Playbook Data Analysis Tool: Collecting Interaction Data from Extremely Remote Users

Bob Kanefsky[1*], Jimin Zheng[1], Ivonne Deliz[2], Jessica J. Marquez[3], and Steven Hillenius[3]

[1] San José State University Research Foundation
[2] ASRC Research & Technology Solutions
[3] NASA Ames Research Center, Moffett Field, CA
{bob.kanefsky, jimin.zheng, ivy.deliz, jessica.j.marquez, steven.r.hillenius}@nasa.gov

Abstract. Typically, user tests for software tools are conducted in person. At NASA, the users may be located at the bottom of the ocean in a pressurized habitat, above the atmosphere in the International Space Station, or in an isolated capsule on a simulated asteroid mission. The Playbook Data Analysis Tool (P-DAT) is a human-computer interaction (HCI) evaluation tool that the NASA Ames HCI Group has developed to record user interactions with Playbook, the group's existing planning-and-execution software application. Once the remotely collected user interaction data makes its way back to Earth, researchers can use P-DAT for in-depth analysis. Since a critical component of the Playbook project is to understand how to develop more intuitive software tools for astronauts to plan in space, P-DAT helps guide us in the development of additional easy-to-use features for Playbook, informing the design of future crew autonomy tools.

Keywords:
usability · human-computer interaction · planning and scheduling · software

1. Introduction

When developing products, it is common for user researchers to iteratively test them using user-research techniques dictated by their evaluation needs and accessibility of target users. Our team, the Human-Computer Interaction Group at NASA Ames Research Center, follows a similar process of user-centered design but has adapted these techniques to work with our unique users and constraints. Since we are a user-focused software development team, one of our core goals when building features and products is to ensure that they are easy to use and able to assist users in the work they need to accomplish. User testing is a standard evaluation technique that we use to iteratively improve our products. Ideally, we would sit side by side with users to evaluate usability. This is adequate for initial evaluations in a lab or where the users' work takes place in an accessible environment, such as an office desk or ground-based mission control facility. But for users in remote, inaccessible locations, this is not an option. In the past, we balanced co-located, lab-based usability tests with product use in operational missions where feedback is only received through indirect comments, post-mission debriefs, or surveys. Gathering detailed, operational, in-use usability data from our remote users – as is typical during in-person user tests – has not been possible. As our products moved to mobile devices and became primarily used for remote mission operations, this became a growing blind spot in terms of accurate usability evaluation.

Operational usability data is critical, as it provides the most realistic product usage information in the context of the operational pressures of a mission; many of the features and tools that we develop have drastically different use patterns when deployed in the environment of an active mission. For example, a software feature that may indicate no major usability issues in a lab setting may have significant usability problems in an operational environment. Notably, these users may be extremely inaccessible, onboard a spacecraft such as the International Space Station (ISS) or in a remote isolated habitat such as the NASA Extreme Environment Mission Operations (NEEMO) mission where several astronauts and engineers live underwater for weeks at a time. Bandwidth is also a major concern; network resources and disk space are limited for spacecraft or remote mission operations, so using typical screen capture or video recording techniques are not practically viable, as operational mission needs take priority. The common recording strategies mentioned earlier also add to the overhead of gathering data and require manual analysis. In addition, since our products and users are mobile, typical physical-recording solutions involving a mount of some sort or the setup of recording hardware in a fixed location would significantly disrupt mission operations. Remote user testing is not new, but the unique user testing constraints required us to develop a new software solution to gather remote usability information without disrupting operational usage, both from the perspective of accuracy of collected data and mission operations.

In this paper, we present the development of P-DAT, the Playbook Data Analysis Tool, an unobtrusive, web-based, structured, bandwidth-efficient, and mobile-friendly tool developed to gather video-like playback of true operational usability data from our mobile planning tool, Playbook.

