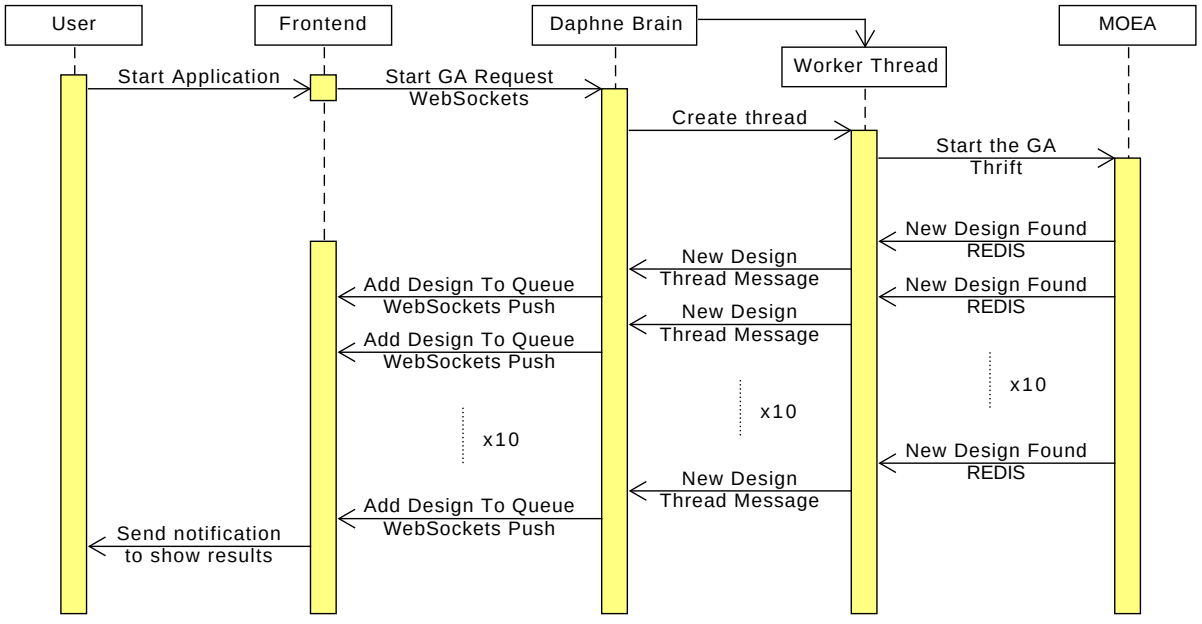

Fig. 2 Background search interaction timeline

As shown in Figure 2, the first request is done asynchronously through WebSockets when the user logs into Daphne, and that triggers the creation of a thread inside the Daphne Brain which will be in charge of handling the actual background search. One could think of that thread as the embodiment of the Explorer. That thread then creates an instance of the MOEA backend and sends the current design set as an input. Immediately afterwards, the same thread starts listening on a Redis (Remote Dictionary Server)* queue. Redis, among other things, is an in-memory data store and a message broker. It can store volatile data accessible through keys and notify listeners of changes in that data. In our use case, we need to be able to save designs from the background search and notify the Daphne Brain of that, and this tool fulfills both needs. The queue where the Daphne Brain thread is listening is where the MOEA instance will keep sending the new architectures it finds. Then, every time the thread finds an architecture, it notifies the main thread of the Daphne Brain, which is responsible for communicating with the interface. This main thread then sends the new architecture to the frontend using what we described as Websockets Push, where it is stored in a queue of pending architectures until the user decides to show them on the plot.

The second task of the Explorer is fostering diversity in design space exploration. This functionality has been named "Diversifier". In past experiments pending publication, we noticed that users tend to focus on optimizing small regions of the tradespace, thus missing possible ways of improving designs. Thus, the Explorer tries to incentivize the users to try and work in new areas of the design space by keeping track of the crowding distance in all the points in the non-dominated set. This same set is divided in 10 smaller sets by partitioning the objective space uniformly from the minimum objective values to the maximum. This creates 10 sets which can have different numbers of designs in them. When the maximum crowding distance on some of the groups is higher than in the others, an action is triggered. The Explorer then sends a message through WebSockets Push to the interface suggesting to the user to start looking at one of the subsets with the higher mean crowding distance.

