Marcel Llopis, Xavier Franch, and Manel Soria. *Assessing the Usability of Two Declarative Programming Languages to Model Geometric Events.* Journal of Aerospace Information Systems, September 2023, 20(9): 546-554.
https://doi.org/10.2514/1.I011207

Assessing the usability of two declarative programming languages to model geometric events in space

Marcel Llopis[1]
NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, United States

Xavier Franch,[2] and Manel Soria[3]
Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain

When space missions plan scientific actions for robotic spacecraft to execute, they frequently do so within a geometric context called an opportunity. While there are geometric software libraries that let users write code to search for opportunities, they require knowledge of algorithms and imperative programming languages, which is a condition that might exclude a potentially large population of scientists. Additionally, there might be more user-friendly software systems for scientists to model and search for opportunities, but those might exclude other missions due to export concerns, or an inability to maintain such software due to lack of staff or funding. To address these concerns, we designed two different computer languages to model opportunities. In this paper, we present these two languages, our study to evaluate their relative readability and usability, and results obtained in our research along with an interpretation of the same. The metric for this study has been a questionnaire with active exercises, statements with corresponding responses on a Likert scale, and open-ended questions to elicit qualitative responses. The study's quantitative results provide us with relative and absolute quantification of the usability and readability of each language, while the study's qualitative results help us direct future language design decisions.

I. Introduction

Usability, defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [1], has been a concern for NASA since the

[1] Deputy Section Manager, Flight Software and Avionics Systems Section, AIAA Senior Member. Corresponding Author; marcel.llopis@jpl.nasa.gov
[2] Professor, Software and Service Engineering Group (GESSI).
[3] Professor, Aerospace Engineering Division.

early stages of space exploration. The practicality of this concern is evident, in that increased usability should result in user performance gains, as both ground personnel and astronauts can complete more successful tasks per unit of time. Beyond practicality, it is also a risk mitigation strategy, in that more usable systems reduce the likelihood that operation errors will impact the crew or the spacecraft. Space mission software, however, especially *mission planning software* for robotic unmanned space missions, has only recently started to enjoy a more inclusive application of usability assessment and guiding techniques. This is due to the fact that the idea of multi-mission software, that is, software that can be reused across missions with minimal to no changes, is also a new development for governmental space agencies. This can be exemplified by the fact the Human-Computer Interaction software group within NASA's Jet Propulsion Laboratory (JPL) was only stood up in the mid-2010's. While this is a recently-formed group, the return it has provided to the area of mission planning software has been quite remarkable as it has played a key role in improving the design and architecture of software, implementing successful design principles within mission processes, and developing interfaces that mission staff use to interact with mission software. Today at JPL it is common practice to staff software projects with usability experts and to have engineers apply industry-accepted usability practices. In this manuscript we will provide an example of our research that integrates usability practices within the design and development of mission planning software at JPL. Specifically, we assess the usability of two different textual languages to enable engineers and scientists to model geometric events in space effectively, easily, and satisfactorily. In this specific context, "geometric events" refers to relevant geometric constraints between bodies in space that must be satisfied to facilitate the planning of spacecraft maneuvers.

In order to accurately frame our research, we need to spend a few paragraphs describing the process wherein human staff use software to command a robotic spacecraft. This process is named mission planning and it is composed of two distinct repeating phases named *uplink* and *downlink*. The overarching purpose of the uplink process is to turn scientific goals and mission constraints into orders that are executable by the spacecraft's hardware. Whereas the downlink process is concerned with obtaining (i) scientific data, e.g., pictures, spectroscopic data, altitude measurements, and (ii) engineering data, i.e., telemetry, from the spacecraft after the uplink orders have been executed. These uplink and downlink processes involve several sub-processes. Most notably, the *science planning* step is an uplink sub-phase common to many space missions. In this phase, a team of scientists decide what actions the spacecraft's science hardware, such as visible spectrum cameras or mass spectrometers, need to execute within a geometric context in order to obtain the desired data to partly or fully satisfy the scientific goals of the mission.

The science planning process for *robotic orbital missions*, that is, space missions where a spacecraft orbits a target body, typically starts with the definition of certain geometric events that are of interest to scientists. This is necessary because, in order to plan the actions spacecraft instruments need to execute, there has to be a geometric context that is pertinent to the data-gathering aspirations of such space missions. As an example, if a spacecraft is orbiting Jupiter, scientists might want to know when Jupiter's moon Europa is not occulted by Jupiter in order to be able to take a series of pictures. These events of interest to scientists are called *opportunities* and the action of finding when they occur is called *opportunity search*.

Opportunity search is a recurring topic, not only within science planning, but also in other parts of the uplink process. Depending on the level of automation of the uplink process and the stage in which a mission is in the uplink process, opportunity search might have a human trigger, as in a scientist trying to search for an event of interest [2], but it can also be driven by automation software that will try to schedule activities when it makes sense from a geometric perspective [3]. In both cases, there is a need to standardize opportunity search from a software perspective. This was discussed in [4], where we described the Tychonis framework, which provides software developers with the ability to describe (i) geometric events in code, and (ii) the algorithms to find them in a structured and extensible manner via a metamodel [5]. Tychonis can be integrated with existing software in a way that the addition of new searchable geometric events into the framework will automatically propagate to software that uses the framework without having to modify such software.