2. Background

2.1. Playbook's Role in Future Crew Autonomy

NASA conducts research on a variety of issues that affect human performance in spaceflight. In particular, the Human Research Program (HRP) is investigating methods to mitigate the potentially detrimental effects on crew performance caused by inadequate or poor interaction between astronauts and the complex systems used during long duration space missions. For exploration-class missions beyond the Moon, astronauts will need to have more autonomy, as Earth flight controllers will not be immediately available once communication lag is measured in minutes instead of seconds. Therefore, we have been developing and evaluating software aids that specifically support crew autonomy.

An astronaut's schedule in space is usually busy; aboard ISS, their day is filled with science experiments, maintenance tasks, public outreach, as well as exercise and other activities for their well-being (eating, talking to family, etc.). As they go through their day, they may get ahead or fall behind on their schedules, depending on the complexity of assigned tasks. Currently, mission controllers monitor astronauts through video feeds, while astronauts frequently talk to mission controllers, updating them on progress. Though effective now, these methods will be inefficient during future long-duration missions to deep space. Enabling astronauts to manage their own schedules may prove to be a more efficient way of conducting operations.

In order to investigate crew autonomy and this new concept of operations, our team has developed and evaluated Playbook [1, 2], an easy-to-use, mobile-based, timeline planning tool. Playbook has several key views – Timeline, Mission Log (a messaging interface), Task List, and a Procedures list – that support crew members' daily spaceflight operations. Playbook's main function is to integrate and visualize the schedules of multiple crew members and mission control in one view (Fig. 1). One of the advantages over current timeline tools is that Playbook allows users to self-schedule, i.e., to edit and manipulate activities in their timelines, rescheduling activities through simple drag and drop interactions. However, many activities cannot be freely moved, because their associated constraints are only met during certain time ranges. Spaceflight timelines must abide by dozens of these operational constraints; hence, activities are currently scheduled by a team of planners to meet those constraints. In order to accommodate this, Playbook provides visual feedback to the user about where an activity could be rescheduled, e.g., to avoid a predicted communications gap in the middle of a video conference. Additionally, Playbook allows crew members to create new activities and add "task list" activities, which are activities that can be completed if the astronauts have free time in their day.

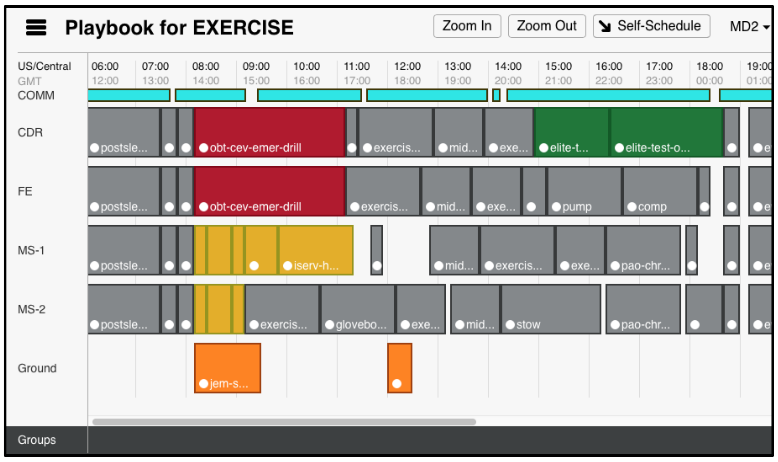

Fig. 1. Playbook screenshot, showing example timelines for four crew members and mission control as well as availability of communication channels.

Our research goals for Playbook are to: 1) design and develop a technology platform that enables the assessment of the feasibility and effectiveness of crew self-scheduling and 2) evaluate crew autonomy and self-scheduling through the use of Playbook in settings analogous to spaceflight environments. Over the last several years, we have developed various Playbook features aimed at making self-scheduling easy and intuitive, user-testing these feature implementations in Earth-based analog missions along the way to verify their efficacy.

