Daphne: An Intelligent Cognitive Assistant for the design of Earth Observing Satellites

---

A Thesis presented to the
Escola Tècnica Superior
d'Enginyeria de
Telecomunicació de
Barcelona Universitat
Politècnica de Catalunya

In Partial Fulfillment of
the Requirements for the
Degree of Master in
Telecommunications
Engineering

---

by
Arnau Prat i Sala

Dr. Daniel Selva, Thesis Supervisor

Dr. Eduard Alarcón, Reporting Advisor (ponent)

SEPTEMBER 2017

ABSTRACT

This thesis introduces Daphne, an Intelligent Cognitive Assistant (ICA) designed to help engineers architect Earth observing satellite systems. The purpose of this thesis is threefold: 1) develop the interfaces through which the user interacts with the system, 2) develop the basic intelligence of the system and 3) perform a preliminary experiment to evaluate the effectiveness of the system in improving design outcomes. Three different interfaces have been developed which include different modes of interaction, visual interfaces (including virtual reality) as well as a physical embodiment for non-verbal interaction. An artificial intelligence has been developed to help users explore the design space by providing criticism and suggestions about the design the user is currently working on. Finally a preliminary experiment has been performed to measure the effectiveness of the system. Ultimately, the goal of Daphne is to enable mixed-initiative human-computer design of complex systems, in which humans and Intelligent Cognitive Assistants engage in balanced collaborations: not dominated by the human or the computer but rather engaging in a meaningful dialogue. This project is part of a research grant awarded by the United States NSF (National Science Foundation).

ACKNOWLEDGEMENT

I would like to thank all the people that made this work possible. Foremost, I would like to express my sincere gratitude to Prof. Daniel Selva and Prof. Eduard Alarcón for their help and for giving me this opportunity. Also I would like to thank all the good people from Ithaca, I hope to come back to the US very soon!

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER I: INTRODUCTION

Introduction

As systems become more and more complex there is a need to develop tools to help engineers with their design. This thesis introduces Daphne, an Intelligent Cognitive Assistant specially designed to support engineers during the system design process, i.e. process of defining the architecture, modules and interfaces for a system to satisfy specific requirements.

To test the efficacy of Daphne to support the design of a complex system, we used a real-world problem of architecting an Earth observing satellite system to perform operational monitoring of the Earth's climate, as described by Selva [1]. Specifically, the goal is to design a constellation of satellites that maximizes a set of 400 climate-related measurement requirements while minimizing the life-cycle cost.

The design problem is formulated as an assignment problem between instruments and orbits. There are a total of 12 candidate instruments (A, B, C, D, E, F, G, H, I, J, K, L) and 5 candidate orbits (1, 2, 3, 4, 5). In order to simplify the task the real names of the orbits and instruments were substituted by number and letters (see appendix A). Because every instrument can be assigned to any orbit or not, there are a total of 2 to the 60 possible designs.

The goal of Daphne is to help engineers by reducing their cognitive load and lead them to better designs. The author's contributions to this project and therefore the purpose of this thesis are: 1) develop the interfaces through which the user interacts with the system, 2) develop the basic intelligence of the system and 3) perform a preliminary experiment of the system.

Background

Intelligent Cognitive Assistants (ICA) are intelligent agents that manage and perform tasks on behalf of humans to reduce routine tasks and the cognitive load of the user [2]. It is important to note that the main goal of these systems is not replacing the human, but to complement human capabilities both in performing routine tasks and solving complex problems [3].

Many of the current ICAs focus on performing routine everyday tasks such as organizing emails, scheduling meetings or answering general questions. A good example of ICA is Siri [4], Apple's virtual personal assistant. Another example is Mycroft [5], an open source virtual

personal assistant based on Raspberry Pi and Arduino. The users can interact with Mycroft through its embodied interface using natural language.

While larger focus has been put on developing Intelligent Cognitive Assistants that perform everyday tasks such as Siri or Mycroft, there also exist intelligent assistants that target more domain-specific tasks. One example is HealthPal [6], a personal medical assistant that monitors various health conditions and alerts the user when there are abnormal indications. HealthPal also utilizes natural language to interact with the user.

Intelligent assistants have also been developed and tested for aerospace and defense applications. A good example is the Cognitive Cockpit Assistant Systems (CASSY/CAMA) [7], an ICA that fosters the pilot's situation awareness during a flight. The agent is capable of understanding the flight situation and goals and understands the intent of the pilot and his or her possible errors. Then based on those observations, the virtual assistant initiates human-like communication with the pilot to help with their situational awareness.

One of the important capabilities of ICAs discussed so far is the ability to provide useful information to the user at the right time. Daphne can reduce the cognitive load of the user by providing information, which is relevant to the current process of the search. This is done through the Critic Agent, which will provide criticism and feedback to the user about the design at hand.

System Architecture

The basic architecture of Daphne is presented in the figure below. Daphne has a modular structure and it's intended to be scalable, i.e., it has to be relatively easy to add new capabilities and adapt it to other complex tasks other than the design of constellation of satellites.

The dashed line separates the front-end front the back end. The front end consists of different interfaces that the user can use to interact with Daphne. The back end consists of 1) the Daphne Brain, which serves all the Daphne interfaces, 2) the Critic Agents, which provide feedback to the user about the designs he or she propose, 3) the Analyst Agent, a question-answering (QA) System which answers general questions about the system, 4) the Data Mining Agent, which runs machine learning algorithms on the dataset for insight generation and 5) the Architecture Evaluation agent, which encapsulates the objective functions and evaluates the designs. The architecture also features three sources of knowledge (expert knowledge, historical database, current dataset), which are also part of the back end, and the rest of the modules use.

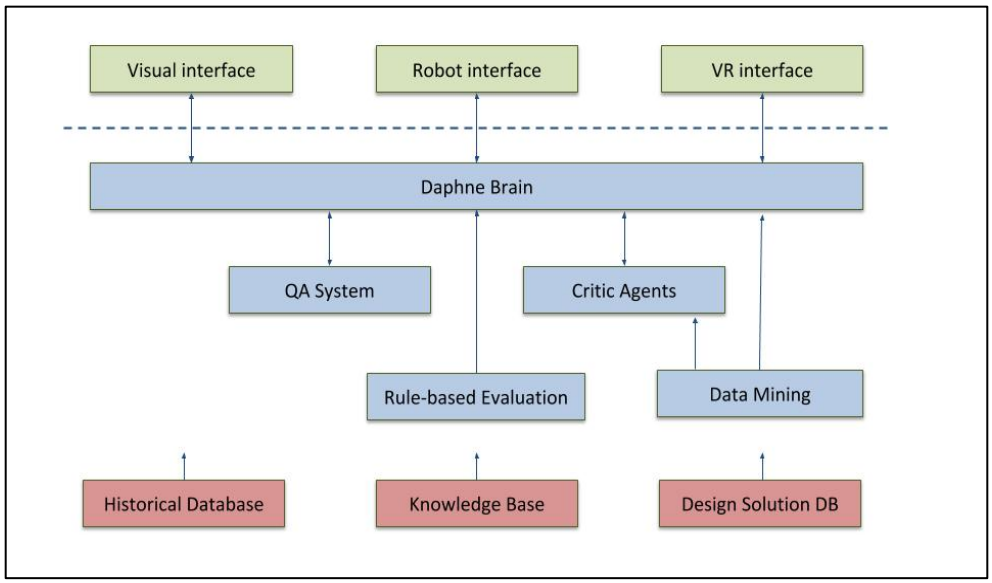

Figure 1: Daphne System Architecture

Gantt Diagram

Below is the Gantt Diagram illustrating the start and finish days of the main tasks of the project. There are 7 main tasks: design the software architecture, develop the robot interface, develop the VR interface, develop the visual interface, develop critic agents, prepare experiment of the system and perform experiment of the system.

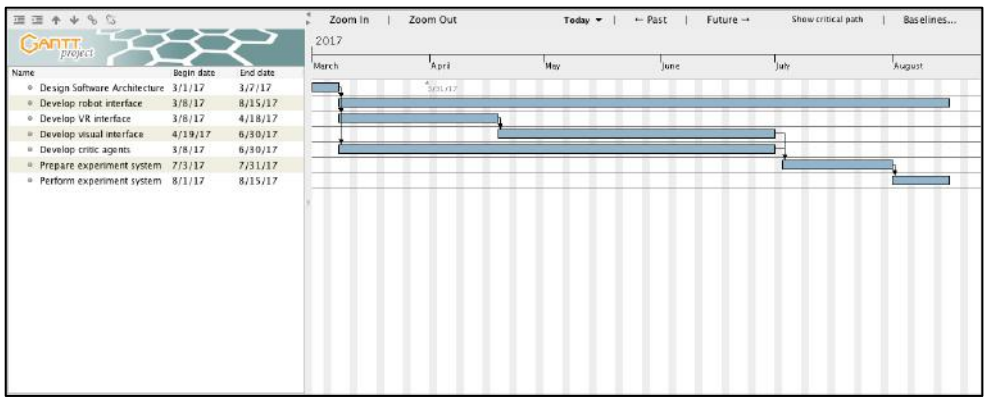

Figure 2: Gantt diagram of the project

Cost Analysis

Below is an approximation of what it has cost to develop a prototype of the system; mainly the most important parts are: the cost of the materials to build the Daphne robot, the cost of a VR headset and the cost of the Amazon Web Services (AWS) server for one year contract.