While Tychonis proved that creating an extensible and reusable opportunity search framework is possible, there are hurdles to using it by individuals who are not experienced software developers. If there is a need to search for a geometric event, one can write custom code that uses the Tychonis framework, compile the code, and execute it. The outcome of the execution step would be a time window in which the geometric event of concern takes place. In this case, the code produced could have an ephemeral existence as it might only be useful to search for just one event. Another option is to integrate a geometry software library such as Tychonis with a host application that will templatize geometric events. With such application, the user would be able to trigger the execution of a multitude of event searches through the host software. In fact, this pattern was described for one of NASA's software tools, the Science Opportunity Analyzer (SOA) [6] in [7]. SOA users model their events via a user interface (UI), and once satisfied with the definition of such an event, they make a request to the software to search for it with Tychonis. However, space missions frequently encounter themselves in cost-constrained environments that result in limited availability of

staff, and this might cascade in science teams not having access to software developers that can write code to search for a specific event. It is also possible that science teams might not have access to elaborate science planning software such as SOA to search for events in a templatized manner.

For the reasons above, we propose the idea of a textual computer language that can be used to model and search for geometric events. The aim is to design a language that (i) can provide univocal textual representations of geometric events in space, and (ii) can be used by space mission scientists and engineers with *just enough* programming experience. In the pursuit of this goal, we designed two approaches for a textual language and compared their usability through the use of a survey. The purpose of this manuscript is to detail our approach to the two language options, the design of the survey, the results from the same, and its practical implications. Section 2. Background will describe the reasoning behind our language paradigm choice along with a discussion on prior usability assessment ideas for textual languages. Section 3. Language Options will cover the differences between the two languages we designed and why it would be useful to evaluate usability of these two specific languages. Section 4. Study Design will describe the instrument used in our evaluations, the user populations involved in the study, how the study was executed, the numerical analyses we conducted, and known possible threats to validity going into the study with our instrument choices. Section 5. Results will provide a descriptive view of the results both on the quantitative and qualitative planes. Finally, Section 6. Discussion will include an interpretation of the results, a higher-level interpretation of the consequences of our findings, and a retrospective on the study.

II. Background

There are two aspects of prior research to be considered in this paper. One is in regards to applicability of different programming language paradigms, whereas the other is the usability assessment of software systems.

A. Programming Paradigm

The declarative programming paradigm promotes the idea that there are types of computer programs that are more apt to be modeled in terms of the description of a problem than in terms of the algorithm needed to solve the problem [8]. The former vision is known as declarative programming and can be exemplified by functional, logic, and constraint languages, but also by Domain-Specific Languages (DSL) such as the Structured Query Language (SQL) [9], which found objective success to model data operations within the realm of Relational DataBase Management Systems (RDBMS) [10]. The latter vision is known as imperative programming and it is promoted by procedural and

object-oriented programming general-purpose programming languages. Typical examples of these sub-approaches to imperative programming include storied languages such as Pascal, FORTRAN, C/C++, Java, and Python.

The declarative vs. imperative programming debate has been ongoing for generations, and at the moment, it is accepted that for scoped and suitable problems, the declarative approach produces more understandable, learnable, accessible and communicable programs. The author of [11] defends that "declarative programming involves stating what is to be computed, but not necessarily how it is to be computed"; the author also explains that the kind of programs that benefit from declarative programming are the ones in which (i) deduction is left to the system - that is, there is no requirement to explain how knowledge will be used by the system; and, (ii) knowledge is composed of independent facts – that is, control flow is not known by the facts stated in a program. The complement of the latter assertion is embodied in imperative programs by their need to orchestrate the use of functions, variables, classes, etc. within their control flow. From this and the use of the Kowalski equation 'algorithm = logic + control' [12], the lack of control leaves logic alone, which is declarative by nature, and could still be the minimal expression of a geometric event that doesn't incorporate its search method. On the other hand, [11] explains the real advantages posed by imperative programming manifest themselves (1) when capturing processes, (2) when second order knowledge need to be expressed, and (3) in order to capture heuristic knowledge. In our case, we are not trying to capture a process and certainly not second order nor heuristic knowledge, as we believe a geometric event to be a description of an event that "just is" as a logical statement, with no algorithmic interpretation. A more practical discussion thread, by proof of contradiction, is that if we were to accept that an imperative language is more suitable to model geometric events in space, we would end up developing a language that provides little value over existing languages such as C combined with a library such as SPICE [13]. SPICE is a C library that provides a collection of low-level functions to perform geometric calculations in space. Learning such language would be a lofty requirement for scientists and engineers working for space missions, just as it would be a lofty requirement for them to learn both C and SPICE. Plus, resulting programs would entangle the definition of a geometric event with the algorithmic method to search for the time window in which that event takes place. In all, reading and writing events in such language would require specialized training in computational geometry and programming, as users interested in modeling an event would need to know how to algorithmically describe it.

As a consequence, in this study we are not considering designing an imperative language due to the fact that the action of describing a geometric event in space is a declarative action in nature. Besides, an imperative language would

not improve on the state of the art and would alienate a portion of the user base we would like to target. From the realm of separation of concerns, how an event is searched for with algorithms is one responsibility and the modeling of an event is another. Frameworks such as Tychonis [4] or libraries such as SPICE own the algorithmic search concern whereas the day-to-day modeling of opportunities by mission staff shall be a concern owned by the declarative language. In effect, the intent of our declarative approach is only to describe and communicate geometric events that could be interpreted and searched for further downstream by more down-to-the-metal modeling and algorithmic resolution engines. are two aspects of prior research to be considered in this paper. One is in regards to applicability of different programming language paradigms, whereas the other is the usability assessment of software systems.

B. Programming Language Usability

The author of [14] noted that the usability of a system is "the capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and user support, to fulfill the specified range of tasks, within the specified range of environmental scenarios". Similarly, the ISO 9241-11 standard [1] considers usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". These two definitions overlap in explaining that usability does not only consider the completion of a set of tasks by a user, but also effectiveness, ease of use, and satisfaction. Our use of the term 'usability' adheres to this doctrine throughout our research and this document; in other words, the aim of our language is for space mission scientists and engineers to effectively, easily, and satisfactorily define geometric events in space.