3. Live Recommender System

The second significant addition to Daphne to make it more proactive has been named the Live Recommender System. This system is tasked with showing advice to the user when he/she is in the process of creating a new design. By default, the system generates all the possible suggestions but they are not shown to the user unless he/she activates the visualization. If they are not being shown and a few sets of suggestions have already been triggered, a prompt will appear asking the user if he/she wants to see advice on improving his/her design. This advice can come from many different sources, as seen in Table 1. Each advice has different triggers and is shown in a different way. The following paragraphs describe how each of the different suggestions are generated and shown.

The Engineer suggestions work in a similar fashion to what was already shown as part of the Aggregator feedback: each design is analyzed and a rule-based program is run to check whether a set of rules of thumb are being satisfied or violated by the current design. The trigger for generating these suggestions is when the user makes some changes to the architecture in the Design Builder space. For now, that is fixed to 3 changes under 60s before triggering. When the role is triggered it activates the rule-based system and returns all the advice, which is shown as a text list.

In the case of the Historian, the role compares the current design with past missions and assesses how similar the design is to past missions. If it finds a significant similarity, the role returns a sentence explaining the similarities between the current design and that of similar historic missions, while if there is no similar mission in the database, the Historian will advise the user to be cautious, as this might be an untested and thus risky design for a satellite constellation. For example, most radar altimeters have been flown on non-SSO orbits, in order to better model the diurnal cycle. Thus, if the user puts a radar altimeter in an SSO, the Historian will flag this decision as "rare" and warn the user that there might be a reason for the absence of missions with that feature. The trigger for generation is, again, 3 changes in the current design under 60s by the user, which activates the comparison between the current design and past missions. The suggestions from this comparison are shown to the user as a list of similar missions for each spacecraft/orbit, as well as another list with similar constellations.

The Analyst is a little different compared to the other roles acting on the Live Recommender System. To begin with, instead of just acting on a trigger, it is continuously running in the background. Every 30 seconds, it finds sets of driving features from designs with Pareto ranking up to 3 using classification association rule mining. The metrics on which these features are chosen are specificity (a.k.a. recall) and coverage (a.k.a. precision). Specificity is the fraction of designs with the feature that are in the desirable set (e.g., Pareto ranking up to 3). Coverage, on the other hand, measures the fraction of designs inside the desirable set that have the feature. Thus, a good feature is one with good specificity and coverage. Those metrics are usually at odds, so a trade-off appears. Then, four of those extracted features

*https://redis.io/

are selected using the Minimum Redundancy Maximum Relevance (mRMR) [28] algorithm. Once the four features are chosen, they are suggested to the user by showing a list with a check or a cross next to each feature depending on whether the feature is present on the current design or not. If the user hovers over each of the missing features in the list, a set of "phantom" instruments will tentatively appear and some existing instruments will tentatively disappear in the design builder to create a configuration that is as similar as possible to the current one and has the feature. Also, when hovering over all of the features in the list, all designs that have these features will be highlighted in the trade-space plot in a similar way to when the Analyst role is run in a reactive way.

The Aggregator role, as mentioned in Table 1, has been transformed into the Live Recommender System for the proactive parts of Daphne.

4. Changes in the interface

There are four main changes in the Daphne user interface, which is shown in Figure 3.

