UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Department of Computer Science

Computer Science Honours Final Paper 2016

Title:	Usability in Astronomy Visualisation Software Interfaces
Author:	Laurisha Rampersad
Project abbreviation:	AstroVis2
Supervisor:	Michelle Kuttel

Category	**Min**	**Max**	**Chosen**
Requirement Analysis and Design	0	20	10
Theoretical Analysis	0	25	-
Experiment Design and Execution	0	20	20
System Development and Implementation	0	15	-
Results, Findings and Conclusion	10	20	20
Aim Formulation and Background Work	10	15	10
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Overall General Project Evaluation	0	10	-
Total marks			80

Usability in Astronomy Visualisation Software Interfaces

Honours Project

Laurisha Rampersad
University of Cape Town
Student Number: RMPLAU001

ABSTRACT

Usability is one of the most important aspects of software development. In the development of scientific software, however, usability practices are challenging and barely used. As scientific software is becoming more complex and relevant, the usability of this software is a necessary and important research topic. The reasons for the challenges to usability stem from the nature of the complex subject domain. Developers of scientific software lack either domain-specific knowledge or software development experience. A possible solution might be persistent collaboration with domain experts. One type of scientific software, astronomy visualisation software, is a relatively new pursuit and usability practices in this field are under-utilised. The common trends indicate that usability is challenging and thus interfaces are not effective. This study carries out a User Centred Design (UCD) approach to designing an astronomy visualisation interface. Design decisions are based on guidelines made popular by Shneiderman and Nielsen. The methodology is iterative and users are involved as much as possible. In each iteration, a prototype interface is designed and then evaluated by users through different methods. Frequent consultation with domain experts produces a more usable, useful and effective astronomy visualisation interface. These findings may be applied to other scientific software fields, as implementing UCD may result in more usable software.

Keywords

Usability; Scientific Software; Astronomy; Visualization; Design; User-Centred Design; Interface Design

1. INTRODUCTION

It is necessary to pay attention to usability considerations when developing software to allow for more efficient interaction between the software and the intended users. Usability has been described as "The capability of the software product to be understood, learned, used, and attractive to the user, when used under specified conditions." [18]. One of the ways in which usability can be improved is through User Centred Design (UCD), which is a term describing the design processes during which the final product is influenced by end users [1]. Despite the wide range of approaches this concept can be implemented in, the integral notion is that intended users must be involved in some way.

Usability measures have an inherent degree of subjectivity, so it is not surprising that there is confusion about the concept. As a result of different groups of software engineers and usability experts developing their own models in isolation from one another, there is no consistent knowledge base to address usability [39]. To add to the inconsistency, Brooke's notion is that usability is a concept which exists with reference to a specific context [4]. This implies that usability standards (which are already fraught with variation and are difficult to measure [39]) will be further disparate depending on the situational context of the software. Scientific software is a particularly complex context.

The usability and UCD practices in commercial software domains (such as mobile applications and web interfaces) have improved over the years [29]. However, those working in scientific fields often deal with software which is poorly designed or which will often need to be patched together from available software [29]. Usability is difficult to achieve in complex domains which involve in-depth domain knowledge, expertise and experience [44], such as scientific domains. Additionally, for many complex and domain specific software projects (such as scientific software), measurement of usability is often challenging [6]. Whether the scientific software is designed for improved workflows or for research, the software still needs to be usable and useful to scientists [29].

Despite the growing importance of scientific software development, there has been little research on software standards in this field [16]. In designing software, certain traits are common in scientific fields. Usually, the developers of scientific software are either software developers with little knowledge of the highly specific subject matter, or instead are domain experts with recent, limited knowledge about software development [29, 41].

In the first scenario, the customer and end user is the research scientist. A typical problem of this scenario is that it is rare that the exact requirements of the project are known in advance [40]. This is a disadvantage in a field where developers are not acquainted with the domain expert activities. Costabile et al. separate these activities into two classes [8]. Class 1 consists of activities involving tailoring the software to choose between existing behaviours/interactions such as

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

parametrization (instructing the software on how to handle different data) or annotations. Class 2 involves some level of programming such as creating models based on the data. Without an understanding of the tasks the end user will perform, usability criteria like learnability and ease of use will not be addressed adequately.

In the second scenario, the domain expert/scientist is the developer. Here, learning the required software development skills may take a lot of effort. This effort will increase for the quality of the software to be better than adequate. Good software for science should be able to be extended/adapted for unforeseen requirements and be verifiably accurate. The latter is difficult due to the fact that scientists lack test oracles (the data with which to test the accuracy of the output of the software) [41]. So, while the scientist may have a better understanding of the use cases, the software might not undergo sufficient testing, especially usability testing when the end-user and developer have the same persona (a hypothetical user classification). In the case where end users would be fellow scientists working in the same fields, domain experts who develop software often omit usability testing and user documentation entirely [22].

One of these scientific fields, astronomy is particularly important to South Africa. South Africa houses large scale radio telescopes (and arrays of telescopes) which store observational data as 3-dimensional cubes. MeerKAT is one of these telescopes which is part of a larger project called the Square Kilometer Array (SKA). The SKA project is an international venture involving 17 countries using their resources to build a radio telescope with one million square meters of collection area in effect [14]. The volumes of data gathered from these new instruments will increase massively, and files from the new telescopes will reach up to a petabyte (10^{15} bytes) in size. This prompted Hassan and Fluke to coin the phrase Petascale Astronomy Era [15]. Processing such large files presents a unique challenge when creating software for astronomy. The goals of this type of software vary from statistical analysis as computed by AstroStat [23], collaboration as would be facilitated by sharing data in CyberSKA [24] to visualizing the multidimensional data with tools and libraries such as Karma [12], VisIVO [3] and GIPSY [36]. Despite the variation in the functionality of such tools, astronomy software shows poor usability in general - especially with regards to user interfaces [7].

Astronomy software projects often under-estimate and under-budget user interfaces resulting in numerous technical engineering panels instead of intuitive interfaces [7]. There was a call for improving astronomy user interfaces by the Virtual Astronomy Observatory (VAO) Science Council's Recommendations (in 2010) [10] as well as the suggestion of adding increased user support to astronomy software tools. It is clear that usability has long been an issue in software for astronomy. However, as there is a wide range of software in astronomy, it is impractical to hold every tool to the same set of usability standards. An astronomy tool for communication would be far easier to make more usable compared to an analysis tool processing a petabyte of data per task. Of this range of tools, we will focus on astronomy visualisation software.

Scientific visualization involves presenting data with three or more dimensions visually in a way which is easier to inspect by eye [15], so as to reduce the complex cognitive load on the brain and enable problem solving and pattern finding [32]. In radio astronomy, three dimensional cubes of data are visualized with tools to aid exploration of the data in the hopes of finding unexpected knowledge [13].

Visualization enhances the usability of a software tool. Usability heuristics of the visualization (and not just the interface) contribute to overall ease of use and includes feedback, consistency and error checking with the added perceptual heuristics of colour, Gestalt Laws and aesthetics [38].

We aim to determine whether the use of User Centred Design (UCD) principles in the design of a software interface for astronomy visualization, will produce a design that is useful, usable and effective. This GUI will be designed to accommodate the functionality of a radio astronomy visualisation tool such as Karma [12]. We aim to optimise the final prototype by capitalising on expert user recommendations and insights throughout a User-Centred Design process. We expect that the time and resources expended on involving users in the design stages, will result in a superior and more usable experience for users.

Our methodology involves first gathering user requirements and then using iterative stages of prototyping (with both design and evaluation phases). The final prototype is evaluated for usability and effectiveness through user evaluation, namely; a User Test and System Usability Scale (SUS) survey. The aim is to generate an approved design for development of an effective user interface for visualisation software.

2. BACKGROUND

The following subsections will address the theory behind the chosen methodology. They will also make note of related work in the usability of scientific software as well as the usability of available astronomy visualisation tools.

2.1 User Sampling

There are many methods which could be used to design a software prototype. UCD integrates users in some way when carrying out these methods [1]. When deciding on an approach to recruit users for this purpose, there are many options, but not all will be suitable. Dumas and Redish have clarified the difference between selecting users for a scientific research study and for a usability study [9]. They argue that when conducting a research study to prove or disprove a certain phenomenon, users are selected with a degree of randomness to support the statistical calculations which rely on a random sampling. In contrast, the goal of a usability study is to uncover errors in the product. Thus Dumas and Redish stress that participants in user studies need to be actual intended users of the product, rather than random members of the population. These intended users are often sourced though Convenience Sampling - where intended users whom you happen to find and are available to you are chosen [9].