The authors of [15, 16] describe several methods to evaluate the usability of a software system. Those are Laboratory Testing, Thinking Aloud, Formal Modeling, Guidelines/Checklists, and Heuristic Evaluation. In [17], the authors explain that these methods "are either difficult to apply, or dependent upon the evaluators' expertise", a statement with which we would concur in regards to the needs of our study. The authors of [18] also recognized these issues, and added that "often, all that is needed is a general indication of the overall level of usability of a system compared to its competitors or its predecessors" and provided an alternative 'quick and dirty' method known as the System Usability Scale (SUS), which has found widespread use within the software industry due to its low cost and high returns.

The SUS is a questionnaire composed of ten statements, where five of those are stated in a positive manner, such as "I think that I would like to use this system frequently" and the remaining five are stated in a negative manner, such

as “I found the system unnecessarily complex”. SUS respondents rate each statement on a Likert 1-5 scale, where 1 is “Strongly Disagree” and 5 is “Strongly Agree”. The SUS method then proposes a calculation that produces a number on a 0-100 scale, where higher numbers denote more usability. This number is obtained through arithmetic operations described in [18] that penalize higher numbers on the Likert scale for answers to ‘negative’ statements, and favor higher numbers on the Likert scale for answers to ‘positive’ statements. [19] concluded that their analysis of a large number of SUS scores showed that “the SUS is a highly robust and versatile tool for usability professionals”. The industry sentiment about the SUS is overall positive, and it would seem sensible to consider the SUS as a tool to use in our study. Besides, in our case, where we compare programming language options, the SUS presents two unique advantages besides low cost of execution and other usually-debated factors: (i) it can provide unambiguous quantitative measures to compare different language options via the Likert scale responses and the overall 0-100 score, and (ii) it can also be a tool to obtain qualitative thoughts *à la* “thinking aloud” if we ask potential users if they would like to volunteer thoughts as they complete the study.

III. Language Options

In our research, we considered two distinct options as declarative languages to model geometric events. One follows in the footsteps of SQL, in that it provides a natural language approach, similar to the English language, to define such events. The other option is more structural in nature, based on key-value tuples, and resembles JavaScript Object Notation (JSON) [20]. These options were chosen for two reasons. Firstly, both SQL and JSON were designed to capture human-readable parameterized statements, and although JSON is less constrained when not used with a schema [21], SQL’s stricter grammar leads the user to follow a constrained pattern to describe a database query. While in our case we are not defining a database query, there is a value in defining an event with a strict grammar, as this would be conducive towards a concise and unambiguous geometric event. Secondly, we are adapting to what scientists and engineers are used to in their daily work within a space mission. On the one hand, all scientists and engineers communicate geometric events to each other in a natural language; hence, a SQL-like more natural language would prove similar to the way they are used to express themselves amongst each other. On the other hand, many functions in space missions are performed by scientists and engineers who operate mission planning software on a daily basis. Those are the types of users who are used to reading XML [22], JSON, or YAML [23] files, which are based on the same overall structural principles; hence, the JSON paradigm is not unknown to our target user base.

Let us consider the event "*Between the start of year 2000 and the end of year 2005, the distance between the Earth and the Moon is less than 400,000 km.*" This is a logically complete distance range event between two bodies. Its semantics are defined via the use of the English language, it contains the two bodies as nouns, and its units are explicitly defined[1]. The way this event would be modeled with the natural language approach, which we will name Natural-Language-Based (NLB), and with the more structural key-value approach, which we will call Key-Value Pair (KVP), is captured in Table 1. At first glance, one can notice preliminary advantages and disadvantages of each option. For instance, while NLB seems to be more readable, it makes event parameters, i.e., "Moon", "Earth", "400000km", and "01/01/2000:12/31/2005", potentially less visible. In contrast, KVP makes event parameters explicit, whereas understanding the event holistically might take more effort. We believe the fact these languages are complementary in their qualities makes them appropriate for a usability study in which users shall be able to explain what language they believe to be more useful for their purpose.

Table 1. The event "*Between the start of year 2000 and the end of year 2005, the distance between the Earth and the Moon is less than 400,000 km.*", modeled with the two proposed languages.

NLB	KVP

`DEF event1 AS`
`  DISTANCE FROM Moon TO Earth`
`  LESS THAN 400000km`
`  DURING 01/01/2000:12/31/2005`
	`DistanceQuery event1{`
	`  observer: "Moon"`
	`  target:  "Earth"`
	`  test: <`
	`  amount: 400000km`
	`  start_time: 01/01/2000`
	`  end_time: 12/31/2005`
	`}`

IV. Study Design

Following the Goal-Question-Metric [24] paradigm, let us state that (i) our goal is to determine a usable language approach to model geometric events in space; (ii) the question is *what is the relative usability, including readability, of two distinct declarative languages*?; and (iii) the metric is a questionnaire with active exercises, statements with corresponding Agree-Disagree responses on a Likert 1-5 scale, and qualitative statements from the respondents. Aspects (i), (ii), and (iii) will be discussed in this section.

[1] Note that in this study we concentrate on the events in terms of structure, and less so on the on the concretization of their parameters, which in this specific example would include units for distance (km, mi, AU, etc.), and when exactly the start or end of a year is.

A. Instrument

We designed our study around a survey whose objective is to obtain quantitative and qualitative data that can speak for the usability of the two different language options. These data will ultimately aid in directing our future choices to implement a language to model geometric events. To that end, the survey is implemented as a questionnaire with these distinct parts: (1) a first page - Section 1 - where respondents fill out demographics information; (2) two pages - Sections 2 and 3 - where respondents, given two examples in each language, perform an exercise where they model events similar to the examples; and (3) a fourth page - Section 4 - where respondents are asked if they think each language option is readable by scientists and engineers, and respond on a Likert 1-5 scale, where 1 is "Strongly Disagree" and 5 is "Strongly Agree". This question is particularly relevant to the study because we believe that readability plays a key role in the success of such language, as there will be stakeholders within a mission who will not write events in the language, but will be involved in determining whether an event makes sense within a mission's scientific context. Finally, after the fourth page, respondents start a SUS questionnaire - Sections 5 to 14 - with all its ten usual questions, which are asked for each language option. The questionnaire, in its entirety, can be found in [25].

