Interaction design challenges and solutions for ALMA operations monitoring and control

Emmanuel Pietriga, Pierre Cubaud, Joseph Schwarz, Romain Primet, Marcus Schilling, Denis Barkats, Emilio Barrios, Baltasar Vila Vilaro

Interaction Design Challenges and Solutions for ALMA Operations Monitoring and Control

Emmanuel Pietriga[a,b], Pierre Cubaud[c], Joseph Schwarz[d], Romain Primet[a], Marcus Schilling[d], Denis Barkats[e], Emilio Barrios[e] and Baltasar Vila Vilaro[e]

[a]INRIA, Orsay, France;
[b]INRIA Chile - CIRIC, Santiago, Chile;
[c]CNAM, Paris, France;
[d]ESO, Garching, Germany;
[e]Joint ALMA Office, Santiago, Chile

ABSTRACT

The ALMA radio-telescope, currently under construction in northern Chile, is a very advanced instrument that presents numerous challenges. From a software perspective, one critical issue is the design of graphical user interfaces for operations monitoring and control that scale to the complexity of the system and to the massive amounts of data users are faced with. Early experience operating the telescope with only a few antennas has shown that conventional user interface technologies are not adequate in this context. They consume too much screen real-estate, require many unnecessary interactions to access relevant information, and fail to provide operators and astronomers with a clear mental map of the instrument. They increase extraneous cognitive load, impeding tasks that call for quick diagnosis and action.

To address this challenge, the ALMA software division adopted a user-centered design approach. For the last two years, astronomers, operators, software engineers and human-computer interaction researchers have been involved in participatory design workshops, with the aim of designing better user interfaces based on state-of-the-art visualization techniques. This paper describes the process that led to the development of those interface components and to a proposal for the science and operations console setup: brainstorming sessions, rapid prototyping, joint implementation work involving software engineers and human-computer interaction researchers, feedback collection from a broader range of users, further iterations and testing.

Keywords: ALMA, Radio-telescope, Operations Monitoring and Control, Human-Computer Interaction, Information Visualization, Control Room

1. INTRODUCTION

The ALMA radio-telescope will be the largest millimeter/submillimeter radio astronomy observatory in the world, at an altitude of 5,000 meters in the high Atacama desert of Chile. With spacial and spectral resolution two orders of magnitude better than that of existing facilities, ALMA will give astronomers unprecedented views of star and planet formation in our galaxy and of the early Universe.

ALMA is a very advanced instrument that presents many technological challenges. From a software perspective, one critical issue to address is the design of graphical user interfaces for operations monitoring and control that scale to the complexity of the system and to the massive amounts of data operators and astronomers will be faced with. Early experience operating the telescope with only a few antennas has shown that conventional user interface technologies based on the WIMP model (Windows, Icons, Menus, Pointer)[1] are not adequate in this

Further author information: (Send correspondence to Emmanuel Pietriga)
Emmanuel Pietriga: E-mail: emmanuel.pietriga@inria.fr, Pierre Cubaud: E-mail: pierre-henri.cubaud@cnam.fr, Joseph Schwarz: E-mail: jschwarz@eso.org, Romain Primet: E-mail: romain.primet@inria.fr, Marcus Schilling: E-mail: mschilli@eso.org, Denis Barkats: E-mail: dbarkats@alma.cl, Emilio Barrios: E-mail: ebarrios@alma.cl, Baltasar Vila Vilaro: E-mail: bvilavil@alma.cl

context. They consume too much screen real-estate, require many unnecessary interactions to access relevant information, and fail to provide users with a clear mental map of the instrument. They increase extraneous cognitive load, impeding tasks that call for quick diagnosis and action. Operators and astronomers lose valuable time rearranging windows and searching for the relevant tab, when they could instead be troubleshooting the system and addressing potentially critical situations that may have a significant impact on the observatory's safety and overall performance in terms of actual observing time. This is even more true given that the instrument's control room is separated from the antennas as well as other hardware including the correlator by approximately 40 kilometers.

Advances in the field of Human-Computer Interaction (HCI) research can help address the critical issue of safe and efficient operations. Generally speaking, the goal of HCI research can be stated as trying to make computers easier to use while augmenting users' capabilities, enabling them to deal with more complex problems, larger datasets, as efficiently as possible, in single-user or cooperative work contexts. A more formal description is that HCI is about designing systems that lower the barrier between the human's cognitive model of what he wants to accomplish and the computer's understanding of the user's task. HCI is concerned with the design, implementation and evaluation of computing systems that humans interact with. It is a highly multidisciplinary field of research, involving experts in computer science, cognitive psychology, design, engineering, ethnography, human factors and sociology. In the more specific context set here, HCI research relates to the design and development of interactive visualization components that will help operators and astronomers better comprehend and manipulate the massive amounts of data they will be faced with when operating the telescope and performing observations.

Novel interface paradigms that go beyond the WIMP model*, coupled with advanced information visualization techniques, will benefit operators and astronomers by making virtual navigation in the system more fluid, by providing them with more relevant and more scalable representations of the system's status, and by enabling them to quickly relate different views of the system's various components. This includes: multi-scale interfaces leveraging spatial memory, dynamic queries, coordinated and multiple views, adjacency matrices for the representation of dense baseline networks, treemaps for the representation of hierarchical hardware and software structures, and advanced time-series visualizations of monitoring points.

This paper describes the iterative user-centered process that led to the design and development of those interface components, and to a proposal for the science and operations workstation hardware setup: brainstorming sessions, rapid prototyping, joint implementation work involving software engineers and human-computer interaction researchers, feedback collection from a broader range of users, further iterations and testing.

2. METHOD

The WIMP model that the Operations Monitoring and Control software (OMC)'s graphical user interface originally relied upon, was unlikely to be suited to the complexity of this very advanced instrument. It was sufficient to implement the relatively basic interfaces of, e.g., the Very Large Array in New Mexico, but such interfaces would not scale to the new instrument. Realizing this, ALMA's software division sought the expertise of human-computer interaction researchers starting in 2009. First for consultancy, and then in the framework of an official collaboration that involved researchers and software engineers in human-computer interaction on one side, and software engineers, astronomers and operators from ALMA on the other side.

As this initiative started relatively late in the ALMA software development project, discarding the existing OMC user interface framework in favor of a fully post-WIMP solution was not an option. A lot of effort had already been devoted to user interfaces by different teams, and the OMC was already in heavy use for development, testing and commissioning. Fortunately, the OMC's architecture based on the concept of independent, pluggable software components, enables us to add novel features to the OMC in an incremental, highly flexible manner, without disturbing day-to-day operations.

Our method is based on a user-centered design approach. It involves:

*Often termed post-WIMP interfaces.[2]

- *in-situ* interviews with operators and astronomers, who are the domain-expert end-users of the OMC software;
- participatory design workshops organized by the HCI researchers, involving a small group of end-users (operators, astronomers, etc.) and software engineers strongly committed to the project;
- rapid software prototyping of ideas resulting from these workshops;
- demonstration of these prototypes to a larger group of end-users, gathering feedback and new ideas;
- high-quality implementation of the ideas identified through this process as software components shipped as OMC plugins, integrated in the general development lifecycle.