Analog missions are conducted in *relatively* safe and accessible locations such as the ocean floor and volcanic terrain that serve as stand-ins for asteroids, the Moon, and Mars. Analogs play a significant role in experimenting with space mission designs, and some even perform Earth science in their own right. One such analog is the NASA Extreme Environment Mission Operations (NEEMO), where astronauts live in an underwater habitat for roughly two weeks. Our team has incrementally deployed and evaluated Playbook for the purposes of studying crew autonomy in several NEEMO missions [3]. Over the last several years, we have also deployed the software for purposes of exploring crew autonomy in the Human Exploration Research Analog (HERA) and the Biologic Analog Science Associated with Lava Terrains (BASALT) missions. As of this paper's writing, Playbook is also in the middle of being evaluated aboard ISS through a series of technology demonstrations in collaboration with the ISS Program and ISS Ops Planners (mission controllers who focus on timeline planning and integration).

In order to meet our research goals, we need to not only collect general feedback from our product users, astronauts and various analog participants, but also establish a systematic method to collect usability and usage data unobtrusively within the various extreme environments during realistic mission operations.

2.2. Collection of Usability Data

HCI practitioners have at their disposal a variety of research methods to collect user feedback in the form of usability data. Typically, these methods fall into one of two categories: in-person or remote. As the name would suggest, in-person research requires researchers to be co-located with users. Many of these methods – including contextual inquiry, participatory observation, and in-person usability testing – are beneficial in that they allow researchers to directly interact with (and adapt to) users, and give researchers a chance to collect additional in-situ contextual information [4]. In contrast, remote research methods can be much more hands-off and do not require researchers to be present at all. What remote methods sacrifice in immediate access to users is made up for by having the benefits of being flexible and unobtrusive. Oftentimes, remote methods use software to capture data in the background with no impact to users. Many modern methods, including A/B testing, web analytics, and screen capturing, can be utilized across many time zones and introduced seamlessly into the tool to allow for these usability experiments at a larger scale.

Due to limitations of our team's ability to conduct in-person usability research in Earth-based analogs and in space, we established a remote data collection mechanism in Playbook, P-DAT (Playbook Data Analysis Tool). In principle, we are leveraging automated software logging to gather use data for post-processing. By continuously capturing browser content and events, we are able to play back a session of data as if the interaction had been recorded as a video. As HCI researchers and practitioners, we are able to review Playbook interactions from any time users conducted self-scheduling. We are able to observe potential usability problems, like repeatedly attempting to drag a fixed activity. Additionally, we might observe software bugs users might have encountered while using the software tool.

Unobtrusively collecting software interaction data may also open the door to systematically determining usability metrics and patterns of interactions. For example, one useful metric might be the number of times the timeline was zoomed per day or the distance activities were dragged per rescheduling event. Information like this might provide further insight with regards to the overall usability of the tool over time. Furthermore, if information is captured in the context of a mission, it may also serve a way to quantify performance in spaceflight or analog environments. As the timeline shows scheduled activities, knowing when an activity was interacted with (e.g., opening linked documents or marking complete) may provide data about time on task, whether particular tasks are difficult to complete on time, or even if certain times of day are less optimal schedules for certain tasks. Future research may focus on certain patterns of interaction which in turn may be correlated to individual or group behaviors. For the moment, we are focusing on providing video-like playback of the data we have collected and on basic features to help find pertinent recordings.

3. P-DAT User Interface

Users interact with P-DAT using a graphical user interface that is composed of two primary components: the replayer, and a timeline overview of all recorded usage, which we call MetaPlaybook since it uses Playbook as a viewing tool.

3.1. Replayer

The replayer component of P-DAT is a user interface for playing back user sessions that occurred in Playbook. The UI was designed specifically to allow its users to watch playbacks of actual Playbook sessions as if they were videos recorded by aiming a video camera at the screen or using screen capturing software. The controls are designed to be similar to video playback software, augmented with timelines to give users the ability to find specific events that occurred during the sessions. This allows P-DAT users to identify trends in behavior and analyze actual interactions to help discover Playbook usability issues.