Name	Link	Cost ($)
Robot 3D parts	https://www.3dhubs.com/	164,03
Robot electronics (appendix B)	https://www.adafruit.com/ https://www.amazon.com/	187,81
VR headset	https://www.amazon.com/VOX-Headset-Virtual-Compatible-Smartphone/dp/B01BTXADM6/	19,99
AWS server (t2.large / 1 year)	https://aws.amazon.com/	916
		1287,83

Table 1: Cost analysis of the project

Project Background

This project is part of a research grant awarded by the United States NSF (National Science Foundation) under the title: Improved Human-Computer Interaction for Design of Complex Systems. More information about the award can be found in the following webpage: https://www.nsf.gov/awardsearch/showAward?AWD_ID=1635253.

The author of this thesis worked under the supervision of Prof. Daniel Selva, who is the principal investigator of the research grant mentioned above. The project has two other co-investigators: Prof. So-Yeon Yoon and Prof. Guy Hoffman and several undergraduate and graduate students. However the work presented in this thesis corresponds only to contributions made by the author.

CHAPTER II: DAPHNE INTERFACES

This section presents the three front-end interfaces that have been developed for the Daphne Cognitive Assistant: Visual Interface, VR interface and Robot Interface. These many interfaces will allow designer to interact with the system in many different ways.

Visual Interface

The Daphne visual interface allows the user to create, evaluate and get feedback about any design (i.e. satellite constellation) and it aims to be the main interface with which the user interacts with the system. A screenshot of the interface is shown in the figure below.

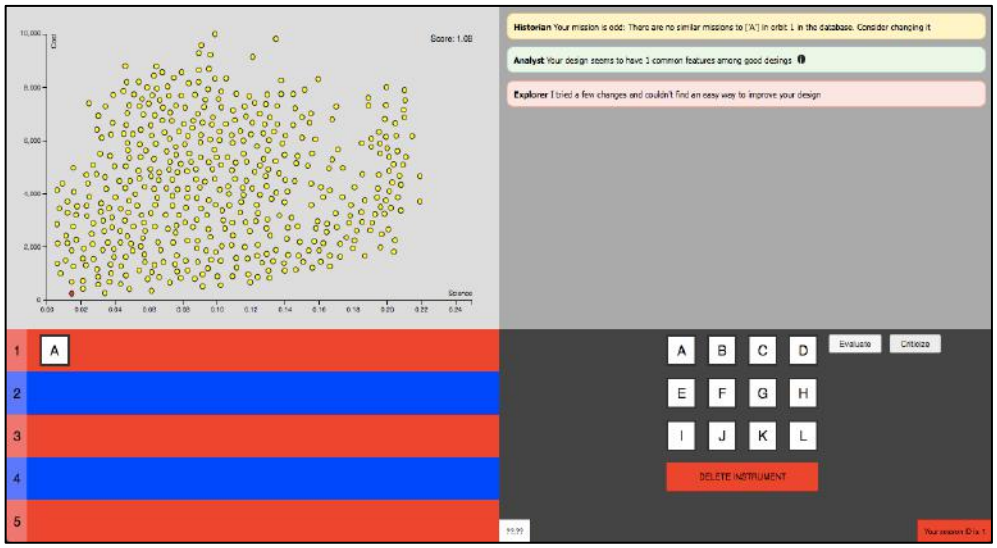

Figure 3: Daphne Visual Interface

The interface consists of 4 different sections: the top left corner contains a scatter plot representation of the design trade-space (i.e. science benefit and cost of the different designs discovered by the user). The top right corner displays the output produced by the Critic Agent. The bottom left corner represents the design the user is currently working on, where each row represents an orbit and each block an instrument. Finally, the bottom right corner contains the instruments to build a new design and the "evaluate" and "criticize" buttons.

A new design is created by drag-and-drop of different instruments (blocks) to different orbits (rows) and/or removing them if needed. When the user is happy with the design, he or she can evaluate it (i.e. calculate its science benefit and cost) by pressing to the button "evaluate". The result of the evaluation will appear in the scatter plot as a red dot.

In order to improve a design, the user can get feedback about it by pressing to the button "criticize". The critic provides specific suggestions to the user about how to improve a given design. There are four different types of critic agents: the expert, the historian, the analyst and the explorer, each of which draw on a different means of analysis to suggest improvements. More details about their implementation are presented in Chapter III: Critic Agents.

As stated above, each design has a science benefit and cost. The science benefit is a number that tells us how much value each design provides to the climate monitoring community. The cost is a measure of how much it is going to cost to design, implement, launch and operate those spacecraft. Naturally, low-cost and high-science designs are desirable. Both these values are computed by the Architecture Evaluator agent, which is found on the backend.

The following figure shows a possible design and the science benefit and cost associated with it (red dot). The goal of the designer is to come up with a design that is as close to the Pareto Front or Pareto Set [8] (set of non-dominant solutions where no objective can be improved without sacrificing at least one other objective) (blue curve) as possible.

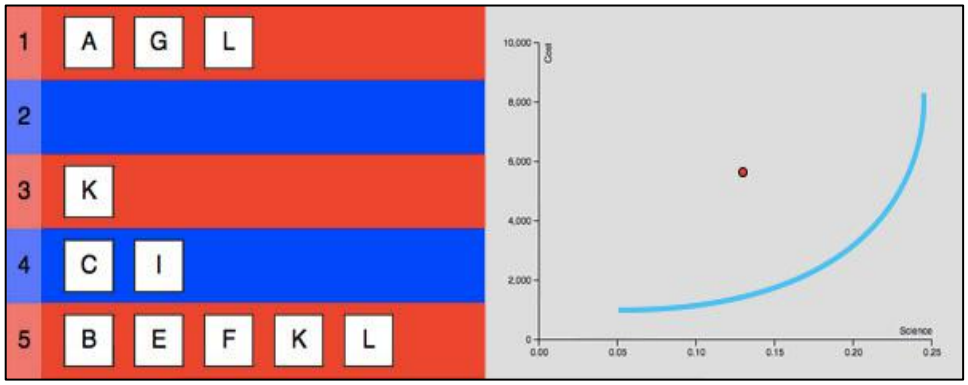

Figure 4: A possible design and its science benefit and cost

The Daphne visual interface is built using HTML, CSS, and JavaScript. Visualizations of the scatter plot were made using a JavaScript library called D3 [9], a library for visualizing data using web standards, while the interactive interface was made using a library called jQuery [10], a library designed to simplify the client side scripting of HTML. As the interface is based on web technology, it can run in any modern web browser on any device.

Communications with the interface and the server is done using Web Socket, which allows for bidirectional asynchronous communication. The visual interface only receives user inputs and visualizes data, while the server does all the data handling and processing. Both the VR and robot interface also use Web Socket for communicating with the backend [11] [12].

VR Interface

The Daphne VR (Virtual Reality) interface aims to be an alternative to the visual interface and it is intended to explore the benefits of working in a Virtual Reality environment compared to a traditional Desktop interface. A screenshot of the interface is shown below.

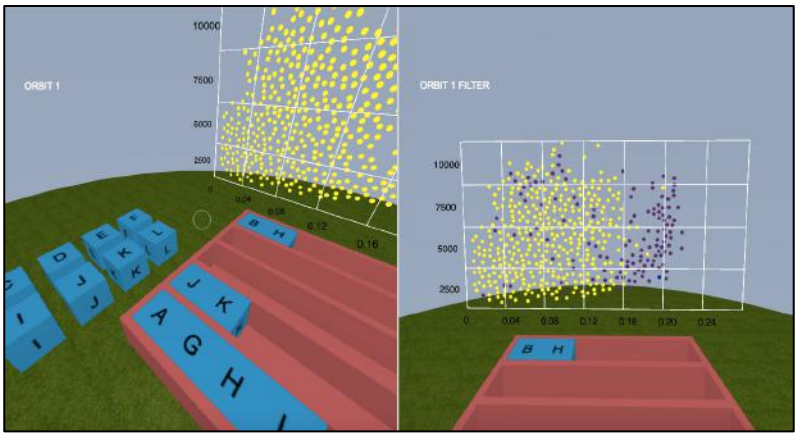

Figure 5: Daphne VR Interface

The interface can run in any modern web browser but the end user needs to have access to a Google Cardboard, GearVR or similar in order to enjoy the full Virtual Reality experience. Apart from a mobile VR headset the user does not require any external controllers or equipment.

The user interacts with the environment by looking at the object that he or she wants to select and then pressing the action button located on the headset. To add an instrument to an orbit for example, first we would need to select the orbit in which we want to place the instrument and then we would select the instrument that we want to add. For evaluating or criticizing a design we would just select the corresponding button.

To evaluate a design after changing the instrument assignment, the user will have to select the "Update" button that will appear when the design is changed. After a few seconds the result of the evaluation will appear in the scatterplot. To criticize a design, the user will need to select the button "Criticize" located in the back. After a few moments the result will appear below.

The VR interface also features a filter system, which can be turned on and off pressing a box located in the right side. The system allows 4 different types of filters by stacking instruments in different places. The filters available are: "Instrument in any orbit", "Instrument in orbit N", "instruments together" and "instruments separate". This allows for the user to discover driving features, which may be common among good design (e.g. most good designs may have in common that instrument A is always in orbit 1 and instruments B and C are never together).