Each category of actors plays an essential role. Operators and astronomers have significant domain expertise, they are well aware of the context-of-work, the associated requirements, and the needs of end-users. They contribute ideas, and can quickly provide feedback about the relevance of different options. HCI researchers are best aware of the state-of-the-art, and can suggest possible solutions to a given problem that other members of the group who are less familiar with the latest research and development advances in the field might not be aware of. Finally, ALMA engineers also participate in the brainstorming, suggesting solutions and later in the process providing feedback about the feasibility of the considered options. The participatory design workshops are organized in sessions that focus on well-identified topics. They are necessarily limited to a small number of people, typically 6 to 8, and usually last 1 to 1.5 days, so as to ensure that participants are highly committed and actively contribute to the discussions and idea generation process.

So far, the workshops have been held every 6 months, on site. Work is typically organized as follows to optimize time spent on site, including face-to-face meetings and cooperative software development[†], and minimize travel costs given that many participants, end-users excluded, live in the northern hemisphere and are not all collocated on the same site (Paris, France and Garching, Germany for the most part):

- right after arriving on site, participatory design sessions for one day or one day and a half (brainstorming, low-fidelity prototyping, feasibility assessment);
- intensive prototyping of the ideas selected from the sessions for 3-to-5 days (includes short iterations and clarifications with end-users who have resumed their usual activities after the design sessions);
- demonstration of prototyped ideas to a larger group of end-users, gathering feedback;
- debriefing with the smaller group of workshop participants, identification of next steps, task assignments;
- cooperative development over the next 6 months of high-quality implementations (for production use) of the ideas originally prototyped on site, at ESO headquarters in Garching and at INRIA in Paris (with some travel between the two sites) using virtual machines simulating the ALMA environment;
- back on site, further iterations, testing, debugging.

This ongoing HCI initiative[3,4] has led so far to the design of multiple advanced visualization components for operations monitoring and control, to the development of coordination mechanisms between components of the OMC, and to a proposal for the hardware setup of the science and operations workstations where operators and astronomers on duty will sit and work cooperatively. The following sections give an overview of these various developments, as well as the ones anticipated for the near future.

[†]ALMA software engineers and HCI researchers have very different but complementary expertise; collocated coding sessions have proven to be extremely productive.

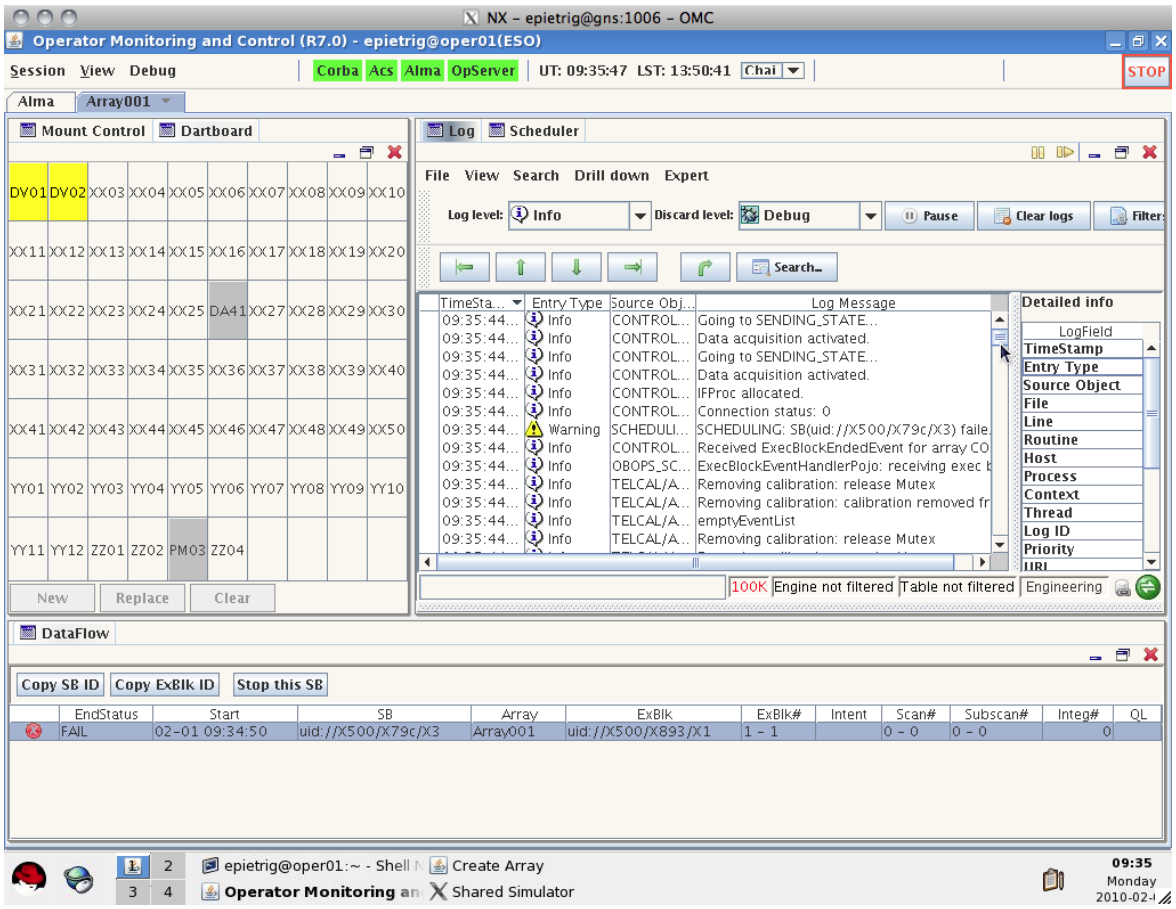


Figure 1. OMC UI components representative of the original WIMP-based interface. The original antenna selector (so-called *chessboard*) is shown on the left-hand side.

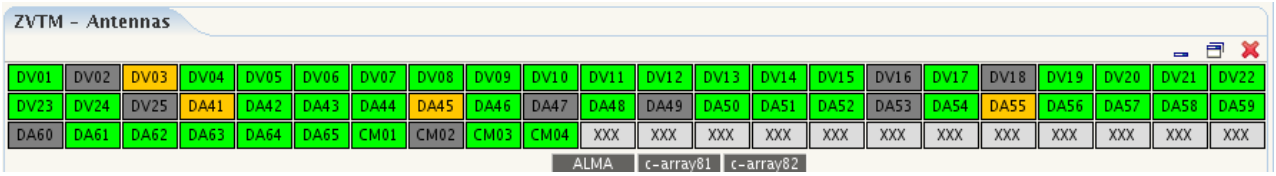
3. NOVEL VISUALIZATION COMPONENTS AND INTERACTIVE FEATURES

The original OMC user interface was relying on a very conventional WIMP design, illustrated in Figure 1, which provided users with a relatively simple and unified interface for navigating in the system, but that failed to provide users with a clear mental map of the instrument. Moreover, it required a lot of window management operations and much mouse pointing and clicking to access a given piece of information. While such low-level actions are not very cognitively demanding, they impede navigation and divert users from their primary tasks: monitoring the system, evaluating the quality of observations underway, identifying and possibly anticipating potential problems, troubleshooting them, coordinating the actions of the different actors in the control room at 2900 meters altitude and on the high site at 5000 meters.

