Redesigning Olin’s Room Reservation Software

Problem Statement
As a small school with limited space resource, Olin requires a room reservation system to ensure that meetings are not double-scheduled in the same location. Olin’s current room scheduling software is clunky, unintuitive, and downright horrible. It appears as though it is designed for a much bigger institution where people don’t know about the possible rooms in which they could schedule their meeting. The reservation process is inflexible, and does not allow multiple sensible paths towards room reservation (i.e.: picking a time first or picking the room first). The order in which data must be entered seems arbitrary and the quantity of required data is excessive. Further, there is no apparent notification or confirmation process after a room is reserved. In addition to difficulties with room scheduling, the interface for gaining information about any particular room could certainly use improvement, since the presentation of information is unintuitive and leads to a dead-end (i.e.: once a desirable room is found, one cannot progress to reserving it from that location). Finally, though some aspects of display of the campus’s schedule of events in the software are well designed, these features are buried under difficult-to-find drop-down menus. We hope to create a system that intuitively caters to the needs and conceptual models of the many users on a small college campus.

Solution Overview
We decided to implement a graphical/textual non-linear room scheduling interface. The primary features of this interface that are what we believe to be improvements on Ad Astra are its tabbed-based navigation, non-linearity of room reservation, ability to cancel or reschedule an event, and graphical searching and scheduling. Navigation through the reservation process is straightforward and intuitive, using tabs as links from page to page. This provides a means for the user to quickly search for rooms that are appealing for a desired event, or pick rooms based on building and location in that building. Further, in the “specify a time” tab, it allows the user to either using buttons to enter a desired time, or “click and drag” on the room’s schedule to choose a desired time slot.

We are also making a couple of key assumptions for our prototype. First is that the system is linked to the school’s login system, so that contact information about the current user can be transferred into the program via login. We are also assuming that this provides all functionality to work with the back-end users, and that the system will have an up-to-date database of all schedulable rooms available for reservation. Finally, we assume that conflicts that come about post-room reservation can be resolved on the back end, and that an individual can be notified, by email, that his or her reservation needs to be changed (which is something that couldn’t happen before) to accommodate someone higher up in the “chain of command” at Olin.

Persona
Based on our interviews, we chose to create three personas to represent the three classifications of users we observed. We noticed a distinct difference between those who had and those who had not used the school’s current room scheduling software. In addition, of those who had not yet used it (or used it very infrequently), there was another distinct difference between those who were power computer users, and those who were not. Interviewees who were admittedly strong computer users noticed many nuances of the software, and were much better at articulating what annoyed them and why it annoyed them (and sometimes even how to improve it). Those who were not as experienced and knowledgeable about computers and programming were frustrated by some aspects of the software, but simply accepted some of the software’s quirky features.
Both of these user types knew what they wanted, but those with vast computing experience were much pickier about features and intuitive interface than those who were not. This distinction created Larry and Mark. Participants who fell into the third user type were very familiar with the school’s current software, since they were incredibly involved in school activities and had reserved many rooms in the past. They knew how the software worked, and simply breezed through the tasks we asked them to perform, knowing exactly where to click to accomplish any given task. They had already dismissed and were no longer bothered by some of the nuances of the software that annoyed inexperienced users. Further, they were also more likely to use features of the program with which they were familiar, despite the fact that other features probably would have led them to task completion sooner. This third distinction resulted in the creation of Ann.

Persona: Modest Mark

Eighteen-year-old Mark is a new student at Olin. He has spent most of his life in his native Hawaii, but attended a technical high school in California. During his first year of college he is taking mostly the required intro courses (ICB’s etc.), but he was able to cross-register for an Asian Philosophy course at Wellesley. Mark favors mind-expanding activities; his hobbies include playing go, playing board games and puzzle games with the game club and flexing his culinary muscle with the cooking club. To really relax, he will often watch comedies in his floor lounge on weekends. As of yet, he has not assumed any leadership roles in student organizations, but he hopes to take on more responsibilities once he’s better organized and better adapted to Olin life.