B. Population and Sample

The projected population for a textual language to model geometric events includes space mission engineers and scientists who plan the actions spacecraft will need to perform to achieve the goals of the mission. Within this population, one of the main factors to take into account is what level of computer programming literacy these individuals possess. From our experience within NASA's Jet Propulsion Laboratory (JPL), individuals who graduated from college more recently have an increased ability to use a programming language compared to individuals who graduated from college less recently. We believe this is not due to the fact that those skills are lost over time, but rather because current science and engineering college programs stress software usage and development more compared to programs of yesteryear.

Given the trend continues to favor a growing user base with more software knowledge, we placed an emphasis on designing a sample that will consider the kind of users who would be using such language for the next 20 to 30 years. This could include recent JPL hires, JPL interns, or college students with a relevant background who are not associated with JPL but could be hired into it. In our study, we picked samples from two different sources. One was JPL, where we selected recent space mission hires and interns, most of them with an Aerospace Engineering background. All of these individuals worked within JPL's Planning and Execution Systems (PES) section, and were involved, directly or

indirectly, in the planning of science or engineering spacecraft activities. The PES section's charter is to staff and manage large-scale space missions during the operations phase, particularly the uplink and downlink processes. More notably, the PES section contains the Science Planning group, which uses geometry as an input for the planning of spacecraft actions and where most of the users of a proposed language to model geometric events would originate from. In consequence, if we know the Science Planning group currently contains 11% of the personnel within the PES section, and we assume most mission operations staff will be provided by the PES section, we can project around 10% of mission operations staff can benefit from a textual language to model geometric events. We expect a similar proportion of mission staff would benefit in missions not managed by JPL or NASA.

The source for the other sample was students within the Aerospace Engineering program at the Universitat Politècnica de Catalunya (UPC). This sample was similar in terms of interests, background, and training to the JPL sample, with the difference that the individuals in the UPC sample had less knowledge of space mission planning. Another characteristic is that the native language of the individuals within the UPC sample was not English, although they had a very good professional command of the language. This enriched the overall sample as there should not be an expectation that users a proposed textual language shall be native English speakers.

In all, these two samples are representative given the experience and educational caliber and background of the individuals make them candidates for a role in an institution where they would model geometric events for a space mission. An observation we can make between these samples and the current population in an institution such as JPL is that the individuals in our samples have a deeper background in computer programming compared to graduates of the past. This is by design since we see a trend where relevant university graduates are entering the workforce with a more substantial programming background anyway; hence it is salient to perform our study with individuals that are representative of future space mission staff and that will be using such language for years to come.

C. Execution

In order to prepare for the execution of the study, we piloted the survey with five JPL interns with a relevant background. Note that these five individuals were not part of the actual study, but were only used for this preliminary phase. This phase was conducted as a form of dry run, but also to find faults in the survey and, to some degree, assess the quality of the survey format. The survey questions were assembled as a web-based questionnaire, and the questionnaire itself was conducted live via one-on-one video conference. One of the authors held the responsibility to act as a guide for the survey during the video conference, for which we allocated one hour with each respondent. In

this process, the guide explained the objectives of the study, the background for the research, then proceeded to (1) send the web link for the survey response interface to the respondent, (2) observe the respondent as they were completing the answers for each question, and (3) ask, for each question, if the respondent had any qualitative feedback about the questions as they appeared in the current survey section. Note that the objective of this piloting phase was not to obtain data that would be incorporated into the results of the study, but, as noted, to practice the execution of the interview, and to acquire qualitative data about the survey itself in case that it needed to be modified to improve it. The most relevant results obtained in the piloting phase were:

(1) Respondents noted the definition of the word "use" in Sections 5, 7, 8, 11, and 12 might be ambiguous and that it could mean (a) writing geometric events with the language, (b) reading geometric events described with the language, or (c) a combination of both writing and reading. From this comment, it was agreed to explain to future respondents that the term "use" entailed spending 50% of the time using the language to write geometric events, and 50% of the time using the language to read geometric events written in the language.

(2) Respondents asked whether they would be using an Integrated Development Environment (IDE) to write the statements in real life, as they used or knew of IDEs that could auto-complete JSON-based text based on a JSON schema, but they didn't know if such capability was available for a custom language. From this it was agreed to state, to all future respondents, that both languages would be used along with intelligent code completion capabilities that were equal in both languages.

More than one respondent in this phase provided these comments. These were significant enough that action was taken to augment the responsibilities of the survey guide during the execution phase of the study. The guide would now need to explain these points as the respondent arrived at the relevant sections, all in order to reduce assumptions or bias, and to normalize mental models going into each question.

After the piloting phase was completed, we began to run the questionnaire with the JPL and UPC samples based on the availability of each respondent. The format for the study followed a pattern similar to the piloting phase. It was a web-based questionnaire completed via video-conference, where the guide introduced the purpose of the study and observed the respondent as they were completing each question. If there was any doubt about the questions, the guide would assist to clarify. Similarly, if there was any qualitative comment from the respondent, the guide would listen and take note of it. At the end of the survey, the guide asked if there were any additional qualitative comments that

the respondent would like to provide about the two language options or the survey. At the end of the questionnaire, the guide was also empowered to ask questions about anything specific to answers to questions in the questionnaire, e.g., *why do you think this language was much more readable than the other?* All comments and responses were captured and transcribed by the guide.

