UniCal Architecture and Module Design

Official Design Document
Version 1.0

ICS 52
Instructor: Dr. R. N. Taylor
Winter 2004
# Table of Contents

1. Introduction ................................................................................................................................. 4  
2. Architecture .................................................................................................................................... 4  
3. Component Design ............................................................................................................................ 5  
    3.1. Calendar ADT Module .................................................................................................................. 6  
        3.1.1. Purpose .......................................................................................................................... 6  
        3.1.2. Provided Interface .......................................................................................................... 6  
        3.1.3. Required Interface ......................................................................................................... 7  
        3.1.4. Constraints ..................................................................................................................... 7  
    3.2. Alarm Module ............................................................................................................................ 8  
        3.2.1. Purpose .......................................................................................................................... 8  
        3.2.2. Provided Interface .......................................................................................................... 8  
        3.2.3. Required Interface ......................................................................................................... 8  
    3.3. Category Manager Module ........................................................................................................ 9  
        3.3.1. Purpose .......................................................................................................................... 9  
        3.3.2. Provided Interface .......................................................................................................... 9  
        3.3.3. Required Interface ......................................................................................................... 9  
    3.4. Event Manager Module ............................................................................................................. 10  
        3.4.1. Purpose .......................................................................................................................... 10  
        3.4.2. Provided Interface .......................................................................................................... 10  
        3.4.3. Required Interface ......................................................................................................... 11  
    3.5. Task Manager Module .............................................................................................................. 12  
        3.5.1. Purpose .......................................................................................................................... 12  
        3.5.2. Provided Interface .......................................................................................................... 12  
        3.5.3. Required Interface ......................................................................................................... 13  
    3.6. Calendar UI Module ................................................................................................................ 13  
        3.6.1. Purpose .......................................................................................................................... 13  
        3.6.2. Provided Interface .......................................................................................................... 14  
        3.6.3. Required Interface ......................................................................................................... 14  
4. Message Types ............................................................................................................................... 14  
    4.1. Calendar Events ....................................................................................................................... 15  
        4.1.1. EventInfo ......................................................................................................................... 15  
        4.1.2. EventRemoved ............................................................................................................... 15  
        4.1.3. AddEvent ...................................................................................................................... 15  
        4.1.4. UpdateEvent .................................................................................................................. 15  
        4.1.5. RemoveEvent .................................................................................................................. 16  
        4.1.6. EventError ...................................................................................................................... 16  
        4.1.7. EventConflicts ............................................................................................................... 16  
    4.2. Tasks ......................................................................................................................................... 16  
        4.2.1. TaskInfo ......................................................................................................................... 16  
        4.2.2. TaskRemoved .................................................................................................................. 16  
        4.2.3. AddTask .......................................................................................................................... 17  
        4.2.4. UpdateTask ..................................................................................................................... 17
1. Introduction

This is the official design document for version 1.0 of the UniCal calendaring application. It is based on a subset of the functionality in the official requirements document for the same application. Future versions of the product will incorporate the missing functionality that is not currently in this design.

This design assumes that there is only a single user; therefore there are no administrators, no login, and no sharing and importing of categories. Additionally, the UI has been simplified significantly, and there are no recurrences of events. However, you still need to ensure data integrity (see section 4.2.2.3.1 and 4.2.5.4 in the requirements document).

All questions concerning this design should be directed to Scott Hendrickson shendric@uci.edu.

2. Architecture

The architectural style chosen for the UniCal application is C2. There are many reasons for using this architectural style. Some of the key reasons are:

- It is flexible, supporting the integration of additional functionality at a later time,
- It can be easily distributed without the need to rewrite the whole application, and
- It supports dynamism allowing users to integrate new functionality while the system is running.