Some of his goals include:

- Be able to reserve a room that serves the needs of his organized activities.
- Accomplish tasks on his own, without asking others for too much help.
- Not feel stupid or inadequate in his skills or knowledge.
- Become more organized and adapted to Olin life.
- Become a bigger part of Olin’s community.
Persona: Linux Larry

Linux Larry is a dedicated Computer Science TA at Olin College that loves his job and loves the Olin community. He is very much into open source software, and is currently teaching a Co-Curricular activity in open-source software directed towards Linux users. He has very strong opinions, especially regarding computers, electronics, and programming, and gets rather openly upset at poorly programmed or incompatible software and GUIs. He is very much a gadget person, and owns a PDA, Blackberry, and Tablet PC in addition to his laptop computer. Time efficiency is critical to him. His schedule is very organized, available to anyone who wants to see it, and also linked to his phone and PDA. He depends on his gadgets to keep him up to date as to when things are happening, and does pretty much everything electronically; he hates to print. At home in his apartment, his desk and physical space aren’t quite as organized as his schedule, but that doesn’t bother him too much. He loves to play WoW and FPS games and can even sometimes be found joining Friendly Fire during one of their Friday night events. In addition to computers, he loves west coast swing dancing, and shooting it up on the basketball court.

Some of his goals include:

- Schedule a recurring meeting for his co-curricular activity quickly and efficiently.
- Operate and navigate software interfaces his way, on his own concept model, without difficulties.
- Eventually have his entire life synched with his electronic gear so that he doesn’t have to worry about remembering his many meetings.
- Become a Computer Science professor at a small college, where he can spread the joy of open-source software.
- Wants to rid himself of all unintuitive, poorly programmed software mainly by boycotting all software he doesn’t like
Persona: Assiduous Ann

Assiduous Ann is a junior planning to earn a degree in general engineering at Olin College. Like most of her peers, she is trying to complete her degree requirements by taking Partial Differential Equations, Bio-materials, Sustainable Design and Japanese. However, unlike her peers, Ann does not want to pursue a career in engineering once she gets her degree. Instead, she wants to go to law school after Olin to become a public defender. Since Ann feels that being a public defender demands strong organization skills and exceptional leadership skills, she, to hone up these skills as well as to serve the Olin community, is currently the president of the martial arts club and the junior class representative. In addition, Ann has an eclectic mix of hobbies and interests. Particularly, she is interested in Japanese culture. Beside being an active member of the Japanese club, Ann is a Japanese film and anime buff. Furthermore, Ann is interested in theater and plans to audition for the next FWOP production. While all of these activities make Ann love her college experience, they also require that Ann be efficient with time. She openly admits, trying to balance her classes and her activities is hard.

Some of her goals include:

- Be able to schedule meetings in various locations and rooms quickly and efficiently.
- Needs quick responses from the system so she can let others know when and where the meetings are.
- Not make any embarrassing mistakes that will affect others.
- Become extremely organized and efficient with time.
- Become a public defender to serve the community.
Reserving a Recurring Event

After receiving an e-mail from the registrar that his co-curricular activity had 8 people register for it, Larry would like to schedule a room this activity every week for the rest of the semester. First, he goes to the school's room scheduling software and logs in. In looking for a room, he wants to foster a closer environment for the participants, and he feels that a large classroom will be too distracting. Not knowing exactly which room he wants, he scans through pictures of the rooms in the academic center until he finds one of the small end-of-the-hall rooms that is intended for about 10 people. Having selected this room as the one he wants, the then chooses the day of the week (Wednesday) and the time of day (8-9 PM) that his activity will take place. He then chooses to make this a recurring event, and indicates that the event will recur weekly, on Wednesdays, until the end of the semester. He quickly double-checks that the information he entered is correct (desired room, day of week, time of day, recurrence), then enters in his contact information, and also indicates that the event is a Co-Curricular Activity in Open-Source Computing. Finally, he submits his request for the room, and gets on with the rest of the work he has to do for the day.

Changing Reservation

While viewing her schedule in outlook for the next week, Ann receives an email from Kim, the Dean of Faculty's assistant. It seems there is a potential conflict for Ann's martial arts club next week - the room Ann is using for practice will be used for an outside visitor's reception. She decides that she wants to move her meeting to the following day. After login into the room scheduling software, she quickly views her list of reservations. After finding the reservation for the martial arts club, she edits the reservation information and confirms the changes. The updated schedule is sent to her club members.