2.2 Design Methods

Brainstorming is a method often used in the early stages of design and Kelly stresses the need to approach brainstorming as session dedicated to producing a large quantity of ideas using a playful approach without strict rules [21]. Another aspect of brainstorming is the concept of Build-and-Jump [21]. Build-and-Jump describes choosing either to build on an existing idea or to jump to a different direction (or perhaps even backwards) to boost the quantity of

ideas are generated [21].

Once an idea is selected, it can be prototyped. Prototypes can be produced through both high fidelity (hi-fi) and low fidelity (lo-fi) methods [35]. The choice of which method to use depends on the goal of the prototype and the stage of the project's progression. A prototype created with the goal of giving users a feel of the product or conducted during requirements gathering (early) phases are well suited to low-fidelity methods. Paper prototyping is a useful lo-fi method as the prototypes are disposable, changes are easy to make and users comment on bigger issues rather than fit-and-finish issues [34]. Rudd suggests that high fidelity prototypes are suited to later stages of design, by trading off speed (time taken to develop the prototype) for accuracy of the prototype being tested. He also describes Horizontal-Prototypes as those which demonstrate high level functionality across the prototype while avoiding lower level (in-depth) details [35]. These prototypes are often interactive and computer based when designing software.

2.3 Usability Evaluation

Usability testing is more than just a once-off activity at the end of a software project. It is through continuous interaction with end-users (from the start of a software project) that the final product will have functionality which is likely to be used [9]. One model of continuous usability testing, as described in the Omero project [29], is a weekly user evaluation cycle. At the beginning of each week, user testing is taped and analysed. The findings are sent to the software development team who then work on tailoring the software to the updated requirements/suggestions.

Miller and Jeffries have advised that the effectiveness of Usability Evaluation can also be improved by using different methods to uncover different problems (and this can be done at various stages of the project) [30]. There are three types of usability evaluation methods: user-based, expert-based and model-based [37]. User-based methods have users as the source of evaluation methods and thus take advantage of domain specific expertise of the users to ensure that the software is more usable in context [37]. These methods include user testing, interviews and walkthroughs. Expert-based methods rely on usability experts having knowledge of best practices. While these best practices are generalised, usability expert knowledge will find many errors in the software [37]. Examples are heuristic evaluations, guideline reviews and cognitive walkthroughs. Finally, model-based methods simulate human interactions and perceptions using a task based approach. While this approach may be cost-effective, it is often time consuming. In a domain-specific context, not including users during evaluation might be detrimental by losing valuable insight into the domain. A range of evaluation methods are described below.

2.4 Evaluation Methods

A Think Out Loud evaluation process involves the user performing tasks using the prototype and talking the evaluators through these tasks while observations are recorded [28]. This method can be used to ensure that the prototype is fulfilling user needs in terms of tasks, as well as testing the flow and usability of the prototype.

Cognitive walkthroughs set up a scenario of a task a user might need to perform, and talking through all the actions necessary to complete the task. This exercise can be performed with either users or usability experts. The goal is to determine difference between user expectations (and current level of understanding) and the reality of the interface through exploration[30].

User Testing is an empirical test of the software. The tests take place either under real-world or controlled settings, with the test possibly being recorded on video or using computer logs. Users have to complete a set of tasks using the software with little assistance from the test conductors, after they have received the appropriate level of training for using the system. Jeffries et al. found that user testing often finds the most critical problems in the software [30].

Another method is to check a set of requirements against usability standards by a usability expert (or even a software engineer) to find potential problems in the software. The advantage of this approach is that very common mistakes may be avoided and it is relatively simple for an expert or software developer to check against criteria such as responsiveness, error detection and feedback. The effort required is thus minimal. However unique and relatively severe errors might not be picked up, diminishing this method's effectiveness [30]. For this reason, this method might not be suitable. The problems we need to find in these prototypes will be domain specific and not necessarily common to conventional user interfaces.

Heuristic Evaluation involves inspection of software by individuals with experience in usability or interface design. Jeffries et al. found that heuristic evaluation not only finds the most problems, it is also relatively low cost [30]. The main disadvantage is the need (or rather preference) for many evaluators with usability expertise, as well as subject domain knowledge. However, the likelihood of a domain expert (such as a doctoral-level expert) deciding to specialise as a usability expert is quite low [6] and thus this combination of expertise is rare. The key factor in terms of effort for this method would come down to the number of evaluators used, as the effectiveness of heuristic evaluation increases with the number of evaluators [30]. For this reason, standard Heuristic Evaluation is not suitable for this study as we do not have access to multiple specialised usability experts.

Another method is for users to complete a set of questions (qualitative and quantitative) about the software in the form of a survey or questionnaire. This can occur either in the beginning of development, as a requirements gathering exercise, or at the end, as a response to using the software. Surveys are often used to complement other evaluation methods [25]. Surveys are easy to use and interpret, however, one needs to ensure that all the relevant questions are asked, or users may not be able to suggest improvements or detect errors. One of the more notable surveys is the 10 question System Usability Scale (SUS) Survey in which participants answer on a scale from one to five. The SUS Survey is know to be a robust and reliable measure of the Usability of a system [2].

Model based evaluations are particularly time consuming as a model needs to be constructed to mimic human behaviour, and then tested to ensure validity [37]. Models are low cost as, once a model has been proved valid, it may be used to test multiple iterations of designs. Designs are constructed at a task level and so the added time of creating a good task for modelling is cumbersome. An example of a model based evaluation is the GOMS (Goals, Operators, Methods and Selection) model which is based on human cog-

nition abilities [19]. Due to the time consuming nature of model based evaluation, this method is not suitable for this study.

2.4.1 Discussion

A systematic review of the most popular forms of usability evaluations in papers since 2012 found that User Testing was the second most frequently used (14.14%), following Surveys/Questionnaires (26.26%) [33]. Despite its apparent advantages, Heuristic Evaluation was the third most popular with a frequency of 12.63% and Cognitive Walkthroughs and Checklist verification had frequencies of less than 3%.

Furthermore, of the software applications that underwent usability testing, only 2.03% of the came from the domain of expert systems. This supports the conclusions of Chilana et al. that usability practices and testing are challenging in complex domain software development [6]. Følstad compared the use of domain experts for a cognitive walkthrough against the use of usability experts [11]. The domain experts found fewer problems than the usability experts, however, the problems found were more critical. Although heuristic evaluation (a method with less user testing) seems to be the most popular evaluation method, it can be seen that user involvement in usability is very important when the domain of the software is complex. A combination of user knowledge, software engineering skill and usability expertise seems to be necessary for achieving the best possible results. A further combination of different evaluation methods achieves better results than one channel of evaluation [30].

2.5 Usability and Scientific Software

Despite the difficulties of incorporating UCD practices when developing scientific software, there are some cases where this has been studied. In order to mitigate the problems when either a scientist is the only developer or a developer has no domain knowledge, Segal suggests that a collaboration of software developers and research scientists should create the software - with the scientist in the hybrid role of peer-programmer and end user [40]. Involving research scientists (the end user) in the design is a step in the direction of UCD. Segal also notes that a problem may arise when the developer is a software engineer rather than an end user developer [41]. The problem is that incredibly extensive requirements would be necessary, however, as stated - these are not always available at the start. It can be seen that a persistent partnership between domain experts, software engineers and usability developers is required [6], although this may be unrealistic in software development which is not commercial, due to funding and scope of the project.

2.5.1 Case Study: Omero

In the Omero/Usable Image Project, a team of usability experts (Usable Image project) and software engineers collaborated on a life sciences software project named Omero[1] [29]. The combined team used weekly evaluation cycles involving iterative user testing and software development. This model exposed many usability flaws: from an easy-to-miss labelling problem in the search interface, to a substantial hierarchy question. Software engineers initially allowed images to belong to categories. Through usability practices, they found that scientists preferred a system closer to tagging the images (similar to tagging in social media) and this more optimal method was used in the final design. The usability practices of the Usable Image Project team involved creating user testing sessions, design workshops, design research sessions, usability inspections and user guides/training materials. This case study concluded that integrating usability practices into scientific software development would substantially improve the software, and that these usability practices would need to be flexible and tailored to the specific project.

[1]Omero is a Java based client-server system for visualizing, managing, analysing, and annotating microscope images and metadata [29]

2.6 Usability of Astronomy Visualization Tools

The following tools visualise astronomy data and will be critiqued based on usability.