D. Statistical Analysis

Part of our numerical scrutiny is performed on the answers to questions in Sections 4-14, which include data that are statistically analyzable, as these questions provide responses on a Likert ordinal scale. For each individual question we could interpret results as a distribution, e.g., "20% of the respondents agree with the fact that language NLB is readable by a scientist or engineer". In our review of the Likert-based results, however, we focused on the central tendency of the overall sample and the subsamples (JPL and UPC) for each question and language option. Note that Likert data is ordinal in nature and this prevents the use of parametric analyses such the statistical *mean* [26], which we did not use in this study, instead, we used the statistical *mode* per question, per language option, for the overall sample and for the subsamples. The other part of our statistical analysis is based on the SUS scores, which are presented as a number within the 0-100 range for each study participant. We use arithmetic means for the SUS scores for the overall sample and for the subsamples. Arithmetic means are acceptable in this case due to the fact that the SUS is an interval score for usability.

E. Threats to Validity

One area of the study in which we believe there was a potential internal threat to validity is regarding the progression and mental framing of the respondents as they complete the questionnaire. More specifically, this threat involves a conscious or subconscious preference for one language versus another as a result of the structure of the questions being asked. At the SUS level, this is handled by alternating positive-framed questions, i.e., questions where "Strongly Agree" is positive for usability, with negative-framed questions, i.e., questions where "Strongly Agree" is negative for usability. As a result, no action was taken from the point of view of our study design in relation to SUS question ordering. An action was taken, however, regarding the fact that our study considers the same sequential questions for the two language options. In order to prevent a possible situation in which a respondent might think a language option is better because it was presented first or second, we chose to alternate the language options within each question, e.g., the questionnaire will ask Question 1 for language NLB first, and KVP second, and then the

questionnaire will ask Question 2 for language KVP first, and language NLB second, and so on. Note that in the questionnaire we did not use the names NLB and KVP for the languages, but we did name languages as "Language A" and "Language B" within each section, and alternated what Language A and B represented as language options. In effect, "Language A" in Section *n* would be different from what "Language A" was in Section *n+1*. This was done to avoid name affinity or likability, and recency bias.

A criticism could be made in relation to the fact that we did not distribute language options across the sample, that is, we did not randomly assign language options across the sample in a way that one respondent responds to questions about only one language option, unaware there is another option. We believe this to be a sensible comment on validity, however, we also believe there is value in respondents being exposed to the two options, as they not only can factor in a scoring in their answers that is relative to each language option against the other, but can also provide qualitative comments about the two options in relation to each other.

In terms of the sample used, there were threats about target population representation for the languages we are evaluating. We initially believed that using the JPL sample was enough to depict the kinds of individuals involved in modeling geometric events in space. However, as the study design evolved, our ideas also evolved. We thought that while the JPL sample was well-versed in the uses our languages would need to operate in, JPL as a whole is a microcosm of a larger realm that can involve users from different organizations and nations outside of the United States. To that end, we recruited individuals from UPC in Catalonia, Spain with a relevant background who could also be users of the languages in their professional or research lives. This action increased sample diversity, making a more robust sample, and implied the benefit that qualitative comments would be expected to be richer as a whole. One point of view is that this could create a threat in case the two subsamples provide quantitative results that substantially differ from the other subsample, but if that occurred, that would be an input to iterate on more research to design a better language proposal.

V. Results

In the following sections of the manuscript we will review the results of each survey section, which include Section 1 - Demographics, Sections 2 and 3 - Exercises, Section 4 - Readability, followed by Sections 5-14, which comprise the SUS questions.

A. Section 1 - Demographics

Respondents were asked to provide their name, college major for their highest degree, educational level, time since graduation, and years of experience in the aerospace domain. For major, 77.78% of all respondents either graduated or were en route to graduate from an Aerospace Engineering program, 14.81% from a Computer Science program, and 7.41% major or majored in Applied Mathematics. In terms of educational level, 70.37% of the respondents graduated or were en route to graduate from a Bachelor's-level program, whereas 29.63% responded they graduated or would be graduating from a Master's-level program. 66.67% of the respondents had not graduated yet, and for the remaining 33.33% who already graduated, the average time since graduation was 1.67 years.

In regards to experience in the aerospace domain, we qualified the question by explaining to respondents that experience meant "involvement with" the aerospace domain in general. As such, internships, work experience, and their college education, if related to aerospace, did count in terms of experience. The response, taking into account both samples, was that the average level of experience in the domain was 3.61 years. In regards to software development background, we wanted to measure experience as a binary and more subjective value for each respondent; that is, they either think of themselves as software developers or not. This is because the sample contained individuals coming from different majors and with different interests, thus measuring experience in terms of length of time could provide a large range of values, as programming can be part of a career or major, but also an interest or a hobby. Besides, only some level of programming experience is expected for the languages considered. Measuring programming knowledge in more detail would have involved designing more questions that would be less relevant to the study. The result we obtained was that 92.59% of the respondents thought of themselves as software developers. More complete data can be found in Table 2.

Table 2. Demographics of the total sample and the two subsamples.

	Total	JPL Sample	UPC Sample
Sample size	27	18	9
Time since graduation, arithmetic mean (years)	0.63	0.78	0.33
Time since graduation, range (years)	[0-5]	[0-5]	[0-1]
Experience in the aerospace domain, arithmetic mean (years)	3.61	3.33	4.17
Experience in the aerospace domain, range (years)	[0-9]	[0-9]	[3-5]
Think of themselves as software developers (sample %)	92.59%	100.00%	77.78%

B. Sections 2 and 3 - Exercises

The purpose of these two sections was to increase, via two exercises, the understanding of each language option and to develop critical opinions about their usage. In Section 2, respondents were provided with the event "*Between the start of year 2000 and the end of year 2005, the distance between the Earth and the Moon is less than 400,000 km*", which was modeled with the two different language options. Starting from this example event and its modeling with each language option, respondents were asked to model another event related to the example, which was: "*In all of 2021, Earth and Mars are at a distance of more than 70M km from each other*". In this exercise, all respondents provided an answer that was objectively correct for each language option. Sample responses to the question are provided in Table 3.