Reserving a Room

Modest Mark has just been elected VP of the newly-formed debate club. As one of his first official duties, he is being asked to help arrange a conference with the heads of other local college debate clubs so that they can plan for a regional tournament. Mark's time commitment to the club is as yet uncertain, and he is concerned about doing a good job with what time he has available. Scheduling the room is something he knows he can figure out. He hopes it will be easy, and not too time-consuming, but above all he wants to be sure he's done it properly. He logs into the site, selects the "Crescent Room" (which the club intends to use for their conference), and after selecting the day of the event, he is presented with a view of the room's usage on that day. Once he has determined availability, he reserves the room for the desired open timeslot and sends the reservation information to the president of his club.
Final Interface Design

Our prototype attempts to streamline the non-linear room scheduling process, while still including both a textual and graphical means of accomplishing most tasks.

Prototype Walkthrough

After login (which is tied to Olin’s windows login so that user contact information can be attained without requiring additional input), the user starts to create a new reservation, indicated by the mode icon (Figure 1). This is default mode since most users use this software to schedule for new events. The user is on “Find a Room Page”. This is the default view since searching for a room before entering event information fit the conceptual model of all of our initial interviewees.

On the “Pick a Room” screen, the user can search for a room by graphically search by clicking a building on an image of a campus map (see Figure 2a). In addition, the user can use a criteria based “filter” on the left (see Figure 2b). If a specific room is not specified in the initial (textual or graphical) search, a list of all matching rooms is displayed on the same screen with the ability to select any number of rooms via checkboxes next to the room descriptions (which include picture and textual data). In addition, floor plans of the buildings that contain these rooms are displayed, in case the user wants to pick a room by his or her memory of that room’s location in a particular building (which is not uncommon for Olin community members).

If the user wishes to enter event information before choosing a room, he or she can click the “Specify a Time” tab. This link will take the user to a page that requests an event name, asks if the event will be private (via a check box), allows the user to enter in the start time and duration or end time of the event by drag and drop or by clicking on buttons and allows them to specify recurrence. At any time, the user can click on a “Pick a Room” button that returns them to the above mentioned screen. Any data entered on this page will be taken into account in room
selection, and potential conflicts will be highlighted to avoid double-scheduling a room.

Once a room is selected, the user is taken to a room calendar screen, which provides the room’s schedule for a given day. This day can be specified either on this page or previously in the event information page. The schedule is designed to be very similar to Outlook’s schedule grid, and displays all other meetings that have reserved that room in blue (with the meeting/event name if it is not private), a conflict in red (if entered data conflicts with an already scheduled event), or the current user’s reservation in green (if entered date does not conflict with another scheduled event). Here, the user is presented with the option of entering or editing his or her event time and date; and this data can be input either by clicking on buttons or graphically by clicking and dragging on the schedule grid or calendar, or typing the text in fields on the right side of the screen. Once all the information is valid, the user can click a link on the bottom of the page that asks them to review their information prior to submission.

The final confirmation screen displays all entered and relevant data. In addition, the user can define people to which the user wishes to send e-mail notifications. If any data are missing, the software will mark the missing fields and the user will not be able to continue until the missing information is entered (Figure 5). After the user submit the request. The system provides a dialog box that informs that their request has been submitted (Figure 6). The user can either create another new event by clicking on continue, or view the current reservation in the system by clicking on My Reservations.

In addition to the above described process, the user always has the option to review his or her reservations. He or she can click the “My Reservations” Mode Button toward the top of the screen to view the reservations that he or she has made (see Figure 6). The user can then choose to edit those reservations (and will be taken to the reserved room’s calendar screen), or even cancel a reservation outright. This feature is important, since meetings at Olin are frequently changed or relocated for various unexpected reasons.
Below is an graphic overview of the system:

![Interaction Flow Diagram](image)

**Figure 7: Interaction flow**

**Functionality**

For the limited range of data it uses, our current prototype is about 95% functional. It does all the filtering steps involved in selecting a room and, although it allows only one selection at a time, it uses the data from that room consistently when scheduling and confirming the reservation. It is able to inform users of a scheduling conflict; it also informs them if they have not included sufficient information about their event to reserve the room. Drag-and-drop scheduling is available, and recurrence, though not fully featured, can be demonstrated.

For editing reservations, the essential steps of the task appear exactly as they should. The difference is that the My Reservations page is static, so edits and deletions do not actually change its content.