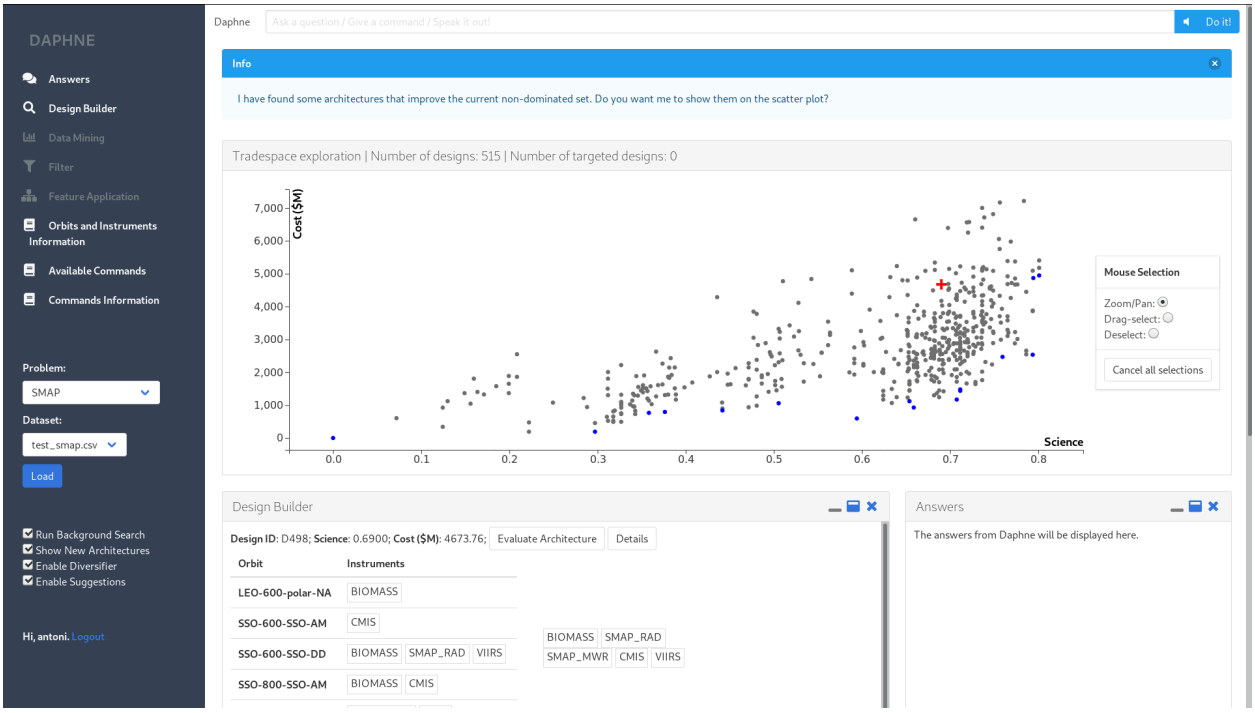

Fig. 3 Daphne screenshot with a notification and all the settings for proactive Daphne

The first change is the blue information block. This new block is where all the notifications from proactive Daphne appear. For example, if there are some new designs from the background search in the queue, or the Live Recommender System is waiting to show its recommendations to the user, both of them will issue a notification which will be shown in this message block. The user can then accept the suggestion, dismiss the notification, or dismiss it and ask to not be shown that again.

Another change is on the left menu: a new array of settings related to the new functionalities has been added there, including the background search, whether or not its results are immediately shown, the exploration assistant (or Diversifier), and the Live Recommender System. The user can manually activate or deactivate each feature depending on his/her needs.

A third change is on the scatter plot, where the deep blue dots now represent the new architectures found by the background search, thus differentiating them from the initial architectures in the dataset. The blue dots turn to the same color as the rest after 5 minutes, to make sure that only the newest architectures are highlighted at any given point.

Finally, the last change is in the Design Builder, where all the suggestions will pop up as described in past sections. They appear in different ways, including text lists on the right side, names of similar missions next to each orbit or a checklist with missing features for the current design.

III. Use Case Scenario: A Constellation for Measuring Soil Moisture

The user is tasked with architecting an Earth Observation Satellite System with the objective of measuring Soil Moisture at the surface. Specifically, the goal is to find a partition of a set of 5 instruments to be assigned in different satellites (all in one satellite, each in its own satellite, or anything in between) so as to maximize satisfaction of a set of 176 measurement objectives related to soil moisture and other secondary measures which are closely tied to it. Measurement objectives are associated with different stakeholders, which include: Weather, Climate, Ecosystems, Water, and Applications. Assuming a classical set partitioning problem with no additional decisions, the number of partitions of a set with 5 elements is only 52, so brute force (full factorial enumeration) could be used instead of a genetic algorithm. In this problem, though, each subset in the partition (a satellite) needs to be assigned to an orbit, thus increasing the dimensionality of the problem. To keep the problem manageable, we limited the possible orbits to 5. With this formulation, the number of possible architectures is 12880. The orbits and the instruments being used are described in Table 2 and Table 3.