Table 3. Potential solutions for exercise in Section 2. Respondents were asked to model and search for the event "*In all of 2021, Earth and Mars are at a distance of more than 70M km from each other*".

NLB	KVP

`DEF event_name AS`
`  DISTANCE FROM Earth TO Mars`
`  MORE THAN 70Mkm`
`  DURING 01/01/2021:12/31/2021`
`SEARCH FOR event2`
	`DistanceQuery event2{`
	`  observer: "Earth"`
	`  target:  "Mars"`
	`  test: >`
	`  amount: 70Mkm`
	`  start_time: 01/01/2021`
	`  end_time: 12/31/2021`
	`}`
	`SEARCH FOR event2`

In contrast, Section 3, instead of presenting an example event, showed what could qualify as potential documentation, in the form of a language grammar, to be read by users of each language, to model an occultation event in each language option. To clarify, an occultation is when the line of sight between two bodies is broken by a

third body. For instance, a solar eclipse is a type of occultation, in that the line of sight between Earth and the Sun is interrupted by the Moon, either fully or partially. This documentation was followed by the request to model the event "*Earth and Mars are at a distance of more than 70M km, and at the same time, Mars and Earth are in full solar conjunction (solar conjunction is when the Sun is between two bodies). Use the 2010-2020 interval.*" Modeling this event required respondents to use the documentation in Section 3, but also to inspect their previous answers in Section 2, which was possible through their use of the web-based form. As it also occurred in Section 2, none of the respondents failed to provide the right answer for the exercise within Section 3. Sample responses to the question in Section 3 are provided in Table 4.

Table 4. Potential solutions for exercise in Section 3. Respondents were asked to model and search for the event "*Earth and Mars are at a distance of more than 70M km, and at the same time, Mars and Earth are in full solar conjunction (solar conjunction is when the Sun is between two bodies). Use the 2010-2020 interval*".

NLB	KVP

DEF event1 AS
FULL OCCULTATION
BETWEEN BACK BODY Mars
AND FRONT BODY Sun
OBSERVED BY BODY Earth
DURING 01/01/2010:12/31/2020
 	
DEF event2 AS
DISTANCE FROM Moon TO Earth
MORE THAN 70Mkm
DURING 01/01/2010:12/31/2020
 	
SEARCH FOR event1 AND event2
	DistanceQuery event1{
	observer: "Earth"
	target: "Mars"
	test: >
	amount: 70Mkm
	start_time: 01/01/2010
	end_time: 12/31/2020
	}
	OccultationEvent event2{
	type: "full"
	back_body: "Mars"
	front_body: "Sun"
	observer: "Earth"
	start_time: 01/01/2010
	end_time: 12/31/2020
	}
	SEARCH FOR event1 AND event2

C. Section 4 – Language Readability

Section 4 provided the statement "*I think the language above is readable by a scientist or engineer*" along with sample events in each language. Respondents were asked to select an answer on a Likert 1-5 scale, where 1 was "*Strongly Disagree*" and 5 was "*Strongly Agree*". The NLB results mode for the overall sample and both subsamples was "*Strongly Agree*". The mode for KVP for the overall sample and the JPL subsample was "*Neutral*", and a tie between "*Agree*" and "*Neutral*" in the UPC sample. Mode results can be found in Table 6 whereas full results from this question can be found in [27].

F. Sections 5-14 – System Usability Scale (SUS)

For the SUS, NLB obtained an average score across all respondents of 89.81. The average NLB result for the JPL sample was 90.00, whereas for the UPC sample, the average SUS score was 89.44. For KVP, the average SUS score across all respondents was 90.00. For this option, the average SUS score within the JPL sample was 89.72, and for the UPC sample it was 90.55. From this, we understand respondents favored KVP's usability by a very small margin (0.19 points). Interestingly enough, the JPL sample gave a higher average score to NLB, also by a small margin (0.28 points), whereas the UPC sample gave a higher score to KVP by a larger, but still small margin (1.11 points). These results can be found in Table 5.

Table 5. SUS Scores per sample as arithmetic means along with the difference magnitude between means. High scores in each row are marked in bold lettering.

Sample	NLB, Mean SUS Score	KVP, Mean SUS Score	Difference Magnitude for Mean SUS Scores
Total	89.81	**90.00**	0.19
JPL	**90.00**	89.72	0.28
UPC	89.44	**90.55**	1.11

When looking at the modes in the responses for the SUS questions, we saw that both subsamples provided similar answers. As a matter of fact, out of ten questions for two language options, resulting in a total combination of 20 questions, the JPL and UPC samples only diverged in their response modes twice. In order to validate these divergences through inferential statistics, we performed a Mann-Whitney U test on the responses from both samples. Our null hypothesis (H_0) was the two samples originate from the same population, hence similar results shall be expected. The null hypothesis, considering a significance level (α) of 0.05, was rejected on both identified divergences. No other descriptive (via mode analysis) nor inferential (via Mann-Whitney U test) divergences were found. The two divergences are as follows:

(1) Question #1, "*I think that I would like to use the language above frequently*", for NLB. The JPL sample's mode was "Strongly Agree" and for the UPC sample it was "Agree". Mann-Whitney U test *p-value*=0.040.

(2) Question #8, "*I found the language above very cumbersome (awkward) to use*", for NLB. The JPL sample's mode was "Strongly Disagree" and for the UPC sample it was "Disagree". Mann-Whitney U test *p-value*=0.035.

Mode responses for the SUS questionnaire can be found in table form in Table 6 whereas full results can be found in [27].