The architecture is presented in Figure 1. Two components, the Alarm and Calendar UI component, have been contracted out to another development team. Your job is to implement the other four components. The interfaces and behaviors of all components in the architecture are described in this document.
3. Component Design

This section describes the interfaces and behavior of each component. Figure 2 shows the diagram used for each component. Any requests sent to, or by, the component are on the left side of the diagram; notifications are on the right. The arrows pointing away from the component mean that the component will emit these requests or notifications, while the arrows pointing towards the component mean that the component receives these requests or notifications.

**Purpose**

The purpose of the component is discussed here.

**Provided Interface**

The requests and notifications that the component responds to (or cares about) are listed here along with a description of how the component is expected to respond to them. The internal format of a message is described in Section 4. For example:

*NewGame* - The request or notification name is listed here along with a brief description of what the component does in response to it.

**Required Interface**

The requests that the component sends out are listed here. For example:

*UpdateScore*
3.1. Calendar ADT Module

3.1.1. Purpose

The purpose of the Calendar ADT module is to store all the data of the UniCal calendar. Any changes to calendar data must be made through this module.

3.1.2. Provided Interface

3.1.2.1. AddCategory - adds the category to the data store. This results in a CategoryInfo notification that contains all the information of the newly added category along with a unique id assigned by this module.

3.1.2.2. UpdateCategory - updates the category information in the data store. This results in a CategoryInfo notification that contains all the information of the category before and after it was updated.

3.1.2.3. RemoveCategory - removes the category information from the data store. This results in a CategoryRemoved notification that includes all the information of the category just removed.

3.1.2.4. AddEvent - adds the event to the data store. This results in an EventInfo notification that contains all the information of the newly added event along with a unique id assigned by this module.
3.1.2.5. **UpdateEvent** - updates the event information in the data store. This results in an *EventInfo* notification that contains all the information of the event before and after it was updated.

3.1.2.6. **RemoveEvent** - removes the event information from the data store. This results in an *EventRemoved* notification that includes all the information of the event just removed.

3.1.2.7. **AddTask** - adds the task to the data store. This results in a *TaskInfo* notification that contains all the information of the newly added task along with a unique id assigned by this module.

3.1.2.8. **UpdateTask** - updates the task information in the data store. This results in a *TaskInfo* notification that contains all the information of the task before and after it was updated.

3.1.2.9. **RemoveTask** - removes the task information from the data store. This results in a *TaskRemoved* notification that includes all the information of the task just removed.

3.1.3. **Required Interface**

3.1.3.1. **CategoryInfo**

3.1.3.2. **CategoryRemoved**

3.1.3.3. **EventInfo**

3.1.3.4. **EventRemoved**

3.1.3.5. **TaskInfo**

3.1.3.6. **TaskRemoved**

3.1.4. **Constraints**

3.1.4.1. All Ids’ created must be unique. In other words, any *CategoryId* cannot be equal to any other *CategoryId*, *EventId*, or *TaskId*.

3.1.4.2. Ids may never be recycled. In other words, even after an event associated with a particular *EventId* has been removed; it may not be used again for a future event.
3.2. Alarm Module

3.2.1. Purpose
This module acts as an alarm by sending out a notification when a particular time has been passed. Multiple alarms may be set concurrently.

3.2.2. Provided Interface
3.2.2.1. SetAlarm - sets an alarm for a given time. This results in an AlarmInfo notification that contains all the information of the newly added alarm. The AlarmId used in the SetAlarm request is used in the AlarmInfo notification.

3.2.2.2. ClearAlarm - cancels a previously scheduled alarm. This results in an AlarmRemoved notification. The AlarmId used in the ClearAlarm request is used in the AlarmInfo notification.

3.2.3. Required Interface
3.2.3.1. AlarmTriggered - this notification is sent when a scheduled alarm occurs. This notification is always followed by an AlarmRemoved notification.

3.2.3.2. AlarmRemoved
3.3. Category Manager Module

3.3.1. Purpose

This module keeps track of all the categories of the calendar.

3.3.2. Provided Interface

3.3.2.1. AddCategory - if the category information is valid, then it forwards the request on. If not, it results in a CategoryError notification that contains the category information with a description of the problem.

