Requirements Document for UniCal

1. Introduction

Since the opening of the campus, the University of California, Irvine (UCI) has held many academic and non-academic events for its many students, faculty, staff, and visitors. Until now, UCI has informed people of these events using a haphazard collection of e-mails, web pages, flyers, and word of mouth. However, as the campus grows, it becomes increasingly more difficult to inform people of the many different UCI events. UCI now wishes to supplement this process by creating a university calendar system, called UniCal, to inform interested parties of campus events.

Because UCI is a research oriented institution, it does not want to burden its professors or students with production oriented software engineering projects. Consequently, it has hired Retro Software Inc. to develop this requirements specification for UniCal, as well as its subsequent design and implementation. The information in this requirements specification is based on interviews conducted with the relevant parties at UCI, by the requirements engineers at Retro Software Inc. All information in this document was determined to be accurate at the time of this writing.

This document contains the detailed requirements specification for UniCal that Retro Software Inc. is developing for UCI. The document should serve as the official basis for any further development of UniCal.

This document contains the following sections:

- Executive Summary
- Application Context
- Functional Requirements
- Environmental Requirements
- Software Qualities
- Other Requirements
- Time Schedule
- Potential Risks
- Future Changes
- Glossary
- References

2. Executive Summary

UniCal aims to provide a solution for scheduling campus events and publicizing their information to UCI faculty, staff, students, and visitors. The application allows customers to schedule, share, and view events via any computer running UniCal. UniCal facilitates

communication by unifying and providing easy access to event information. Changes in events will be updated to users' schedules so that everybody is well informed. By maintaining consistent and reliable event information, UniCal enables effective event arrangement and collaboration among users. Retro Software anticipates that the overall effect of adopting UniCal will result in more accurate, efficient and convenient schedule coordinating among UCI personnel.

UniCal provides the following key features:

- **Scheduling and sharing events** - A user can schedule events and group them into specific categories using UniCal. These events can be seen by other users of the system.
- **Viewing events** - A user can view all of the events in their calendar in either a weekly or monthly view.
- **Event reminders** - UniCal will provide reminders for a user by playing a sound and displaying a pop-up window with details of the event at a customizable time interval before each event.
- **Task list** - Users can also create a task list with deadlines and reminders to help them meet goals.

Some of the most important risks posed by the development of UniCal are the following:

- **Lack of adoption** - Because there are many other calendaring systems currently on the market, it is possible that students, faculty, and staff will not adopt UniCal into their normal routine. This would effectively render the system as no more useful than current approaches to informing people of events.
- **Usability** – To help encourage adoption of the system, UniCal must be extremely intuitive and easy to use.
- **Rapid development** – The schedule according to which UniCal will be developed is extremely aggressive to ensure that another company will not develop a similar product first and claim the market. Other qualities cannot be sacrificed as a result of this rapid development.

Although the contents of this document were thoroughly verified, it may still occur that some requirements are inconclusive or ambiguous. If it is so determined, it is requested that Retro Software Inc. be contacted via e-mail at shendric@uci.edu to resolve the issue.

3. Application Context

UniCal will be used in the institutional environment, UC, Irvine, where scheduling and coordinating events can be difficult because of the large numbers of people involved. The introduction of UniCal should reduce the effort required to coordinate schedules among multiple people by automating much of the effort of keeping everyone's schedule up to date. Adopting UniCal will result in substantial change for UCI. It will require that the application be installed on each person's computer, and that the application be used as the primary means of scheduling and coordinating events on campus.

Users of UniCal fall into two different roles: administrators, and users: Administrators of UniCal have all of the same functionality as a user, but in addition are responsible for managing the different accounts of the system. They may create, delete, and edit user accounts, as well as other administrator accounts. Should users forget their password, it is the responsibility of an administrator to reset their password so that they may once again gain access to the system. Should an administrator forget their password, they may have the password e-mailed to the e-mail address associated with their account.