Table 6. Mode for each response in Sections 5-14. The "Statement" column, besides containing the statement provided to the respondents in each section for each language option, also includes the SUS question number, except for Section 5, which is not part of the SUS questionnaire. The "Responses" columns contain the mode for each sample, JPL and UPC, but also the mode for the two samples together in the "Total Mode" column. Note that discrepancies between samples have been marked in bold lettering style.

Statement	NLB Responses			KVP Responses		
	Total Mode	JPL Mode	UPC Mode	Total Mode	JPL Mode	UPC Mode
Section 5: I think the language above is readable by a scientist or engineer	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Neutral*	*Neutral*	*Agree/ Neutral*
Section 6 (SUS #1): I think that I would like to use the language above frequently	***Agree***	***Strongly Agree***	***Agree***	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*
Section 7 (SUS #2): I found the language above unnecessarily complex	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*
Section 8 (SUS #3): I thought the language above was easy to use	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*
Section 9 (SUS #4): I think that I would need the support of a technical person to be able to use the language above	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*
Section 10 (SUS #5): SUS #5: I found the various functions in the language above were very well integrated	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*
Section 11 (SUS #6): I thought there was too much inconsistency in the language above	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*
Section 12 (SUS #7): I would imagine that most people would learn to use the language above very quickly	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*
Section 13 (SUS #8): I found the language above very cumbersome (awkward) to use	***Strongly Disagree***	***Strongly Disagree***	***Disagree***	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*
Section 14 (SUS #9): I felt very confident using the language above	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*	*Strongly Agree*
Section 15 (SUS #10): I needed to learn a lot of things before I could get going with the language above	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*	*Strongly Disagree*

VI. Discussion

One of the main takeaways from the study is that respondents considered that NLB was, by a substantial margin (see Table 6, Section 5), a more readable implementation of a textual language for our purpose. An expectation, while designing the two language options, was that NLB would be considered more readable overall due to its similarity to the English language, but there was also doubt as to whether, given the software development experience of the samples and their potential more comprehensive exposure to data exchange formats like JSON or XML, that KVP could also be considered quite readable. One thought we are currently considering is the fact the positive evaluation of NLB in terms of readability also makes it more communicable within a group, and only not at the level of just one individual. In other words, scientists and engineers, with no previous training, can look at events written in it and understand their meaning and debate whether an event makes sense or not within the context of a mission. This debate is a key point, as we have learnt that scientists and engineers, especially during the remote work spells of the COVID-19 pandemic, exchange textual information via email and chat facilities more frequently than when they are physically co-located. While this is not a core point to our research, it makes an argument that the ability to understand or read an event might be more important than the ability to write an event. In essence, more readability might imply higher-quality group communication and decision-making via digital-textual means.

Another reflection is that the differences between NLB and KVP in relation to the SUS scores results are small and the SUS scores are overall very high, indicating that both options are highly usable according to the sample and the methods used. Delving into the details of the scores, the two samples combined gave a higher score to KVP, by a very small margin; however, the JPL sample gave a higher score to NLB, and the UPC sample gave a higher score to KVP. Because both samples are small and the differences in scores are small, this could be construed as part of survey "noise", but we believe there is something indicated by this small divergence. Our thesis is that native English speakers favor NLB, and the totality of the JPL sample is indeed composed of native English speakers. In contrast, the UPC sample was composed, in its totality, by non-native English speakers, and while they all had very good command of the English language and never asked for clarification about the grammar or vocabulary that was part of the NLB option, a slight hesitation towards a language they are not fully comfortable with, when compared to their native languages, might be encoded in their responses. As we saw for the statements "I think that I would like to use the language above frequently" (Table 6, Section 6) and "I found the language above very cumbersome (awkward) to use" (Table 6, Section 13), the mode results indicate the UPC sample was slightly less enthusiastic about NLB than it

was for KVP relative to the JPL sample. While we are not considering developing an NLB option in different languages, one way to validate this thesis would be to perform the same study in which both KVP and NLB use words and constructions in the native language of the respondents. In any case, and as a corollary for the SUS scores, we (1) do not consider usability for one language option to be substantially better than for the other, and (2) both language options are highly usable as evidenced by the high SUS numbers.

VII. Conclusion

From our analysis, we have determined it is more sensible to proceed with research on the implementation of a textual language that adheres to the NLB philosophy. This research would include the completion of the grammar for a variety of frequently used geometric events, the design of a unit system for values such as distance, and a mature way to define dates and times. In order to implement the language, one route to take is to leverage the Xtext language development framework [28], as it provides a path to transform text into instances of a metamodel, which aligns with the design of the Tychonis framework [3]. Another benefit of Xtext is that, depending on the environment in which it is used, it provides auto-complete and other guard rails for languages implemented with it. Notably, the need for such capability was a comment provided by respondents during our dry run of the survey.

Additional future research should also consider the automatic expansion of the language grammar based on independent augmentations of the Tychonis framework. This would imply that if a developer adds a new type of searchable geometric event to Tychonis, the textual language would automatically be augmented without having to involve a language developer to explicitly augment the grammar, as the grammar would be implicitly encoded in the Tychonis metamodel. To elaborate, one of the main tenets of the Tychonis framework is separation of concerns, which are (1) the description, or modeling, of families of geometric events, (2) the description of the algorithms to search for a family of geometric events, and (3) the capture and provisioning of the geometric event search results. In regards to a textual language, a fourth concern should consider how the parameters of a family of geometric events would manifest within the context of a given textual language such as the ones described in this manuscript.

Acknowledgments

The authors would like to thank the respondents in the study for generously offering their time in support of the research described in this paper. These valuable results would not have been obtainable without their involvement.

The authors would also like to thank the Planning and Execution Systems Section within JPL for its dedication to improving the state of the art of mission planning software. The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Bevan, N. "ISO 9241: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)-Part 11: Guidance on Usability." *Tc*, Vol. 159, 1998, p. 61.

