A LEAN UX APPROACH FOR DEVELOPING EFFECTIVE MONITORING AND CONTROL USER INTERFACES: A CASE STUDY FOR THE SKA CSP.LMC SUBSYSTEM

V. Alberti*, INAF-OATs, Trieste, Italy
C. Baffa, E. Giani, G. Marotta, INAF-OA Arcetri, Firenze, Italy
M. Colciago, I. Novak, Cosylab Switzerland, Brugg, Switzerland
G. Brajnik, University of Udine and IDS, Udine, Italy

Abstract

The Central Signal Processor Local Monitor and Control (CSP.LMC) is a software component that allows the flow of information and commands between the Telescope Manager (TM) and the subsystems dedicated to signal processing, namely the correlator and beamformer, the pulsar search and the pulsar timing engines. It acts as an adapter by specialising the commands and associated data from the TM to the subsystems and by exposing the subsystems as a unified entity while monitoring their status. In this paper, we approach the problem of creating a User Interface (UI) for such a component. Through a series of short learning cycles, we want to explore different ways of looking at the system and build an initial set of UIs that can be refined to be used as engineering UIs in the first Array Assembly of the Square Kilometre Array. The process heavily involves some of the developers of the CSP.LMC in creating the dashboards, and other ones as participants in informal evaluations. In fact, the opportunities offered by Taranta, a tool to develop web UIs without needing web-development skills, make it possible to quickly realise a working dashboard that can be promptly tested. This also supports the short feedback cycle advocated by a Lean UX approach and maps well in a bi-weekly sprint cadence. In this paper, we will describe the method and present the results highlighting strengths and pain points where faced.

INTRODUCTION

The SKA Observatory (SKAO) occupies a prominent role in the framework of modern, ambitious scientific projects [1]. The two telescopes that comprise the observatory will provide the scientific community with powerful magnifying glasses to observe the universe at radio frequencies. Being a complex system, the SKA project adopted a staged programme to incrementally deliver the telescope and reduce the risk of not identifying issues promptly, therefore providing the scientific community with a sub-optimal instrument. The goal of the first stage, Array Assembly 0.5 (AA0.5), which is planned to last until the end of next year, is to demonstrate a working implementation of the architecture and supply chain as early as possible. The system being verified consists of 4 dishes, 6 stations, and all the necessary infrastructure positioned in the Karoo and Murchinson deserts, respectively. The telescope software delivery is coordinated by leveraging the Scaled Agile Framework (SAFe) [2], whose 3-month heartbeat[1] helps synchronise the work developed by about thirty teams. As systems grow in complexity, the need for clear, intuitive, and efficient user interfaces becomes of paramount importance, especially in sectors where precision and rapid responses are critical. In the context of growing agile methodologies, Lean UX has emerged as an approach that blends product and interface design processes into iterative cycles. It integrates feedback loops and constant iteration, ensuring that the UI evolves in tandem with user needs and system requirements [3]. This study presents an exploration of Lean UX principles applied to the CSP.LMC subsystem's Monitoring and Control UIs. Key to this implementation was the utilisation of Taranta [4], a tool that allows for creating web-based GUIs for the TANGO Control System [5] devices. The Lean UX method had to be tuned to take into consideration the specific context of the study: the final user developed the dashboards himself, thanks to the opportunity given by Taranta; the team involved isn't a UI team but rather the control software development team; the selected approach limited the initial scope of the dashboards, with plans to increase it later on. Through this paper, we aim to detail our journey, methodologies employed, and lessons learned. Our hope is that, by sharing our experiences, we can offer insights and facilitate the use of Lean UX principles and methods in the context of other complex control systems and teams developing them.

CSP.LMC

The Central Signal Processor (CSP) is a core component of the SKA software, responsible for processing data received from the antennas in order to enable further scientific analysis [6]. It comprises three primary instruments, hereafter referred to as "subsystems", that are devoted to specific data processing:

- The Correlator and Beam Former (CBF) processes raw antenna data to produce visibility[2].
- The Pulsar Search (PSS) identifies potential candidates for pulsar research.

* valentina.alberti@inaf.it

[1] SAFe bases its cadence on periodic events among which there are Program Increments (PIs): a PI is a time frame during with an Agile Release Trains plans, and progressively releases a working system. the length of a PI can vary between 8-12 weeks, but in the SKA project case it has been extended to 3 months.