GIPSY is an interactive system (developed by the Kapteyn Institute) which reduces and displays astronomy data [36]. GIPSY contains a UI named HERMES which has interactive and non-interactive versions as well as a GUI named Ggi. The interface is quite rudimentary, technical and contains multiple similar looking text fields making the it overwhelming and difficult to use. Ruiz et al. state that GIPSY needs work on the user interface as it is currently not transparent nor user friendly enough for non-expert users [36].

KARMA is toolkit for image and signal processing applications, with a library of tools for visualisation. The KARMA toolkit [12] uses a modular approach, with many widgets to perform different functions. This greatly extends the usefulness of the toolkit and library. However, having several open window based control panels might add to the cognitive complexity of tasks. This adds to the designs usability flaws. There are a host of difficulties and usability issues one encounters when using Karma. Every time the program is started, one needs to add axes to the viewing panel, change the colour gradients and perform various adjustments to the visualisation interface before interacting with the data as these adjustments are not saved. Submenus in Karma are displayed as separate windows and this tends to clutter the workspace. Further, closing one of these secondary or tertiary windows will close the entire program and the entire process will need to be started again.

VisIVO is a cross-platform multi-dimensional application for visualisation [3] and its accessibility is enhanced by the fact that it is a cross-platform, an important consideration for usability [15]. VisIVO allows for integration, interactivity, navigation and collaboration across its many platforms (mobile application, an easy to use web portal, desktop application). This tailoring of the software could afford users more control over their experience, increasing usability. The interface screenshots show a simplistic GUI which requires multiple clicks, fields and buttons to be addressed in order to complete a task. This is a common trend throughout most of the astronomy software analysed.

Both Iris [26] and SKIRT [5] are astronomy tools dealing in highly complex functions and manipulations of the data. As such, they allow the option for a greater degree of customization of the existing interface (with limited graphical usability elements) through code. This comes at the cost of an existing user interface which is more technical than would be optimal. However, while the user interface may not be graphical, it hides complexity through an interaction mechanism allowing for the non-graphical user interface to

still be somewhat user friendly [5].

3. METHODOLOGY

UCD involves many ideas and practices which culminate in the goal of designing with the user in mind. This paper will address achieving this goal by splitting UCD into two processes. The first process investigated will be the use of industry recommended design guidelines in order to design for the user. These guidelines include tested interface design principles such as providing feedback for user actions [31]. The second process is the involvement of users in the design stages. This will be investigated by involving users frequently in the design process (requirements gathering and evaluation meetings) to ensure that the final prototype is usable, useful and effective.

3.1 Design and Usability Guidelines

In astronomy visualisation software, it is ironic that software which is designed with the purpose of visualising data is often lacking in basic visualisation principles. These principles have been modelled by visualisation experts in terms of the mantras such as *overview first, zoom, filter, details on demand* by Shneiderman [42]. However in especially cognitively complex contexts (scientific fields such as radio astronomy), users actions are often non-linear over time, with many means to perform an action. Thus it becomes difficult to pin-point a static formula for how the user interacts with the tool for each task [38]. Sedig et al. suggest taking a granular approach and evaluating the mapping between user actions, system reactions and user perceptions of those reactions [38].

These principles are important for complex data, such as radio astronomy data cubes, and this study aims to keep these principles in mind throughout the design process. This can be facilitated by choices in visual cues such as spatial structure, colour, depth, icons and so forth. (need to get the book from library for references) For example, visualising a radio data cube by using a rainbow colour scale would be problematic for showing an accurate overview of the data. This is because yellow lies in the middle of this scale and indicates middle results, but due to the highlighting effects of the colour yellow, these middle values might be subconsciously seen as extreme values [43]. A better colour scale might be the Linearized Optimal Colour Scale which is designed to maximise noticeable differences between colours on the scale while maintaining an intuitive order (e.g. black, dark red, green, cyan) [27].

Finally, generalised user-centred interface design principles need to be leveraged for the user to be able to efficiently explore the data. These 10 explained heuristics as described by Nielsen are seemingly obvious but have been shown to collectively account for up to 95% of usability problems during evaluations of the system [31]. Due to the high cognitive load of the data being processed by astronomy visualisation software, the most important principle to leverage is to allow the users to point and click rather than needing to type or remember information. Nielsen found that this principle accounted for 22% of all serious usability problems assessed.

A summary of all the visualisation and usability principles to follow are listed in the Guideline checklist in Table 1. This checklist is consulted during the design phase of each iteration.

3.2 Usability Study Methodology

This process involves users in the design stages to ensure that the final product is usable. We adapted the continuous user testing model as described in the Omero Project [29]. The process begins with a requirements gathering meeting to determine the current flaws in astronomy visualisation software and user needs from the system. Once this is done, the astronomy visualisation prototype is developed through iterative design-evaluation cycles. There are three iterations of prototype design and expert user evaluation. The aim is to maximise end user recommendations and input, by seeking as many reviews with expert users as possible, within time constraints. As the iterations progress, the fidelity of the prototypes evolve and become more complex.

3.3 Requirements Gathering

The goal of the requirements gathering phase is to discover the chief concerns and needs of a typical user of astronomy visualisation software. A typical user will perform various tasks to achieve different goals within a predominantly visual context (for example, observing the 3D data cube in slices). The users for this study were chosen through Convenience Sampling as experts in the field were needed and these experts were chosen by association with the previous projects in this field.

This initial meeting took place with two expert end users who use astronomy visualisation software regularly (once a day) to frequently (10-20 times a day). In order to gain a contextual understanding of how this software is used, the environment was the expert users' standard work environment: a shared office setting. A typical set of user tasks using KARMA [12] and DS9 [20] (another visualisation tool) were observed, while notes on the usability of the software were taken along with user comments and suggestions.

3.4 Design Iterations

3.4.1 First Iteration

The first iteration used brainstorming and paper prototyping for the design process. Brainstorming with paper sketches allows for rapid ideation with disposable sketches so that the design space was not constricted by sticking to old ideas, but rather building on these ideas or suggesting new ideas altogether. This is known as Build and Jump [21] and it allowed for two alternative designs to be created for the users to compare. Paper prototyping was used due to its disposable and flexible nature. This method allows the end users and designers to see the design as temporary and thus easy to change in terms of the functional aspects and task flow, instead of resisting changes [34]. It also avoids users commenting on fit and finish issues, such as font or colour, instead of work flows and general layout.

Instead of using a Heuristic Evaluation (which excludes domain experts) or User Testing (which relies on formal testing of a functional system), evaluation in this cycle took place using a Cognitive Walkthrough method [30]. The prototypes are described to the users in terms of tasks and features. The expert users then verbally review and compare the two proposed paper prototypes. This session took place in a controlled office environment with two expert users. The expert users responded to the proposed designs with comments and suggestions which were recorded and occasionally drawn onto the prototypes or additional paper. By allowing

Table 1: Design Guidelines as in [31, 42]

Principle	**Description**
System Status	Are users always informed of what is happening through feedback?
Error handling	Are errors prevented and/or can users easily diagnose and recover from errors?
User control	Are undo and redo functions supported?
Consistency	Are actions consistent both within the system and when compared to standard conventions?
Flexibility	Can the system cater to novice and expert users through tailoring and shortcuts?
Help	Is there sufficient documentation to guide users when they are confused/new?
Cognitive load	Are icons and actions visible for users to rather recognise instead of recall information?
Aesthetic design	Is the design pleasing and non-straining for a high volume user?
Learnability	Are aspects that are not immediately intuitive, learnable for users?

users to speak freely during the review of the prototypes (instead of following a set format) further brainstorming could take place at this stage. The two expert users were chosen through Convenience Sampling.

3.4.2 Second Iteration

User Experience (UX) Prototyping Software was used during the second iteration. While prototyping software takes longer to implement than a paper prototype, at this stage of the study, a higher fidelity prototype is necessary to convey the complex workings of the interface. The Indigo Studio (by Infragistics) [17] software tool was used as it is intuitive to use and the available features allow for recording User Testing metrics such as correct and incorrect clicks, as well as added user comments. Based on user feedback, this prototype combined elements from both initial paper prototypes.

Users evaluated the prototype in this phase by completing a task-based User Test online and leaving comments wherever necessary. This user test was created with the goal of determining whether or not the design was intuitive. The feedback from the test indicated whether users would click on the correct buttons or look in the correct places to achieve a task.

Choosing to complete this task online occurred due to external factors which made it difficult to meet users in person at this phase. However the online test was conducted using the website IndigoDesigned which is affiliated with the prototyping software Indigo Studio. The prototype was hosted online and users completed a set of tasks on the prototype while all their movements, clicks and interactions were recorded. User comments created a constructive way to suggest improvements as being able to type up responses anonymously removes some of the inhibition of the participant. One of these users had difficulty completing the online test and requested to meet in person to provide adequate feedback. Three users were selected for this task by using the two existing participants and selecting a third through Convenience Sampling.