Users of the UniCal system use the system to schedule and share events. Users may be faculty, staff, students, or even offices who wish to publicize office events. Users may create events, schedule reminders for events, specify recurrences for events, and group events together into categories. The categories that users create may be shared with other users of the system and included on their calendars. Users may also maintain a private task list of tasks that need to be completed.

It is the hope of UCI that UniCal will eventually be used by other campuses. Such a change would require the application to be highly scalable and likely require instances of the UniCal system to interoperate since, due to political issues, it is unlikely that all campuses will share one instance.

4. Functional Requirements

4.1. ADMINISTRATOR FUNCTIONALITY

4.1.1. Administrators have the following functionality in addition to regular users (see 4.2) of the system

4.1.2. Login

4.1.2.1. The administrator should log in to UniCal using their username and password

4.1.2.2. The administrator password must be changed the first time the administrator logs in to the system

4.1.2.3. There should be one default administrator account with the username and password of ‘admin’ and ‘admin’ respectively. This account cannot be deleted

4.1.3. Adding/deleting/editing administrator accounts

4.1.3.1. An administrator should be able to add, delete, or edit other administrator accounts through the user interface

4.1.3.2. An administrator can enter the following information for each administrator. All information must be entered for an administrator account to be added to the system

4.1.3.2.1. **Username:** must be between 5 and 20 characters and should consist only of upper case, lower case, and numeric characters

4.1.3.2.2. **Password:** must be between 5 and 20 characters and should consist only of upper case, lower case, numeric, and punctuation characters

4.1.3.2.3. **Full Name:** must be less than 100 characters

4.1.3.2.4. **E-mail:** must conform to section 3.4 of RFC 2822 (http://www.ietf.org/rfc/rfc2822.txt?number=2822).

4.1.3.3. The administrator should be able to edit and save their password, full name, and e-mail address

4.1.3.4. An administrator should be able to reset the password of administrator accounts that they created or that were directly or indirectly created by accounts that they created

4.1.3.5. An administrator should be able to delete only administrator accounts that they created or that were directly or indirectly created by accounts that they created

4.1.4. Adding/deleting/editing user accounts

4.1.4.1. An administrator should be able to add, delete, or edit all user accounts through the user interface

4.1.4.2. An administrator can enter the following information for each user. All information must be entered for a user account to be added to the system

4.1.4.2.1. **Username:** see 4.1.3.2.1

4.1.4.2.2. **Password:** see 4.1.3.2.2

4.1.4.2.3. **Full Name:** must be less than 100 characters

4.1.4.3. All usernames in the system must be unique at all times

4.1.4.4. An administrator should be able to reset a user's password when requested by a user. However, an administrator should not be able to see any users current password

4.1.5. User Interface

4.1.5.1. UCI has few restrictions on the user interface for administrator functionality except that it should be functional and user friendly. The layout of the user interface for administrator functionality is left to the designers

4.2. USER FUNCTIONALITY

4.2.1. Login

4.2.1.1. A user logs in to UniCal using a username and password (see 4.1.4.2.1, 4.1.4.2.2)

4.2.1.2. The user password must be changed the first time a user logs in to the system

4.2.2. Adding/deleting/editing categories

4.2.2.1. A user should be able to add, delete, or edit categories through the user interface

4.2.2.2. A user can enter the following information for each category. All information must be entered for a category to be added to the users account

4.2.2.2.1. **Name:** a single line description used to identify the category (50 characters max)

4.2.2.2.2. **Color:** the color in which all events in the category are displayed

4.2.2.2.3. **Description:** a multi-line description of the category

4.2.2.3. All category names for a given user must be unique at all times.

4.2.2.3.1. If a change in the system will cause a category name to be identical to another category name, then the user should be notified of this and given an opportunity to change the name before continuing with the function

4.2.2.4. Once a category is created, events (see 4.2.5) may be added to the category

4.2.3. Importing categories

4.2.3.1. A user may include another users category (and the events therein) in their calendar by selecting it from a list of all user categories stored on UniCal

4.2.3.2. A user should be able to use a local name and color for an imported category, and set a local reminder (see 4.2.5.2.8) for each event (see 4.2.5) in an imported category

