Usability in Scientific and Astronomy Software

Honours Literature Review

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 of domain experts and usability professionals. This would take place throughout development and include stages of usability evaluation. There are many methods of evaluating usability. After analysis of these methods, it is suggested that user testing and heuristic evaluation would be the most productive for the domain of scientific software development. One type of scientific software, astronomy software, is a relatively new pursuit and usability practices in this field are under-utilised. Different software tools in the field are analysed in this paper. The common trends and standards indicate that user friendly software seems to be highly challenged in the field of software for astronomy. A focus on introducing usability practices in this field of software development through design and evaluation could result in more effective software.

Keywords

Usability; Scientific Software; Astronomy; Visualization

1. INTRODUCTION

Usability considerations are necessary during software development for people. They allow for a greater degree of efficient interaction between the software and the intended users. Usability has 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 isn't a consistent knowledge base to address usability [27]. The fields of Human Computer Interaction (HCI) and User Centred Design (UCD) endeavour to study the many methods to implement and evaluate usability. Spanning from user satisfaction to learnability, there are a range of attributes to consider in making software more appealing and useful to users.

In conventional domains (such as mobile applications and web interfaces), there is widespread proliferation of usability methods. However, in complex and domain specific software projects, usability is often challenging and not established by research [4].

These complex, domain-specific contexts include the field of scientific software - for improved workflows and research [18]. Brooke's notion is that usability is a concept which exists with reference to a specific context [2]. This implies that usability standards (which are already fraught with variation and are difficult to measure [27]) will be further disparate depending on the situational context of the software.

This review critically analyses the use of usability practices and evaluations in scientific software development, with a focus on software in Astronomy. Software development in Astronomy is a rapidly expanding field with a focus on Astronomy visualization, analysis and collaborative software.

Section 2 will discuss usability in terms of Scientific Software with reference to the unique goals, requirements and challenges in this field. Section 3 discusses methods of evaluating usability and how effective these evaluations are. In section 4, Astronomy software is explored. This software is divided into one of three categories: analytical, collaborative or visualisation. The various Astronomy software packages are then commented on and compared using observation and the appropriate literature. Finally, section 5 gives an overview of the most relevant findings of the review, and also suggests where further research and work is needed to address to issues of usability in software for complex domains such as Astronomy.

2. SCIENTIFIC SOFTWARE

Interviews by Kelly and Sanders [24] found that scientific software is developed for three main purposes, namely: research, training and external decision support (software moved into new industries to support further research). These purposes overlap and are connected by the cognitive complexity of the subject matter. Despite the growing importance of this field of software development, there has been little research contributing to software standards in this field [12]. In designing software, certain traits are common in scientific fields.

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

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 [18, 29].

In the first scenario, the customer and end user is the research scientist. A typical problem of this scenario is that it is rare for the exact requirements of the project to be known in advance [28]. This is a huge disadvantage in a field where developers aren't acquainted with the domain expert activities. Costabile et al. separates these activities into two classes [6]. Class 1 consists of activities involving tailoring the software to choose between existing behaviours/interactions such as 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 understanding the tasks the end user will perform, usability criteria like learnability and ease of use won't be addressed adequately.

In the second scenario where the domain expert/ scientist is the developer, learning the required software development skills may take a lot of effort. This effort will increase for the quality of the software to be more than just adequate. Good software for science should be able to be extended/adapted for unforeseen requirements and 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) [29]. 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). This was found in a survey by Kelly and Sanders [13]. The survey interviewed scientific domain experts who were developing software. The survey found that interviewees would often omit usability testing and user documentation entirely, if the end users would be fellow scientists working in the same field.

A study by Segal [28] suggested that a collaboration of software developers and research scientists should create the software - with the scientist in the role of a hybrid peer-programmer and end user. Segal also notes in another paper [29] that a problem may arise when the developer is a software engineer rather than an end user developer. The problem is that incredibly extensive requirements would be necessary, however, as stated - these are not always available. It can be seen that a "persistent partnership" [4] between domain experts, software engineers and usability developers is required, although this may be unrealistic in software development which isn't commercial, due to funding and scope of the project.

2.1 Case Study

In the Omero/Usable Image Project [18], a team of usability experts (Usable Image project) and software engineers collaborated on a life sciences software project named Omero[1]. 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. The conclusions of this case study involved the notion 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 [18]

3. USABILITY EVALUATION

Usability testing is more than just a once-off activity at the end of a software project. It is only through continuous interaction with end-users that a software project will transcend being merely a tool that does the bare minimum functionality. One model of continuous usability testing, as described in the Omero project [18], 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. This is quite an intensive method and not all software projects have the budget or scope for this level of testing.