[2] Robinson, T. D., Maltagliati, L., Marley, M. S., and Fortney, J. J. "Titan Solar Occultation Observations Reveal Transit Spectra of a Hazy World." *Proceedings of the National Academy of Sciences*, Vol. 111, No. 25, 2014, pp. 9042–9047. https://doi.org/10.1073/pnas.1403473111

[3] Maillard, A., Chien, S., and Wells, C. "Planning the Coverage of Solar System Bodies Under Geometric Constraints." *Journal of Aerospace Information Systems*, Vol. 18, No. 5, 2021, pp. 289–306. https://doi.org/10.2514/1.I010896

[4] Llopis, M., Soria, M., and Franch, X. "Tychonis: A Model-Based Approach to Define and Search for Geometric Events in Space." *Acta Astronautica*, Vol. 183, 2021, pp. 319–329. https://doi.org/10.1016/j.actaastro.2021.01.057

[5] Paige, R. F., Kolovos, D. S., and Polack, F. A. "A Tutorial on Metamodelling for Grammar Researchers." *Science of Computer Programming*, Vol. 96, 2014, pp. 396–416. https://doi.org/10.1016/j.scico.2014.05.007

[6] Llopis, M., Polanskey, C. A., Lawler, C. R., and Ortega, C. The Planning Software behind the Bright Spots on Ceres: The Challenges and Successes of Science Opportunity Analyzer. 2019. https://doi.org/10.1109/SMC-IT.2019.00005

[7] Llopis, M., Franch, X., and Soria, M. Integrating the Science Opportunity Analyzer with a Reusable Opportunity Search Framework. In *ASCEND 2020*, 2020, p. 4221. https://doi.org/10.2514/6.2020-4221

[8] Lloyd, J. W. Practical Advantages of Declarative Programming. 1994.

[9] Chamberlin, D. D., and Boyce, R. F. SEQUEL: A Structured English Query Language. 1974. https://doi.org/10.1145/800296.811515

[10] Codd, E. F. Relational Database: A Practical Foundation for Productivity. In *ACM Turing award lectures*, 2007, p. 1981. https://doi.org/10.1145/358396.358400

[11] Winograd, T. Frame Representations and the Declarative/Procedural Controversy. In *Representation and understanding*, Elsevier, 1975, pp. 185–210. https://doi.org/10.1016/B978-0-12-108550-6.50012-4

[12] Kowalski, R. "Algorithm= Logic+ Control." *Communications of the ACM*, Vol. 22, No. 7, 1979, pp. 424–436. https://doi.org/10.1145/359131.359136

[13] Acton Jr, C. H. “Ancillary Data Services of NASA’s Navigation and Ancillary Information Facility.” *Planetary and Space Science*, Vol. 44, No. 1, 1996, pp. 65–70. https://doi.org/10.1016/0032-0633(95)00107-7

[14] Shackel, B. “Usability–Context, Framework, Definition, Design and Evaluation.” *Interacting with computers*, Vol. 21, Nos. 5–6, 2009, pp. 339–346. https://doi.org/10.1016/j.intcom.2009.04.007

[15] Nielsen, J. *Usability Engineering*. Morgan Kaufmann, 1994.

[16] Lansdale, M. W., and Ormerod, T. C. *Understanding Interfaces: A Handbook of Human-Computer Dialogue*. Academic Press Professional, Inc., 1994.

[17] Lin, H. X., Choong, Y.-Y., and Salvendy, G. “A Proposed Index of Usability: A Method for Comparing the Relative Usability of Different Software Systems.” *Behaviour & information technology*, Vol. 16, Nos. 4–5, 1997, pp. 267–277. https://doi.org/10.1080/014492997119833

[18] Brooke, J. and others. “SUS-A Quick and Dirty Usability Scale.” *Usability evaluation in industry*, Vol. 189, No. 194, 1996, pp. 4–7.

[19] Bangor, A., Kortum, P. T., and Miller, J. T. “An Empirical Evaluation of the System Usability Scale.” *Intl. Journal of Human–Computer Interaction*, Vol. 24, No. 6, 2008, pp. 574–594. https://doi.org/10.1080/10447310802205776

[20] Bray, T. *The Javascript Object Notation (Json) Data Interchange Format*. 2014.

[21] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., and Vrgoč, D. Foundations of JSON Schema. 2016. https://doi.org/10.1145/2872427.2883029

[22] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., and Cowan, J. Extensible Markup Language (XML) 1.0. W3C recommendation October, 2000.

[23] Ben-Kiki, O., Evans, C., and Ingerson, B. “YAML Ain’t Markup Language (YAML)(Tm) Version 1.2.” *YAML. org, Tech. Rep*, Vol. 359, 2009.

[24] Caldiera, V. R. B. G., and Rombach, H. D. “The Goal Question Metric Approach.” *Encyclopedia of software engineering*, 1994, pp. 528–532. https://doi.org/10.1002/0471028959.sof142

[25] Llopis, M., Franch, X., and Soria, M. Questionnaire Used to Evaluate the Usability of Two Declarative Computer Languages to Model Geometric Events in Space. Zenodo, May, 2022. https://doi.org/10.5281/zenodo.6569979

[26] Allen, I. E., and Seaman, C. A. “Likert Scales and Data Analyses.” *Quality progress*, Vol. 40, No. 7, 2007, pp. 64–65.

[27] Llopis, M., Franch, X., and Soria, M. Questionnaire Used to Evaluate the Usability of Two Declarative Computer Languages to Model Geometric Events in Space. Responses from Usability Questions. Zenodo, May, 2022. https://doi.org/10.5281/zenodo.6596564

[28] Eysholdt, M., and Behrens, H. Xtext: Implement Your Language Faster than the Quick and Dirty Way. 2010. https://doi.org/10.1145/1869542.1869625