4.2.4. Updating

4.2.4.1. A user should be allowed to update their calendar so that their categories are available for others to import (see 4.2.3), and a user's imported categories are updated with changes made to the events in the imported category

4.2.4.2. Users may edit their calendar while not connected to the internet

4.2.5. Adding/deleting/editing events

4.2.5.1. A user should be able to add, delete, or edit events through the user interface

4.2.5.2. A user can enter the following information for each event. All information must entered for the event to be added to a category

4.2.5.2.1. **Name:** a single line description used to identify the event (50 characters max)

4.2.5.2.2. **Start time:** the starting date and time of the event formatted as (hours)/(minutes) (am or pm) (month)/(day)/(year) using two digit numbers except for hour and month, and four digit number for the year

4.2.5.2.3. **End time:** the starting time of the event formatted as (hours)/(minutes) (am or pm) (month)/(day)/(year) using two digit numbers except for hour and month, and four digit number for the year

4.2.5.2.4. **Location:** the location of the event (using previously entered values or entering a new value)

4.2.5.2.5. **Description:** a description of the event

4.2.5.2.6. **Recurrence:** the recurrence for the event set with respect to the start time of the event (see 4.2.7)

4.2.5.2.7. **Category:** the category that contains the event (see 4.2.2)

4.2.5.2.8. **Reminder:** the reminder for the event set with respect to the start time of the event (see 4.2.8)

4.2.5.3. An event must belong to exactly one category

4.2.5.4. The end time of the event must occur after the start time of the event

4.2.5.5. The start and end time may have a minimum granularity of 1 minute

4.2.5.6. Event names do not need to be unique

4.2.6. Adding/deleting/editing tasks

4.2.6.1. A user should be able to add, delete, and edit tasks through the user interface

4.2.6.2. A user can enter the following information for each task. All information must entered for the task to be added

4.2.6.2.1. **Name:** a single line description used to identify the task (50 characters max)

4.2.6.2.2. **Description:** a description of the task

4.2.6.2.3. **Due:** the date and time the task is due formatted as (hours)/(minutes) (am or pm) (month)/(day)/(year) using two digit numbers except for hour and month, and four digit number for the year

4.2.6.2.4. **Reminder:** the reminder for the task set with respect to when the task is due (see 4.2.8)

4.2.7. Setting a recurrence

4.2.7.1. When setting a recurrence, a user should be able to select any of the following recurrence settings provided by the user interface

4.2.7.1.1. **No recurrence:** the event only occurs once at the specified time

4.2.7.1.2. **Weekly recurrence:** the user may specify which days of the week for which the event reoccurs each week

4.2.7.1.3. **Monthly recurrence:** the user may specify which day of the month for which the event reoccurs each month

4.2.7.1.4. **Yearly recurrence:** the user may specify the day of the year for which the event reoccurs each year

4.2.7.2. The following functionality is required for events with recurrences

4.2.7.2.1. The recurrences of an event should start on or after the start date of an event

4.2.7.2.2. The user must specify an end date after which the event will not occur

4.2.7.2.3. Where there are conflicts between an event's information and the event's recurrence, the recurrence settings should take precedence

4.2.8. Setting a reminder

4.2.8.1. When setting a reminder, a user may schedule the reminder at any of the following time intervals before the event: 0, 5, 10, 15, or 30 minutes, 1 to 12, or 18 hours, 1 to 6 days, or 1 to 2 weeks

4.2.8.2. If the reminder is set for something that has a recurrence, then the reminder will occur for each instance of the recurrence

4.2.8.3. The reminder will be triggered when the system's time and date are greater than the reminders scheduled time, and the reminder has not been dismissed

4.2.8.4. When a reminder is triggered, a popup window will be displayed with all event details and a sound will be played

4.2.8.5. Dismissing a reminder

4.2.8.5.1. When a reminder is triggered, a user may dismiss the reminder, in which case the reminder window closes

4.2.8.6. Snoozing

4.2.8.6.1. When a reminder is triggered, a user may press snooze, in which case the reminder window will close and the reminder will be reschedule to 5 minutes after the time of pressing snooze