3.4.3 Third Iteration

The third iteration of the prototype was designed using the same UX software as used in the second iteration, Indigo Studio. This software was very useful in the second iteration by allowing fast prototyping of a functional interface.

The users involved in evaluation at this stage were the same users from the previous iteration. This prototype was evaluated using three methods.

The users interacted with the prototype in an individual task-based User Test. The setting were office spaces with minimal interference. Users attempted to complete the tasks in the order assigned to them. These tasks range from simple interactions to multiple steps. The users were encouraged to Think Out Loud and were recorded talking through the process of completing the tasks [28]. This method can be used to ensure that the prototype is fulfilling user needs in terms of tasks, as well as testing the flow and usability of the prototype.

The next method of evaluation was a System Usability Scale (SUS) Survey in which users rated aspects of their experience on a scale from 1 to 5. The interpretation of the results is gathered by adding the users' answers in a certain way (Odd answer - 1 and 5 - even answer) and then multiplying the sum by 2.5 to get a percentile.

Finally users commented in a conversational setting on their overall experience with the prototype in relation to the current software available.

4. RESULTS

4.1 Requirements Gathering

The expert users clarified the software requirements of an average astronomer working in radio astronomy. A high volume user will open a visualisation tool such as Karma 10-20 times a day to view different date sets. Most commonly experts view a radio data cube frame by frame along the three different axes: x,y, and z. Frames can be played as movie and traversed to specific points. Beyond this, various data statistics and co-ordinate information must be displayed, axes viewed or removed from the data view and analysis of the data can be viewed (such as a histogram of the emissions collapsed along a certain axis).

Expert users described software such as Karma to be very powerful in terms of functionality, but not easy to use. Examples of Karma's shortfalls are: the fact that closing one sub-window exits the entire session, there are multiple windows and steps to address before simple, basic tasks can

be accomplished (such as selecting a preferred colour map, applying axes and file browsing), preferences (colour maps, scaling, axes) need to be respecified every time the program is booted, the interface is highly technical and straining to use for long periods and the multiple windows are small and lead to cluttered screens.

After discussing current problems, experts brainstormed ideal features, including specific features to increase effectiveness as well as an overall improvement in interface usability. Suggestions included; viewing two radio cubes simultaneously for comparison, shortcut keys, standard sub-menus instead of new pop-up windows, an easy access to playing a movie of the data, annotations of certain points in the data, colour scales to represent quantitative meaning and displaying co-ordinate data when hovering over the frame. We attempted to address as many of these requirements as possible in our first design.

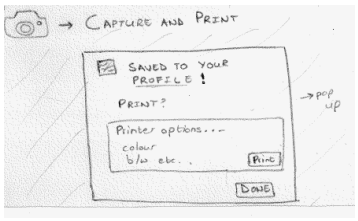
4.2 First Iteration

The first design cycle produced two paper prototypes depicting alternate interface options for the astronomy visualisation software. Designs both provide an overview of the data in the main view, with further detail on demand. Additional (and novel) features of the prototypes include the ability to annotate the data with flagging. The functionality and features designed to increase the usability of the software are the same across both prototypes, for example, swapping axes would produce the same result in both prototypes. Both designs use icons to facilitate user recall. Icons are consistent across both prototypes.

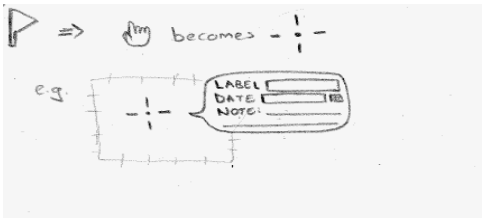
There are eight main features represented by icons. The first feature, represented by a camera, is to capture an image of the data and saving this in a format suitable for publication (Figure 1(a)). A flag icon represents the ability to flag (annotate) the data with comments. Once clicked, the cursor will change to a pointer to allowing the user to select a point on the frame to flag and to save a note or comment on that data (Figure 1(b)). Users can swap axes around (e.g. from XY to XZ) easily using the axes icon (Figure 1(c)). A settings menu is represented by a gear icon to easily change and save preferences for future sessions (Figure 1(d)). Most importantly, a user profile allows a user to save preferences in settings as well as allows for saved captures and flags to be stored (Figure 1(e)). The ability to play through the frames as a movie was seen as a necessary addition to the home screen and is represented in the play icon which changes to a pause icon when clicked and vice versa (not pictured). The calculator icon represents a section where further calculations such as statistical measures and histograms can be found (Figure 1(f)). Finally, a cube with a plus on it represents the ability to split the screen in two and view an addition data cube for comparison (shown in (Figure 2(a)). An additional feature provides an interactive three dimensional cube of the data along with the images viewed by frame to enhance contextual understanding of the data. While the layout is different, all functions, icons and buttons would perform the same actions across both designs. The two designs only differ significantly in the layout of the interface as described below.

4.2.1 Design 1

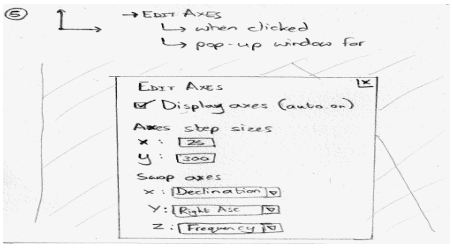
The first design has a large, highly visual screen with an expanding side menu on the right containing all the icons



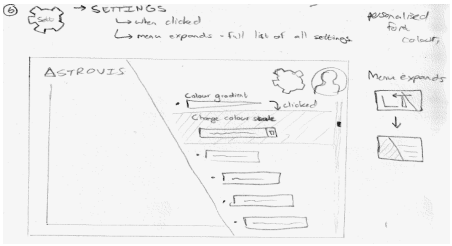
(a) Capture Function Dialog



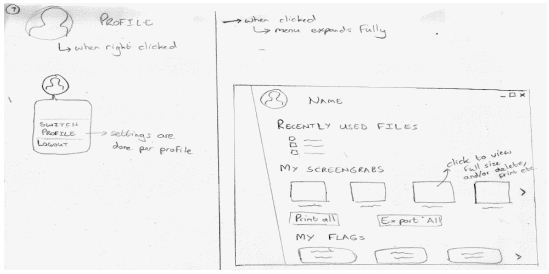
(b) Flag Function Dialog



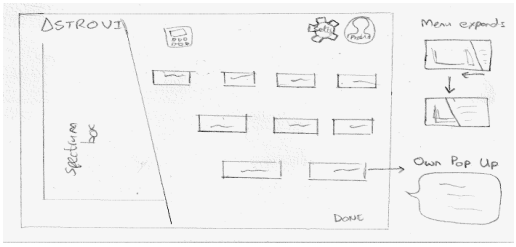
(c) Axes Preferences Dialog



(d) Settings Expanded Menu

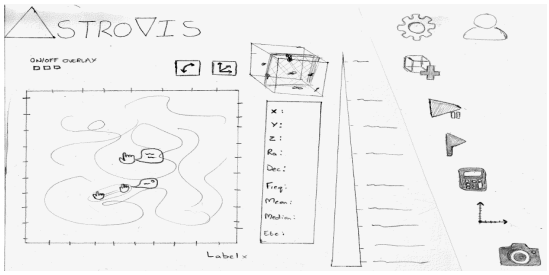


(e) Profile Expanded View

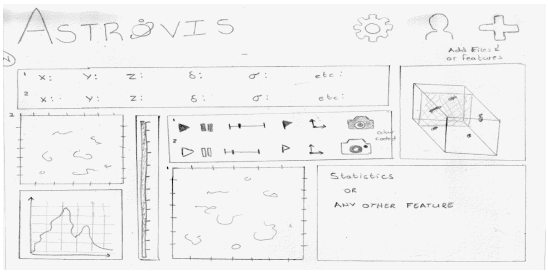


(f) Further Calculations Expanded Menu

Figure 1: Functionality of Suggested Features



(a) First Design



(b) Second Design

Figure 2: Home View of Each Design

representing additional functionality (Figure 2(a)). This is to allow for as much space as possible for users to complete visual queries about the data. Prominent on the side menu (in the middle) is the play button, which allows the main visualisation of the slices of a radio cube to viewed either as stills or played as a film. The side menu allows for less clutter on screen and more space in the visual section of the layout (viewing by frames, three dimensional view). There is also an option to alternate the main focus of the window to an interactive view of the 3D data cube and vice versa. To the immediate left of the side menu is a triangular scale depicting the values corresponding to the colour scale/map used in the main visualisation of the data cube. There is also a space dedicated to showing important updated data (co-ordinate and statistical) which is easy to view without referring to the menus.