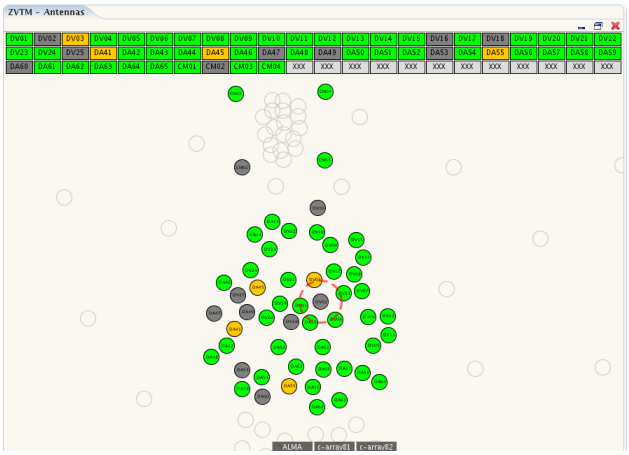
Our first initiative consisted in investigating post-WIMP alternatives to this interface paradigm that would make navigation in the system more fluid and would actually support users in their tasks rather than get in their way. We also designed advanced visualization plugins that were not originally planned as part of the OMC, but that were identified as useful additions during the participatory design workshops. The following sections describe the proposed alternative navigation paradigm, and the main visualization components developed for the OMC.

3.1 Antennas

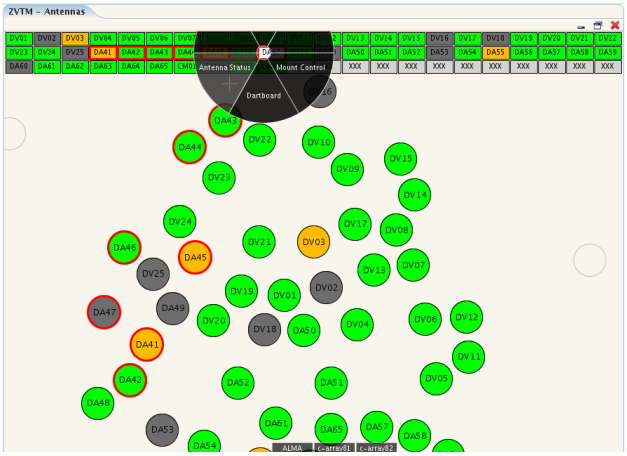
As mentioned earlier, the OMC's original user interface design and implementation was well-advanced when this project started, and end-users were already very familiar with it. At this point in the project, we did not want



(a) Compact version showing only the name and status of antennas.



(b) Expanded version also showing a zoomable map of antennas. When hovering a chessboard cell (here DV02), the corresponding antenna is highlighted with a red dashed circle on the map to help users locate the antenna.



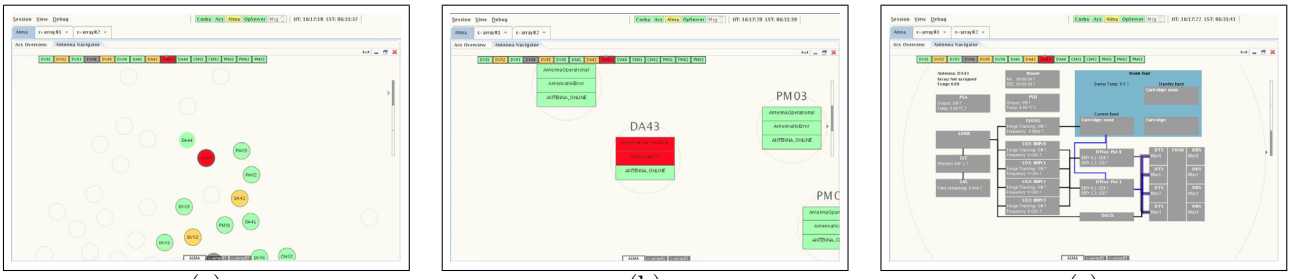
(c) Contextual pie menu invoked after antennas DA41 through DA47 have been selected in the chessboard (note that the antennas also get automatically selected on the map). Using this pie menu, users can invoke any of three plugins that provide more detail about devices that make up the antenna.

Figure 2. Enhanced chessboard.

to force radical changes on users. Entirely discarding the original interface paradigm was neither desirable nor practical; we thus had to find a way to make both paradigms coexist.

The OMC provides a general framework in which independent components are plugged-in and accessed through a relatively unified interface. For instance, most antenna-centric plugins are accessed by first selecting the corresponding item in the top-level *View* menu, choosing the corresponding antenna in a grid of antennas called the *chessboard* (left-hand side of Figure 1) that pops-up in a new window or tab. This eventually creates yet another window instantiating the requested plugin for this specific antenna. This can be a view showing pointing information, a view showing a block diagram of the main antenna hardware components, a view showing mount control, etc. For instance, double-clicking Antenna DV02 in the chessboard of Figure 1 would pop-up a new tab containing the Dartboard plugin for that antenna. While this is simple to understand and provides a unified mechanism to access many elements of the user interface, it is also cumbersome to operate, requiring many user actions to access a particular piece of information, as each individual plugin features its own chessboard.

We first worked on an enhanced version of this chessboard. We wanted this new chessboard to be a single, persistently-displayed entry point to all antenna-centric plugins, from which most, if not all, antenna plugins could be readily invoked for a given antenna or group of antennas. The compact version of this chessboard, illustrated in Figure 2-a, provides functionalities similar to the original chessboard. However, the order of operations is reversed: whereas in the original chessboard users first selected the type of plugin to instantiate and then specified what antenna to show, the enhanced chessboard requires users to first select antennas and then select what plugin to invoke for those antennas. One notable difference with respect to the old chessboard is that multiple antennas (cells) can be selected at once before invoking a given plugin, resulting in a corresponding



(a) (b) (c)

Figure 3. Semantic Zooming on Antenna DA43. As the user zooms-in, additional details are revealed.

number of tabs to be instantiated, one for each antenna, all at once. The selection settings are not lost, letting users invoke multiple plugins for the same set of antennas with a single point-and-click action using a contextual pie menu (as detailed later).

The main enhancement, however, is that this enhanced chessboard's window can be resized, revealing a zoomable geographical map of the antennas as illustrated in Figure 2-b. This map lets users navigate in the system using a zoomable user interface.[5] Zoomable User Interfaces (ZUI) are based on the concept of a large 2D canvas that contains arbitrary graphical objects, including vector graphics, images, control widgets, text and documents. They are often coupled with *semantic zooming*[6] capabilities: the visual representation of an object depends on the amount of pixels available to display it. While pure geometric zooming will simply magnify an object, semantic zooming changes its appearance, typically showing additional information as the screen real-estate assigned to it increases. This is illustrated in Figure 3, where the user smoothly zooms-in on Antenna DA43: at low magnification (Figure 3-a), the antenna is represented as a simple circle labeled with the antenna's name. The circle's color gives high-level information about its status. Zooming-in (Figure 3-b), the circle gets replaced by a stack of rectangles giving more detailed information about the antenna's status. Further zooming-in (Figure 3-c), the rectangles get replaced by a more elaborate block-diagram representing the main hardware components of the antenna.

This interface paradigm has several advantages: it enables more fluid and efficient navigation,[7] as drilling down into the data requires less pointing and clicking, and almost no window management operations. But most importantly, it helps users built a strong mental map of the system. The representation of the system is much more concrete, enabling users to more easily navigate the interface thanks to their innate spatial orientation abilities.