[2] Visibilities are complex flux measurements in spatial frequency space

- The Pulsar Timing (PST) determines the period and irregularities of identified pulsar candidates.

On top of these three subsystems, another one, i.e. the Local Monitoring and Control (CSP.LMC) acts as the primary interface for the collective CSP apparatus, presenting it as a unified unit to clients (TM) hiding its complexity. Like all software components in the SKA project, it is developed using the TANGO Control Framework and comprises multiple "devices", each fulfilling a unique role. At AA0.5 the basic functionalities of only 2 of the three engines will be in place, namely the CBF and the PST. Within the SKA comprehensive software, the CSP.LMC receives commands from and conveys essential details for monitoring the CSP subsystems to the Telescope Manager. It also offers the necessary interfaces for subsystem configuration. The CSP.LMC's coding is undertaken by a dedicated Agile group known as the CREAM team, within the SKA software's SAFe development approach. Teams release increments of prioritised functionalities in two-week-long sprints during the course of a Program Increment.

TARANTA

Taranta is a web application that provides the user with the ability to create a graphical user interface to interact with Tango devices [4]. It comprises two main components. The frontend component is a React client that provides a generic device view, similar to the Jive desktop application [7], and a visual editor to create new dashboards and run existing ones. The backend implements the GraphQL API to Tango devices. Embracing the no-code paradigm [8], Taranta gives the end user the ability to create their own dashboards, significantly reducing the user requirements gathering phase and de facto merging the design and implementation steps typical of UI creation process. Provided with a set of widgets, the UI creator drags and drops them on the canvas and configures them with the correct Tango devices focusing on very specific use cases and workflows. If needed, the ability to customise the look and feel of a widget by means of Cascading Style Sheets (CSS) properties is also available when developing a dashboard. In addition to the dashboard development process, the Taranta development team owns the analysis of user needs in terms of new widgets and improvements in style, functionality, and performance. Once understood and prioritised, the team proceeds with the design and implementation of the essential building blocks of the UIs, called Taranta widgets.

PROCESS AND TECHNIQUES, A THEORETICAL INTRODUCTION

Initially proposed by Jeff Gothelf and Josh Seiden [3], Lean UX is a mindset and a set of techniques that aim at making the user experience a focal point of the development of a product. It creates a deeper shared understanding of the user needs by investigating the root causes that generate them. Having as part of its foundation the Agile software development and the Lean Start-Up Cycle, it promotes very fast learning cycles that allow for frequent validation and adaptation of proposed design solutions. This fits well with SAFe and the biweekly cadence of team iterations. The process starts with a number of prioritised assumptions and hypotheses that need to be validated through the collection of pieces of evidence [9].

To properly adopt the Lean UX approach we relied on a set of well-known techniques such as semi-structured interviews, user profiles, prototypes and user testing. We introduce all of them in the following.

- **Semi-structured Interviews** A semi-structured interview is a qualitative research method in which appropriately selected stakeholders are interviewed 1-to-1, following a general set of open questions. The interview script is not rigid and can be altered when the opportunity to further explore a theme or a response presents itself. The goal is to collect design-relevant information that can include the characterization of the users and the context in which they operate [10].
- **User Profiling**. Users are the beneficiaries of the implemented interface. To better provide them with a meaningful tool we have to be able to figure out what user-related aspects are relevant for the design. Their needs, expectations and behaviours in relation to the system are fundamental pieces of information and are well captured as role profiles which include a description of the context in which some tasks are performed, (workflow, physical environment, social situation, external sources, background, etc.), of characteristics of the performance of the role (frequency, intensity, duration, complexity, etc.) and design objectives that are important for the role (ease of learning, efficiency, reliability, accuracy, etc.) [11].
- **Prototyping**. Prototyping is the activity of creating artefacts supporting or suggesting a certain set of actions to be performed with the system in question. The intended audience is a crucial parameter in identifying the suited fidelity level and choosing the best prototyping method. Prototypes can be characterized in terms of fidelity with respect to the final system. In particular, McCurdy et al. [12] identified 5 dimensions along which fidelity can be defined: 1. Visual refinement, whose extremes are hand-drawn sketches vs pixel-accurate mock-ups; 2. Breadth of functionality, corresponding to the number of implemented features; 3. Depth of functionality, which is the level of implementation detail of each feature; 4. Richness of interactivity, defined in terms of the set of interactive elements captured and represented in the prototype; and 5. Richness of data model, related to how much the data manipulated by the prototype is representative of the actual domain. Taranta is a very effective tool that supports mixed-fidelity prototypes that can be built very quickly. For example, prototypes of dashboards with poor visual fidelity, but wide breadth, high depth, richness of data and of interactivity. This kind of mixed fidelity prototype are very effective for UIs of control systems.