3.3.2.2. UpdateCategory - if the category information is valid, then it forwards the request on. If not, it results in a CategoryError notification that contains the category information with a description of the problem.

3.3.2.3. RemoveCategory - Simply forwards the request.

3.3.2.4. CategoryInfo - Simply forwards the notification.

3.3.2.5. CategoryRemoved - removes all that events that were part of the category through a series of RemoveEvent requests.

3.3.2.6. EventInfo - monitored in order to remove events from a removed category

3.3.2.7. EventRemoved - monitored in order to remove events from a removed category

3.3.3. Required Interface

3.3.3.1. AddCategory

3.3.3.2. UpdateCategory
3.3.3.3. RemoveCategory
3.3.3.4. RemoveEvent
3.3.3.5. CategoryInfo
3.3.3.6. CategoryRemoved
3.3.3.7. CategoryError

### 3.4. Event Manager Module

![Event Manager Module Diagram]

#### 3.4.1. Purpose

This module keeps track of events and verifies that they are valid (i.e. they meet requirements). It also schedules and clears alarms for events that have reminders.

#### 3.4.2. Provided Interface

3.4.2.1. **AddEvent** - if the event information is valid, then it forwards the request on. If not, it results in an **EventError** notification that contains the event information with a description of the problem.

3.4.2.2. **UpdateEvent** - if the event information is valid, then it forwards the request on. If not, it results in an **EventError** notification that contains the event information with a description of the problem.

3.4.2.3. **RemoveEvent** - simply forwards the request.

3.4.2.4. **ReminderSnoozed** - schedules a new alarm using a **SetAlarm** request. The **ReminderId** is used as the **AlarmId**.

3.4.2.5. **ReminderDismissed** - ignored.
3.4.2.6. *EventInfo* - forwards the notification and sets or clears any alarms if there is a reminder. This may result in a *SetAlarm* or *ClearAlarm* request. The *EventId* is used as the *AlarmId*. This will also result in an *EventConflicts* notification with the latest conflicts.

3.4.2.7. *EventRemoved* - forwards the notification and clears any alarms if there was a reminder. This may result in a *ClearAlarm* request. The *EventId* is used as the *AlarmId*. This will also result in an *EventConflicts* notification with the latest conflicts.

3.4.2.8. *AlarmTriggered* - if the *AlarmId* refers to an *EventId*, then this results in a *ShowReminder* notification for that event and an *UpdateEvent* request to remove the reminder for the event. The *EventId* is used as the *ReminderId*.

3.4.2.9. *AlarmRemoved* - ignored.

3.4.3. **Required Interface**

3.4.3.1. *AddEvent*

3.4.3.2. *UpdateEvent*

3.4.3.3. *RemoveEvent*

3.4.3.4. *SetAlarm*

3.4.3.5. *ClearAlarm*

3.4.3.6. *EventInfo*

3.4.3.7. *EventRemoved*

3.4.3.8. *EventConflicts*

3.4.3.9. *EventError*

3.4.3.10. *ShowReminder*
3.5. Task Manager Module

3.5.1. Purpose
This module keeps track of tasks and verifies that they are valid (i.e. they meet requirements). It also schedules and clears alarms for tasks that have reminders.

3.5.2. Provided Interface
3.5.2.1. AddTask - if the task information is valid, then it forwards the request on. If not, it results in a TaskError notification that contains the task information with a description of the problem.
3.5.2.2. UpdateTask - if the task information is valid, then it forwards the request on. If not, it results in a TaskError notification that contains the task information with a description of the problem.
3.5.2.3. RemoveTask - simply forwards the request.
3.5.2.4. ReminderSnoozed - schedules a new alarm using SetAlarm. The ReminderId is used as the AlarmId.
3.5.2.5. ReminderDismissed - ignored.
3.5.2.6. TaskInfo - forwards the notification and sets or clears any alarms if there is a reminder. This may result in SetAlarm or ClearAlarm requests. The TaskId is used as the AlarmId.
3.5.2.7. TaskRemoved - forwards the notification and clears any alarms if there was a reminder. This may result in a ClearAlarm request. The TaskId is used as the AlarmId.
3.5.2.8. *AlarmTriggered* - if the *AlarmId* refers to a *TaskId*, then this results in a *ShowReminder* notification for that task and an *UpdateTask* request to clear the reminder for the task. The *TaskId* is used as the *ReminderId*.