The chessboard and map are superimposed on different layers in the same plugin window. The two views are tightly coupled: antenna selection can be performed and modified from both the chessboard and the map. Right-clicking on an antenna on the chessboard and on the map provides users with the same options to invoke plugins. Double-clicking an antenna cell in the chessboard smoothly animates the map to the corresponding antenna. The zoom factor is not changed, meaning that it is possible, e.g., to jump from block diagram to block diagram with a simple double-click action in the chessboard.

This enhanced chessboard, as several of the other plugins described in this paper, is implemented with the ZVTM Java toolkit.[8] The toolkit provides off-the-shelf components and features, such as smooth panning and zooming, magnification lenses,[9] pie menus and other post-WIMP features that require only limited effort to implement[‡]. The toolkit also makes it easy to create graphical animations of most visual variables that define graphical objects, which helps decrease users' extraneous cognitive load by providing a high-level of perceptual continuity[10] between the different states of the interface and the system.

Pie menus are used extensively in the novel plugins, as illustrated in Figure 2-c. In a pie menu, "*the items are placed along the circumference of a circle at equal radial distances from the center. Pie menus gain over traditional linear menus by reducing target seek time, lowering error rates by fixing the distance factor and*

[‡]Implementing such features using WIMP toolkits such as Java Swing is not possible, and usually requires using lower-level drawing APIs such as Java2D, making the development and code maintenance effort much more costly.

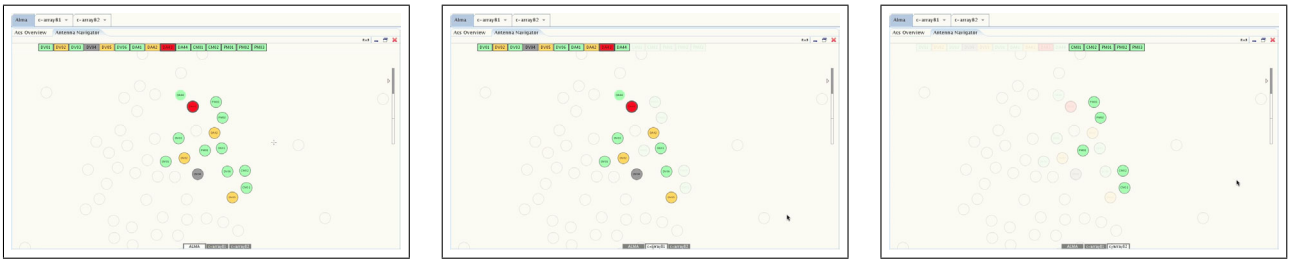


Figure 4. Visually filtering antennas based on subarray assignment: (a) showing all ALMA antennas, (b) showing only antennas assigned to c-array81, (c) showing only antennas assigned to c-array82.

increasing the target size [...]"[11] Overall, when the number of items is small – typically less than eight – pie menus improve command selection performance over the traditional linearly-arranged menus found in most WIMP user interfaces.

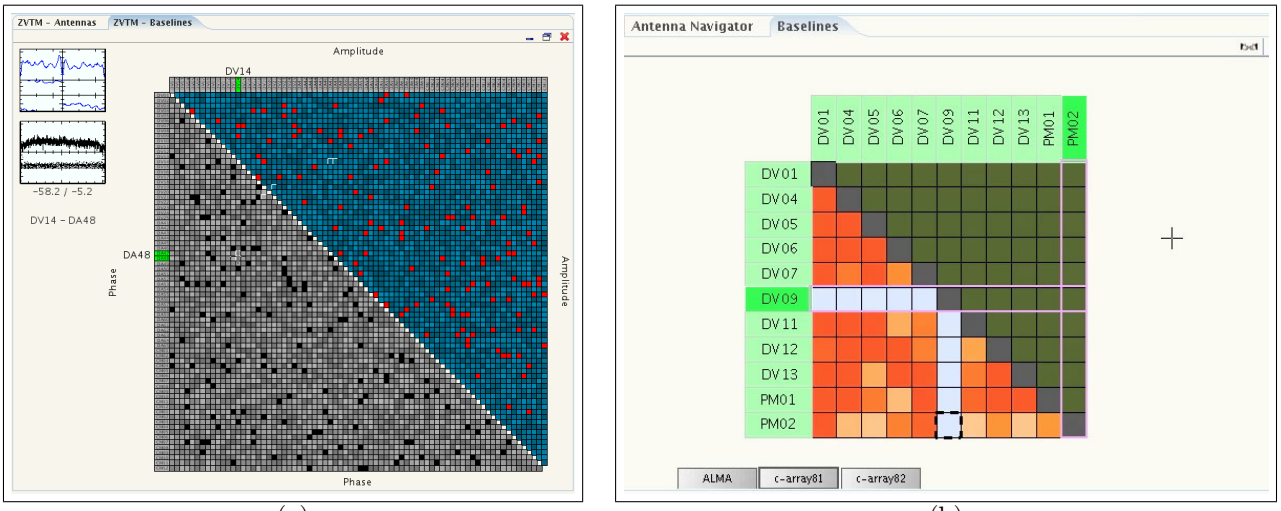
Additional noteworthy features of the enhanced chessboard include visual filtering and layout adjustment. Visual filtering lets users choose what antennas to show in the chessboard and map. While ALMA will comprise more than sixty antennas, all antennas will not be assigned to the same array. The instrument will typically be split into at least two different arrays, with antennas assigned to one of those arrays only at any given time. The enhanced chessboard lets operators and astronomers visualize either all antennas, or the antennas assigned to a specific array, as illustrated in Figure 4. Antennas that are not part of the selected array are not removed from the representation, but rendered with low-contrast colors, as if they were almost transparent. Those antennas no longer clutter the screen, enabling users to focus on the antennas assigned to the array of interest. Nevertheless, users are still aware of their presence and status thanks to the translucent rendering.

The antenna map's layout is based on the actual geographical coordinates of the antennas, to help users topologically relate antennas and possibly other entities (weather stations, antenna transporters, fiber optics paths), and better orient themselves in the system. The map also provides information about antennas shadowing one another, which is an important piece of information during observations. The layout, however, even if based on the actual geographical coordinates of antennas at the high site, is not a simple projection of those coordinates on a 2D plane. While such a projection is available, it will in most cases be very inefficient in terms of screen real-estate usage. There are more than 200 pads on which antennas can be positioned, with many pads at the center of the array separated by only a few dozen meters and forming a compact core, and many other pads scattered along "arms", the longest distance between two pads attaining 16 kilometers. To optimize screen real-estate usage, a distortion algorithm is applied to create a focus+context representation.[12] This distortion can be parameterized depending on the actual array configuration. Indeed, in some configurations antennas are assigned to pads very close to one another, while in other configurations they are scattered on pads that are much more distant from one another. Adjusting the layout by modifying the distorted projection's parameters helps users get a better, more legible map.

3.2 Baselines

ALMA being a radio interferometer, an essential aspect of the instrument are the baselines formed by each pair of antennas of a given subarray. Several variables are associated with those baselines, including signal phase and amplitude, that astronomers want to monitor to detect potential issues and assess the quality of the currently running observation.