You can also log in with whatever name-password combination you want, but the system will pretend that you are Michael Wu. All rooms are on the same schedule, so nothing is available at 11-12 or 1-2. You also can’t specify recurrence parameters. We feel that what’s left is quite sufficient to demonstrate the workings of the site as applies to our core tasks, however.

**Tools We Used**

The primary format of the prototype pages was standard HTML with CSS styles and limited JavaScript. Whereas the first version was static, however, significant PHP server scripting was involved in the second and third prototypes. This enabled us to make data consistent between pages and resulted in richer and more realistic interactions. It also made it considerably easier to
extend the list of rooms on campus, giving a better cross-section of what’s on Ad Astra. The basic content was written using Macromedia Dreamweaver, while the PHP was written in using only a text editor and Mozilla Firefox (for testing and debugging).

While these are both excellent ways to build web content, they sometimes came into conflict. Websites written to view well in Firefox (as ours was) don’t always display correctly in Dreamweaver’s design browser. Moreover, the replacement of static HTML components with server-scripted components (which Dreamweaver does not compile) meant that some of the content could only be edited by changing the PHP source code directly. This proved to be limiting, since we each have different talents, and PHP, simple as it is, does involve a learning curve. Also, changes to the HTML content might adversely affect the way that the PHP content is displayed, and vice versa. Our solution to this was to plan our changes jointly, and decide who should implement which parts and in what language. By doing so, we avoided about 99% of conflicts, and achieved consistently good aesthetics.

**Design Evolution**

Our team’s interface underwent a series of drastic changes during the initial design phases, and then some more subtle, but still important changes towards the end of the project. An analysis of the changes of each aspect of the room scheduling system follows:

**Overall Design: Non-Linearity**

Our initial goal with the room scheduling software was to create a completely non-linear system. Thus, users could enter in whichever data they wanted whenever they wanted, and reach the end goal through the path that they wanted to take. One of our initial designs involved a simple display of all of the data input necessary to reserve a room, the team felt as though this completely nonlinear approach was not user-friendly, and would overwhelm beginning users. However, from preliminary interviews, the team discerned that most people prefer to pick a room prior to picking a time. Thus, we decided that, by presenting the user with the room select option first, but still providing the ability for non-linearity, we would both accomplish one of our primary goals of a non-linear system without overwhelming the user. Thus, we decided to divide up the interface by functionality (room select, info entering, confirm, etc…), and present the users with links to allow them to freely navigate between data information screens. However, we found that the links were not quite obvious enough, and that beginning users really wanted the guidance of a linear system. Thus, we adapted to change the links to more obvious tabs, and include “continue” buttons (that did nothing but link to the next page in the linear process most commonly used by preliminary users) at the bottom of every information entry screen. This semi-nonlinear design was the one we ended up using in our final prototype. Basically, it guides users who feel they need the guidance, without inhibiting advanced users from taking the path that best suits them.

Further, from our paper prototype, we noticed that we had no way of indicating modes. The user had no idea whether they were in the “create a new reservation” mode or the “edit a past reservation” mode. Thus, we implemented an indicator for this via color and text next to our tabs.

Figure 1. The team’s initial idea for non-linearity, the status bar.
These changes are best illustrated by the change from our status bar (which was also eliminated because users really didn’t care to have all of the information they had entered continually presented to them) with links to different pages to the tabs that can be found atop our current prototype.

Figure 2. Tabbed navigation allows the user to quickly and easily switch between room and time edit. Further the colored section on the left will indicate whether the user is in “Create New Event” or “Edit a Reservation” mode.

Room Selection Screen:

The room selection screen actually didn’t go through many revisions, primarily because it was developed directly out of recommendations from our users. The biggest change in the screen was from a text-based “search” for rooms to a criteria based “filter.” This change was made because none of the users trusted that the system would be able to properly interpret the keyword for which they wanted to search. All other features, from the check-box selection of specific rooms to the drop-down menu selection of building and room number, to the graphical search method on the right remained almost identical. A comparison of the low-fidelity to the high-fidelity system can be seen below.

Figure 3. A comparison of the team’s paper prototype (left) and final prototype (right) for the room selection screen.