- **Usability Testing**. Usability Testing is an empirical technique aimed at gathering data to identify usability deficiencies existing in a product. The sessions are carried out by a facilitator using the "think aloud" protocol and observing the participants while they perform some tasks via the UI. The participant is asked to describe what they are doing and what outcome they expect when certain actions are performed as well as verbalise their doubts or feelings while using the UI. The facilitator assures the quality of the data by guiding the participants without accidentally influencing their behaviour. The choice of the tasks is critical and reflects activities that would be realistically performed by the user in real life. Based on the observed behaviour the facilitator identifies the strengths and weaknesses of the UI under test and then translates them into recommendations for the next iteration. Usability testing is an extremely powerful and cheap technique to revise and improve the utility and usability of a specific design [13].

APPLICATION OF THE PROCESS TO UIs OF THE CSP.LMC

The goal of this study was to incrementally create and quickly validate a set of UIs that can support the monitoring and control of the CSP.LMC in AA0.5 through the adoption of a Lean UX approach tweaked to our specific needs when necessary. In particular, we had to take into account that: the user is also the dashboard designer; the teams' work is planned at PI boundaries which makes it possible to squeeze in the sprints only small changes; and we targeted a specific time frame in the near future to contain the scope of work and create dashboards that can be relatively stable for a certain amount of time. The Lean UX method is characterised by very fast learning cycles that allow for frequent validation and adaptation of proposed design solutions. The process we followed is articulated in stages:

- **Discovery phase**: a conceptual step to identify the target user roles and implementation time horizon. This allowed us to analyse the most relevant use cases and to derive the dashboards' conceptual model;
- **Implement-Revise-Adjust phase**: a step that included the realization of a set of dashboards and a usability testing session followed by a debriefing meeting and by the adaptation of the UI as per the received feedback. When it wasn't possible to immediately implement changes, they were added as UX technical dept stories to either the CSP.LMC or the Taranta backlogs.

The study has been possible thanks to the involvement of the following participants: one person with experience in UX to guide the process and some design choices, a CSP.LMC developer who also acted as the main dashboard designer and creator, and two CSP.LMC developers and the team's Product Owner involved as participants in the usability testing sessions. Moreover, an expert in usability testing conducted the design validation step. All the sessions were recorded to allow for future reviews and notes were taken during the probing sessions to capture the interviewer's observations, the user's opinions and feelings and the user's suggestions. These sessions lasted one hour each and were followed by a debriefing session during which the testing expert, the UI developer and the UX person discussed what was learnt and identified immediate and future improvements. Paper notes and rough sketches have been a useful support during the entire process. Taranta was the only tool used to develop the dashboards.

Discovery Phase Methodology and Results

The goal of this phase was to reach a consensus between the UI expert and the dashboard developer on the scope of work and the process to be followed to reach the result. In our specific situation, the dashboard creator was also a representative of the target user and played both roles in the process. Following the semi-structured interview technique it has been possible to characterise the target user role, the time frame and the main use cases that the dashboard had to support. The target user role has been characterised following Constantine and Lockwood's checklist [11]. It represents a system expert such as CSP.LMC developers and testers. Details are provided in the Appendix.

The identified focal use cases are: to be aware of the current status of the CSP.LMC, its subsystems and the associated subarrays; to be able to interact (command and monitor) with the subarrays independently to perform all the happy path commands (from switching on the subsystems to execute a scan); to receive the monitoring information relevant for verifying that the system transitioned to the expected states; to access lower level information on subsystems and subarrays; to be able to recover from faulty conditions; to run sequences of commands; to be able to debug failures; to be able to quickly create short-lived custom dashboards for diagnosing specific problems.