Some of these variables are monitored as time-series using the QuickLook software. However, given the total number of baselines (e.g., for 50 antennas, 50*49=1225), time-series visualization fail to provide an overview of this aspect of the system, making it impossible for end-users to evaluate at a glance its status. Representing the network formed by all baselines on the zoomable map of antennas introduced earlier using even the most advanced node-link diagramming techniques is not an option. Indeed, baselines linking all antennas pairwise for a given sub-array, the resulting fully-connected graph would eventually get rendered as a hairball that users would have difficulty making sense of.



(a) (b)

Figure 5. Displaying baselines using an adjacency matrix: (a) early proof of concept showing 60+ antennas; (b) actual implementation (under development) showing a subarray with 11 antennas only. The light blue column/row indicates a problem with antenna DV09.

We thus investigated and started prototyping an alternative representation that often works well for dense networks: adjacency matrices.[13] Adjacency matrices put the nodes of the network as row and column labels. Edges correspond to the cells. The main advantage of this representation is that all edges are visible, no matter how dense the network is. For fully-connected networks, the matrix is completely filled, but all cells are visible, providing a legible overview of very dense networks and making it possible to spot outliers, as illustrated in Figure 5-a.

Users can choose to visualize the matrix of all antennas, or smaller matrices containing only the antennas assigned to the same subarray. The latter option, illustrated in Figure 5-b, will usually make more sense, given that there are no active baselines between antennas that belong to different subarrays.

In the case of undirected networks such as that formed by ALMA's baselines, the matrix is symmetric. This means that two cells are available, one on each side of the diagonal, to display information about each baseline. We are currently in the process of defining two representative figures and their mapping to relevant color schemes. While several ideas have been discussed during participatory design workshops, their implementation proves somewhat challenging. Beyond the theoretical problem of defining such representative figures, the choice is also constrained by technical limitations: the data needed to compute the visual representation are not necessarily accessible from the OMC and require changes in remote parts of the software developed by other groups.

Similar technical limitations constrain the implementation of ideas that arose from the brainstorming sessions. We had initially intended to use the adjacency matrix as an efficient selection mechanism that would have let users quickly brush through baselines, select several of them and access the corresponding time-series plots all at once. Users could have overlaid the plots or displayed them as small multiples,[14] which would have made baseline compare-and-contrast tasks much more efficient. Unfortunately, the data can currently only be obtained for one baseline at a time, and the next request can only be processed at the next integration.

The component is thus limited to the display of plots for one baseline at a time for now. These plots, generated by the QuickLook and CorrGUI software, can be either displayed next to the matrix, or can be embedded in it thanks to the same semantic zooming mechanism as used in the antenna map. Zooming-in on a cell automatically reveals the plots associated with the corresponding baseline.

These two navigation and selection methods are clearly more efficient than the baseline selection mechanisms originally implemented (drop-down menus of antennas), but are still quite far from what we had originally envisioned. Such limitations on the functionalities that can be offered in the interface are not caused by limitations

(a) (b)

Figure 6. (a) Synthesized view of hardware devices as a treemap, for monitoring purposes. (b) Zoomed-in view focusing on a particular device and its sub-hierarchy, that features a faulty component (colored red).

in terms of interaction or visualization, but because of technical limitations upstream, some of which might not exist had the need for this particular feature been identified earlier. This is a typical example that highlights the need to involve end-users and start designing the human-computer interfaces of such advanced and complex instruments much earlier in the project.[15,16]

3.3 Monitoring Component Hierarchies

Several parts of the system are organized into hierarchies of components. This includes hardware devices (antennas, correlator, other computing systems) and the main software components. Such hierarchies are typically represented as expandable trees that let users open specific branches of the tree they might be interested in. This kind of representation is fairly ubiquitous in computing systems and is often employed to browse hierarchies such as, e.g., file systems. Most user interface toolkits make it relatively straightforward to implement such tree components (e.g. Java Swing's JTree).

This type of representation has advantages: the initial view can be relatively compact depending on the breadth of the tree at its top-level, and users can expand and contract branches on demand depending on their particular goal. However, it is relatively ill-suited to contexts in which the visual representation of the tree is designed for monitoring purposes. It is only possible to visualize a very small subset of nodes at a time, and users need to perform many point-and-click actions on very small items to navigate in the tree.

We designed an alternative visualization component to monitor ALMA's hardware devices. As illustrated in Figure 6-a, we use a treemap[17] to represent the hierarchy. While most visualizations of hierarchies represent parent-child relationships by drawing links between the nodes laid out in 2D space according to their depth in the tree (wasting a lot of screen space in the process), treemaps nest child nodes inside their parent, creating a space-filling visualization that optimizes screen space. The technique can display large hierarchies[18] and is particularly well-suited to broad-but-shallow hierarchies as that of ALMA's devices. We lay out hierarchies using the squarified treemap algorithm[19] and make the visualization zoomable, integrating advanced navigation features[20] so that users can focus and get more detail about a particular component and its sub-hierarchy via a simple click on the corresponding node (Figure 6-b) and easily go back up in the hierarchy.

Given the size of ALMA's device hierarchy, it is possible to monitor every single component at any depth in the structure without performing any navigation action, something that was not possible with the original JTree-based UI component. As shown in Figure 6-a, it is easy to spot faulty components thanks to the color coding. Node labels might not all be readable at this scale depending on the size of the enclosing window, but this information can be easily accessed either by hovering the node and dwelling (this pops-up a toolitp), or by clicking on the node in which case the view smoothly pans and zooms on that node, revealing additional information (Figure 6-b).