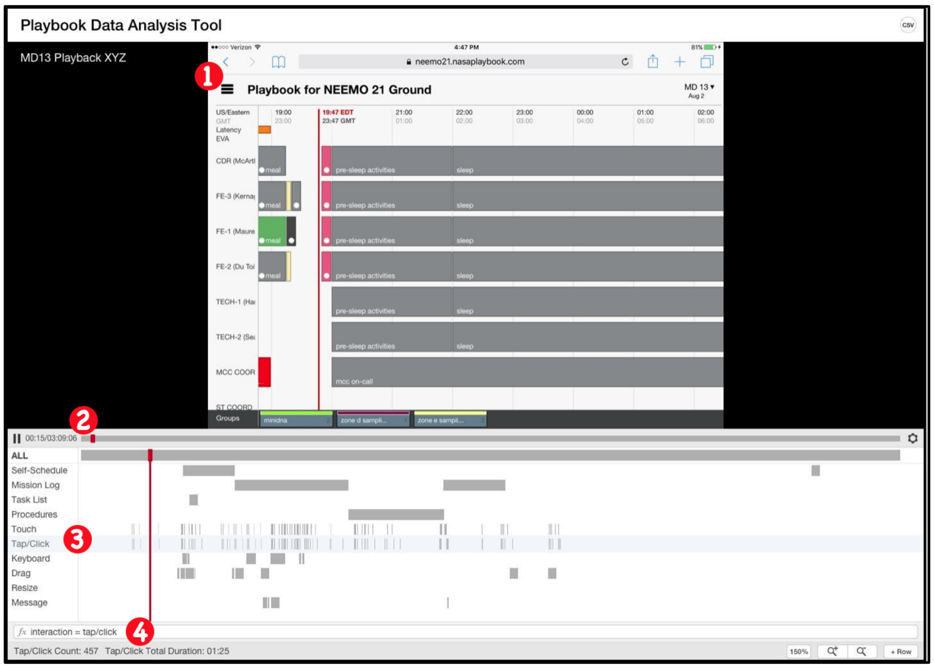

Fig. 2. P-DAT User Interface screenshot.

In Fig. 2, the primary components of the replayer are: (1) Playback area, (2) Progress bar with playback controls, (3) Event timeline, and (4) Query bar. Using the logs of a user's interactions in Playbook, P-DAT reconstructs what the user saw while using Playbook. This reconstruction is then replayed in the playback area (1), with conventional play and pause controls adjacent to a progress bar used for scrubbing through the playback (2).

The data has additional information beyond what is needed for reconstructing this playback. It includes a log of interactions and on-screen elements, enabling the tool to display where certain events occurred during the entire sequence of the Playbook session. The event timeline (3) section of the interface visualizes these events and allows a researcher to use P-DAT to easily identify where certain actions occurred. The rows of the event timeline contain Playbook user interaction events such as taps/clicks, keyboard inputs, drags, etc. These rows can also show which of Playbook's specific views were displayed: Timeline, Mission Log, Task List, Procedure, etc.. Along each of these rows, the time where these events occurred is marked in gray. This allows researchers to see precisely at what point in the session the actions occurred. Clicking on any of these marked areas in the event timeline will reposition the playback to that specific area, showing the researcher the actions that occurred at that time. For increased fidelity, the event timeline can be zoomed in and out.

To extend the functionality of the event timeline, P-DAT has a query language. Each row is generated from a predefined query, and the user may edit it, or add new ones, using the query bar (4) at the bottom. This feature is analogous to the function bar in spreadsheet software, where the formula for a specific cell is displayed when a particular cell is selected. Clicking on any of the rows in P-DAT's event timeline will populate the query bar with the query corresponding to that row, and the user can edit the formula to adjust what actions are highlighted. This feature also allows users to construct complex queries to find specific events in session playbacks. Examples of complex queries are:

- `Click followed by drag followed within 10ms by drop without any tap or click`
- `During view=Timeline, orientation=portrait`
- `During mode=edit, drag followed by drop`

This query language allows users of P-DAT to define specific events that are relevant to their research questions. For example, in Playbook, dragging anywhere in the Timeline view normally scrolls the entire view so the user can see activities later or earlier in the day that did not fit within the window, but if the Self-Scheduling button is pressed first to enable editing mode, then dragging an activity moves it within its timeline, where it can be dropped at a new time. That combination of button state and user action defines a self-scheduling event that can be queried or filtered. This allows finding P-DAT recordings, or sections of recordings, that are of importance to researchers from a usability perspective. For example, we can search for mode errors: cases in which a user appears to be attempting to move an activity but inadvertently scrolls, or vice versa, and immediately reverses course.