In fact, the cost, effort and experience required for a particular usability evaluation method may affect which one is used.

There are three types of usability evaluation methods: user-based, expert-based and model-based [25]. User-based methods have users as the source of evaluation methods and these methods include user testing, interviews and walkthroughs. Expert-based methods rely on usability experts having knowledge of best practices. Examples are heuristic evaluations, guideline reviews and cognitive walkthroughs. Finally, model-based methods simulate human interactions and perceptions using a task based approach.

3.1 Evaluation Methods

3.1.1 Cognitive Walkthroughs

Cognitive walkthroughs involve setting up a scenario of a certain task a user might need to perform, and going (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 the reality of the interface through exploration[19]. This also takes note of the prerequisite and learned knowledge of the potential user. This method's effort and cost depend on the number of people evaluating the software and the type of expertise these people have.

3.1.2 User Testing

This is an empirical experiment of the software. 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. It was found in a study by Jeffries et al. [19] that user testing often found the most critical problems in 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.

3.1.3 Guidelines or Checklist

A set of requirements and usability standards are checked against 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 unexpected errors might not be picked up, diminishing this method's effectiveness.

3.1.4 Heuristic Evaluation

This involves inspection of the software by individuals with experience in usability or interface design. Studies in the paper by Jeffries et al. [19] found that heuristic evaluation not only found the most problems, it is also relatively low cost. The main disadvantage would be 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 [4] 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 according to Jeffries et al. the effectiveness of heuristic evaluation increases with the number of evaluators.

3.1.5 Surveys and Questionnaires

A set of questions (qualitative and quantitative) are answered by users about the software. 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 have often been used to complement other evaluation methods [16]. 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.

3.1.6 Models

Model based evaluations are particularly time consuming as a model needs to be constructed to mimic human behaviour, and then tested to ensure validity [25]. 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. Examples of model based evaluations include the GOMS (Goals, Operators, Methods and Selection) model which is based on human cognition abilities and the EPIC (Executive-Process/Interactive Control) system which models human perception and behaviour.

3.2 Discussion

In a recent paper[22], a systematic mapping review analysed the most popular forms of usability evaluations in papers since 2012. User Testing was the second most frequently used (14.14%), following Surveys/Questionnaires (26.26%). Despite it's apparent advantages, Heuristic Evaluations was the third most popular with a frequency of 12.63%. Finally cognitive walkthroughs and checklist verification had frequencies of less than 3%. Further findings from the review found that of software applications that underwent usability evaluations, only 2.03% came from the domain of expert systems.

This seems to support one of the conclusions of Chilana et al. [4], of finding usability practices and testing challenging in complex domain software development. Følstad compared the results of using domain experts for a cognitive walkthrough against the results of using usability experts. 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. Out of the usability methods described, heuristic evaluation and user testing have the highest rates of finding problems in the software. Thus, evaluation should use both methods, with a focus on user testing closer to the beginning of the project. This can be done with low fidelity prototypes as well, to ensure that the big conceptual notions and requirements are understood fully before the software cannot be changed without incurring great cost. Continual user evaluations with all three parties will be conducive to an end product which satisfies user expectations.

Despite the general recommendation for usability evaluation, there is some debate over the misuse of this evaluation. In a field such a scientific software development, there is a large scope for creating innovative and novel software to aid further knowledge discovery. A field like Astronomy Software development (and even Astronomy Visualisation software) is relatively young and tasks and outcomes are often unknown at the outset of developing software. Greenberg and Buxton [10] argue that usability evaluation may stifle projects in this area of innovative development as we are not aware how cultural adoption of this software might unfold.

4. ASTRONOMY SOFTWARE

Astronomy involves studying objects and processes which exist outside of Earth's atmosphere. In order to gather this data, large scale telescopes and arrays of telescopes (such as MeerKAT[2]) are used. The volumes of data collected are massive, and file can reach up to a petabyte (10^{15} bytes) in size. This prompted Hassan and Fluke [11] to coin the phrase Petascale Astronomy Era. This data is used for a range of purposes and processing such large files leads to a unique challenge when creating software for astronomy. The goals of this type of software vary from statistical analysis as computed by AstroStat [14], collaboration as would be facilitated by sharing of the data in CyberSKA [15] to visualizing the multidimensional data with tools and libraries such as Karma [8], VisIVO [1] and GIPSY [23].