### 3.5.3. Required Interface

3.5.3.1. *AddTask*
3.5.3.2. *UpdateTask*
3.5.3.3. *RemoveTask*
3.5.3.4. *SetAlarm*
3.5.3.5. *ClearAlarm*
3.5.3.6. *TaskInfo*
3.5.3.7. *TaskRemoved*
3.5.3.8. *TaskError*
3.5.3.9. *ShowReminder*

### 3.6. Calendar UI Module

#### 3.6.1. Purpose

This module presents the UniCal user interface to the user. The user interface displays calendar information to the user and allows the user to manipulate events, categories, and tasks.
3.6.2. Provided Interface

3.6.2.1. CategoryInfo - updates the UI with this category information.
3.6.2.2. CategoryError - prompts the user to correct the category information. This may result in an AddCategory request with the corrected information if there was no CategoryId, an UpdateCategory request if there was a CategoryId, or a RemoveCategory request if there was a CategoryId but the user canceled.
3.6.2.3. EventInfo - updates the UI with this event information.
3.6.2.4. EventError - prompts the user to correct the event information. This may result in an AddEvent request with the corrected information if there was no EventId, an UpdateEvent request if there was an EventId, or a RemoveEvent request if there was an EventId but the user canceled.
3.6.2.5. TaskInfo - updates the UI with this task information.
3.6.2.6. TaskError - prompts the user to correct the task information. This may result in an AddTask request with the corrected information if there was no TaskId, an UpdateTask request if there was a TaskId, or a RemoveTask request if there was a TaskId but the user canceled.
3.6.2.7. ShowReminder - displays a reminder box to the user with the option of selecting snooze. This will result in a ReminderSnoozed or ReminderDismissed request.
3.6.2.8. EventConflicts - highlights the events that are involved in a conflict.

3.6.3. Required Interface

3.6.3.1. AddCategory
3.6.3.2. UpdateCategory
3.6.3.3. RemoveCategory
3.6.3.4. AddEvent
3.6.3.5. UpdateEvent
3.6.3.6. RemoveEvent
3.6.3.7. AddTask
3.6.3.8. UpdateTask
3.6.3.9. RemoveTask
3.6.3.10. ReminderSnoozed
3.6.3.11. ReminderDismissed

4. Message Types

Event-based systems use events to transfer information between components in the architecture. In this architecture messages consist of a name, and a set of name-value pairs called parameters. Each type of event is listed in this section and grouped by purpose. The name of an event is specified by the third level section heading (for example, EventInfo and EventRemoved are event names). For each event, the name of each parameter, the parameter’s data type, and a short description of the parameter is listed after the event name. All parameters must have a value other than null unless otherwise noted.
4.1. Calendar Events

4.1.1. EventInfo

4.1.1.1. EventId (Object) - the unique event id
4.1.1.2. Name (String) - the name of the event
4.1.1.3. CategoryId (Object) - the category id that contains the event
4.1.1.4. Description (String) - a description of the event
4.1.1.5. StartTime (long) - starting time of the event
4.1.1.6. EndTime (long) - ending time of the event
4.1.1.7. Reminder (long) - milliseconds before the StartTime (a value less than zero indicates no reminder)
4.1.1.8. OldName (String) - the previous name
4.1.1.9. OldCategoryId (Object) - the previous category id
4.1.1.10. OldDescription (String) - the previous description
4.1.1.11. OldStartTime (long) - the previous starting time
4.1.1.12. OldEndTime (long) - the previous ending time
4.1.1.13. OldReminder (long) - the previous reminder value