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 2 Candidate instruments

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

Table 3 Candidate orbits. SSO = Sun-Synchronous Orbit. LTAN = Local Time of the Ascending Node

In this Demo Scenario, the user is exploring the set of candidate architectures described above and is considering architectures within a broad range of lifecycle costs, between $1000M and $5000M. The goal of the user is to find the best possible set of architectures in the shortest amount of time. For example, we can assume that the user has 2 hours to work on this task.

Daphne starts by generating a random dataset of around 500 architectures satisfying all constraints, in order to warm-start the search. With the new proactive functionality, the background search also starts working at the same point.

After 1 or 2 minutes, which is what it usually takes for the background search to find 10 new designs which improve the initial non-dominated set at the beginning of the search (it takes longer as the search progresses), a message will appear for the user asking if he/she wants to see those new solutions. If the user agrees, the background search results will be shown in real time in the scatter plot from now on.

The user then goes back to work on an architecture close to the minimum required cost, which perhaps simply flies the L-band radar in a a 600km morning SSO and CMIS in a 600km polar orbit ($1,059M and 0.22 science score). After some time thinking, he may ask Daphne how to improve that design. Daphne, through its Aggregator role, may come back with a suggestion to move the radar to a dawn-dusk orbit to reduce power consumption and thus cost, and to replace the CMIS instrument with the L-band radiometer in the same orbit to improve sensitivity. Note that these changes are likely to make some improvements to the architecture ($758M for a science score of 0.34), but they will also likely degrade other aspects of the architecture, such as temporal resolution in this case - since we go from 2 orbits to 1 orbit. Here is where the interaction with the user is critical, since they can understand those effects and weigh the relative desirability of the architectures including aspects that are not necessarily captured in the model. At this point, the user may try some of the suggested changes. After some time, the Live Recommender System will have accumulated

enough feedback that another message will pop out asking the user if he/she wants to see all the suggestions in real time. Again, in case the user agrees to it, all these suggestions will start showing in the Design Builder block in the different ways we have already described.

After some time of the user working with the dataset, chances are that some regions in the space have been more extensively searched than others, so the Diversifier will send a message, suggesting the user to start looking in a relatively unexplored area of the scatter plot. For example, it might recommend to start searching designs in the $4000M range, as there might be fewer designs in that area compared to the rest. This helps the user in the given task of finding designs for a range of lifecycle costs.

Thanks to a combination of these 3 new tools, the user has, on the one side, an automatic way of finding better architectures which runs without any intervention, and he/she can refine those results with a live system that keeps suggesting improvements. On the other side, if he/she focuses too much on improving a small part of the dataset, the Diversifier helps shuffle the focus to more undersearched areas, thus increasing the probability of finding a better set of solutions.

IV. Conclusion and Future Work

In this paper, we have explained the first steps of turning Daphne into a proactive virtual assistant from its reactive origins by adding a new way of communication between the Daphne Brain and the user and adding new features which use this new channel. These features include a background design space search algorithm, a program that tries to get the user to explore more of the dataset, and a live recommender system that gives advice to the user in real time to guide them towards better designs faster.

We have described how all the already existing pieces have changed to adapt to this new system as well as how the new components have been programmed and integrated into the general system. We also explained how we are making Daphne more general and thus more useful for more users and tasks, and a specific use case illustrating the utility of the changes implemented.

In future work, we plan to run an experiment with human subjects, in which we will compare the performance of the same subject when using the reactive Daphne versus the proactive Daphne. At the same time, we will ask the subjects to assess the usability of both systems and their emotions when using them, as it has been shown that emotions play an important role in software adoption and user performance [29].