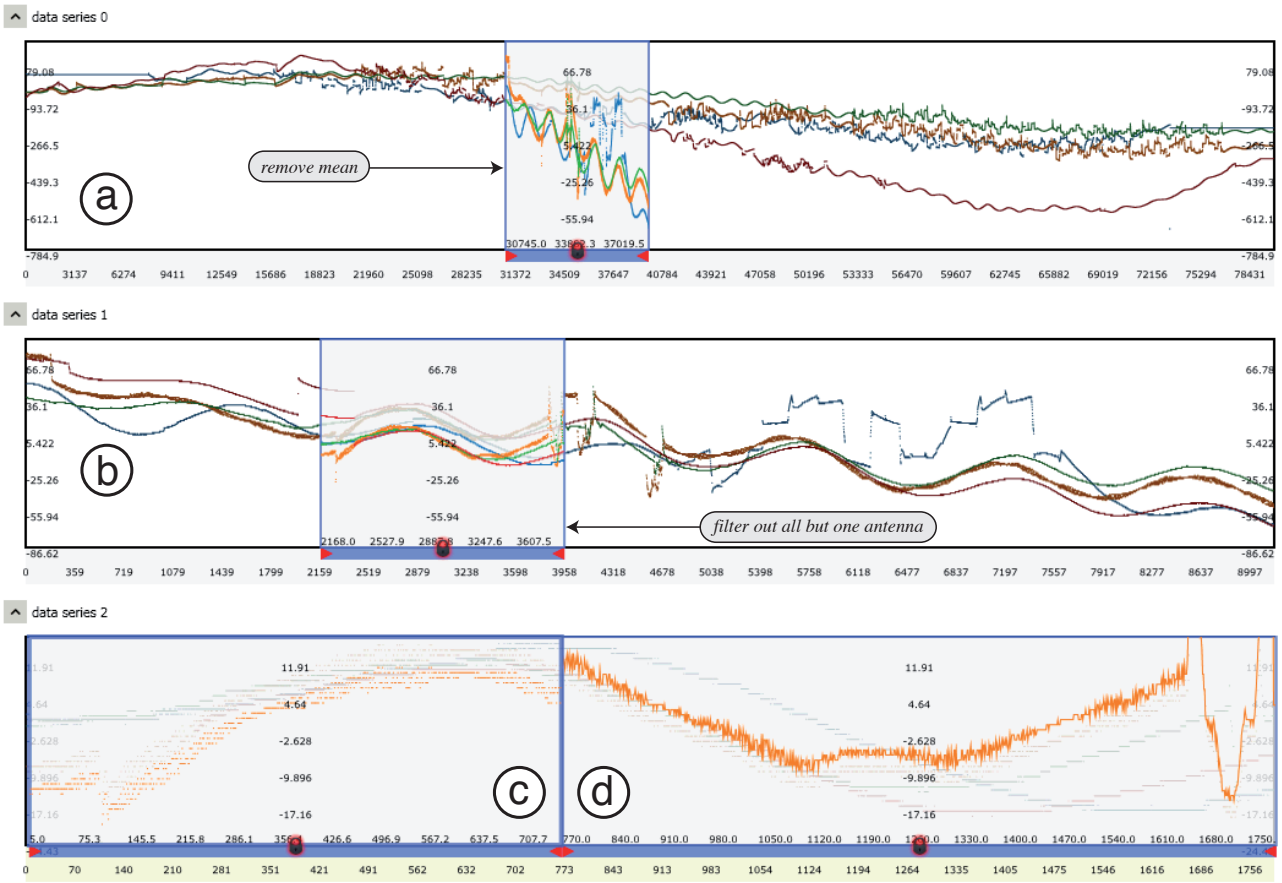


Figure 7. Exploring ALMA Line Length Correction Stretcher Voltage plots for four antennas: (a) two-day overview at a sampling rate of one second; (b) magnification of the 5 hours seen through the *remove mean* lens applied in (a); magnification of the 50 minutes for a single antenna seen through a filtering lens, rendered in scatterplot mode (c) and line-plot mode (d).

This visualization component provides users with a synthesized view of hardware devices. The main treemap corresponds to the antennas. The smaller treemaps on the right-hand side correspond to Power Distribution Units, Antenna Bus Masters, and Correlator Nodes.

3.4 Time-series and Charts

Time-series visualizations are used by both operators and astronomers for a variety of tasks, ranging from checking some of the thousands of monitor points in the system to performing scientific data quality assurance during observations. Users might want to visualize monitor points in real-time, or perform offline analyses on logged data to identify trends and recurring patterns.

Many off-the-shelf packages exist that can plot time-series. However, most of these provide fairly limited functionality in terms of human-computer interaction, and do not scale to the number and size of the time-series that ALMA generates. The following describes a relatively basic scenario that illustrates the benefit of using more advanced multi-scale time-series visualization.

When combining the signals coming from the different antennas taking part in a given observation, ALMA's correlator must know the length of the path traveled by the signal through the fiber optics cables with an accuracy of hundredths of a millimeter. A round-trip laser signal gets sent to all antennas in order to continuously monitor the length of the optical fibers, as the latter can expand and contract due to temperature variations. These changes in path length must be compensated in real-time. This is achieved by the Line Length Correction

(LLC) system, which ensures that path lengths are all stabilized to about 1 micron. Operators are interested in monitoring the LLC stretcher voltage for each antenna, and observing any deviation of an antenna's LLC stretcher voltage compared to that of the other antennas.

From a time-series visualization perspective, different things can be happening at different time scales: from several days to a few minutes. Efficient multi-scale and multi-focus visualization is thus an essential feature. Figure 7-a plots voltage against time for 2 days at a sampling rate of 1 second.

Starting from this 2-day overview, the operator is first interested in finding out whether all antennas are behaving normally or if one or more are somehow deviating. All antennas are expected to behave more or less similarly. Direct visual comparison between the plots, overlaid or stacked, is sufficient to see the deviation of one antenna during the second day (Figure 7-a).

To better see the smaller and shorter fluctuations, the operator creates a lens spanning a 5-hour period, applying a *remove mean* operator (Figure 7-a). The lens is dragged and dropped to create panel 7-b. The content of that panel is a stretched version of what is seen through 7-a's lens. The operator can clearly see that the fluctuations are in phase, which implies that they all come from a single source. She could then plot various monitoring points simultaneously based on system or environmental components likely to cause these fluctuations and create lenses defining a cross-correlation operator (not shown here), eventually tracing the source to temperature fluctuations in the local oscillator room.

Finally, zooming in further to display just 50 minutes (Figure 7-c), the operator can see some odd features in the signal for one of the antennas. She filters out antennas that behave normally using a lens defining the appropriate filter operator. She also switches from a scatterplot to a line-plot rendering inside the lens focus (Figure 7-d), revealing fast oscillations at a much finer scale that occur on top of the slow ones observed earlier for this particular antenna. This antenna-specific issue eventually gets traced to a device repeatedly turning on and off in the antenna.

We recently started working on a tool that will provide a subset of the functionalities offered in the ChronoLens framework,[21] in order to support such scenarios. We are also considering providing alternate representations of time-series that will enable users to get an overview of a large number of time-series simultaneously and access the details as line graphs for a subset of them, as in Line Graph Explorer[22] which *"encodes the y-dimension of individual line graphs with color instead of space"*.

In addition to time-series, ALMA software such as QuickLook use 2D chart drawing packages to create scatterplots, bar charts, etc. One of those software is QuickLook, that relies upon JFreeChart to display the data on screen. JFreeChart supports multiple types of charts, but features poor interactive capabilities, which severely limit its usability when displaying more than a few variables on the same chart. This is a significant problem as users of QuickLook will typically have to deal with charts for more than 60 antennas, and hundreds (if not thousands) of baselines. To partially address this problem, we made enhancements to QuickLook and JFreeChart, that enables users to visually highlight the values of one particular variable on multivariate plots when brushing over the corresponding labels, or filter out, i.e., hide either temporarily or permanently, variables considered as not relevant to the task at hand. The charts themselves have been made pickable, so that users can click on a particular value and unambiguously relate it to the corresponding variable when color is not sufficient to differentiate the different plots overlaid in the same window.

3.5 Coordinated and Multiple Views

The OMC provides users with a series of plugins that lets them inspect different aspects of the system and control it: antenna control, event logs and alarm panels, block diagram of the hardware in each antenna, information about the correlator, software components, etc. One guiding principle in terms of software architecture was that plugins should be independent from one another. This has clear advantages in terms of software development (no dependencies) and robustness of the system. However, it also means that plugins are not aware of each other and that they cannot communicate with one another. One consequence in terms of user interface design is that the views provided by the different components, while providing complementary views on the system, cannot easily be coordinated.

Figure 8. Top: Three independent user interface components that give complementary views of the system. Bottom: the antenna map can be coordinated (a) with both the AntennaStatus panel and the logging panel (a). Then, whenever the user selects an antenna in the chessboard or the map (here DA43), the AntennaStatus panel gets automatically updated to show DA43's block diagram (b), and the logs get filtered to only show entries related to that antenna (c).

Coordinated and Multiple Views[23] are a very powerful mechanism that makes navigation in a complex user interface potentially much less tedious, saving users valuable time when troubleshooting the system. They are often used in the exploratory visualization of large and complex datasets to help users in their sense-making process.