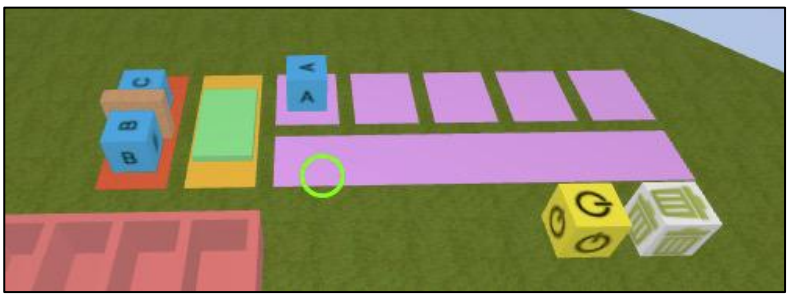

Figure 6: Filter section of the VR Interface

The image below is a top view of the virtual environment. The environment can be divided in 4 different parts: 1) Design space: represents the design the user is currently working on, 2) Scatter plot: displays the science benefit and cost of the different designs discovered by the user, 3) Critic Result: displays the output of the Critic Agents and 4) Filter section: allows the user to filter the designs on the scatterplot.

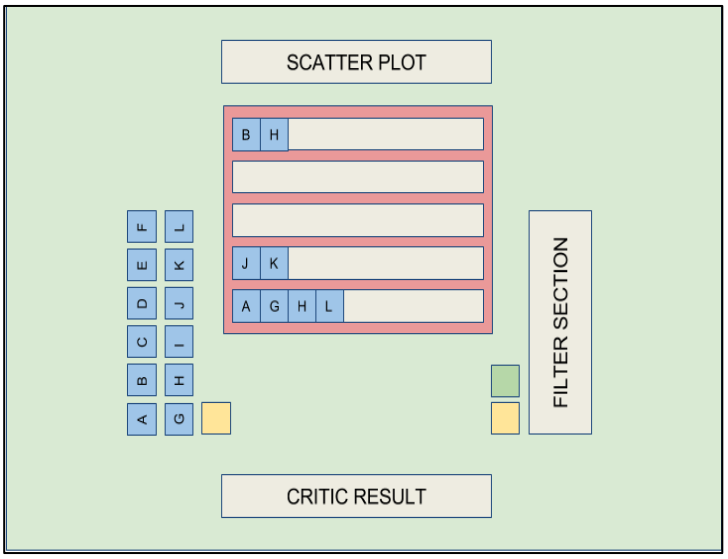

Figure 7: Top view of the Virtual Interface

As stated before, this experimental interface is intended to test whether designers work better on a computer screen or when immersed on a virtual design environment. The interface has gotten good feedback from early users and some of them have shown more engagement than with the visual interface. Future experiments are expected to be performed comparing both interfaces.

The Daphne VR interface was implemented using Three.js [13], a cross-browser JavaScript library used to create and display animated 3D computer graphics in a web browser. Although we initially intended to implement the interface using Unity and Oculus Rift, we decided to use a web-based platform for its versatility and inexpensive cost.

Robot Interface

The design of the Daphne robot interface is presented in the figure below. The goal of the robot interface is to improve the human-machine communication by using non-verbal channels and embodied interaction to enable mixed-initiative human-computer design. The robot is built using 3D printing technology and commercial of the shelf (COTS) components and its design will be made available to the community as an open source project, which means that anyone will be able to assemble one given time and very simple to buy electronics.

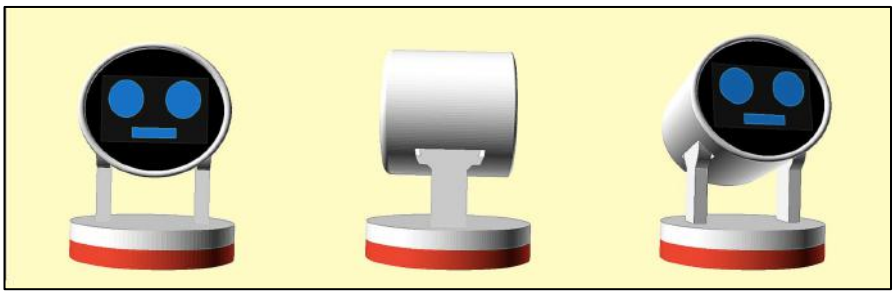

Figure 8: Daphne Robot Interface

Among its main components there is an LCD screen, which can be used to either show facial expressions or display any type of data (e.g. images and graphs). The robot also has smooth, variable speed, pan and tilt moves. These traits will be used to track the user, but also to increase its expressiveness (e.g. nod its head). These kinds of features can help improve the interaction and/or human performance in the task to be performed.

The hardware architecture of the robot is presented in the figure below. Between its main components are a Raspberry Pi, an Arduino Uno, a 7'' LCD screen, a camera, a microphone and a speaker. Due to the fact that Raspbian (the Operating System (OS) on the Raspberry Pi) is not a Real Time OS and the board does not have a dedicated module for the PWM, we were forced to use an Arduino to control the motors instead of using the GPIOs on the Raspberry Pi in order to avoid erratic behavior. The rest of the electronics (camera, microphone, speaker…) are controlled directly with the Raspberry Pi.

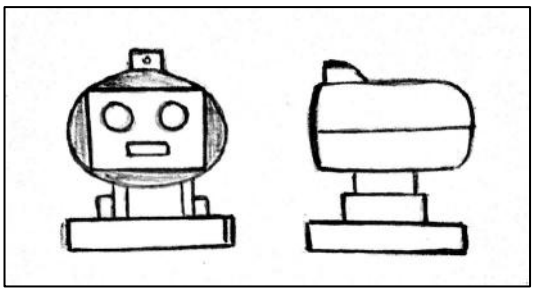

Figure 9: Early design concept of the Robot Interface

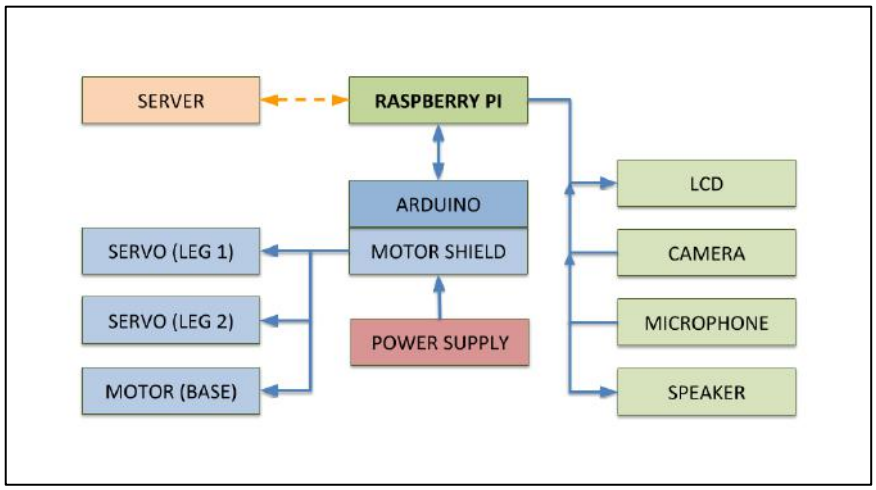

Figure 10: Daphne Robot hardware architecture

The Raspberry Pi interacts with the server through Wi-Fi and with the Arduino using serial communication. The connection with the rest of devices is done through USB. The robot is powered through the Arduino using a 9 VDC power adapter. The Arduino then distributes the power to the rest of the electronics including the Raspberry Pi. This configuration allows having a single power supply and it can support the peaks of power coming from the servos, which sometimes is not possible using batteries.

The software architecture of the robot is presented in the figure below. The code on the Raspberry Pi is written in Python and it consists of 5 main threads. 1) Main.py: it acts like a hub between all other threads; 2) Screen.py: It controls what is shown in the LCD screen; 3) Conversation.py: Manages the Question and Answer system including Speech to Text and Text to Speech modules; 4) Movement.py: controls the movements of the stepper motor (pan move) located in the base and the two servomotors (tilt move) located in the head of the robot.

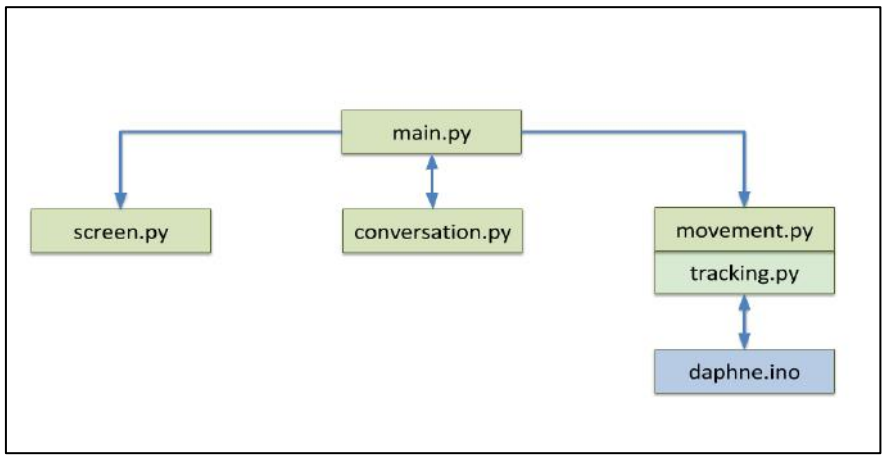

Figure 11: Daphne Robot software architecture

The Daphne robot has been designed using OpenSCAD [14], a free software application for creating solid 3D CAD (computer-aided design) objects. Unlike other programs like Solidworks or Autodesk Inventor, OpenSCAD is script based (i.e. it does not use an interactive 3D interface for modeling, but a scripting language). Below is an example of a simple object designed using OpenScad (left: code, right: rendered object).

```
difference() {
  cube([10,10,10],center=true);
  translate([0,0,1])
     sphere(r=6,center=true,$fn=50);
}
```

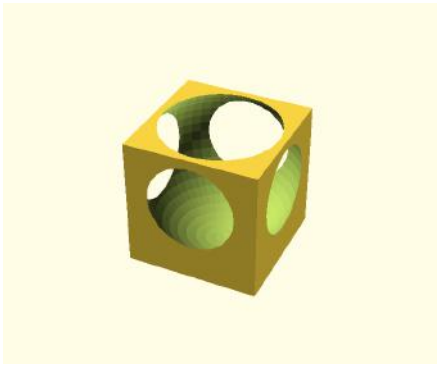

Figure 12: OpenSCAD model example

The final design of the Daphne robot has more than 500 lines of code and it consists of 6 different parts: head_top.stl, head_bottom.stl, shelf_top.stl, screen_cover.stl, leg.stl, base_top.stl and base_bottom.stl. These parts were printed using the 3D printing service 3D Hubs [15]. This service was much cheaper than other available services such as the Cornell Rapid Prototyping Lab (RPL). The different parts of the robot are attached to each other using bolts and clamps, which makes the robot very robust. Below are an image of the OpenSCAD and the printed version of the robot, which is completely operational.