4.1.2. EventRemoved

4.1.2.1. EventId (Object) - the unique event id
4.1.2.2. OldName (String) - the previous name
4.1.2.3. OldCategoryId (Object) - the previous category id
4.1.2.4. OldDescription (String) - the previous description
4.1.2.5. OldStartTime (long) - the previous starting time
4.1.2.6. OldEndTime (long) - the previous ending time
4.1.2.7. OldReminder (long) - the previous reminder value

4.1.3. AddEvent

4.1.3.1. Name (String) - the name of the event
4.1.3.2. CategoryId (Object) - the category id that contains the event
4.1.3.3. Description (String) - a description of the event
4.1.3.4. StartTime (long) - starting time of the event
4.1.3.5. EndTime (long) - ending time of the event
4.1.3.6. Reminder (long) - milliseconds before the StartTime (a value less than zero indicates no reminder)

4.1.4. UpdateEvent

4.1.4.1. EventId (Object) - the unique event id
4.1.4.2. Name (String) - the name of the event
4.1.4.3. CategoryId (Object) - the category id that contains the event
4.1.4.4. Description (String) - a description of the event
4.1.4.5. StartTime (long) - starting time of the event
4.1.4.6. *EndTime* (long) - ending time of the event

4.1.4.7. *Reminder* (long) - milliseconds before the *StartTime* (a value less than zero indicates no reminder)

### 4.1.5. RemoveEvent

4.1.5.1. *EventId* (Object) - the unique event id

### 4.1.6. EventError

4.1.6.1. *EventId* (Object) - the unique event id (may be null if unknown)

4.1.6.2. *OldName* (String) - the previous name

4.1.6.3. *OldCategoryId* (Object) - the previous category id

4.1.6.4. *OldDescription* (String) - the previous description

4.1.6.5. *OldStartTime* (long) - the previous starting time

4.1.6.6. *OldEndTime* (long) - the previous ending time

4.1.6.7. *OldReminder* (long) - the previous reminder

4.1.6.8. *ErrorDescription* (String) - human readable reason for the error.

### 4.1.7. EventConflicts

4.1.7.1. *EventIds* (Object[]) - the events that conflicts with another event

### 4.2. Tasks

#### 4.2.1. TaskInfo

4.2.1.1. *TaskId* (Object) - the unique task id

4.2.1.2. *Name* (String) - the name of the task

4.2.1.3. *Description* (String) - a description of the task

4.2.1.4. *DueTime* (long) - due time for the task

4.2.1.5. *Reminder* (long) - milliseconds before the *DueTime* (a value less than zero indicates no reminder)

4.2.1.6. *Completed* (boolean) - indicates whether the task is completed

4.2.1.7. *OldName* (String) - the previous name

4.2.1.8. *OldDescription* (String) - the previous description

4.2.1.9. *OldDueTime* (long) - the previous due time

4.2.1.10. *OldReminder* (long) - the previous reminder

4.2.1.11. *OldCompleted* (boolean) - the previous completed value

#### 4.2.2. TaskRemoved

4.2.2.1. *TaskId* (Object) - the unique task id

4.2.2.2. *OldName* (String) - the previous name

4.2.2.3. *OldDescription* (String) - the previous description

4.2.2.4. *OldDueTime* (long) - the previous due time

4.2.2.5. *OldReminder* (long) - the previous reminder

4.2.2.6. *OldCompleted* (boolean) - the previous completed value
4.2.3. AddTask
   4.2.3.1. Name (String) - the name of the task
   4.2.3.2. Description (String) - a description of the task
   4.2.3.3. DueTime (long) - due time for the task
   4.2.3.4. Reminder (long) - milliseconds before the DueTime (a value less than zero indicates no reminder)
   4.2.3.5. Completed (boolean) - indicates whether the task is completed