3.2. MetaPlaybook

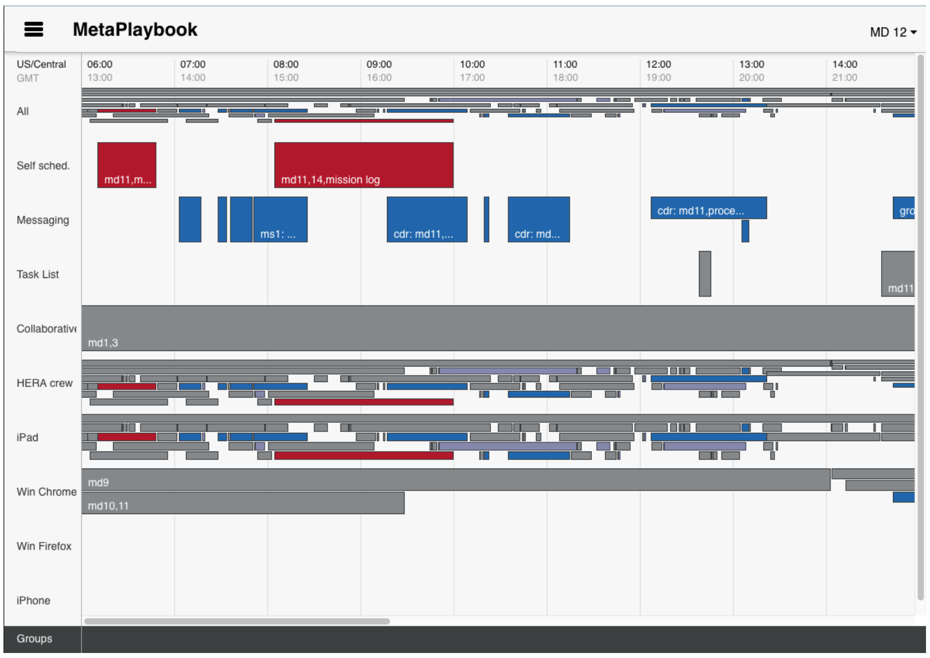

Fig. 3. MetaPlaybook consists of Playbook with data representing recordings as "activities."

When Playbook is being used in operations, users open and close it throughout the day and it can be used by several users at a time, which results in a large number of recorded sessions for each day of a mission, necessitating a way to find relevant recordings. We have chosen to use Playbook itself as a tool to visualize past Playbook recordings in timeline form. A script generates a "plan" in which each "activity" is a recorded session, and each row is a distinguishing characteristic of a recording (e.g. whether certain features were used or Playbook was accessed from a specific browser), and the activities link to that recording via the unique id of the recording to open in P-DAT for analyzing. When we use Playbook to show this overview of recordings Playbook for a mission, we call it MetaPlaybook.

4. Technical Development of Data Collection

Since the primary requirement for P-DAT was to record and replay a session as if a video camera had been aimed at the screen, we considered three approaches that one might take to achieve this. One could record the actual pixel values of the screen in a series of rapid screen shots, many times per second, and store it in a compressed video format. There are several commercial and free screen-capture tools available to do exactly that. We did not pursue this option because video files, even compressed, tend to be large, and communication bandwidth from orbit is a limited resource.

Alternatively, one could record every input event (every click or tap, drag gesture, and so on) received by Playbook, along with the name of the event handler. (An event handler, also called a listener, is a subroutine that is called each time an input is received. For example, Playbook's search button handles a click or tap by toggling its colors and displaying or hiding a search-result sidebar.) Then, in order to replay the events, one would start Playbook and call the same handlers with the same timing. Since software behaves deterministically, this would theoretically produce exactly the same visual results the user saw, provided one started with *exactly* the same application data (the mission plan and history of previous changes) and replayed it with *exactly* the same version of Playbook, among other variables. This was the approach we initially implemented, but we realized it was a fragile one, since a single unrecorded step would send the playback in the wrong direction. It would also have required us to preserve snapshots of every version of the data and software used. Moreover, it would have been burdensome to maintain, since the hooks in the software for recording events would have had to keep up with the rapid development of new features in Playbook.