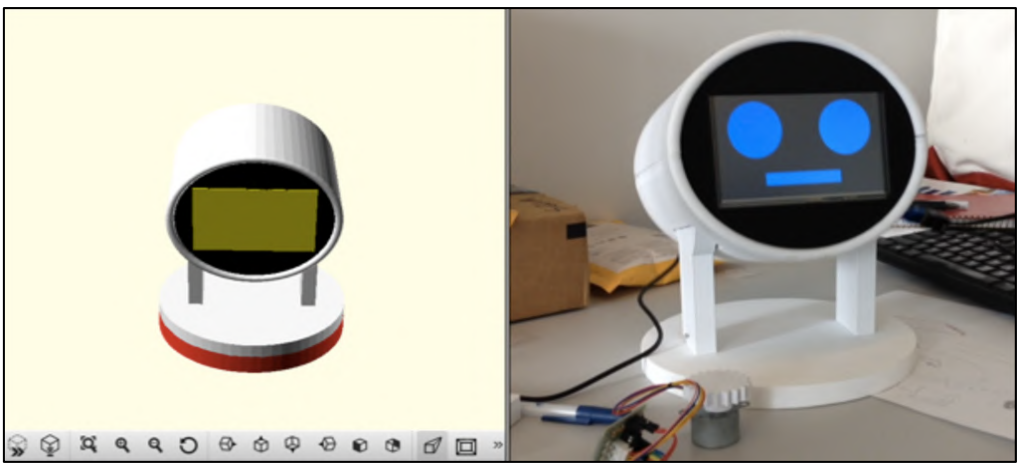

Figure 13: Daphne Robot OpenSCAD and physical implementation

The visual interface and the robot interface are expected to work together and complement each other. A mockup of how the interaction with the user will look like is presented in the figure below. This configuration will allow mixed-initiative design, in which humans and intelligent design assistants engage in balanced collaborations. Not dominated by the human or the computer but rather engaging in a meaningful dialogue. This is done by having verbal and nonverbal interactions as well as embodied interactions.

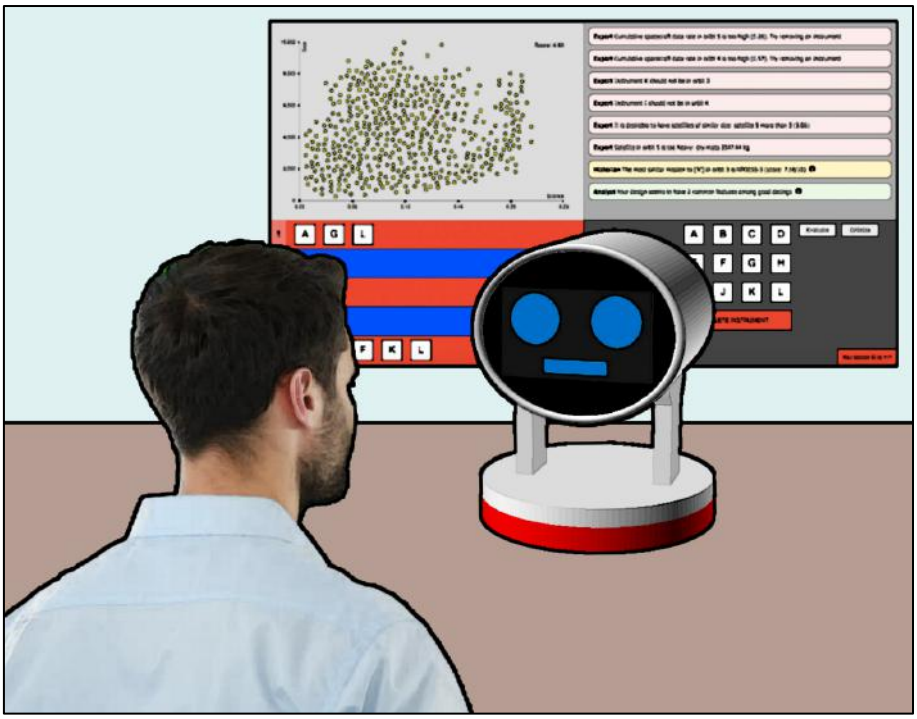

Figure 14: Mockup of human interacting with Daphne

CHAPTER III: CRITIC AGENTS

Introduction

The aim of the Critic Agent is to criticize an architecture proposed by the user (i.e. to provide feedback to the user about the strengths and weaknesses of that design). In addition to the criticism, it also provides specific suggestions to the user about how to improve a given architecture. The Critic Agent is an essential part of the system and it plays a key role in improving design outcomes. An example of the output of the Critic Agent is shown below.

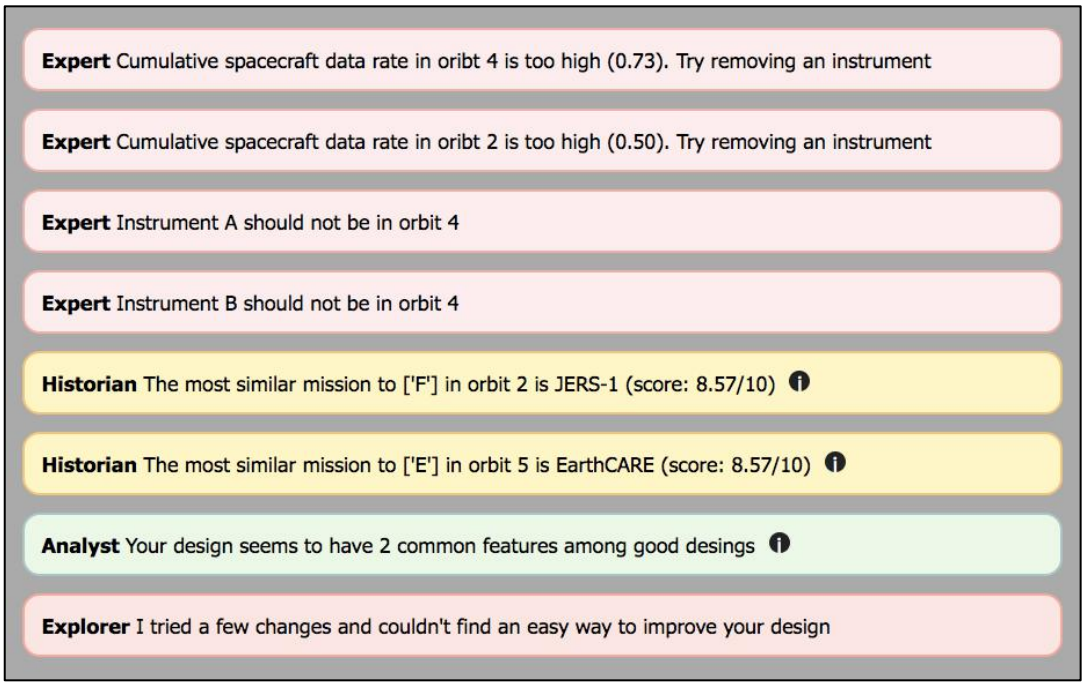

Figure 15: Example of Critic Agent

Using four different sources of information, there are four different types of critiques or recommendations: those based on expert knowledge (rule-driven), those based on past missions (legacy-driven), those based on knowledge extracted from design solutions (data-driven) and those based on exploring the design space (model-driven).

Each type of critique agent can be seen as an expert in his or her field and can be characterized as follows. The expert has many years of experience designing spacecraft for climate monitoring. The historian maintains a database of past missions and thus can recommend things based on what was done in the past. The analyst uses statistical techniques to look for trends in the current dataset and recommend changes to the user's design based on that. The explorer searches the design space and alerts the user if it finds any good designs.

Expert (Rule-based)

The expert notifies the user if a given design violates a set of rules defined a priori. These rules can be considered as basic design principles or heuristics that domain experts use for coming up with good designs (e.g. two instruments using the same frequency should not be used in a same spacecraft if at least one of them is active).

These rules are encoded into the system using Jess (Java Expert System Shell), an inference engine to develop expert systems in Java. Below is an example of one of these rules implemented in this language, which is triggered if a passive optical instrument is flown on a Dawn Dusk (DD) orbit. This rule is based on the fact that instruments of this type don't work well in these types of orbits because of the poor illumination conditions.

```
(defrule CRITIQUE-PERFORMANCE::passive-optica-instrument-in-DD-orbit
    "Passive optical instruments should not be in DD orbits"
    (CAPABILITIES::can-measure (in-orbit ?o) (instrument ?instr) (orbit-type SSO) (orbit-RAAN DD))
    (DATABASE::Instrument (Name ?instr) (Illumination Passive))
    =>
    (call ?*p* addElement (new java.lang.String
        (str-cat "Instrument " ?instr " should not be in orbit " ?o ))))
```

Figure 16: Example of a rule implemented in Jess

In total, up to 11 rules have been encoded into the system, which can be divided into 2 types: performance and cost. Below are depicted each of them and the category where they belong. The idea is that new rules can be added in order to further help the user.