For this range of software, 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 usable compared to an analysis tool processing a petabyte of data. However, there is a general trend of poor usability in astronomy software - especially with regards to user interfaces [5]. Astronomy software projects often under-estimate and

[2]A radio telescope in South Africa, part of a larger project called the Square Kilometer Array. Further information may be found at http://www.ska.ac.za/meerkat/

under-budget user interfaces resulting in numerous technical engineering panels instead of intuitive interfaces. This call for improving astronomy user interfaces was echoed by the Virtual Astronomy Observatory (VAO) Science Council's Recommendations (2010) [7] as well as added to with the suggestion of increased user support. It is clear that usability has long been an issue in software for astronomy, however, most measures to remedy this seem to rely largely on improved user interfaces.

4.1 Analytical Tools

The analytical tools CASA and AstroStat will be compared and their usability will be commented on. CASA (Common Astronomy Software Applications) [30] is a software package for analysis, calibration and imaging of astronomic data. It was developed by the National Radio Astronomy Observatory in conjunction with international scientists from the European Southern Observatory (ESO), the National Astronomical Observatory of Japan (NAOJ), the CSIRO Australia Telescope National Facility (CSIRO/ATNF), and the Netherlands Institute for Radio Astronomy (ASTRON), It offers a wealth of online support and user guides through its website and Wiki, however the package operates with a command-line input instead of a graphical user interface (GUI). Compared to GUIs, this adds a burden on the user to learn a new (Python based) language and would most likely increases the cognitive complexity of user tasks. However, this also allows for an increase in the range and power of the many available features and makes this software more robust.

In contrast, AstroStat [14] (which also provides analysis of astronomic data) makes use of a simple and intuitive graphical user interface - GUI - which will aid learnability and make use of the features more accessible to Astronomers who don't have as much experience with code. To rival the extensive user guides of CASA, AstroStat contains 'Help' and 'Example' buttons on the GUI, which is comparatively a far more user friendly option than pouring through a guide document and Wiki pages. AstroStat also categorises functionality based on experience (exploratory, advanced and expert) to provide some degree of tailoring. There is an apparent trade-off between an extensive range of features and usability between CASA and AstroStat.

4.2 Workflow and Collaboration Tools

CyberSKA is an online collaborative portal [15] which allows astronomers as domain experts and end users to share resources and input. The interface layout borrows from social networking sites and this is an advantage as it makes use of learned symbols to create an intuitive platform. There is a degree of clutter and some elements could have been hidden, however some of those features add to the usability of the software - such as a readily viewed 'Upcoming Events' tile. The integration of features to compress, extract, segment and visualize astronomy data sets caters to the specific tasks of the domain-experts making this tool more than just collaborative.

An Android application has been developed to notify users of new astrophysical events (such as discovering an extra-solar planet) [31]. This as-yet unnamed application developed by Zhao et al. makes use of a mobile platform to enhance usability due to its accessibility and ingrained usability features (e.g. multi-touch screen). The notifications are displayed in an organised and systematic interface which makes use of multimedia to enrich the content of with images and other relevant files.

Both of the tools described in this section are intrinsically more usable than the analytical tools mentioned above due to differing levels of complexity for the user tasks at hand.

4.3 Visualization Tools

Scientific visualization involves taking data with three or more dimensions and representing this visually in a way which is easier to inspect by eye [11]. This is done in order to reduce the complex cognitive load on the brain and enable problem solving and pattern finding [21]. In astronomy, three dimensional cubes of data are visualized, and this is a tool to aid exploration of the data in context of its surroundings in the hopes of finding unexpected knowledge[9].

Visualization lends itself to enhancing the usability of a software tool. Usability heuristics of the visualization (and not just the interface) contributes to overall ease of use and includes feedback, consistency and error checking with the added perceptual heuristics of colour, Gestalt Laws and aesthetics [26]. The following tools visualise astronomy data and will be critiqued based on usability.

4.3.1 Tool Analysis

GIPSY is an interactive system (developed by the Kapteyn Institute) which reduces and displays astronomy data. Ruiz et al. [23] stated that GIPSY needs work on the user interface as it was currently not transparent nor user friendly enough for non-expert users. The visualization tool GIPSY contains a UI named HERMES which has interactive and non-interactive versions as well as a GUI named Ggi [23]. From observation, the interface seems quite rudimentary, technical and contains multiple similar looking text fields making the interface overwhelming and difficult to use.

KARMA is toolkit for image and signal processing applications with a library containing tools for visualisation. The KARMA toolkit [8] uses a modular approach utilizing 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 any task in Astronomy. From observation, the interface seems quite outdated yet more intuitive than that of GIPSY.