Starting from this material, we created a set of hypotheses that had to be probed into during the usability testing sessions. They included the following:

1. The layout should be optimised for a relatively large monitor and the use of a mouse and keyboard as this is often the workstation layout for developers.
2. Given the level of expertise of the target user and the specificity of the use cases, the structure of the dashboards, naming conventions and mental model should not diverge greatly from the underlying system. This is normally not a good practice but it is reasonable in the case of low-level engineering UIs used to support the development and debugging of the code implementing the specific devices being monitored. In fact, adding an abstraction level that isn't needed may have the opposite effect.
3. The top-level dashboard should act as an information entry point and allow for assessing the overall state of the CSP.LMC, its subsystems and subarrays at a glance. It should also support the drill-down capability by providing a means to reach other dashboards.

4. Commands should always be available and grouped in a panel separate from different dashboards. The position of the commands panel shouldn't be fixed to allow for more flexibility in supporting user preferences. Moreover, if the control panel is placed on the side of the dominant hand, the interaction time can be reduced by minimising the length of the cursor movement. Direct interactions with the UI will happen through this panel most of the time.
5. Tables are the best compromise between clarity and effectiveness when a finite number of state attributes needs to be shown and compared for each device. Although tables may become cumbersome when the number of devices is very big, they allow the user to effectively compare values across different devices and identify outliers when the number of rows is limited. This is the case for the current and short-term implementation of CSP.LMC. In the long run, CSP.LMC will comprise many more devices and a different design approach should be followed.
6. The timing at which key devices perform a certain state transition is critical for decision-making during the supervision of a system and should be part of the top-level dashboard.
7. Lower-level dashboards should provide all the information needed when debugging CSP.LMC. We don't aim at creating detailed dashboards for the subsystems because they'll be created by the responsible teams for AA0.5.
8. Taranta supports the fast implementation and revision of dashboards.

During a preparatory session focused on testing, the validation criteria to be satisfied in order to consider the assumptions solid have been fleshed out. The task assigned to the developers during the probing session was to perform some exploratory testing [14] with the aim of discovering potential flaws in the CSP.LMC code. In the case of the Product Owner (PO), the session would focus on the ability and easiness to drive the system under test to complete all the steps required to perform a scan and interpret the results. The difference in the proposed tasks reflects the difference in the two roles. Although assuming a detailed knowledge of the CSP.LMC is still appropriate, the goal of the PO while using the UIs may differ from those of the developers. Rather than debugging the software the PO is likely more involved in demoing and validating the subsystem behaviour. Moreover, the PO has generally a less detailed knowledge of implementation aspects but a more accurate systemic view of future developments. In this sense, we are increasing a bit the scope of work to include a slightly different role and we expect the UI to be missing some functionalities.

It has to be noticed that going into the details of the various Taranta widgets and of the Tango devices that had to be configured wasn't necessary at this stage, given the high expertise of the designer in both CSP.LMC and Taranta.

Implement-Revise-Adjust Phase

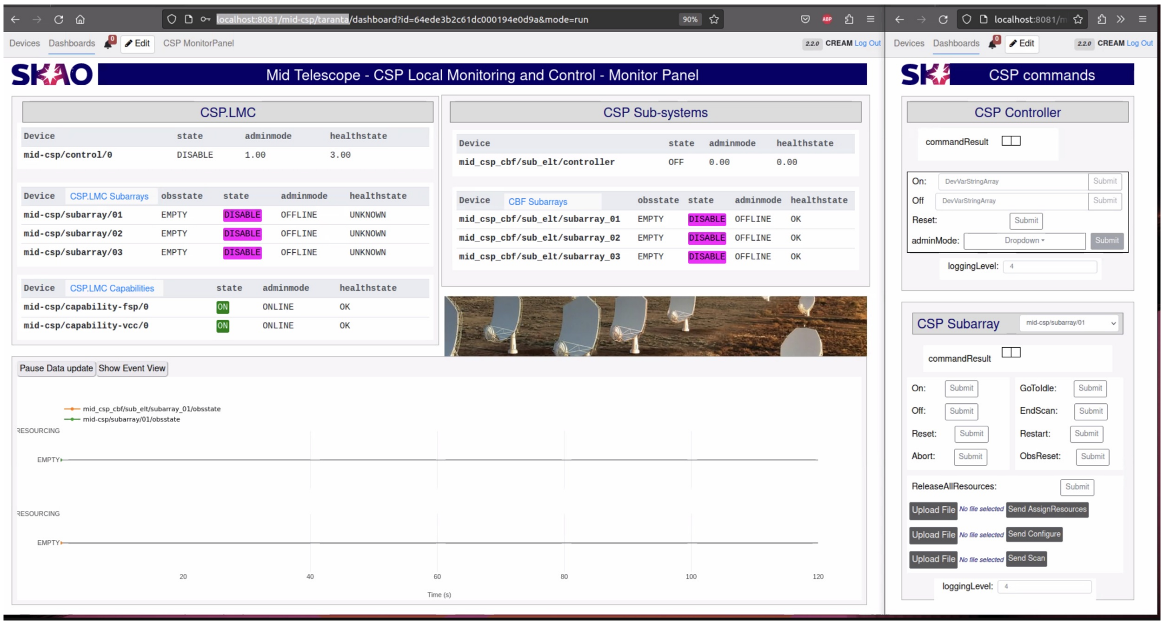
After conceptualising the UIs we followed a typical agile method: first of all we created a working prototype that consisted of one control panel, one high-level monitoring panel and 2 lower-level interfaces to capture detailed views of subarrays and of CSP resources and capabilities. Secondly, we conducted the usability tests to validate our assumptions and third, we implemented the changes. The 2 sessions with the developers used the original design while for the session with the PO small improvements have been made to the dashboard. Regarding the original hypotheses, we could derive the following conclusions:

Hypotheses 1, 2 and 3 are overall verified for the developers. In all cases, the provided information was clear and sufficient to test the CSP.LMC and the interaction with the subsystems. Less technical roles or new developers may find it beneficial to be presented with links to the documentation or diagrams that guide them to perform the correct sequence of actions and help to correctly interpret the combination of state values. Moreover, important quantities to be monitored in the future have been suggested. An example is data packets being sent/received along with the packet rate and the number of dropped packets. A possible effective visualization includes a time-based chart where peaks, lows, and sloping trends could be quickly spotted and associated with possible problems. Finally, the assumption on the screen real estate setting may not be always valid for non-development roles that could sometimes use their laptop. A different layout should be considered to better support this scenario.

Hypothesis 4: in those cases when the user was interacting with the dashboard directly, they naturally positioned the control panel on the side of their dominant hand, supporting our initial assumption. The separation of the control panel from the other UIs was received well by the developers. More quantitative data should be gathered to measure the effective reduction in the interaction time. Following the received feedback the sequence of commands has been adjusted and a few text fields have been added to better support less experienced users. Adding tooltips to Taranta is felt as a good way to ease the understanding of the UI by displaying informative text.

Hypothesis 5, 6 and 7 have been verified. There have been refresh issues in one session and the data weren't updated in real time. This greatly diminished the usability of the UI causing some frustration especially when using the timeline widget to visualise the timing of state transitions. A ticket was created to investigate the problem. As the system is growing additional information will have to be added to the detailed dashboards in the medium term but the current content is sufficient for AA0.5.

Hypothesis 8 Taranta measured up to the expectations in terms of easiness of use and modifiability of the UI. Challenges can be faced in case of missing widgets or functionalities as their implementation needs to be prioritised at a higher level in the 3-month cycle.



Device	state	adminmode	healthstate
mid-csp/control/0	DISABLE	1.00	3.00

Device	CSP.LMC Subarrays	obsstate	state	adminmode	healthstate
mid-csp/subarray/01		EMPTY	DISABLE	OFFLINE	UNKNOWN
mid-csp/subarray/02		EMPTY	DISABLE	OFFLINE	UNKNOWN
mid-csp/subarray/03		EMPTY	DISABLE	OFFLINE	UNKNOWN

Device	CSP.LMC Capabilities	state	adminmode	healthstate
mid-csp/capability-fsp/0		ON	ONLINE	OK
mid-csp/capability-vcc/0		ON	ONLINE	OK

Device	state	adminmode	healthstate
mid_csp_cbf/sub_elt/controller	OFF	0.00	0.00

Device	CBF Subarrays	obsstate	state	adminmode	healthstate
mid_csp_cbf/sub_elt/subarray_01		EMPTY	DISABLE	OFFLINE	OK
mid_csp_cbf/sub_elt/subarray_02		EMPTY	DISABLE	OFFLINE	OK
mid_csp_cbf/sub_elt/subarray_03		EMPTY	DISABLE	OFFLINE	OK

Figure 1: First implementation of CSP.LMC monitoring dashboard. The command panel is visualised on the right.