Rule #	Rule type	Rule description
1	Performance	passive-optica-instrument-in-DD-orbit
2	Performance	atmospheric-chemistry-instrument-in-AM-orbit
3	Performance	side-looking-instrument-in-less-400-km-orbit
4	Performance	two-lidars-working-at-the-same-frequency
5	Performance	too-many-instruments
6	Performance	resource-limitations-datarate
7	Performance	resource-limitations-power
8	Cost	mass-check "limits the dry-mass of a satellite"
9	Cost	satellite-size-comparison
10	Cost	satellite-cost-comparison
11	Cost	launch-packaging-factors

Table 2: Rules implemented for the expert

The figure below represents the number of warnings displayed for different designs. It can be clearly seen that the closer the design is to the Pareto Front the lower the number of warnings. The expert is one of the agents that can be more useful to the user because the number of warnings is directly related with the performance of the design.

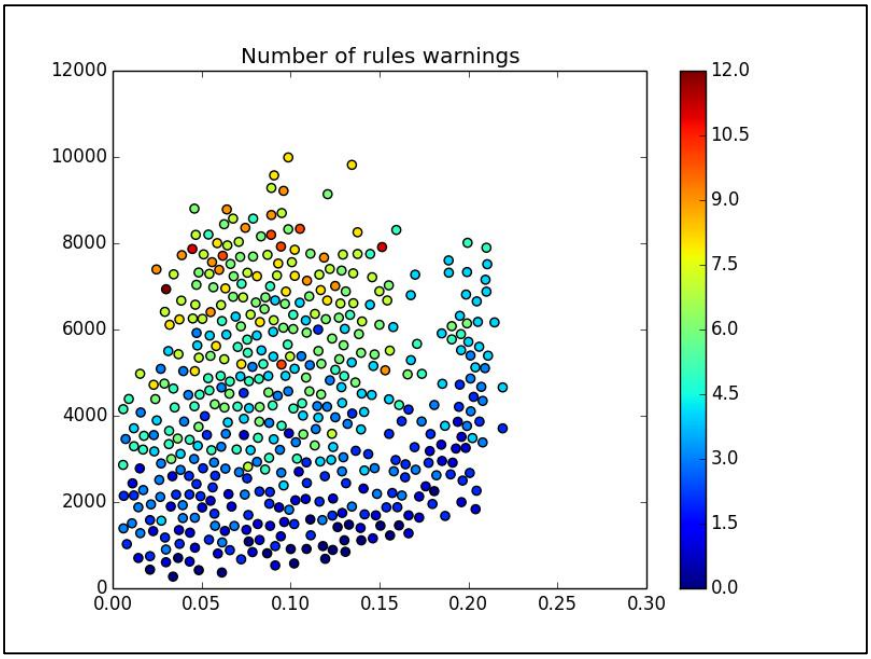

Figure 17: Number of warnings for different designs

Historian (Legacy-driven)

The historian will notify the user if a mission with a similar configuration (similar types and classes of instruments) has been flown in a similar orbit in the past. This would indicate that since a similar type of mission has been proven successful, the given design is also likely to perform well. The motivation for this approach is similar to that of the case-based reasoning [15], which is one of the popular reasoning methods used in artificial intelligence. Conversely, if a similar mission has never been flown in 50 years of space-based Earth observation, this is also a powerful argument suggesting that perhaps there is a flaw in the design of such mission.

The historian uses as its main source of knowledge the CEOS Missions, Instruments and Measurements database [16]. This database is maintained by the European Space Agency (ESA) and it is updated annually. The database contains more than 500 missions and 800 different instruments and it is a very good tool to be considered when designing future missions.

Figure 18: CEOS database and ESA logos

The historian compares all the missions in the database with the 5 missions (orbits) of the design proposed by the user and it computes a similarity score for each of them. This score is computed taking into account the similarity of the orbit in which the mission is flown (3 points) and the average similarity of the instruments of the mission (4 points). The instruments score is only computed if the orbit score is bigger than 1. If the orbit score is above the threshold the total score (orbit + average instruments) is normalized over 10 and if not it is 0. This process has to be done for each mission in the database. Finally, the database mission with the highest score (the most similar to the one of the user) is the one shown to the user along with its score. Below is the code of the Mission Similarity function.

```
def missionsSimilarity(self, orbit1, instruments1, missionsDatabase):
    maxScore = -1
    maxMission = None
    # Iterate over all the missions in the database
    for mission2 in missionsDatabase:
        score = 0
        # Get orbits similarity
        score += self.orbitsSimilarity(orbit1, mission2)
        # If score bigger than a threshold
        if(score > 1):
            # Get instruments similarities
            score += self.instrumentsSimilarity(instruments1, mission2.instruments)
        if score > maxScore:
            maxScore = score
            maxMission = mission2
    # Return result
    return [(maxScore*10)/7, maxMission]
```

Figure 19: Mission similarity function

In order to compute the orbit score the orbitsSimilarity function takes 3 parameters into account: type (e.g. sun-synchronous, polar…), altitude (e.g. 600 km, 800 km…) and LST (e.g. AM, PM or DD). In order to compute the average instrument score the instrumentsSimilarity takes into account 4 parameters: technology (e.g. atmospheric lidar, imaging radar…), type (e.g. atmospheric chemistry, hyperspectral imagers…), geometry (e.g. nadar viewing, side-looking…) and waveband (e.g. UV, MWIR…). If the mission has more than one instrument and any of these instruments has more than one technology, type, geometry or waveband, the scores will be averaged over the total number of instruments and the instruments parameters.

The figure below represents the average mission similarity score for different architectures. While in the case of the expert a clear pattern could be seen between the number of warnings and the distance to the Pareto Front, here it cannot. Although at first glance no direct relationship can be seen, it has been found that the historian can also be very useful in improving design outcomes, especially for more expert users.

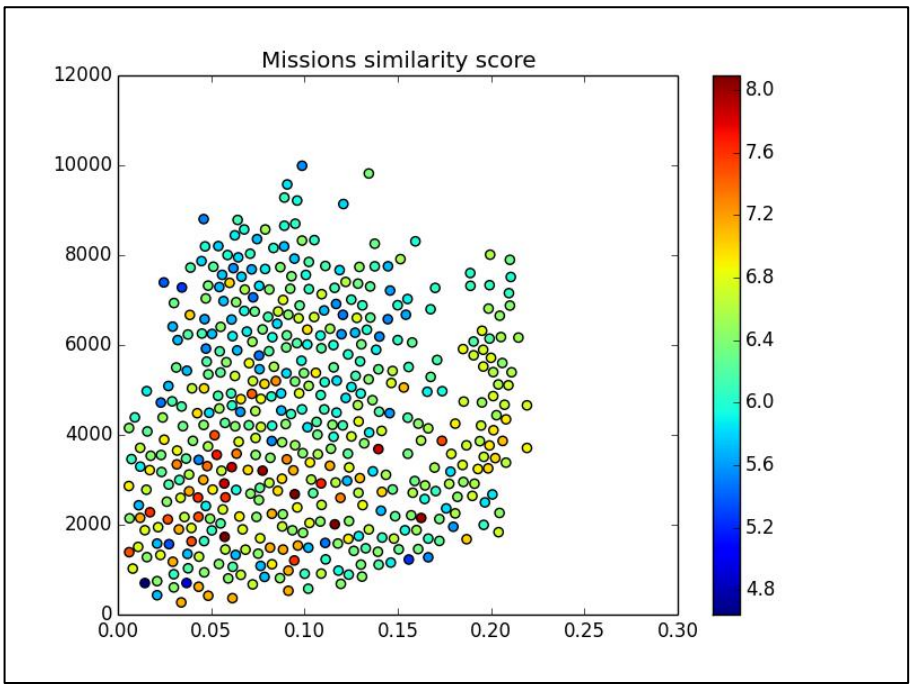

Figure 20: Average mission similarity score for different designs

Analyst (Data-driven)

The analyst notifies the user if a given architecture shares the same features that are found frequently among good designs. Again, while sharing a certain design feature with most good designs is no guarantee for success, it can be helpful information. Similarly, Daphne can use this information to suggest changes to a user-defined design, based on the good features found so far.

These features are extracted from the designs discovered by the user using a Data Mining tool developed previously in the lab [17]. This tool mines the current population for driving features (combination of architectural variables), which drive designs towards good regions of the design space. Each feature is composed by a combination of 10 types of primitive features (present, absent, in orbit, not in orbit, together, together in orbit, separate, empty orbit, number of orbits used and number of instruments). These "good" features are extracted periodically and saved to a file called EOSS_features.csv, with the format presented in the figure below.

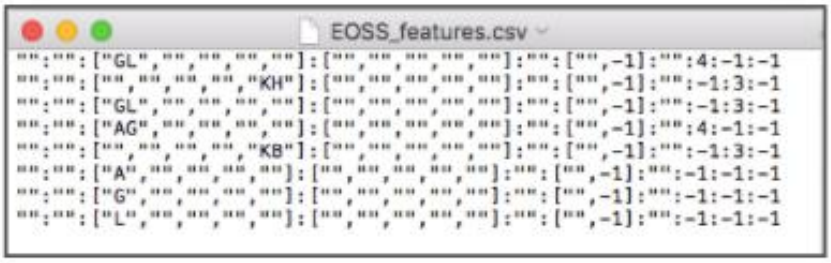

Figure 21: EOSS_features.csv file

The job of the analyst is to find out if any of the features in the file are present in the design proposed by the user and if they are display a message. The analyst does this by parsing the EOSS_features.csv file and looking if any of the features are in the current design. The more features a design has in common with those found by the Data Mining tool, the better (in theory) the architecture will be.