While we acknowledged the strength of the approach adopted in terms of plugin independence, we proposed to add a lightweight coordination mechanism that would let plugins publish *interaction events*. Other plugins could subscribe to these events and react to the corresponding callbacks based on user settings and on their interpretation of the event.

This lightweight coordination mechanism has been implemented at the level of the OMC framework, and each plugin is now free, but not required to, implement it. Various types of events have been defined, such as *antenna selected*, *antenna unselected*, *baseline selected*, etc. Different plugins will fire different subsets of events, and can listen to a different subset of events. More types of events can be added in the future. The approach taken is very open and non-binding.

Thanks to this mechanism, navigation in the system can be made dramatically much more efficient. Figure 8 illustrates a simple example. The top part shows three independent components: a map of the antennas, an AntennaStatus panel showing the block diagram of one particular antenna, an logging panel showing all errors and warnings logged by the system.

In the absence of the plugin coordination mechanism, a user wanting to troubleshoot Antenna DA43 (painted red on the map to indicate that it is faulty) would have had to manually select the antenna in the AntennaStatus window, and to perform many tedious actions to setup a filter in the logging panel so as to only see the log entries related to that particular antenna.

With the coordination mechanism that was recently implemented, users only have to setup a link between the three plugins once using the *handshake* icon (Figure 8-a) to specify that whenever an antenna gets selected on the map or in the chessboard, the AntennaStatus panel should automatically be updated to show that antenna's block diagram (Figure 8-b), and the logging panel should only show entries related to this same antenna[§] (Figure 8-c). This way, all components get updated with a single click.

A similar behavior could be implement for the general logging system, and we anticipate that other plugins will be upgraded to support the coordination mechanism, possibly defining new types of events. For instance, we have started working on *baseline selection* events, that can be used to coordinate, e.g., the antenna map and the baseline adjacency matrix. Selecting a particular baseline in the matrix displays a straight link between the two corresponding antennas on the map, possibly panning and zooming the view so as to make both of them visible, thus quickly giving users an idea of that baseline's length. Baseline selection events can also be used in the future to filter QuickLook views to show only (or highlight in context) particular baseline time-series thanks to the extensions to JFreeChart briefly described in Section 3.4.

One noteworthy subtlety is that coordination links established between views are not symmetric by default. For instance, the link set from the antenna map to the AntennaStatus window in Figure 8 means that whenever the user selects an antenna on the map, the AntennaStatus diagram will update automatically. However, manually selecting another antenna in the AntennaStatus window does not impact the antenna map in any way. The converse link can of course be created (provided it makes sense), but has to be set explicitly.

4. SCIENCE AND OPERATIONS WORKSTATION HARDWARE SETUP

The OMC software user interface components described above, along with all other existing components that run within or outside of the OMC, are hosted on the science and operations workstations located in the control room. A station is typically manned by an operator and an astronomer on duty (AoD), who interact with the system through different UI components according to their respective tasks, but also share a common subset of interface components that are relevant to both of them.

ALMA being a very complex instrument, operators and astronomers are faced with large amounts of information that have to be displayed either permanently, e.g., critical monitor points and information about currently running observation, or temporarily, e.g., a console to launch scripts during a troubleshooting session or a Web browser to access documentation. The early answer to the problem of screen real-estate posed by the large number of UI components to be displayed was first addressed by adding more screens. In December 2011, the main science and operations workstation consisted of eight 23" screens laid out horizontally. Obviously, this approach will not scale indefinitely, but more importantly, it is far from optimal from an ergonomics perspective.

[§]This particular plugin does this by hiding the rows that do not match the query. An alternative would be to keep all rows visible and visually highlight the rows that match it.

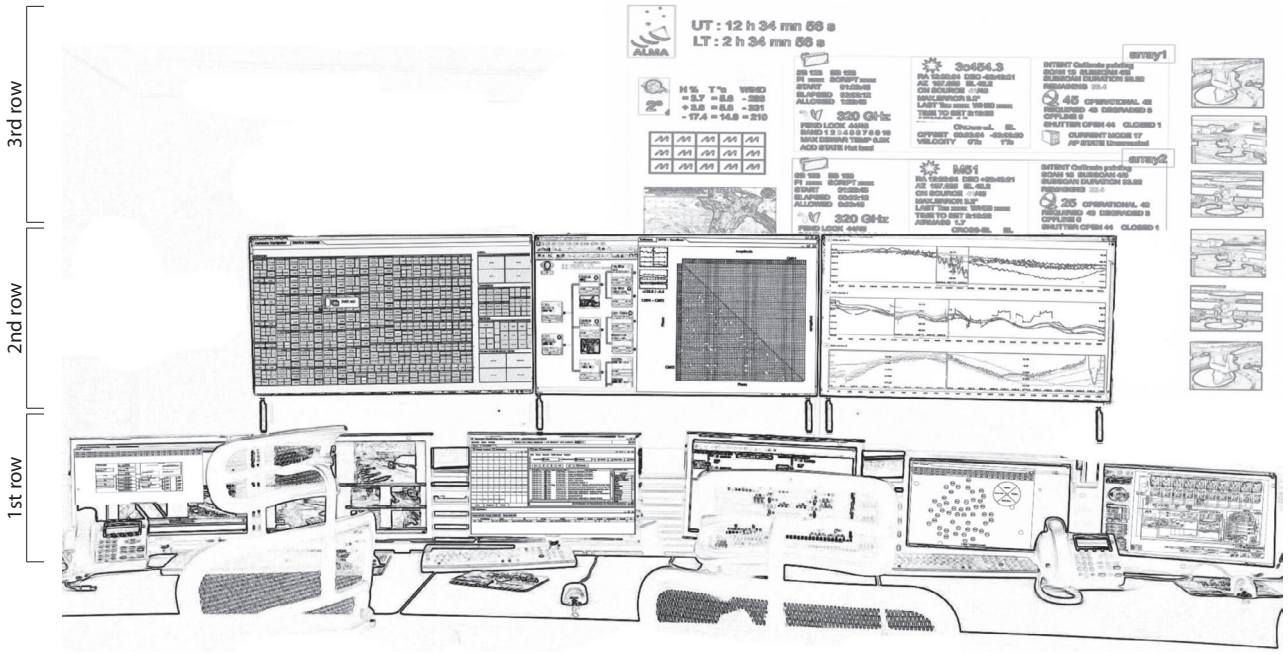


Figure 9. Sketch of the proposed science and operations hardware setup. The first row contains UI components that users interact with frequently. The second row contains UI components that users interact with infrequently. The third row contains UI components that users never interact with. This display can be seen by everyone in the room and its content is configured statically. It provides a high-level overview of the system, current observation(s) and observing conditions.

In December 2011 we began working on an alternate, more elaborate hardware setup for the workstation. We started with a list of all existing user interface components that are displayed on this station to assess the amount of screen real-estate required to display them efficiently.

As mentioned earlier, UI components can be standalone applications or plugins running within the OMC. They can run permanently or be invoked for a limited period of time. An additional characteristic of these components is how much users interact with them. Some components are for monitoring purposes mostly, and users essentially look at them from time to time, but do not interact with them. Other components are more interactive, and take input (keyboard and mouse) from the user on a regular basis. It is important to note that this is not a binary classification, but a continuum, that can nevertheless inform the layout of UI components and their assignment to the various screens that compose the workstation.