Additionally, the usability testing sessions allowed us to uncover some pain points of the design. They spanned all areas from visualization enhancement and coherence of the information to code implementation improvements. For example, using multiple tables showing a different set of states caused columns to be misaligned which reduced their readability. The problem was solved by rearranging the devices displayed in different tables as well as the position of the tables themselves. The visualization gained in simplicity. A second observation pointed out the presence of inconsistencies: in navigating between dashboards, it wasn't possible to reach the main monitoring boards from the lower-level ones; in the naming conventions, mixed use of uppercase and lowercase for the same family of attributes, different ways of indicating missing values (null, no data etc), in the implementation of the same command for different devices. The missing links have been added and CSP.LMC code improvement items were identified and included in the backlog. Similarly, bugs as well as suggested improvement tickets, such as increased logging functionalities, were created in Taranta backlog when necessary. Overall, the current design leaves the screens real estate relatively empty. This is because we didn't try to optimise the space but rather to organise the information to support the major tasks. An alternative design could be considered that increases the information density in each dashboard at the probable cost of breaking consistency in the current drill-down approach.

Figure 1 shows the first implementation of the top-level monitoring dashboard. It shows the relevant states of the CSP.LMC devices, namely the CSP controller, the subarrays and capabilities, and of one of the subsystems, the CBF, and its subarrays. Since no commands have been sent and the system is in its initial state, the state transition plot at the bottom is empty. The command panel is visible on the right side of the image. It encompasses two distinct areas dedicated to the controller and to the subarray, respectively. (TBC)

CONCLUSIONS

In this paper, we discussed our approach to include Lean UX techniques in the development of CSP.LMC dashboards for the first SKA release, AA0.5. The CSP system to be represented will comprise of the CBF and PST in its initial shape. The CSP.LMC element will act as a potential entry point in debugging problems in the lower-level signal chain (excluding TM). In order to support this activity, we decided to create dedicated dashboards that allow a comprehensive view on the state of the CSP devices. Thanks to the adoption of some Lean UX techniques it was possible to detail the characteristics of a specialist interacting with the system under consideration and the main use cases that the dashboards should support. Expanding the discussion to the PO, we enriched the initial UI with some explanatory tips and identified the way forward to support additional roles at AA0.5. We believe that the current set of dashboards could be employed in a usability testing session with Array Integration and Verification personnel to verify the need for additional use cases or different ways of looking at the system.

Although the use of Taranta greatly simplified the design step, we will find the discovery phase beneficial because it fosters a deeper understanding of the boundaries within which the user operates along with goals and motivations. The usability testing sessions proved to be a effective tool to uncover gaps, improvements, suggestions regarding both the implementation of the dashboards and their design, the available widgets and the implementation of the underlying system. All in all, we believe that the adopted approach is sustainable for a team whose primary focus isn't the development of UIs but that can greatly benefit from their use during the everyday work.

APPENDIX

User role: CSP.LMC developer (or any subsystem expert such as code maintainers or testers)

Responsibility: to develop, test, validate and debug CSP.LMC code.

Context (within which the role is played):

- Overall job, workflow or activity within which the role is played: CSP.LMC developers are in charge of implementing and testing new functionalities and troubleshooting problems related to the subsystem.
- Physical environment: their own office with their own workstation. Often, this includes a laptop and a second screen.
- Social situation: mostly alone.
- Relationship with indirect users in role: direct interaction with other developers both in the team and belonging to other SKA teams in case of troubleshooting.
- External sources of information: the screen of their PC.
- Background in terms of training, education or experience: experienced SW developers, graduated, PHD is not strictly required. Proficient in problem-solving.
- System knowledge is expected or required: basic understanding of SKA project and its components is required. Good understanding of CSP.LMC scope and functionalities.
- Domain knowledge expected or required: software development. Telescopes is a bonus.
- Distribution of user skills in terms of novice, intermediate or expert usage patterns: moderate to expert.
- Required or discretionary nature of role: regular job.