The next diagram tries to summarize how the analyst works. A Data Mining tool extracts "good" features from the designs discovered by the user and it saves the result to a file. Then, the analyst looks whether the design proposed by the user contains any of the features saved in the file mentioned above and if it does it notifies the user.

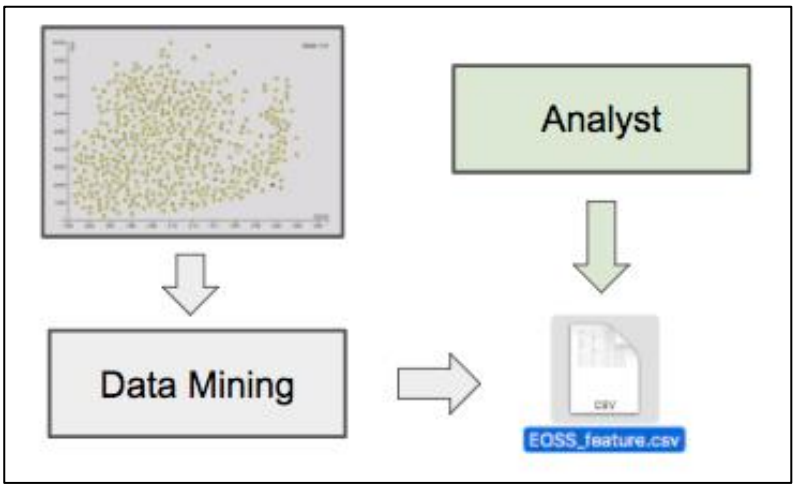

Figure 22: Diagram of the analyst

Explorer (Model driven)

The explorer searches the design space in parallel with the user and alerts him or her if it finds a better design. The explorer performs two type of searches: 1) Local: it searches around the design the user is working on and 2) Global: it searches on all the design space. The first one works by making small changes to the last design evaluated by the user (add or delete an instrument), while the second one uses a Genetic Algorithm to find the best design.

Internally each design is represented by an array of 60 ones and zeros. Every 12 ones and zeros represent an orbit, where the first number represents instrument A, the second instrument B and so on. If a value is set to one it means that such instrument is present in that orbit, and if it is set to 0 it means that it is not present. Although this representation may sound complicated, it makes the implementation of the local and global explorer much simpler.

Local Explorer: As stated above the local explorer searches around the design proposed by the user by making small changes to the design (by adding or deleting an instrument). Using the representation presented above this search only consists in flipping each of the 60 ones and zeros of the array that represents the design one by one. Below is the function, which creates the 60 possible variations of the design.

```
def permute_arch(arch):
    initialArray = arch_to_array(arch)
    res = []
    for i in range(60):
        modArray = initialArray[:]
        modArray[i] ^= 1
        res.append(array_to_arch(modArray))
    return res
```

Figure 23: Function used by the local explorer

The local explorer is useful when the user has already a good design and tries to fine tune it by making small changes to it or in a situation where adding or deleting and instrument can greatly improve a design. In the case the explorer finds a better design before the user changes its current configuration, the explorer will notify the user about its discovery.

Global Explorer As said before the global explorer searches on all the design space by using a Genetic Algorithm [18]. Very shortly these algorithms are based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals using the following basics extracted from the reference:

Individuals compete for resources and mates inside a population
Those individuals that are most successful will produce more offspring
Genes from “good” individuals propagate through the population
Each successive generation will become more suited to the environment

Table 3: Genetic Algorithms basics

The Global Explorer is implemented in Python using a library called DEAP [19] (Distributed Evolutionary Algorithms in Python), which allow easily creating and testing Genetic Algorithms in the Python programing language.

One of the most important functions in these algorithms is the evaluate function, which determines which individuals produce offspring or not, i.e. the genes of the individuals with a higher result in the evaluation function have higher probability to pass on to the next generation. Below is the code of this function, which gives the maximum score to those architectures which have maximum science and minimum cost.

```
def evaluate(individual):
    architecture = array_to_arch(individual)
    result = vassar.evaluateArch(architecture)
    # Maximize scientific benefit and minimize cost
    return result[0],-result[1]
```

Figure 24: Function used by the local explorer

Adaptation

As stated in Chapter I, one of Daphne's requirements is that it needs to be easy to adapt it to other complex tasks other than the design of constellations of satellites. Thus it was important to design the Critic Agent so that it wasn't difficult to make these changes. Below are the modifications that should be made to each of the agent: expert, historian, analyst and explorer in order to adapt them to another task.

The Expert requires the user to define new rules for each new problem. These rules need to be written in Jess and need to be placed inside one of the following files: critique_cost.clp and critique_performance.clp. Having two different files aims to organize the rules depending whether they have to do with the performance or cost.

The Historian requires the user to provide a database that is useful for the problem at hand. The historian also requires the user to define a new similarity function in order to compare the design proposed by the user with the ones in the database. Thus it is critical to correctly define how the similarity score between the design being evaluated and the deigns in the database is computed in order to provide the user with good insights of past designs.

The Analyst does not require many substantial changes from the user. The Data Mining algorithm will mine the design space an identify those driving features which are good among good designs. Then the analyst will simply check if these driving features are present in the design being evaluated and if so it will notify the user.

The Explorer requires the user to redefine the representation of the design so that it can be used by the local and global explorer algorithms. The user also needs to redefine the evaluate function on the global explorer to define which parameters we want to optimize, in our case we want to maximize the science benefit and minimize the cost.

CHAPTER IV: SYSTEM EXPERIMENT

Introduction

This section introduces a preliminary experiment performed with human subjects in order to test the efficacy of Daphne in the design of complex systems. The experiment has a duration of approximately 75 minutes and it consists of 3 parts: a short tutorial in order to familiarize the user with the problem and the interface (approximately 15 minutes), 3 phases with increasing level of difficulty where the user needs to design a different constellation of satellites (15 minutes each) and a small survey at the end (approximately 15 minutes).

The experiment looks for the possible effects on the performance (known as the Dependent Variable) caused by the number of clicks to the criticize button (known as the Independent Variable). The hypothesis of the experiment is that the performance (distance to the Utopia Point of the best design discovered by the subject) is positively correlated with the number of clicks to the criticize button, i.e. users who use the critic agent more get better results.

The experiment starts with a small tutorial consisting of an interactive step-by-step tour of the site. This is created using a library called Sheperd [21] that allows to easily creating popovers to help guide the subject through the interface. Below is an example of how it looks like and in Appendix C it can be found the full text of the tutorial.

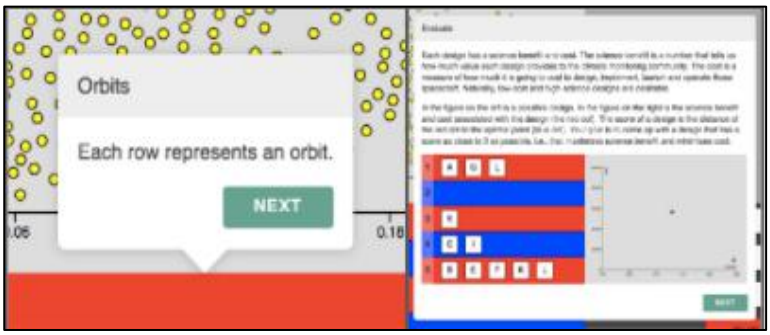

Figure 25: Screenshots of the tutorial

Once the subject has completed the tutorial, the main phase of the experiment begins, which consist on designing three constellations of satellites with increasing level of difficulty. For each phase the subject only has access to a subset of orbits and instruments

and has to try to find the best possible design in 15 minutes. Below are depicted the orbits and instruments available for each stage as well as the total number of possible designs.

Stage	Orbits available	Instruments available	Number of possible designs
1	1, 3, 4	A, B, C, D	4096
2	1, 2, 3, 4	A, B, C, D, E, F	16777216
3	1, 2, 3, 4, 5	A, B, C, D, E, F, G, H;I, J, K, L	1152921504606846976

Table 4: Orbits and instruments available

In order not to overwhelm the subject with the huge number of possible designs when all the instruments and orbits are available, it was decided to implement a system of stages where the user starts with a few instruments and orbits to choose and finishes with all of them available. The image below shows how the user knows which instruments and orbits can use or not. The instruments that cannot be used are darker while the orbits that cannot use will have a white line from side to side.

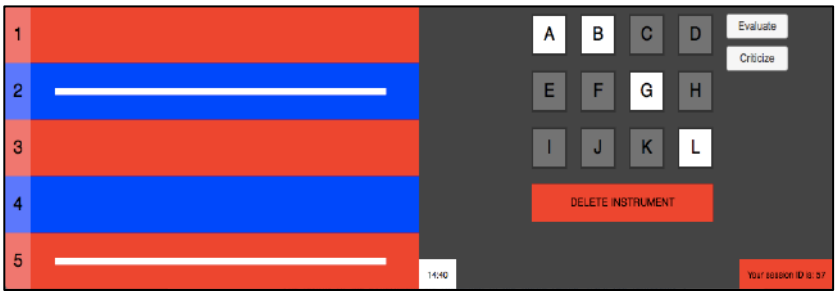

Figure 26: Screenshot of the experiment

The survey is the last part of the experiment and it consist of a small number of questions such as what is the background of the subject (e.g. engineering, physics…), how useful was the critic agent… The goal of the survey is not so much to be used for the experiment but to get feedback about how we could improve the system in order to make it more useful and intuitive for the end user.