4.2.2 Design 2

The second design is a customisable widget based view (Figure 2(a)). The goal of this design is to allow expert (high volume) users to set up their most commonly used features as fixtures on the main screen with as little effort as possible. Each box and object drawn represent different functions and tools which can be dragged on and off screen in order to allow expert users to customise their own experience. Even though this is a customisable interface, the layout drawn is the standard recommended view which would be available the first time the program is used. Thereafter, users could change certain widgets/features to incorporate their most used functions and make them all easily accessible (e.g. some further calculations). The big plus icon in the top right of the screen will be used to add features to the screen. If a widget is dragged towards the plus button, it will turn into an 'x' and releasing the widget over this x removes it from the screen. Next to this plus button is the profile and settings which will be constantly available in the header ribbon.

4.2.3 Evaluation

When reviewing the first design, expert users enjoyed being able to view an overview first and then details on demand. The range of functions (including innovative functions such as flagging the data with comments) were a welcome addition. Many of the intuitive issues with existing software such as Karma are solved by an updated design and having a user profile to save settings such as visible axes and colour scales for viewing the data. Users remarked that the task of viewing and interacting with data cubes will be much faster without all the set up usually required.

Users remarked that using screenshots for capturing the images would not be viable as the images will not be of a high enough quality for published papers. However, they suggested the use of vector graphics or images saved as a PNG. Furthermore, export of images could benefit from the addition of meta data (session information such as file name, time, and custom fields) as tags.

Further expert user suggestions were that: the add button should implement a split screen for two cubes (for comparison) and tabs for three or more cubes; a file browser could be included in the main screen to show files currently visualised and to find more; the flags should be saved separately to the file to avoid messing up the file and so that the flags of comments on a radio file could be overlayed onto an optical file for example; and that dditing the axes via the shortcut menu buttons should give the user the option to save these changes to the profile.

Overall the expert users found Paper prototype 2 to be extremely useful with the ability to save users a few tens of hours per project/goal as the screen can be customised. However, the expert users thought that there might be a problem with adoption of such an unconventional type of interface. Prototype 1 is simpler and more likely to be adopted by more traditional users. The expert users suggested prototype 1 as a home view with prototype 2 as the expert view.

In terms of the methodology, paper prototyping proved to be a very difficult medium to work with when making the prototypes interactive, as the software is complex with many functions. However, using paper and sketches did allow for more innovative designs and the end users enjoyed the ability to manipulate the prototype and point to different aspects when giving feedback. The use of paper also allowed the users to draw on the designs, giving the review sessions the option for occasional co-designed elements. The freedom of the format of the review session also served to enhance the brainstorming and engage the users.

4.3 Second Iteration

The second design phase incorporated feedback from expert users to create an interactive simplified prototype of the GUI [2]. The prototype used dummy data for a mock-up of the kind of input (text, statistics, images and so on) the user would encounter in a functional implementation. Features carried through from the first prototype have the same functionality in this iteration, unless otherwise specified.

4.3.1 Design

This prototype combined both designs from the first iteration (Figures 3 and 4). The basic layout (Figure 3) is

[2]The second design is available at https://indigodesigned.com/share/3mn759zpj0y6

derived from Design 1, with a triangular expanding menu used to house the icon buttons. The main area of the screen is devoted to the viewing panels for the 3D cube, viewing a radio data cube by stills, a file browser and statistics on the data. These panels resizeable and customisable as in Design 2.

A hamburger menu was chosen to save space (Figure 3(a) and expanded in Figure 3(b)). The menu items have been reduced to six items from eight. User profile and settings remain largely unchanged, with more detail on the type of information. The profile option expands the triangular menu fully, to cover most of the screen, and depicts recently used files and recent images captures and flags added (Figure 3(c)). Settings allow the user to change and save preferences (Figure 3(d)), including the presence of axes (a separate shortcut icon in the last iteration).

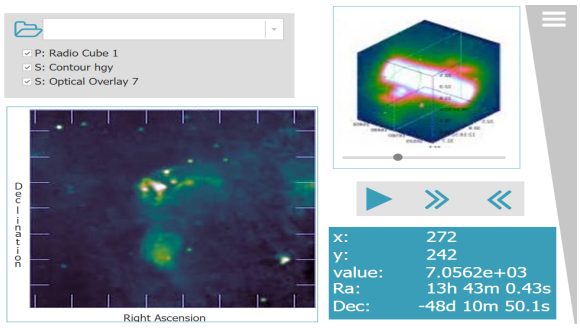
Capturing and flagging images have more specific details included such as exporting formats and metadata tagging for captures (Figure 4(a)) and changing the cursor when flagging (Figure 4(b)). 'Further calculations' is a feature which is not completely explored but rather used as a placeholder for complex statistical data a user might need (Figure 4(c)). Finally, the add button's functionality was changed in this iteration (Figure 4(d)). The intended functionality of this feature is two-fold. The button can add data to the home screen (either as an additional viewing panel or as an overlaying of data onto the primary radio data cube) as well as add other features (such as statistics or a file browser) to the home screen as additional panels and remove existing panels using checkboxes. This is intended to aid the customisation of the screen view.

4.3.2 *Evaluation*

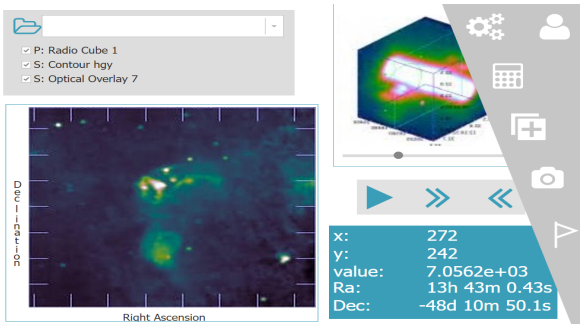
The users liked the prototype overall, especially in terms of the additional features. They liked the function of moving through a cube (the play button and slider) and found elements such as the statistics panel on the home screen and a list of recent captures and flags useful. One user commented that features such as annotation (flagging) of the data in real time and a profile to save customisable settings make this prototype a great improvement on current software. In particular, the inclusion of the 3D cube was appreciated as the user noted that the 3D view provides much more context to the standard view when stepping through frames.

Some feedback was conflicting, however, as one participant liked the icons - calling them intuitive - while another participant found some of the icons unintuitive and suggested leaving the labels on the icons all the time rather than only being visible when hovering over an icon. In fact this participant did not like the hamburger bar and suggested to permanently expose these tools while perhaps giving an option to hide them, while another user liked the flag button but would like it to be more immediately visible in the page.

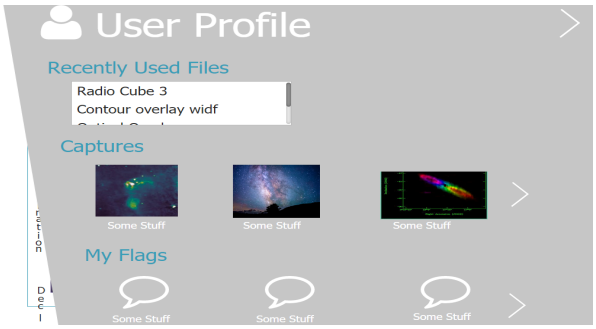
Other criticisms include that it would be easier to find saved Captures (or flags) if they were available under the capture (or flag) icon rather than only in the profile. The user noted that changing this and other small things could make navigation more intuitive. Users questioned whether visualisation options (such as colour maps) should be placed in settings, as they expected only global application preferences (such as font) there. A user suggested a separate menu to house visualisation preferences. Another user originally liked the triangular toolbar but now feels that it takes up