We eventually came up with a proposal where screens are organized in three levels. The first level (lower part of Figure 9) contains the highly interactive UI components. It is made of standard 23" LCD panels at high resolution (approx. 100dpi). The second level (middle part of Figure 9 contains UI components that are mostly used for monitoring purposes, i.e., that display more static content. It is made of larger (46") panels, that could either use back projection or LCD tiles. They are offset in depth and cover a wider field of view than the screens on the first row. Those screens can be read from farther away. Their purpose is to give a relatively detailed view of system status, and to act as a communication medium between operators and astronomers by providing them with a common representation of the system. Both the operator and astronomer can see all three 2nd-row screens from their seat. Other parties in the room can also better see those screen than the smaller screens used currently. It is less important to have a high pixel density on those second level screens. Thus, FullHD resolution is likely sufficient, despite the increased diagonal. The third level (upper-right part of Figure 9) uses projection to create a large display readable from all workstations in the control room. This row is never interacted with. Its layout is static. The purpose of this row is to give a high level overview of system status and other information that is relevant to all groups in the room, including information about running observation(s) and current weather. By looking at this display, any personnel entering the room can quickly get an overview of the situation.[16]

We made a proposal for the assignment of UI components to the different rows and screens of this hardware setup[¶], informed by current practices and by the assessment of optimal dimensions and tolerance for each individual view. As mentioned earlier, the workstation is shared by the operator and astronomer on duty. It is organized so that operator-oriented UI components are on the left side, and astronomer-oriented components are on the right side, with components relevant to both users in the central part.

ACKNOWLEDGMENTS

We wish to thank all astronomers, engineers and operators who participated in discussions and provided us with insightful comments and ideas, including: Ravinder Bathia, Larry Brother, Alessandro Caproni, Stuart Corder, Lindsey Davis, Bill Dent, Philippe Duhoux, Brian Glendenning, Preben Grosbøl, Antonio S. Hales, Jorge Ibsen, Daniel Kent, Alison Peck, Gianni Raffi, Mark Rawlings, Kartik Seth, Debra Shepherd, Nick Whyborn, Honglin Ye.

REFERENCES

1. Definition, "WIMP (computing)." http://en.wikipedia.org/wiki/WIMP_(computing) (Last accessed: May 25, 2012).
2. van Dam, A., "Post-WIMP user interfaces," *Commun. ACM* **40**, 63–67 (Feb. 1997).
3. Schwarz, J., Pietriga, E., Schilling, M., and Grosbøl, P., "Goodbye to WIMPs: A Scalable Interface for ALMA Operations," in [*ADASS '10: Proceedings of the Astronomical Data Analysis Software and Systems*], Evans, I. N., Accomazzi, A., Mink, D. J., and Rots, A. H., eds., *ASP Conference Series*, ASP (2010).
4. Schilling, M., Primet, R., Pietriga, E., and Schwarz, J., "Human Computer Interaction in the ALMA Control Room," in [*ADASS '11: Proceedings of the Astronomical Data Analysis Software and Systems*], *ASP Conference Series*, 137–140, ASP (2011).
5. Bederson, B. B. and Hollan, J. D., "Pad++: a zooming graphical interface for exploring alternate interface physics," in [*Proceedings of the 7th annual ACM symposium on User interface software and technology*], *UIST '94*, 17–26, ACM, New York, NY, USA (1994).
6. Perlin, K. and Fox, D., "Pad: an alternative approach to the computer interface," in [*Proceedings of the 20th annual conference on Computer graphics and interactive techniques*], *SIGGRAPH '93*, 57–64, ACM, New York, NY, USA (1993).
7. Pietriga, E., Appert, C., and Beaudouin-Lafon, M., "Pointing and Beyond: an Operationalization and Preliminary Evaluation of Multi-scale Searching," in [*CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*], 1215–1224, ACM Press, New York, NY, USA (2007).
8. Pietriga, E., "A Toolkit for Addressing HCI Issues in Visual Language Environments," in [*IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*], 145–152, IEEE Computer Society, Los Alamitos, CA, USA (2005).
9. Pietriga, E., Bau, O., and Appert, C., "Representation-Independent In-Place Magnification with Sigma Lenses," *IEEE Transactions on Visualization and Computer Graphics* **16**(03), 455–467 (2010).
10. Robertson, G. G., K.Card, S., and D.Mackinlay, J., "Information visualization using 3D interactive animation," *Communication of the ACM* **36**(4), 56–71 (1993).
11. Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B., "An empirical comparison of pie vs. linear menus," in [*Proceedings of the SIGCHI conference on Human factors in computing systems*], *CHI '88*, 95–100, ACM, New York, NY, USA (1988).
12. Cockburn, A., Karlson, A., and Bederson, B. B., "A review of overview+detail, zooming, and focus+context interfaces," *ACM Comput. Surv.* **41**, 2:1–2:31 (Jan. 2009).
13. Henry, N. and Fekete, J.-D., "Matlink: enhanced matrix visualization for analyzing social networks," in [*Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*], *INTERACT'07*, 288–302, Springer-Verlag, Berlin, Heidelberg (2007).
14. Tufte, E., [*The visual display of quantitative information*], Graphics press Cheshire, CT (1983).

[¶]Since these workstations are multi-user work environments, users should not be allowed to customize the interface too much. A good one-size-fits-all UI component layout should be defined and used by all so that any user can easily orientate herself in the interface if she has to take over in a critical situation.

15. Shneiderman, B. and Plaisant, C., [*Designing the User Interface: Strategies for Effective Human-Computer Interaction (5th Edition)*], Addison-Wesley Longman. (2009).
16. Endsley, M. R. and Jones, D. G., eds., [*Designing for Situation Awareness: an Approach to User-Centered Design*], CRC Press, Taylor & Francis (January 2012). 370 pages.
17. Johnson, B. and Shneiderman, B., "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in [*Proceedings of the 2nd conference on Visualization '91*], *VIS '91*, 284–291, IEEE Computer Society Press, Los Alamitos, CA, USA (1991).
18. Fekete, J.-D. and Plaisant, C., "Interactive information visualization of a million items," in [*Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*], *INFOVIS '02*, 117–, IEEE Computer Society, Washington, DC, USA (2002).
19. Bruls, M., Huizing, K., and van Wijk, J., "Squarified treemaps," in [*In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*], 33–42, Press (1999).
20. Blanch, R. and Lecolinet, E., "Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques," *IEEE Transactions on Visualization and Computer Graphics* **13**, 1248–1253 (November 2007).
21. Zhao, J., Chevalier, F., Pietriga, E., and Balakrishnan, R., "Exploratory Analysis of Time-Series with ChronoLenses," *IEEE Transactions on Visualization and Computer Graphics* **17**, 2422–2431 (2011).
22. Kincaid, R. and Lam, H., "Line graph explorer: scalable display of line graphs using focus+context," in [*Proceedings of the working conference on Advanced visual interfaces*], *AVI '06*, 404–411, ACM, New York, NY, USA (2006).
23. North, C. and Shneiderman, B., "Snap-together visualization: a user interface for coordinating visualizations via relational schemata," in [*Proceedings of the working conference on Advanced visual interfaces*], *AVI '00*, 128–135, ACM, New York, NY, USA (2000).