Instead, we now directly record what Playbook displays, by listening for changes to the HTML Document Object Model (DOM), the underlying data that the web browser renders. We begin the recording log with the initial DOM that appears upon page load. HTML consists entirely of *elements*, which can have *attributes* and contain text or other elements, so the recording process consists of watching for changes to any of these. We log an entry each time an element is added, deleted, or modified, or its text or an attribute is changed. Each of these DOM-change entries is simply a terse line of comma-separated text, referring to elements by previously assigned IDs. To keep the size of the recordings manageable, we throttle the more rapid attribute changes (representing animations) to a reasonable frame rate and omit some of the intermediate stages; there is no need for the replay to look perfectly smooth.

While these DOM-change entries are sufficient to capture almost all visual changes on the screen, there is still the need to record certain additional changes that do not affect the DOM but do affect what the user sees, such as scroll position, window-size changes, and keyboard input. We also record where the user touches the touchscreen, and visualize that by overlaying fingerprint images during playback.

To support real-time playback, a third type of entry is needed in addition to DOM changes and events: the passage of time. Relative timestamps are added between any two entries that had a human-perceptible lag between them – whether a split-second pause between an element flashing red and then disappearing, or a user hesitating in thought for a moment between clicks, or the clock's minute-long pauses between ticks. (To save bandwidth, we omit timestamps between changes that are effectively simultaneous: when the browser is working as fast as it can to redraw several elements moving in lockstep, we don't record the milliseconds that slip by.)

When these DOM changes are reconstructed in a browser, this technique guarantees a pixel-perfect recreation of exactly what the user saw, as if video frames had been captured. (Large static resources, like style sheets and images, are external to the document and not copied into each recording, so we take care to point P-DAT at contemporaneous versions of them.) This approach to recording data for video-like playback has an additional advantage besides data volume savings: it lends itself to automated analysis much better than pixel data would have. The inherent structure of the collected data allows researcher to enter queries as described in the previous section.

In summary, the recorder component of P-DAT runs in the background while users are interacting with Playbook; it listens to the DOM for visual changes, and also records browser events in the DOM. When Playbook is opened in a browser, the recorder writes a snapshot of the initial HTML to a text file on the server and gives it a unique ID. As this content changes, it logs the changes, additions and deletions of the DOM elements until the user closes the Playbook window. Changes to the DOM include changes in any attributes of the elements. Using a URL with the unique ID for each recording session, P-DAT can replay that session in another browser. One way to locate the desired URL is to follow a link from the MetaPlaybook timeline. The query language allows P-DAT users to find specific Playbook interactions in recordings, or in segments of a recording.

5. Usage

We envision P-DAT as a tool for researchers to rely on when evaluating, after a mission, how the crew used Playbook. The crew's experience with Playbook has implications for research questions relating to the larger ideas of crew autonomy and how a minimally-sized crew traveling into deep space will operate outside of contact with mission control on Earth.

Already, P-DAT has been used internally by the team on a regular basis for tasks such as usability diagnoses and software debugging. As Playbook is deployed on various analog missions, like the ones mentioned earlier, the team regularly uses the recordings to verify different usability issues seen in Playbook during mission operations. One particular example: during a NEEMO 21 deployment, members of the Playbook team observed crew members experiencing an issue with a new "add activity" feature in Playbook. Because the Playbook team could not physically be with the crew, who were in an underwater habitat, the team could only vaguely watch the issue unfold via video cameras and screen-sharing software from mission control on the surface. Post-mission, we were able to find the recording of the issue that the crew experienced and view it repeatedly, allowing us to formulate a better understanding of the problem. This insight led to the development of another new feature in the Playbook tool that will potentially see deployment in NEEMO 22.