Results

The results of the experiment are presented in the figure below. For different reasons it was only possible to perform the experiment with 4 subjects and only with stage 3 (all instruments and orbits available). Each dot represents a different subject, where the x-axis represents the number of times that the subject has clicked to the criticize button, while the y-axis represents the performance of the best design discovered by the user in 15 minutes (note that smaller the distance to the utopia point the better).

Although the results are not statistically significant, due to the small number of subject, it shows a trend that the use of the critic may leads to better designs. This trend may serve as a motivation to perform further experiments, either increasing the number of subjects or changing the design of the experiment itself. This could be done by having different groups where some have access to the critic and others not or by redefining completely the design and structure of the experiment presented in this section.

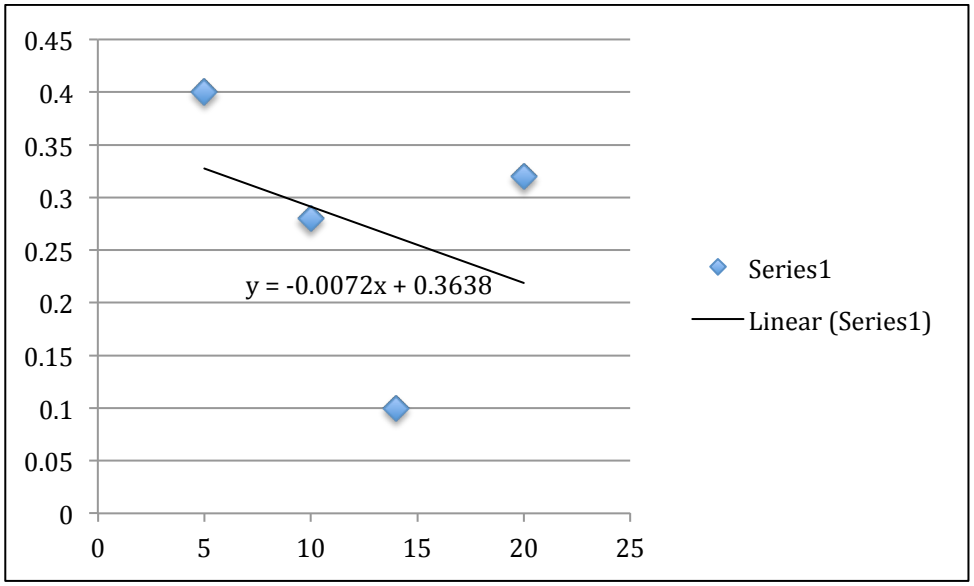

Figure 27: Result of the system experiment

Although the experiment only had 4 subjects, because of the difficulty of finding more volunteers, there exist tools, which will enable to grow very significantly the number of subjects in a very fast way. One of these tools is Amazon Mechanical Turk [22], which allow performing cognition experiments through the Internet.

Although the author of this thesis tried to perform the experiment through this platform this was not possible since it required modifying most of the experiment to adapt it to this platform and there was no time. Another problem we were worried about was that through the Internet there is no easy way to control the conditions in which the subject is doing the experiment (e.g. the subject could be watching a movie while doing the experiment) and this could greatly affect the outcome of the test.

CHAPTER V: CONCLUSION

Conclusions

This thesis presented an Intelligent Cognitive Assistant (ICA) for architecting Earth observation satellite systems. Chapter 1 provided an overview of the system and a background study, Chapter 2 presented the 3 front-end interfaces that have been developed for the user to interacts with the system, Chapter 3 introduced an intelligence able to provide criticism and suggestions about any design proposed by the user. Finally Chapter 4 presented a preliminary experiment of the system.

Three interfaces have been presented: Visual Interface, VR Interface and Robot Interface, which allow the user to interact with the system in many different ways. Also an AI able to criticize any design proposed by the user has been introduced, which allows the user to come up with better designs.

Ultimately the goal of this project is to make the first steps towards mixed-initiative human-computer design, in which humans and Intelligent Cognitive Assistants interact as collaborators in a close feedback loop. Although the results of the experiment presented in Chapter 4 do have a small statistical relevance, due to the small number of subjects, it hints that this kind of interactions may lead to better designs that if it was only the human or the computer who performed the task.

Finally say that the code of this project (front-end interfaces and critic agent) has been made available in Github in the following repositories:

- Visual + critic: https://github.com/seakers/daphne-visual
- VR interface: https://github.com/seakers/daphne-VR
- Robot interface: https://github.com/seakers/daphne-robot

Future work

Concluding this report, it is worth mentioning that the objectives specified have been fulfilled, and the basis for a fully functional system has been established. Besides this, there are still parts to implement and some of the implemented parts have still some ground for improvement. Below are listed some of the possible improvements for the different parts.

Front-end interfaces: add new functionalities to the Visual Interface (e.g. capability to filter architectures in order to identify driving features). Port the VR interface to other platforms such as HTC Vive or Oculus Rift, which will allow for new forms of interaction. Optimize the design of the Robot Interface in order to better fit all the electronics. Finish implementing and testing the software of the Robot interface.

Critic Agents: Right now, the Critic only intervenes if requested by the user. In order to honor the mixed-initiative nature of Daphne, we should also allow Daphne to intervene even when she hasn't been asked to.

System Experiment: Perform the experiment with more subjects and different groups, where some have access to the critic agents and others not. This will allow a between and within subject design and perform an Analysis of Variance (ANOVA) test.

Research contributions

The following are research publications in which the author of this project has contributed. The publications include a Conference Paper, which has been accepted for presentation at the AIAA SciTech Forum 2018 and a poster presented by the author at the AIAA Intelligent Systems Workshop 2017.

- **A. Prat** and D. Selva "Daphne: Design of Visual, VR and Embodied Interface", AIAA Intelligent Systems Workshop, University of Michigan (Ann Arbor), Poster.

- H. Bang, A. Virós, **A. Prat**, D. Selva "Daphne: An Intelligent Assistant for Architecting Earth Observing Satellite Systems", AIAA SciTech Forum 2018, Conference Paper.

APPENDIX A

Orbit names and table of equivalences with CEOS database

Instrument alias	Instrument name
1	LEO-600-polar-NA
2	SSO-600-SSO-AM
3	SSO-600-SSO-DD
4	SSO-800-SSO-DD
5	SSO-800-SSO-PM

Table 5: Orbits alias and names

	Type	Period	Sense	Inclination	Altitude	Longitude	Orbit LST	Repeat cycle
LEO-600-polar-NA	Inclined, non-sun-synchronous	ANY	ANY	90 deg (±5)	600 Km (±50)	ANY	ANY	ANY
SSO-600-SSO-AM	Sun-Synchronous	ANY	ANY	ANY	600 Km (±50)	ANY	7:30 - 11:30 (AM-PM)	ANY
SSO-600-SSO-DD	Sun-Synchronous	ANY	ANY	ANY	600 Km (±50)	ANY	5:30-6:30 (AM-PM)	ANY
SSO-800-SSO-DD	Sun-Synchronous	ANY	ANY	ANY	800 Km (±50)	ANY	5:30-6:30 (AM-PM)	ANY
SSO-800-SSO-PM	Sun-Synchronous	ANY	ANY	ANY	800 Km (±50)	ANY	1:00-3:30 (AM-PM)	ANY

Table 6: Orbits table of equivalence

Instrument names and table of equivalences with CEOS database

Instrument alias	Instrument name
A	ACE_ORCA
B	ACE_POL
C	ACE_LID
D	CLAR_ERB
E	ACE_CPR
F	DESD_SAR

G	DESD_LID
H	GACM_VIS
I	GACM_SWIR
J	HYSP_TIR
K	POSTEPS_IRS
L	CNES_KaRIN

Table 7: Instruments alias and names

	Instrument Type	Technology	Geometry Type	Instrument Wavebands
ACE_ORCA	"Ocean colour instruments"	"Medium-resolution spectro-radiometer"	"Cross-track scanning"	"UV", "VIS", "NIR", "SWIR"
ACE_POL	"Multiple direction/polarisation radiometers"	"Multi-channel/direction/polarisation radiometer"	ANY	"VIS", "NIR", "SWIR"
ACE_LID	"Lidars"	"Atmospheric lidar"	"Nadir-viewing"	"VIS" "NIR"
CLAR_ERB	"Hyperspectral imagers"	"Multi-purpose imaging Vis/IR radiometer"	"Nadir-viewing"	"VIS", "NIR", "SWIR", "TIR", "FIR"
ACE_CPR	"Cloud profile and rain radars"	"Cloud and precipitation radar"	"Nadir-viewing"	"MW"
DESD_SAR	"Imaging microwave radars"	"Imaging radar (SAR)"	"Side-looking"	"MW", "L-Band", "S-Band"
DESD_LID	"Lidars"	"Lidar altimeter"	ANY	"NIR"
GACM_VIS	"Atmospheric chemistry"	"High-resolution nadir-scanning IR spectrometer"	"Nadir-viewing"	"UV", "VIS"
GACM_SWIR	"Atmospheric chemistry"	"High-resolution nadir-scanning IR spectrometer"	"Nadir-viewing"	"SWIR"
HYSP_TIR	"Imaging multi-spectral radiometers (vis/IR)"	"Medium-resolution IR spectrometer"	"Whisk-broom scanning"	"MWIR", "TIR"
POSTEPS_IRS	"Atmospheric temperature and humidity sounders"	"Medium-resolution IR spectrometer"	"Cross-track scanning"	"MWIR", "TIR"
CNES_KaRIN	"Radar altimeters"	"Radar altimeter"	"Nadir-viewing"	"MW", "Ku-Band"