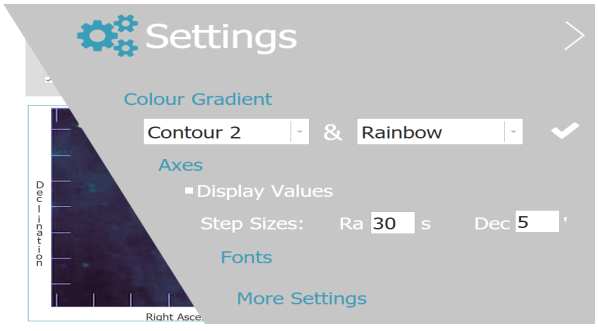
(a) Home View



(b) Expanded Menu

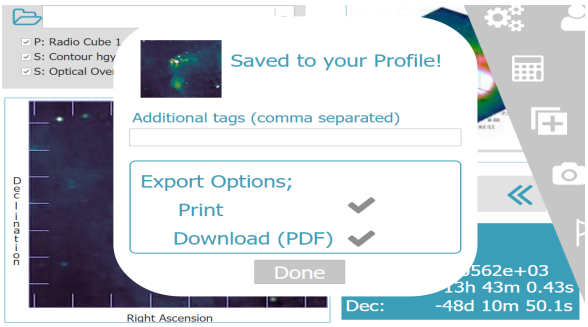


(c) Profile Expanded View



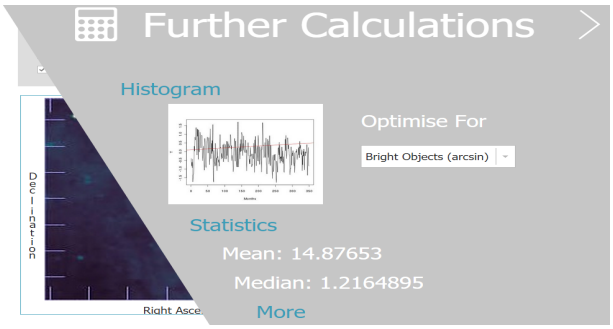
(d) Settings Expanded Menu

Figure 3: Basic Overview

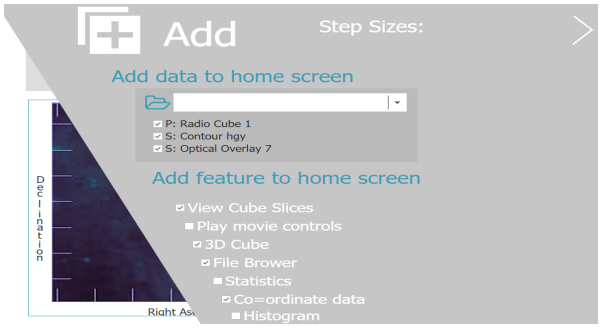


(a) Capture Feature Dialong

(b) Flag Feature Dialog



(c) Further Calculations Expanded Menu



(d) Add Expanded Menu

Figure 4: Functionality of Features

too much space in this prototype.

Further expert user suggestions were to provide access to more tools from the home screen (such as editing contrast, swapping axes, colour maps, scaling schemes etc) and to include shortcuts for high volume users (such as ctrl+click to skip ten frames instead of skipping through one at a time).

Overall the users suggested minor changes to enhance the navigation of the system. However, the functionality was often described as useful and an improvement in the usability of astronomy software. Users liked features such as the 3D cube, annotations in real time and capturing images. Users appreciated features readily available on the main view (without requiring menus to access). However, users had some concerns in terms of the menu taking up too much space.

In terms of methodology, prototyping software was highly useful, although certain aspects of the software were only understood near the end of the design process. For example, the proportions of standard screen sizes as set by the prototyping software resulted in an oddly proportioned prototype, and this was only truly understood near the end of design. An online user test was very useful in terms of proving metrics such as successful and unsuccessful clicks, time taken to complete tasks and allowing for forthright comments from the user. However certain issues arose in the test and resulted in one user needing to rather do a test in person, which skewed the metrics and rendering them misleading.

4.4 Third Iteration

The final prototype is based on all user feedback and the works of Shneiderman [42] and Nielsen [31]. The prototype [3] is designed to be a Horizontal-Prototype [35].

4.4.1 Design

The GUI design removed the possibility of customising the home screen too much. The design goals were to maintain a minimalist design while still reducing the cognitive load on the user by using icons for recognition, sticking to conventional practices, focusing on aesthetic appeal and incorporating functions in an intuitive way. The user profile is simple and has the same functionality as previous iterations (Figure 6). Figure 5(a) depicts the initial view of the system. The triangular side menu has been changed to a horizontal ribbon menu at the top of the screen in order to maximise space for the visualisation aspects. The majority of the screen space is used for the panel which steps through the frames of a radio data cube. To the right of this panel is a control panel with five buttons, a 3D view of a radio data cube and an informational field with tabs for the histogram and statistics. Running down the left of the screen is a file browser which also depicts current files in each of the visualisation panels. An item listed here can be expanded by clicking on the little arrow next to each entry to show statistical data and allows for changing of the colour and scaling schemes. Finally, a toolbar along the bottom of the screen allows the user to skip to certain frames in the cube, to play through the frames like a movie and to scroll to a point in the frames using a slider.

It is possible to interchange the two visualisation panels using the 'swap' button (Figure 5(b)), in order to view both

[3]The prototype is available at https://indigodesigned.com/share/vn61mym8xkc4

visualisations. The 3D cube can be rotated in either position. Figure 5(b) also depicts the histogram tab in the information panel and the reduction of the submenu in the file browser on the left. The XY, XZ and YZ indicate switching to the respective axes for the data being viewed by frame.

Submenus (instead of the multiple pop-up windows as used in Karma) allow users to easily accomplish tasks. Existing flags and captures can now be found under their respective menus (Figure 6(a)). The cursor becomes a pointer when selecting a point in the frame to flag. Once selected, a dialog prompts users to annotate and save the flag (Figure 6(b)). The capture feature works similarly to the flag feature and can directly export and save the captured images (Figure 6(c)). Finally, the new feature, Compare (based on the add feature from the first iteration) allows users to select and compare two data sets (Figure 6(d)). The data sets can viewed by frame or played as a movie simultaneously. The rest of the screen is removed for this feature in order to maximise the ability to solve visual queries (Figure 6(e)).

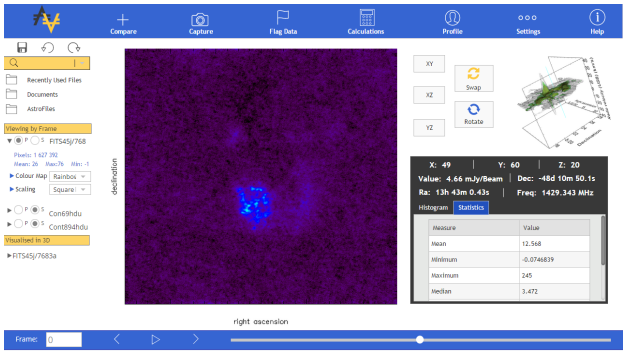
4.4.2 *Evaluation*

Final feedback was largely positive. Users found the system well laid out and enjoyed the clean and modern feel to the design which they felt was consistent with modern software. They also remarked that the clean interface made it easy to find relevant functions. Users particularly liked having both the 3D cube and 2D frame view in one interface to interact with together. They liked the capture feature to export to various formats. The flag feature was described as the reason this software is very effective. Overlay features were very clearly identified in the file browser. Participants liked the data displayed directly by default onscreen, as it would aid in data analysis. In fact, the way data was displayed overall was described as very useful. Most tasks were completed with ease and icons were intuitive.

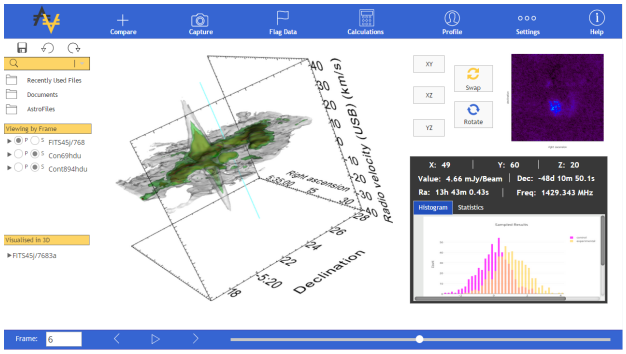
The SUS survey (Table 2) results are: 87.5%, 92.5% and 97.5%. The average of this result is 92.5%, which is a positive result. Bangor et al. found that SUS surveys are robust evaluations [2]. Further, Bangor et al. found that the average score for a system is 70%, however, scores over 90% indicate truly superior products.

There was some confusion about the rotate function needing a click and hold interaction. One user found this unintuitive, another user liked it and the third user felt it was a matter of preference. Most users navigated the control panel (consisting of the XY, XZ, YZ, swap and rotate buttons) quickly. The found it clear and intuitive due to the icons and labelling. They also remarked that the directly accessible buttons would speed up data analysis. However, one user was confused by which visualisation each button refers to (either the 2D viewing by frame, 3D cube or both). This user suggested that the swapping axes buttons (XY, XZ, YZ) should be placed on the frame view and that the rotate button should be on the 3D cube with a rotation element to pop up.