4.2.9. User interface

4.2.9.1. Appropriate menus and toolbars should be available to access all functionality using the mouse or keyboard

4.2.9.2. There should be the following three parts to the user interface arranged from left to right: the category list (see 4.2.10), the calendar view (see 4.2.11), and the task list (see 4.2.14)

4.2.9.3. The category list should initially utilize 10% of the screen

4.2.9.4. The calendar view should initially utilize 80% of the screen

4.2.9.5. The task list should initially utilize 10% of the screen

4.2.9.6. A user should be able to resize the portion of the screen that each section utilizes

4.2.10. Category list

4.2.10.1. This displays a list of imported (see 4.2.3) and created (see 4.2.2) category names sorted alphabetically

4.2.10.2. The list should display each category name to the right of a checkbox (which indicates whether the events of the category are visible in the calendar view (see 4.2.11)). The user should be able to check or uncheck this box to include or exclude the events from the calendar view respectively.

4.2.10.3. The area behind each category name should be colored according to the category color (see 4.2.2.2.2)

4.2.10.4. The user should be able to edit category properties via a context menu for each category in the list

4.2.11. Calendar view

4.2.11.1. This view should consist of a tabbed pane which includes a tab for a monthly calendar (see 4.2.12) and a tab for a weekly calendar (see 4.2.13). The tabs are located at the top left of the view.

4.2.11.2. All calendar views should start on the same day of the week, either Sunday or Monday. The user should be able to set this in system preferences.

4.2.11.3. A user should be able to enter a date, and have any view focus in on that date.

4.2.12. Monthly calendar view

4.2.12.1. This view should display 7 days by 5 weeks using equally sized boxes to represent the scheduled events for each day.

4.2.12.2. The day of the month should be displayed in the top right corner of each box

4.2.12.3. The box representing the first of any month should contain the name of the month in the top left corner and the four digit year between the month name and the day of month

4.2.12.4. The name of each event for each day should be listed in that days box in order of the event starting time

4.2.12.5. Events that span multiple days should be included in each day that they occur

4.2.12.6. The background of each event should be colored according to the color of its respective category (see 4.2.2.2.2)

4.2.12.7. Scrollbars for a given box should be included in the box if all information cannot fit within the box

4.2.12.8. Double clicking on the number of any day should change the current view to the weekly calendar (see 4.2.13). The weekly calendar should include the day that was double clicked

4.2.12.9. The current day (according to the system clock) should be highlighted with a red border at the edge of the box representing that day

4.2.12.10. A day may be selected using the mouse or the arrow keys of the keyboard, the current day is selected by default

4.2.12.11. Page-up will select the date that is one month prior to the currently selected day, Page-down will work similarly

4.2.12.12. The selected day is highlighted with a black border (within the current day highlight (see 4.2.12.9) if necessary)

4.2.12.13. Adding an event to the calendar when in this view should use the currently selected date as the default starting and ending date for the event.

4.2.13. Weekly calendar view

4.2.13.1. This view should display a grid where each day of the week is represented by 7 columns and the time of the day is indicated along the left hand side

4.2.13.2. The date for each day should be displayed at the top of each column and formatted as (Week Day), (Month)/(Day)/(Year) using a the full week day name for the week day, a number for the month, a two digit number for the day, and a four digit number for the year

4.2.13.3. Time increments for the left hand side should be selected from 5, 10, 15, 30 minutes, and 1 hour, the default should be 30 minute increments

4.2.13.4. Each event for a given day should be displayed in the following way

4.2.13.4.1. A 25% translucent rectangle should indicate the starting and ending time and day of each event by covering the area indicated by the start date and time and end date and time of the event. The color of the rectangle should be the color indicated by the events category

4.2.13.4.2. The rectangle should span multiple days for multi-day events

4.2.13.4.3. The name of the event should appear at the top left corner of the translucent rectangle

4.2.13.4.4. Conflicting events should simply overlap

4.2.13.5. The current day (according to the system clock) should be highlighted with a red border at the edge of column representing that day