4.2.4. UpdateTask
   4.2.4.1. TaskId (Object) - the unique task id
   4.2.4.2. Name (String) - the name of the task
   4.2.4.3. Description (String) - a description of the task
   4.2.4.4. DueTime (long) - due time for the task
   4.2.4.5. Reminder (long) - milliseconds before the DueTime (a value less than zero indicates no reminder)
   4.2.4.6. Completed (boolean) - indicates whether the task is completed

4.2.5. RemoveTask
   4.2.5.1. TaskId (Object) - the unique task id

4.2.6. TaskError
   4.2.6.1. TaskId (Object) - the unique task id (may be null if unknown)
   4.2.6.2. OldName (String) - the previous name
   4.2.6.3. OldDescription (String) - the previous description
   4.2.6.4. OldDueTime (long) - the previous due time
   4.2.6.5. OldReminder (long) - the previous reminder
   4.2.6.6. OldCompleted (boolean) - the previous completed value
   4.2.6.7. ErrorDescription (String) - human readable reason for the error

4.3. Categories

4.3.1. CategoryInfo
   4.3.1.1. CategoryId (Object) - the unique category id
   4.3.1.2. Name (String) - the name of the category
   4.3.1.3. Description (String) - a description of the category
   4.3.1.4. Color (Color) - the color for category events
   4.3.1.5. OldName (String) - the previous name
   4.3.1.6. OldDescription (String) - the previous description
   4.3.1.7. OldColor (Color) - the previous color

4.3.2. CategoryRemoved
   4.3.2.1. CategoryId (Object) - the unique category id
   4.3.2.2. OldName (String) - the previous name
   4.3.2.3. OldDescription (String) - the previous description
4.3.2.4. *OldColor* (Color) - the previous color

4.3.3. **AddCategory**
   4.3.3.1. *Name* (String) - the name of the category
   4.3.3.2. *Description* (String) - a description of the category
   4.3.3.3. *Color* (Color) - the color for category events

4.3.4. **UpdateCategory**
   4.3.4.1. *CategoryId* (Object) - the unique category id
   4.3.4.2. *Name* (String) - the name of the category
   4.3.4.3. *Description* (String) - a description of the category
   4.3.4.4. *Color* (Color) - the color for category events

4.3.5. **RemoveCategory**
   4.3.5.1. *CategoryId* (Object) - the unique category id

4.3.6. **CategoryError**
   4.3.6.1. *CategoryId* (Object) - the unique category id (may be null if unknown)
   4.3.6.2. *OldName* (String) - the previous name
   4.3.6.3. *OldDescription* (String) - the previous description
   4.3.6.4. *OldColor* (Color) - the previous color
   4.3.6.5. *ErrorDescription* (String) - human readable reason for the error

4.4. **Alarms**

4.4.1. **AlarmInfo**
   4.4.1.1. *AlarmId* (Object) - the alarm id
   4.4.1.2. *AlarmTime* (long) alarm time in milliseconds

4.4.2. **AlarmRemoved**
   4.4.2.1. *AlarmId* (Object) - the alarm id

4.4.3. **SetAlarm**
   4.4.3.1. *AlarmId* (Object) - the alarm id
   4.4.3.2. *AlarmTime* (long) alarm time in milliseconds

4.4.4. **ClearAlarm**
   4.4.4.1. *AlarmId* (Object) - the alarm id

4.4.5. **AlarmTriggered**
   4.4.5.1. *AlarmId* (Object) - the alarm id
   4.4.5.2. *AlarmTime* (long) alarm time in milliseconds

4.5. **Reminders**

4.5.1. **ShowReminder**
   4.5.1.1. *ReminderId* (Object) - the reminder id
   4.5.1.2. *ReminderText* (String) - the reminder text
4.5.2. *ReminderSnoozed*
   4.5.2.1. *ReminderId* (Object) - the reminder id
   4.5.2.2. *SnoozeTime* (long) - the number of milliseconds to snooze

4.5.3. *ReminderDismissed*
   4.5.3.1. *ReminderId* (Object) - the reminder id

5. **Miscellaneous**

   **5.1. Time Format**

   Unless otherwise noted, all times are reported as the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.