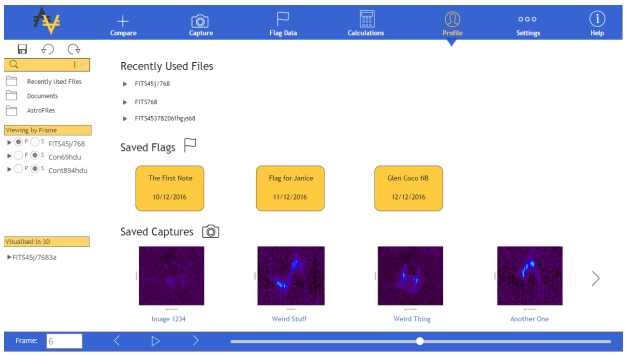
Users had other suggestions including to: add clarity to the scientific analysis; add a description of the format type when saving images; use a dialog box with drop downs rather than submenus (Figure 6(d)) to capture images and to add keyboard shortcuts as astronomers tend to use these often. One user also noted that although the data was displayed in a useful way, there was not much indication as to how this prototype would handle very large data sets.



(a) Home View



(b) Switching Visualisation Panels



(c) Profile View

Figure 5: Final Prototype Basic Overview

Table 2: Raw SUS Results

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
P1	3	1	5	2	4	2	5	1	5	1
P2	4	2	5	1	5	1	4	1	5	1
P3	5	1	5	1	4	1	5	1	5	1

Overall, the users responded positively to this design and found it to be an improvement on current software in terms of usability and useful features.

A participant had this to say about this GUI prototype;

"Very easy to use, nicely intuitive. Overall I think this interface is far superior to some of the existing visualisation software commonly used in radio astronomy in terms of usability (i.e. easy to pick up and use without training) and in terms of usefulness."

Another user remarked;

"It meets all of the basics requirements that astronomers require: data stats, exporting images, etc. This package could easily be used by professional astronomers."

5. DISCUSSION/ANALYSIS

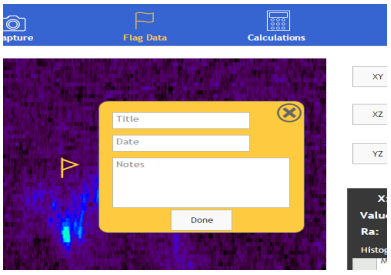
This study aims to determine whether the use of UCD principles in the design of a software interface for astronomy visualization, will produce a design that is useful, usable and effective. This was investigated using an iterative prototyping approach which involved users as much as possible. Findings from this study indicate that domain expert insight is highly valuable when designing for complex domains. It is important to continuously gain these insights as scientific software projects rarely have robust, clear requirements at the beginning of any project. The requirements need to be developed and conveyed to the designer progressively to avoid large misunderstandings, especially early on. Findings indicate that this progressive refinement will result in more useful software. These findings are supported by existing knowledge. For example, the concerns of usability being difficult to achieve in complex domains [44] were confirmed in this study. Furthermore, the Omero project found that continuous collaboration with end users resulted in a great number of (important) problems identified and fixed in the software [29]. Similar results were found in this study, as some of the functionality of prototypes improved drastically over iterations (such as the 'add' function).

The designs presented in this study differ noticeably from the designs of existing astronomy visualisation tools (described in Section 2.6). The innovative functions added were positively received by users in each iteration. These functions included the capability to flag, capture and compare data. The findings indicate that these basic functions would be useful as embedded features in most scientific software, as scientists frequently compute these queries visually using other means (for example, hand-written notes). Icons reduced the cognitive load on users (one of the guidelines in Table 1) as users either found the icons immediately intuitive or easily learnable (another guideline from Table 1). Users were able to point and click rather than type and remember - which Nielson found solved up to 22% of usability problems [31]. This is an important finding as icons were rarely used in the astronomy visualisation tools described above (Section 2.6) and this resulted in a trend of overly technical interfaces which increased cognitive complexity. An important finding for visualisation software is that at every iteration, users enjoyed having an overview of a much data as possible in the home screen (while maintaining a clean layout) and then further detail on demand. This finding makes use of Shneiderman's visualisation mantra; *overview first, zoom, filter, details on demand* [42]. The first iteration's designs received mixed comments and some important problems were identified early on (such as the need for a file browser in the home view). Out of the nine criteria in the Design Guidelines (Table 1), the designs fulfilled only two adequately - low cognitive load and high learnability. Design 2 on its own also provides flexibility through customisation of the interface. In contrast, the third iteration adequately fulfils all Design Guidelines, with findings reflecting that some additional thought is needed for error handling and consistency. This improvement was achieved through continuous communication with end users. The final prototype's design was found to be more usable than existing software, due to the use of UCD principles.

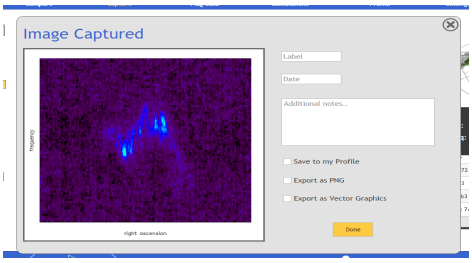
The methodology of this study was largely successful based on the positive evaluation of the final prototype. The key components of this approach were involved users and iterative phases. Without involved users, the UCD approach falls apart and our findings stress the need for domain experts who are intended users. This is because these are the only users who would be able to address misunderstandings and major domain problems in each design. Dumas and Redish also recommended using expert users who would be intended users [9]. Iterative phases were found to be necessary to ensure the progression of the usability of the final product (as can be seen in the difference between the first and final prototypes). Time needs to be set aside for both design and evaluation of prototypes. One design method, Paper Prototyping, proved to be problematic as the complexity of astronomy visualisation software was difficult to represent on paper. However, it highlighted key requirement misunderstandings quite quickly. Interactive, computer based prototypes proved to be far more successful in examining usability and effectiveness. Each evaluation method used yielded many suggestions for improvements through commenting - making each evaluation successful. An important finding is the importance of the SUS survey, as the quantitative results could firmly determine the final prototype as excellent and a success to end users.

The limitations of this study include the feasibility of this design. While many of the features are aesthetic or functional, some might be difficult to accomplish in practice such as loading multiple data sets in different views (frame-by-frame and 3D). A subsequent iteration could address the way extremely large volumes of data would be handled by the prototype. The study could also be limited by bias to-

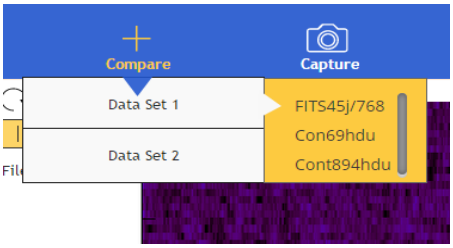
(a) Flag Submenu



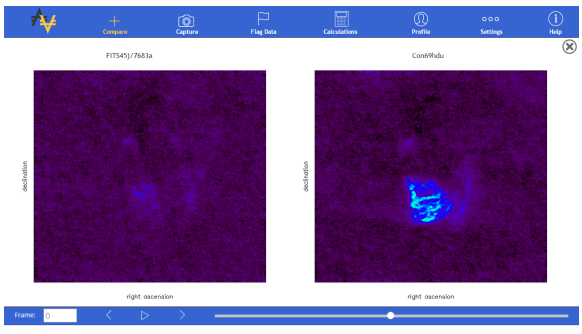
(b) Flagging Dialog



(c) Capture Dialog



(d) Compare Submenu



(e) Compare View

Figure 6: Functions and Submenus

wards the designer in the users feedback. However this was mitigated in the final iteration by recording the user test but leaving users alone (to alleviate pressure) while they completed the SUS survey and provided comments. This study was also somewhat limited by the scarcity of suitable participants, as only domain experts and intended users were sought.

6. CONCLUSIONS AND FUTURE WORK

The usability, effectiveness and usefulness of the astronomy visualisation interface prototype was improved significantly by taking a User Centred Design approach and involving users as much as possible. This improvement refers to the prototype being an improvement on the designs of current astronomy software tools, achieved by involving users continuously in a usability centred study. The improvement also refers to how an iterative approach improved the quality the prototypes over time. This study found that UCD approaches can be adopted for non-traditional domains. In the context of the current standards of scientific and astronomy visualisation tools, this conclusion is valuable. It shows a way for the multiple usability problems facing scientists to be alleviated and to improve the quality of scientific research.