4.2.13.6. An event or time range may be selected using the mouse or keyboard, the selected event or time range is highlighted with a black border

4.2.13.7. Page-up will display the previous week, Page-down will display the next week

4.2.13.8. Adding an event to the calendar when in this view should use the currently selected date and time range as the default starting and ending date and time for the event

4.2.14. Task list

4.2.14.1. The task list should display each task in descending order of due date

4.2.14.2. A user should be able to mark tasks as completed

4.2.14.3. Tasks that are marked as completed should disappear in 24 hours

4.3. MISCELLANEOUS FUNCTIONALITY

4.3.1. Additional user interface requirements

4.3.1.1. All functionality should be accessible through menu items

4.3.1.2. All functionality should be accessible with or without the use of a mouse

4.3.2. Help

4.3.2.1. UniCal should have help documentation provided through the UI that describes each function in this document including an example of how to perform each function

4.3.2.2. UniCal help documentation should also include a short tutorial on how UniCal is used

4.3.3. Error Handling

4.3.3.1. All error handling is left to the designers of the UniCal system

4.3.3.2. All error handling should allow a user to correct the error with minimal effort, and when possible, allowing a user to continue their previous function rather than starting over

5. Environmental Requirements

Since most people on the UCI campus use either Microsoft Windows or the Macintosh OS, UCI has decided that UniCal should be able to run on both platforms.

UCI has also hired a consultant regarding programming languages, and, based on her report recommends that UniCal be implemented in Java™, to ensure portability across platforms as well as easy maintainability. Use of the JDK 1.4 is expected.

6. Other Requirements

Users should interface with UniCal through a stand alone application on their machines.

The cost of development for the UniCal system must not exceed $399,999.98. UCI has consulted with financial analysts to determine that this is the maximum amount the system can cost without becoming unprofitable.

Retro Software Inc. should carefully document all decisions made, especially decisions pertaining to changes in this document (and subsequent documents; always per agreement with UCI).

Retro Software Inc. will deliver detailed user manuals for the UniCal system.

7. Software Qualities

- **User-friendliness** – Because many of UniCal's users are expected to be people with minimal to no time to learn a complicated new system, it is imperative that the system be as user-friendly as possible.
- **Correctness** – Because UCI wants the system to be widely used, it is important that UniCal perform all of its requirements correctly and does not result in the proliferation of inaccurate or incomplete information.
- **Reliability** – Reliability of UniCal is not essential, but nonetheless important. The accepted rate of failure is one crash per month. Any failure rate above this will create an unfavorable impression of UCI towards Retro Software.

- **Performance** – Performance is crucial to UniCal – if the system is too slow, customers may revert to sending e-mails and using word of mouth. Synchronizing events with UniCal should take no more than one second for every fifty events. All other operations must be performed nearly instantaneously.
- **Reusability** – Reusability is important because it is hoped that UniCal will eventually be adopted at all other UC campuses.
- **Extensibility** – Several future enhancements for UniCal have already been proposed, and there will undoubtedly be more forthcoming. Therefore, it is critical that UniCal can be extended with relative ease.
- **Evolvability** – UniCal is expected to undergo significant changes as UCI determines which features are most important to its population. Evolvability is central to this goal.
- **Robustness** – UniCal must not crash if a user enters incorrect or invalid data, or performs some otherwise unexpected action. A reasonable response that allows the application to resume normal operation must be given.
- **Verifiability** – Although it is preferable that the system undergo formal, thorough verification and testing before deployment, this is not feasible with the rigorous time schedule desired by UCI. However, extensive testing of the accuracy of the system's functionality should be performed before release.
- **Maintainability** – UCI anticipates that UniCal will be used over long periods of time, eventually by large numbers of users. Due to this fact, the likelihood of several future enhancements, and the high probability of an equally tight time schedule for their development, it is imperative that UniCal be easily maintainable.
- **Reparability** – Because UniCal is being developed under such a tight time schedule, it is likely that testing will not be done as thoroughly as desired, and some faults will make it into the product. Therefore, the system must be easily repairable in order to fix these defects in a timely manner so as not to disrupt the business of UCI.
- **Safety** – Since UniCal is not a safety-critical application, there are no safety concerns.
- **Portability** – Even though UniCal will be implemented entirely in Java™, a highly portable language, portability will still need to be considered in the design and implementation of the system. It is UCI's desire that no features of UniCal will violate commonly accepted standards for each platform.
- **Understandability** – To support evolvability, reparability, and maintainability, it is imperative that all aspects of UniCal be easily understood, even to future developers who are not currently involved in the creation of the system.
- **Interoperability** – For this current set of requirements, the interoperability of UniCal with any other application is not needed. However, it is of prime importance that UniCal interoperate with other calendaring systems in future versions.