As another example, Playbook was recently deployed on the International Space Station under the Crew Autonomous Self-Scheduling Test (CAST) research experiment during Expeditions 50 and 51. A crew member using Playbook in space reported a connectivity problem that might have prevented the astronaut's last few changes from taking effect or being sent back to Earth. Because of the way data is delivered to and from ISS, the team on Earth could not immediately validate what the crew member saw or determine if it was an unknown bug or an artifact of the network problem. Meanwhile, on Earth, we observed a certain activity incorrectly disappearing and appearing at the beginning of the previous day. The crew member didn't report seeing anything amiss with that activity, and we wondered whether the disappearance was visible onboard and related to the reported problem. We have no opportunity to directly speak to busy users in space to ask what they see on their screens; voice communications have to be relayed through at least two flight controllers. P-DAT was later utilized as a crucial tool in diagnosing and solving the problem. We were able to locate recordings of the Playbook sessions running at the time and determine that the bug we observed on the ground was not visible onboard.

Playbook was deployed in four 30-day HERA missions in 2016. In each mission, the four analog crew members were given a series of self-scheduling (planning and rescheduling) exercises to complete in Playbook. The exercises were varied, from simple rescheduling to more complex, highly-constrained planning. From these missions alone, we accumulated over a hundred unique session recordings, most of which are still being processed and analyzed with the help of P-DAT. Without P-DAT, tracking the number of instances of self-scheduling events for several 30-day missions would have been tedious and complicated. We can leverage P-DAT to not only track the number of times rescheduling occurred throughout the mission, but also the times of day these events happened. We plan to apply this same analysis in the Hawai‘i Space Exploration Analog and Simulation (HI-SEAS), an 8-month Mars analog mission. With P-DAT, we are able to collect detailed Playbook user data with little effort from long-duration missions.

6. Future Work

We currently have the capability to analyze one recorded session at a time. Yet we have collected hundreds of hours of data in thousands of separate recordings. The main tool we have so far that can analyze more than one recording is MetaPlaybook (see §3.2). More sophisticated features, applying the query language to multiple recordings, may include:

- Searching for rare events of interest and playing the corresponding sections of the recordings.
- Creating a montage that replays all of the occasions that match a given query, to help a researcher discover new insights into how a particular feature was used or common task was performed.
- Creating graphs using proposed extended queries such as
    - `For mission NEEMO 21, plot the duration of mode=edit as a function of time`
    - `For mission HERA, plot the duration between drag and drop as a function of time`
    - `For mission HI-SEAS, make a bar chart of the number of times mode=edit per day`
- Generating charts to summarize answers to questions like “After the Self-Scheduling button is pressed, how often is it followed by the Done button and how often by the Cancel button?”

7. Conclusion

P-DAT has demonstrated the capability of discreetly capturing usability data in a manner that is transparent to Playbook’s end-users. In our experience, P-DAT data has already shown its utility, revealing potential usability patterns, helping diagnose software bugs, and identifying metrics and events that are pertinent to Playbook usage as well as spaceflight operations. As we continue to develop this analysis tool, P-DAT may yet provide a method for long-duration, unobtrusive human performance collection and evaluation for mission controllers back on Earth and researchers investigating the effects and mitigations related to future human spaceflight performance.

References

1. Marquez, J.J., Pyrzak, G., Hashemi, S., McMillin, K., Medwid, J., et al.: Supporting Real-Time Operations and Execution through Timeline and Scheduling Aids. 43rd International Conference on Environmental Systems, Vail Colorado (2013)
2. Hashemi, S., Hillenius, S. @NASA: The User Experience of a Space Station. Speech at SXSW Interactive, Austin, Texas. (2013)
3. Marquez, J.J., Hillenius, S., Deliz, I., Kanefsky, B., Zheng, J., Reagan, M.: Increasing Crew Autonomy for Long Duration Exploration Missions: Self-Scheduling. IEEE Aerospace Conference, Big Sky, Montana (2017)
4. Martin, B., Hanington, B. M. Universal methods of design: 100 ways to research complex problems, develop innovative ideas, and design effective solutions. Beverly, Massachusetts: Rockport (2012)

Acknowledgements

This work is funded by the NASA Human Research Program as a Directed Research Project within Human Factors and Behavioral Performance.