Table 8: Instruments table of equivalences

APPENDIX B

BoM of materials for the Robot Interface

NAME	URL	QUANTITY	PRICE (USD)
5" TFT display with touchscreen	https://www.adafruit.com/products/1596	1	39,95
40-pin FPC Extension Board + 200mm Cable	https://www.adafruit.com/product/2098	1	4,50
Adafruit DPi TFT Kippah for Raspberry Pi with Touch Support	https://www.adafruit.com/product/2453	1	19,95
Raspberry Pi 3 - Model B - ARMv8 with 1G RAM	https://www.adafruit.com/products/3055	1	39,95
Micro Servo - High Powered, High Torque Metal Gear	https://www.adafruit.com/products/2307	2	23,90
Arduino UNO R3 ATmega328P	https://www.amazon.com/MakerBest-Quality-Compatible-ATmega328P-Development/dp/B00Q6ZW4NO/ref=sr_1_8?s=electronics&ie=UTF8&qid=1493829320&sr=1-8&keywords=Arduino+uno	1	9,44
28BYJ-48 DC 5V Stepper Motor	https://www.amazon.com/KOOKYE-28BYJ-48-Stepper-ULN2003-Arduino/dp/B019TOJRC4/ref=sr_1_7?s=electronics&rps=1&ie=UTF8&qid=1493829239&sr=1-7&keywords=stepper+motor&refinements=p_85%3A2470955010	1	8,59
External USB Audio Sound Card Adapter	https://www.amazon.com/dp/B000N35A0Y/ref=twister_B0722P1LQ9?_encoding=UTF8&psc=1	1	6,99
Power Gear 98950 PC Microphone	https://www.amazon.com/Power-Gear-98950-Microphone-Detachable/dp/B000BYCNKU/ref=sr_1_4?s=pc&rps=1&ie=UTF8&qid=1493828865&sr=1-4&keywords=microphone&refinements=p_85%3A2470955011	1	4,99
U19-A Night Vision Webcam 12.0MP	https://www.amazon.com/Night-Vision-Webcam-12-0MP-Microphone/dp/B003VY4M42/ref=sr_1_1?s=pc&rps=1&ie=UTF8&qid=1493828993&sr=1-1&keywords=Webcam&refinements=p_85%3A2470955011	1	7,67
M2.5 Brass Spacer Standoff/Screw/Nut Assortment Kit	https://www.amazon.com/HVAZI-Standoff-Stainless-Assortment-Male-Female/dp/B01L0I6CUJG/ref=pd_day0_147_1?_encoding=UTF8&pd_rd_i=B01L06CUJG&pd_rd_r=G3C4XH0PVY1X9YGT2ZRN&pd_rd_w=yQW9a&pd_rd_wg=Yysd5&psc=1&refRID=G3C4XH0PVY1X9YGT2ZRN	1	11,89
Sensor Shield Servo Motor for Arduino UNO	https://www.amazon.com/SainSmart-Digital-Analog-Arduino-Duemilanove/dp/B0076FWAAK/ref=pd_sim_23_6?_encoding=UTF8&pd_rd_i=B0076FWAAK&pd_rd_r=CWPCABR0YJMV4DPRDGSD&pd_rd_w=JTZ0k&pd_rd_wg=czCzB&psc=1&refRID=CWPCABR0YJMV4DPRDGSD	1	9,99
			187,81

APPENDIX C

Full text of the tutorial of the experiment of the system

1. Experiment overview (body)

In this experiment, you will design a constellation of satellites for Earth observation. Don't worry if you don't know much about the topic since all the details have been left out so that you will only deal with simple numbers and letters. A design (i.e., a satellite constellation) is defined by a set of satellites, where each satellite has a set of instruments (sensors) and an orbit (altitude, inclination). A good design has a high science benefit and low cost. There are a total of 5 candidate orbits (represented by numbers) and 12 candidate instruments (represented by letters).

<Image https://selva-research.com/daphne-VR/img/img1.jpg>

2. Orbits

Each row represents an orbit.

3. Instruments

Each block represents an instrument.

4. Your first design

Drag instrument A to orbit 3.

5. Your first design

Click to evaluate your design (i.e., to calculate its science benefit and cost).

6. Your first design

Your design should appear as a red dot in the science-cost scatter plot.

7. Evaluate (body)

Each design has a science benefit and cost. The science benefit is a number that tells us how much value each design provides to the climate monitoring community. The cost is a measure of how much it is going to cost to design, implement, launch and operate those spacecraft. Naturally, low-cost and high-science designs are desirable. In the figure on the left is a possible design. In the figure on the right is the science benefit and cost associated with this design (the red dot). The score of a design is the distance of the red dot to the optimal point (blue

dot). Your goal is to come up with a design that has a score as close to 0 as possible, i.e., that maximizes science benefit and minimizes cost.

<Image https://selva-research.com/daphne-VR/img/img2.jpg>

8. Criticize

Click to criticize your design.

Criticize

The comments and suggestions from the critic agents should appear here.

Criticize (body)

The critic agents will give you valuable information to help you improve your design. There are four types of critic agents: the expert, the historian, the analyst and the explorer. The expert has many years of experience designing spacecraft for climate monitoring. The historian has access to a database of past missions and thus can recommend things based on what was done in the past. The analyst uses statistical techniques to look for trends in the current dataset and recommend changes to your design based on that. The explorer will discover new design for you and it will tell you if it finds someone good.

<Image https://selva-research.com/daphne-VR/img/img3.jpg>

Criticize

The expert suggests moving instrument A to another orbit.

Criticize

Try to move instrument A to orbit 1 and re-evaluate your design.

Criticize

You should see that the science benefit of your design has improved.

Criticize

Try also to take advantage of the historian, analyst, and explorer suggestions.

Timer

You will have 15 minutes to find the best design in each round.

Round 1 (body)

In this round you will have the following orbits and instruments available:

Orbits: 1, 3, 4

Instruments: A, B, G, L

Round 2 (body)

In this round you will have the following orbits and instruments available:

Orbits: 1, 2, 3, 4

Instruments: A, B, C, D, E, F

Round 3 (body)

In this round you will have the following orbits and instruments available:

Orbits: 1, 2, 3, 4, 5

Instruments: A, B, C, D, E, F, G, H, I, J, K, L

Round 4 (body)

Conclusion of the experiment. Thank you for participating!

Questions asked in the survey of the experiment

Question	**Answers**
What is your education background?	Education background (e.g. aerospace engineering).
The critic agent was useful • The expert was useful? • The historian was useful? • The analyst was useful? • The explorer was useful?	Number between 1 to 5 1 Strongly disagree 2 Somewhat disagree 3 Neutral 4 Somewhat agree 5 Strongly agree
What would you improve?	Free form
Comments and suggestions?	Free form

Table 9: Questions asked in the survey

BIBLIOGRAPHY

[1] Selva, Crawley, “VASSAR: Value assessment of system architectures using rules”, IEEE Aerospace Conference, 2013

[2] Myers et al., “ An intelligent personal assistant for task and time management”, AI Magazine, Vol. 20, No. 2, 2007, pp. 47.

[3] Engelbart, Douglas C, “Augmenting human intellect: a conceptual framework”, PACKER, Randall and JORDAN, Ken. Multimedia. From Wagner to Virtual Reality. New York: WW Norton & Company, 2001, pp. 64-90.

[4] Wikipedia, “Siri” [Online] Available: https://en.wikipedia.org/wiki/Siri

[5] Mycroft, “Mycroft AI – Open Source Artificial Intelligence Voice Assistant” [Online] Available: https://mycroft.ai/

[6] Komninos et al., “HealthPal: an intelligent personal medical assistant for supporting the self-monitoring of healthcare in the ageing society”, Proceedings of the UbiHealth, 2006.

[7] Onken et al., “Assistant systems for aircraft guidance: cognitive man-machine cooperation”, Aerospace Science and Technology, 2001, Vol. 5, No. 8, pp. 511-520.

[8] Wikipedia, “Pareto Front” [Online] Available: https://en.wikipedia.org/wiki/Pareto_efficiency

[9] D3.js, “D3.js – Data-Drive Documents” [Online] Available: https://d3js.org/

[10] jQuery, “jQuery” [Online] Available: https://jquery.com/

[11] Django channels, “Django channels – Channels 2.0a1 documentation” [Online] Available: https://channels.readthedocs.io/en/stable/

[12] Nginx, “Nginx wiki” [Online] Available: https://www.nginx.com/resources/wiki/

[13] Three.js, “Three.js – Javascript 3D library” [Online] Available: https://threejs.org/

[14] OpenScad, “OpenSCAD – The programmers Solid 3D CAD Modeller” [Online] Available: http://www.openscad.org/

[15] 3D Hubs, “3D Hubs” [Online] Available: https://www.3dhubs.com/

[16] Xu, Li D, “Case based reasoning”, IEEE potentials, 1994, Vol. 13, No. 5, pp. 10-13.

[17] ESA, “The CEOS database: mission, instruments and measurements” [Online] Available: http://database.eohandbook.com/

[18] Bang, Selva, “iFEED: Interactive Feature Extraction for Engineering Design”, IDETC/CIE ASME International Design Engineering Technical Conferences & Computer and Information in Engineering Conference, 2016

[19] Shepherd, “Shepherd” [Online] Available: http://github.hubspot.com/shepherd/

[20] Amazon, “Amazon Mechanical Turk” [Online] Available: https://www.mturk.com/