Event Information/Scheduling Screen:

The largest change that happened on this screen was that it became only one screen. In our paper prototype, we had divided up the screen in which the user could enter event information from the screen in which the user compared their event information to the schedule of the room that he/she had chosen. However, we found that the former screen was relatively useless, especially if we included all of its functionality on the later. Thus, after our paper prototype, we eliminated the Event Information screen.
Within the series of three prototypes for the Event Information/Room Scheduling screen, a few other changes were made. Initially, we had developed the screen with 24-hour time, for ease of programming. This was not very well received, and frequently users didn’t even notice that they were supposed to use 24-hour time. Returning to the more familiar 12-hour time was move to top priority, and changed for our final prototype. We also added functionality that informs the user that a room has not been selected (assuming this is the case), and provides a link back to the room selection screen, so they can more easily figure out how to “go back” once they have entered event information. This feature improves the effectiveness of the non-linear system, since it helps the user remain comfortable being on a screen without enough information to take advantage of its full usefulness. Some of these feature changes are highlighted below.

Figure 4. Changes from our paper prototype (left) to the final prototype (right) in the Room Schedule/Event Information screen. Features specifically implemented for the final prototype are circled in red.

Confirmation Screen:

The confirmation screen went through a large series of revisions throughout the prototypes. Initially, the confirmation screen was intended to be an area in which users could edit their reservation information, and send an e-mail to any other people that might want to be informed about the event being scheduled. However, from our paper prototype, we realized that users rarely actually used the editable features of this page (since they were happy with their selections), and one user actually pointed out that he would just skip to this page in the future, bypassing the rest of the system. This would not be ideal, since they would be losing the functionality of the room reservation system that makes it important – avoiding scheduling conflicts. Thus, we decided to eliminate any ability for the user to edit their event information on the confirmation screen, and only implemented the e-mail functionality for the high-fidelity prototypes.

During the heuristic evaluations and high-fidelity prototype testing, we received a good amount of feedback on this page, in particular. Users were confused by the “optional” e-mail capabilities, especially that the current user’s e-mail address was already filled into the text field. However, part of the idea of incorporating e-mail into the system was to ensure that the primary user knew
that he/she would receive a confirmation e-mail. Thus, instead of making the e-mail feature optional, we implemented a system in which an e-mail to the user is required, and the user can input other email addresses into the “cc:” field. This made it much more clear that the user must receive an email (the confirmation), and that any other emails were optional. Finally, for the third prototype, we implemented clear warnings for this system to notify the user that information that they had entered was invalid. Without valid information, the user cannot proceed to reserve the event. These features are highlighted in the figure below.

Figure 5. Changes from our paper prototype (left) to the final prototype (right) in the Confirmation screen. Features implemented for the final prototype from reviews of earlier iterations of high-fidelity prototypes are circled in red.

Color:

Probably the largest change as a result of heuristic evaluation was the use of color in our high-fidelity prototype. For our paper prototype, we had used color so that we easily differentiate screens, and make the process go faster. Since we kind of liked the feel of this, we implemented something similar for the first high-fidelity prototype – lots of color to differentiate tasks. However, during the heuristic evaluations, this use of color got poor feedback. It was recommended that we reduce the number of colors used, since it was very distracting to the user. To do this, we first changed the tone of the static bar at the top of the screen from a cold blue to a warm maroon, to help the user feel more comfortable in the system. We also toned down unnecessary colors to shades of gray. Finally, we took advantage of these color changes to further highlight the modal properties of the software. Everything in the “create a new event” mode was changed to a blue shade, while everything in the “edit a reservation” mode was changed to orange. This made a significant difference in the comfort users seemed to have with the prototype. The figure below depicts such changes.
Most Valuable Phase:

It is very clear to us that the most valuable stage of this process was the paper prototype. Most of our major feedback came from that phase, and once we had created the high-fidelity prototype from that feedback, we rarely ever received major criticisms about interface design (other than those that came from features that were not yet implemented). The paper prototype allowed us to really consider the intricacies of the design, and test out some of the creative and new features that we thought would be helpful. While not all of these features were helpful, we were able to evolve them into functionality that was useful for the user, and then tweak that functionality in the high-fidelity prototypes. Things like changing from sidebar navigation to tabbed navigation, and clarity in modes were critical realizations that occurred because of our paper prototypes, and are features that truly make this system user-friendly.