Apart from the human subjects experiment, we plan on improving the new proactive features. For the design space search, more advanced search algorithms can be utilized that facilitate the integration of expert knowledge [30]. For the recommender systems, suggestions coming from the Analyst and the Historian are just scratching the surface of what is possible to do with both roles. For example, instead of using specific instruments to create features for the Analyst, higher level variables can also be used (e.g., High-Power Instruments, Visible and Near Infrared Instruments), which can lead to higher predictive power for features of a given length. Following cognitive style theories [31], these suggestions could also be provided in a more visual way compared to just lists of text, for example by coloring instruments according to a certain scale or suggesting instruments moves between orbits by tentatively showing the instrument in the new orbit in gray and having the user click on the new position to confirm the change. Regarding the Diversifier, instead of just taking into account distance in the objective space, it could also account for distance in the decision space, i.e., look for architectures with decision values which have not been explored enough. Similarly, one can look for diversity in the feature space, meaning the user could be directed to explore architectures containing features which have not been explored.

We will also study frameworks and algorithms for anticipating user goals and intentions and future actions.

Daphne continues to be available online at `https://www.selva-research.com/daphne`, and its code is open, licensed under the MIT license and available at `https://www.github.com/seakers`.

References

[1] Nag, S., Hughes, S. P., and Le Moigne, J., "Streamlining the Design Tradespace for Earth Imaging Constellations," *AIAA Space 2016*, 2016, pp. 1–17. doi:10.2514/6.2016-5561.

[2] Selva, D., "Knowledge-intensive global optimization of Earth observing system architectures: a climate-centric case study," *SPIE Remote Sensing*, Vol. 9241, 2014, pp. 1–22. doi:10.1117/12.2067558.

[3] Bang, H., and Selva, D., "iFEED: Interactive Feature Extraction for Engineering Design," *ASME 2016 International*

Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 2016, pp. 1–11. doi:10.1115/DETC2016-60077.

[4] Bang, H., Shi, Y. L. Z., Yoon, S.-Y., Hoffman, G., and Selva, D., "Exploring the Feature Space to Aid Learning in Design Space Exploration," *Des. Comput. Cogn.*, 2018.

[5] Thompson, R. E., Colombi, J. M., Black, J., and Ayres, B. J., "Disaggregated Space System Concept Optimization: Model-Based Conceptual Design Methods," *Systems Engineering*, Vol. 18, No. 6, 2015, pp. 549–567. doi:10.1002/sys.21310.

[6] Bang, H., Virós Martin, A., Prat, A., and Selva, D., "Daphne: An Intelligent Assistant for Architecting Earth Observing Satellite Systems," *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, 2018, pp. 1–14. doi:10.2514/6.2018-1366.

[7] Ferrucci, D. A., "Introduction to "This is Watson"," *IBM Journal of Research and Development*, Vol. 56, No. 3.4, 2012, pp. 1–15. doi:10.1147/JRD.2012.2184356.

[8] Hauswald, J., Tang, L., Mars, J., Laurenzano, M. A., Zhang, Y., Li, C., Rovinski, A., Khurana, A., Dreslinski, R. G., Mudge, T., and Petrucci, V., "Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers," *Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '15*, 2015, pp. 223–238. doi:10.1145/2694344.2694347.

[9] Baudiš, P., "YodaQA : A Modular Question Answering System Pipeline," *POSTER 2015-19th International Student Conference on Electrical Engineering*, 2015, pp. 1156–1165. URL `http://ailao.eu/yodaqa/yodaqa-poster2015.pdf`.

[10] Wang, D., and Nyberg, E., "CMU OAQA at TREC 2016 LiveQA : An Attentional Neural Encoder-Decoder Approach for Answer Ranking," *Text REtrieval Conference (TREC) 2016*, 2016, pp. 1–6.

[11] Brooke, J., et al., "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, Vol. 189, No. 194, 1996, pp. 4–7.

[12] Allen, J., Guinn, C. I., and Horvtz, E., "Mixed-initiative interaction," *IEEE Intelligent Systems and their Applications*, Vol. 14, No. 5, 1999, pp. 14–23. doi:10.1109/5254.796083.

[13] Luger, E., and Sellen, A., ""Like Having a Really Bad PA": The Gulf Between User Expectation and Experience of Conversational Agents," *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, 2016, pp. 5286–5297. doi:10.1145/2858036.2858288.