There is a gap in the literature on studies of usability in scientific software domains. This needs to be addressed in order to develop software standards for usability in complex fields. Based on the exceedingly positive SUS score and comments, it can be seen that this approach is worth the additional time and effort - if the project values usability, usefulness and effectiveness highly. Based on the trends of poor usability in astronomy visualisation software as described above - it is recommended that usability, usefulness and effectiveness are prioritised in the development of subsequent tools. This recommendation is extended to all scientific software fields. Despite the inherent difficulty in making scientific software more usable, it is possible and worthwhile to do so.

Some design findings are applicable to other scientific software fields such as reducing cognitive load with icons and providing and overview first with details on demand. These are visual findings which can also be applied to any type of visualisation software. Functions such as flagging, capturing and comparing data can be used in any scientific field as these functions aid scientific discovery.

Further research can investigate other design methods to improve usability, can implement a fully functional implementation based on this design or extract the design elements used and adapt them to other scientific software projects. Potentially, more scientific and visualisation software projects will incorporate UCD into the software development process and break the trend of poor usability.

7. ACKNOWLEDGMENTS

I would like to thank my supervisor Michelle Kuttel for exceeding expectations with all her support, input and time - without which the project would not be possible. I would also like to thank expert users Sarah Blyth, Ed Elson and Adrianna Pińska for their invaluable insight and continuous availability. Thank you to the Department of Science and Technology and the Square Kilometer Array.

Finally I would like to thank Premie Rampersad, Vi-

jay Rampersad and Saieshan Vandayar for their unwavering support, help and encouragement.

8. REFERENCES

[1] C. Abras, D. Maloney-Krichmar, and J. Preece. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456, 2004.

[2] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[3] U. Becciani, A. Costa, V. Antonuccio-Delogu, G. Caniglia, M. Comparato, C. Gheller, Z. Jin, M. Krokos, and P. Massimino. Visivo-integrated tools and services for large-scale astrophysical visualization. *Publications of the Astronomical Society of the Pacific*, 122(887):119, 2010.

[4] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[5] P. Camps and M. Baes. Skirt: An advanced dust radiative transfer code with a user-friendly architecture. *Astronomy and Computing*, 9:20–33, 2015.

[6] P. K. Chilana, J. O. Wobbrock, and A. J. Ko. Understanding usability practices in complex domains. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2337–2346, New York, NY, USA, 2010. ACM.

[7] G. Chiozzi, K. Gillies, B. Goodrich, S. Wampler, J. Johnson, K. McCann, G. Schumacher, and D. Silva. Trends in software for large astronomy projects. In *11th ICALEPCS Int. Conf. on Accelerator & Large Experimental Physics Control Systems, Knoxville*, pages 13–17, 2007.

[8] M.-F. Costabile, D. Fogli, C. Letondal, P. Mussio, and A. Piccinno. Domain-expert users and their needs of software development. In *HCI 2003 End User Development Session*, 2003.

[9] J. S. Dumas and J. Redish. *A Practical Guide to Usability Testing*. Intellect Books, 1999.

[10] G. Fabbiano, C. Brogan, D. Calzetti, S. Djorgovski, P. Eskridge, Z. Ivezic, E. Feigelson, A. Goodman, B. Madore, M. Postman, et al. Recommendations of the virtual astronomical observatory (vao) science council for the vao second year activity. *arXiv preprint arXiv:1108.4348*, 2011.

[11] A. Følstad. Work-domain experts as evaluators: usability inspection of domain-specific work-support systems. *International Journal of Human-Computer Interaction*, 22(3):217–245, 2007.

[12] R. Gooch. Karma: a visualization test-bed. 101:80–82, 1996.

[13] A. A. Goodman. Principles of high-dimensional data visualization in astronomy. *Astronomische Nachrichten*, 333(5-6):505–514, 2012.

[14] P. Hall. The square kilometre array: An international engineering perspective. In *The Square Kilometre Array: An Engineering Perspective*, pages 5–16. Springer, 2005.

[15] A. Hassan and C. J. Fluke. Scientific visualization in astronomy: Towards the petascale astronomy era. *Publications of the Astronomical Society of Australia*, 28(02):150–170, 2011.

[16] J. Howison and J. D. Herbsleb. Scientific software production: incentives and collaboration. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 513–522. ACM, 2011.

[17] Infragistics. Indigo studio.

[18] I. ISO. Iec 9126, software product evaluation–quality characteristics and guidelines for their use. 2001. *International Organization for Standardization*.

[19] B. E. John and D. E. Kieras. The goms family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(4):320–351, 1996.

[20] W. Joye and E. Mandel. New features of saoimage ds9. In *Astronomical data analysis software and systems XII*, volume 295, pages 489–490, 2003.

[21] A. T. Kelly. The perfect brainstorm. *Kelly, T. & Littman, J. The Art of Innovation*, 2000.

[22] D. Kelly and R. Sanders. The challenge of testing scientific software. *CAST 2008: Beyond the Boundaries*, 2008.

[23] A. K. Kembhavi, A. A. Mahabal, T. Kale, S. Jagade, A. Vibhute, P. Garg, K. Vaghmare, S. Navelkar, T. Agrawal, A. Chattopadhyay, et al. Astrostat - a vo tool for statistical analysis. *Astronomy and Computing*, 11:126–137, 2015.

[24] C. Kiddle, M. Andrecut, A. Brazier, S. Chatterjee, E. Chen, J. Cordes, R. Curry, R. Este, O. Eymere, P. Federl, et al. Looking towards the future of radio astronomy with the cyberska collaborative portal. In *Astronomical Data Analysis Software and Systems XX*, volume 442, page 669, 2011.

[25] B. Laugwitz, T. Held, and M. Schrepp. *Construction and evaluation of a user experience questionnaire*. Springer, 2008.

[26] O. Laurino, J. Budynkiewicz, R. DâĂŹAbrusco, N. Bonaventura, I. Busko, M. Cresitello-Dittmar, S. M. Doe, R. Ebert, J. D. Evans, P. Norris, et al. Iris: An extensible application for building and analyzing spectral energy distributions. *Astronomy and Computing*, 7:81–94, 2014.

[27] H. Levkowitz and G. T. Herman. The design and evaluation of color scales for image data. *IEEE Computer Graphics and Applications*, 12(1):72–80, 1992.

[28] C. Lewis and J. Rieman. Task-centered user interface design. *A Practical Introductio*, 1993.

[29] C. Macaulay, D. Sloan, X. Jiang, P. Forbes, S. Loynton, J. R. Swedlow, and P. Gregor. Usability and user-centered design in scientific software development. *IEEE Software*, 26(1):96, 2009.

[30] J. R. Miller and R. Jeffries. Interface-usability evaluation: science of trade-offs. *Software, IEEE*, 9(5):97–98, 1992.

[31] J. Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 152–158. ACM, 1994.

[32] P. Parsons and K. Sedig. Distribution of information

processing while performing complex cognitive activities with visualization tools. In *Handbook of Human Centric Visualization*, pages 693–715. Springer, 2014.

[33] F. Paz and J. A. Pow-Sang. Usability evaluation methods for software development: A systematic mapping review. In *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*, pages 1–4. IEEE, 2015.

[34] M. Rettig. Prototyping for tiny fingers. *Communications of the ACM*, 37(4):21–27, 1994.

[35] J. Rudd, K. Stern, and S. Isensee. Low vs. high-fidelity prototyping debate. *interactions*, 3(1):76–85, 1996.

[36] J. Ruiz, J. Santander-Vela, V. Espigares, L. Verdes-Montenegro, and J. van der Hulst. Gipsy 3d: Analysis, visualization and vo tools for datacubes. In *Astronomical Data Analysis Software and Systems XVIII*, volume 411, page 406, 2009.

[37] J. Scholtz. Usability evaluation. *National Institute of Standards and Technology*, 2004.

[38] K. Sedig, P. Parsons, M. Dittmer, and R. Haworth. Human-centered interactivity of visualization tools: Micro-and macro-level considerations. In *Handbook of Human Centric Visualization*, pages 717–743. Springer, 2014.

[39] A. Seffah and E. Metzker. The obstacles and myths of usability and software engineering. *Communications of the ACM*, 47(12):71–76, 2004.

[40] J. Segal. When software engineers met research scientists: A case study. *Empirical Software Engineering*, 10(4):517–536, 2005.

[41] J. Segal and C. Morris. Developing scientific software. *Software, IEEE*, 25(4):18–20, 2008.

[42] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996.

[43] S. Silva, B. S. Santos, and J. Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35(2):320–333, 2011.

[44] J. Viitanen, S. Karjalainen, H. Kautonen, and A. Laukkanen. Challenges in usability evaluation of expert domain products. In *Proc Intl Conf HCI'05*. Citeseer, 2005.