- **Productivity** – Because UCI would like to release the UniCal application as quickly as possible, it is desirable that the productivity of all involved in the development of UniCal be supported and facilitated with quality processes and tools as much as possible. However, limited funding is available for this requirement, so it must be foregone.
- **Size** – Hard drive space and memory are abundant nowadays – size is not an issue.
- **Scalability** – It is important that UniCal be able to scale to support the entire population of UCI, and perhaps, the UC system. Therefore, scalability is an important issue to consider in the development of UniCal.
- **Timeliness** – It is imperative that all artifacts of the UniCal system be delivered on time.
- **Visibility** – Due to the rigorous time schedule, extra time and effort to make the project status visible should be forfeited.

8. Time Schedule

The development schedule for UniCal is as follows:

- Design must be completed by **February 26th, 2004 at 2:00 PM**.
- Implementation and module testing must be completed by **March 9th, 2004 at 2:00 PM**.
- System testing must be completed by **March 18th, 2004 at 2:00 PM**, however this date may be changed.

9. Potential Risks

- **Difficult to use** – UniCal has a lot of features and addresses every kind of user. Furthermore, a significant portion of its expected user base is not computer literate. Hence, it is possible that some users might find it too difficult to use UniCal.
- **Limited schedule** – The schedule according to which UniCal is being developed is extremely aggressive and may result in a product that does not adequately satisfy the needs of UCI.
- **Lack of adequate support processes and tools** – A project of this size would ideally be supported by a number of quality software engineering processes and tools. Unfortunately, UCI lacks the funds necessary to obtain these tools.
- **Privacy issues** – As all categories will be viewable by every user, some users may feel uncomfortable letting other people see their (private) events. This could cause users to refrain from using UniCal.

10. Future Changes

- **Programmatic API for administrator functionality** – In future versions, UCI may want to be able to programmatically access administrator functionality so that the creation and removal of user accounts can be done automatically by computers processing incoming and outgoing students, staff, and faculty.
- **Web & PDA interface** – In future versions, UCI may want to create Web interfaces and PDA interfaces to supplement UniCal.
- **Permissions** – In future versions, UCI may want to introduce permissions to allow users of UniCal to keep certain events private.
- **Interface with other calendaring applications** – In future versions, UCI may want UniCal to interface with other calendaring applications. Minimally this will involve sharing event information such as through importing and exporting .vcal files (see http://www.imc.org/pdi/vcal-10.txt). However, support of more automated techniques such as scheduling meetings may also be desired.
- **Scheduling meetings** – In future versions, UCI may want to have UniCal facilitate the process of scheduling meetings, reserving rooms, and request RSVP information.

11. Glossary

- **Administrator** – One of the two rules in the UniCal System; administrators have the ability to add, delete, and modify user accounts.
- **Category** – A grouping of events into a common theme
- **Event** – A social gathering or activity that the system will help to coordinate.
- **Java™** – A highly portable, general purpose programming language developed by Sun Microsystems in the early- to mid-1990's.
- **UCI** – University of California, Irvine. The client company contracting the development of the UniCal system.
- **User** – One of the two roles in the UniCal system; users are ordinary people who wish to schedule, share, and view different university events.

12. References

- http://www.uci.edu
- http://www.java.sun.com
- http://www.dictionary.com
- http://www.imc.org/pdi/vcal-10.txt
- http://www.ietf.org/rfc/rfc2822.txt?number=2822