[14] Meurisch, C., Ionescu, M.-D., Schmidt, B., and Mühlhäuser, M., "Reference Model of Next-generation Digital Personal Assistant: Integrating Proactive Behavior," *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ACM, New York, NY, USA, 2017, pp. 149–152. doi:10.1145/3123024.3123145.

[15] Hoffman, G., and Breazeal, C., "Effects of anticipatory action on human-robot teamwork: Efficiency, fluency, and perception of team," *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2007, pp. 1–8. doi:10.1145/1228716.1228718.

[16] Shah, J., and Breazeal, C., "An empirical analysis of team coordination behaviors and action planning with application to human–robot teaming," *Human factors*, Vol. 52, No. 2, 2010, pp. 234–245. doi:10.1177/0018720809350882.

[17] Meurisch, C., Jeutter, B., Schmidt, W., Gündling, N., Schmidt, B., Herrlich, F., and Mühlhäuser, M., "An Extensible Pervasive Platform for Large-Scale Anticipatory Mobile Computing," *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1, 2017, pp. 459–464. doi:10.1109/COMPSAC.2017.54.

[18] Bernard, D., "Cognitive interaction: Towards "cognitivity" requirements for the design of virtual assistants," *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 210–215. doi:10.1109/SMC.2017.8122604.

[19] Lim, B. Y., and Dey, A. K., "Investigating Intelligibility for Uncertain Context-aware Applications," *Proceedings of the 13th International Conference on Ubiquitous Computing*, ACM, New York, NY, USA, 2011, pp. 415–424. doi:10.1145/2030112.2030168.

[20] Whitworth, B., "Polite computing," *Behaviour & Information Technology*, Vol. 24, No. 5, 2005, pp. 353–363. doi:10.1080/01449290512331333700.

[21] Armentano, M., and Amandi, A., "Personalized detection of user intentions," *Knowledge-Based Systems*, Vol. 24, No. 8, 2011, pp. 1169 – 1180. doi:10.1016/j.knosys.2011.04.020.

[22] Hitomi, N., and Selva, D., "A classification and comparison of credit assignment strategies in multiobjective adaptive operator selection," *IEEE Transactions on Evolutionary Computation*, Vol. 21, No. 2, 2017, pp. 294–314. doi:10.1109/TEVC.2016.2602348.

[23] Gershman, S. J., "Deconstructing the human algorithms for exploration," *Cognition*, Vol. 173, 2018, pp. 34–42. doi:10.1016/j.cognition.2017.12.014.

[24] Cohen, J. D., McClure, S. M., and Angela, J. Y., "Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration," *Philosophical Transactions of the Royal Society B: Biological Sciences*, Vol. 362, No. 1481, 2007, pp. 933–942. doi:10.1098/rstb.2007.2098.

[25] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, Vol. 6, No. 2, 2002, pp. 182–197. doi:10.1109/4235.996017.

[26] Newman, S., *Building microservices: designing fine-grained systems*, O'Reilly Media, Inc., 2015.

[27] Selva, D., Cameron, B. G., and Crawley, E. F., "Rule-Based System Architecting of Earth Observing Systems: Earth Science Decadal Survey," *Journal of Spacecraft and Rockets*, Vol. 51, No. 5, 2014, pp. 1505–1521. doi:10.2514/1.A32656.

[28] Peng, H., Long, F., and Ding, C., "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 8, 2005, pp. 1226–1238. doi:10.1109/TPAMI.2005.159.

[29] Manfred, T., and Sascha, M., "Usability, aesthetics and emotions in human–technology interaction," *International Journal of Psychology*, Vol. 42, No. 4, 2008, pp. 253–264. doi:10.1080/00207590701396674.

[30] Hitomi, N., and Selva, D., "Incorporating expert knowledge into evolutionary algorithms with operators and constraints to design satellite systems," *Applied Soft Computing*, Vol. 66, 2018, pp. 330 – 345. doi:https://doi.org/10.1016/j.asoc.2018.02.017, URL `http://www.sciencedirect.com/science/article/pii/S1568494618300760`.

[31] Shi, L., Bang, H., Hoffman, G., Selva, D., and Yoon, S.-Y., "Cognitive style and field knowledge in complex design problem solving: A comparative case study of decision support systems," *Des. Comput. Cogn.*, 2018.