VisIVO is a cross-platform multi-dimensional application for visualisation [1] and it's accessibility is enhanced by the fact that it is a cross-platform, an important consideration for usability [11]. VisIVO allows for integration, interactivity, navigation and collaboration across it's many platforms (mobile application, an easy to use web portal, desktop application). This level of 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 analysed astronomical software described above.

Both Iris [17] and SKIRT [3] 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

optimal. However, as stated in the article on SKIRT [3], 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 user friendly.

4.4 Discussion

The interfaces of the analysed software packages with greater functionality were often outdated and seem tedious to navigate (with many menus to traverse). The collaborative tools such as CyberSKA had more visually appealing interfaces which also seemed current and easy to use. As astronomy is such a complex domain, the functionality of the software cannot be sacrificed for a greater degree of usability. However, a user-centred approach to design might result in software which actually lowers cognitive complexity and increases knowledge discovery. A good user interface as well as a more intuitive flow of processes to complete tasks would create software which is not only appealing, but also more useful than the current standard in software for astronomy. It is also difficult to find description about how (if at all) these software projects underwent usability testing. If usability testing was omitted, this could be an approach to implement in order to improve the quality of use of the software.

Alternative methods other than standard mouse/keyboard interaction could also be explored in creating an intuitive interface for astronomy visualizations as was done by Mulumba et al. [20]. This recent project created prototypes to investigate gesture-based tracking and multi-touch interaction with the visualization software. The tasks included visual transformations, sub-volume extractions within the visualization and slicing of the data. It was found that with the gesture tracking, there were some issues with depth cues and position tracking and thus further work is required. Multi-touch interaction seemed promising. This would take advantage of learned pathways of users who currently use touch screens and multi-touch interfaces with similar actions (e.g. pinching for zooming). The paper did not elaborate further on the usability testing conducted, however this could be an avenue for further research.

5. CONCLUSIONS

This paper has explored the unique challenges and attitudes towards usability in highly specific and complex domains. In doing so, approaches to usability in creating scientific software and the development methods which result in the usability (or lack thereof) were analysed.

As a subset of scientific software, astronomy software is equally as specific and faces the same challenges in usability. An analysis of these astronomy tools uncovered trends of outdated and ineffective user interfaces, especially in the sub-field of astronomy visualizations. This was particularly surprising for a field which relies so heavily on visual perceptions. Other trends included tedious and repetitive actions in order to complete domain-specific tasks. It is clear that usability practices in the field of astronomy software development are under-utilized (if used at all) and there needs to be a shift in focus to producing more effective and useful software by incorporating some usability evaluations, as long as the evaluations are not restricting the development of this software.

Avenues for future work could involve investigating alternative interface approaches and designs for a more intuitive and effective user experience. These designs would need to reduce cognitive complexity and improve analytical deductions. There is also a need for creating methods and tools enabling software development in this field to be done with usability in mind. This can be done using evaluation techniques and encouraging user centred design rather than just functional competency. The user evaluation methods recommended would be a combination of user testing and heuristic evaluation in cycles throughout the development process. A combination of user domain knowledge and usability expertise is important for creating software satisfying the needs of users in science and astronomy.

6. REFERENCES

[1] 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.

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

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

[4] 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.

[5] 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.

[6] M. Costabile, D. Fogli, C. Letondal, P. Mussio, and A. Piccinno. Domain-expert users and their needs of software development1. *IST PROGRAMME*, page 6, 2003.

[7] G. Fabbiano and D. Calzetti. Recommendations of the vao-science council. 2010.

[8] R. Gooch. Karma: a visualization test-bed. In *Astronomical Data Analysis Software and Systems V*, volume 101, page 80, 1996.

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

[10] S. Greenberg and B. Buxton. Usability evaluation considered harmful (some of the time). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 111–120. ACM, 2008.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

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

[20] P. Mulumba, J. Gain, P. Marais, and P. Woudt. Scientific visualization of radio astronomy data using gesture interaction. In *Astronomical Data Analysis Software an Systems XXIV (ADASS XXIV)*, volume 495, page 145, 2015.

[21] 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.

[22] 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.

[23] 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.

[24] R. Sanders and D. Kelly. Dealing with risk in scientific software development. *IEEE software*, 25(4):21, 2008.

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

[26] 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.

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

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

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

[30] N. R. A. O. Website. National radio astronomy observatory - casa. Available at https://casa.nrao.edu/, version 1.6.0.

[31] Y. Zhao, I. Bond, and W. Sweatman. An android application for receiving notifications of astrophysical transient events. *Astronomy and computing*, 6:19–27, 2014.