Characteristics (of performance of role):

- Orientation, attitude or emotional state: CSP.LMC developers are responsible of the quality of the code they develop.
- Frequency with which role is played: normal working hours.
- Regularity with which the role is played: the role is played regularly every day.
- Intensity of interaction in the role: increased activity when the troubleshooting is particularly difficult or when testing new functionalities.
- Duration of the interaction: varies with the development stage.
- Complexity of the interaction: the complexity of the interaction may vary with the task performed. For example, trying to find to what extent the CSP.LMC behaviour leads to an overall bug that causes a failure while running system/integration tests can increase complexity and stress levels.
- Predictability of interaction in the role: some tasks are repetitive, for example, some verification or testing procedures. In other cases, the developer autonomously decides the best strategy to follow to test the system.
- Volume and complexity of information handled in the role: relatively small amount of information (generally the subsystem). Complexity is mainly due to the analysis of the root causes of problems.
- Direction of information flow to or from the system: the information flows from the system to the developer when collecting information. The flow is in the opposite direction when performing tests or driving and querying the system during testing.

Usability requirements:

- Ease of learning and memory retention: nothing specific.
- Adaptability: An adaptable UI can be designed that provides new ways to explore related failures and helps the CSp.LMC developers to quickly relate information. It is likely that a degree of customization would be very welcome (at least in terms of colour themes, font size, toolbars, docking panes but also in defining colour-coded labels or tags).
- Fault tolerance/protection: Appropriate confirmation of actions may be needed to avoid making slips. Less experienced users may benefit from some mechanism that prevents them from sending sequences of commands that may lead to undesired states.
- Accuracy: the accuracy of the information that is displayed is very important because it could impact the efficiency with which the CSP.LMC developer understands the cause of a problem.
- Completeness: the CSP.LMC developer has to be able to collect all the information he needs to perform a detailed analysis of the status of a subsystem.
- Efficiency of the user: irritation and frustration may arise if the interaction with the UI is too slow or cumbersome.
- Controllability: CSP.LMC developers need to be able to drive the system in any desired state especially when performing exploratory testing or debugging activities.
- Reliability of the system: developers need to trust the system and therefore the UI should present only relevant information, in a very clear, unambiguous and complete way, with the option to get more relevant details if needed. Moreover, developers should be made aware if the system is busy processing or stuck in a process. This would increase awareness and allow the user to take corrective actions to for example abort the process.
- Attention: needs to be managed while performing some tasks such as discover when a certain thing started to deviate from the nominal behaviour.

REFERENCES

[1] SKA Observatory, `https://www.skao.int/en`

[2] Scaled Agile, Inc., SAFe for Lean Enterprises 6.0, `https://www.scaledagileframework.com/`

[3] J. Gothelf and J. Seiden, *Lean UX: Designing Great Products with Agile Teams*. O'Reilly Media, Inc., 2016.

[4] Taranta, https://taranta.readthedocs.io/en/latest/

[5] Tango Controls, https://www.tango-controls.org/

[6] G. Marotta, E. Giani, I. Novak, A. Söderqvist, and C. Baffa, "Software design for CSP.LMC in SKA", in *SPIE Astron. Telesc. Instrum.*, Montréal, Québec, Canada, 2022. doi:10.1117/12.2630140

[7] Jive, https://shorturl.at/ewAX5

[8] M. Eguiraun *et al.*, "Taranta, the No-Code Web Dashboard in Production", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 1017–1022. doi:10.18429/JACoW-ICALEPCS2021-FRAR01

[9] Interaction Design Foundation, https://www.interaction-design.org/

[10] K. Holzblatt, J.B. Wendell, and S. Wood, *Rapid Contextual Design*. Morgan Kaufmann, 2005. doi:10.1016/B978-0-12-354051-5.X5000-9

[11] L. Constantine and L. Lockwood, *Software for use: a practical guide to the models and methods of usage-centered design*. Addison-Wesley, 1999.

[12] M. McCurdy, C. Connors, G. Pyrzak, B. Kanefsky, and A. Vera., "Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success", in *CHI 2006*, New York, NY, 2006, pp. 1233–1242. doi:10.1145/1124772.1124959

[13] J. Rubin and D. Chisnell, *Handbook of Usability Testing*, Wiley, second edition, 2008.

[14] E. Hendrickson, *Explore it! Reduce risk and increase confidence with exploratory testing*. The Pragmatic Programmers, 2013.