Model-driven Application Design for a Campus Calendar Network

Allison Bloodworth, University of Califonia at Berkeley http://www.berkeley.edu/ <abloodworth@berkeley.edu>

Dr.. Robert J. Glushko, University of Califonia at Berkeley http://www.berkeley.edu/ Center for Document Engineering http://cde.berkeley.edu/ <glushko@sims.berkeley.edu>

Due to the decentralized nature of computing on the University of California, Berkeley campus, different schools, departments, and campus organizations often create applications in an independent and ad-hoc fashion. A striking example of this occurs with event calendars -- at least 80 different calendars exist on the berkeley.edu domain. Each calendar has its own way of describing events, uses different forms for submitting them, and different databases to store them. This means that there is no automated way to cross-post event information among these calendars. Cross-posting is accomplished today using manual data entry forms located on many calendars' websites, or by sending an email with event information to a calendar administrator.

This situation is not novel. Most large organizations struggle with incompatible models and applications for time sheets, expense forms, project schedules, registrations, etc. The problems are also typical of those that arise between enterprises with incompatible catalogs and transactional documents like orders and invoices.

In the fall of 2003 a team of UC Berkeley staff members, led by the first author and advised by the second author, began the process of solving this problem. We developed a standard data model of an Event flexible and scalable enough to accommodate the requirements of most calendars on the Berkeley campus. The group began by selecting 23 campus calendars and harvesting the data elements from each one. We then went through a process of harmonizing and consolidating the data elements into a list of candidate components. The design portion of the process involved using normalization procedures to separate the selected components into functionally dependent aggregates. This resulted in a conceptual model of an event. The group's final step was to begin the implementation process by encoding this conceptual model into an XML schema.

This Event model was then used as the basis for a model-driven event management system, the UC Berkeley Calendar Network. Our team created a web-based calendar that can display multiple views of events conforming to this Event data model. In conjunction with our Calendar Management tool, which allows calendar administrators to both manage the events in their calendar and customize their calendar's appearance, the calendar we provide can be used by many organizations on campus. This tool provides numerous ways to customize the calendar, including the use of cascading style sheets or XSL transforms. Calendars using our system will store their events in a centralized repository also based on our Event data model. However, there are calendar administrators who have specialized web development needs, or a need to maintain their own repository of event information and thus cannot use our calendar and repository. For these users, we outline a process by which they can send event information to, or pull information from the centralized event repository using an XML document that, again, is based on our Event data model.

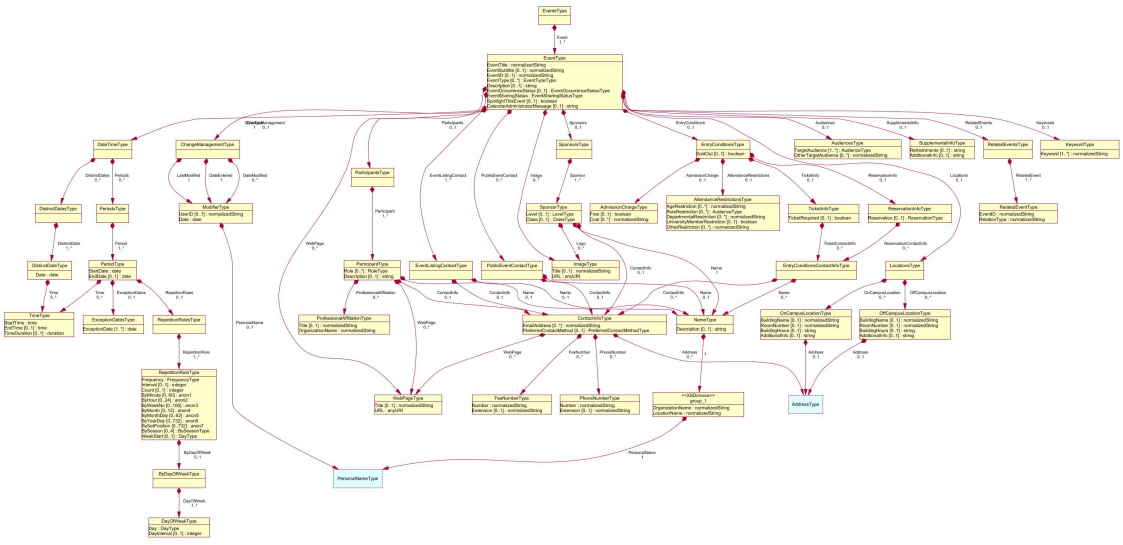
Table of Contents

Overview of the Paper

This paper will outline the process followed by a group of researchers, masters students, and employees at the University of California at Berkeley as they created a standard data model for a calendar event that could be used by all event calendars on campus. It also describes a subsequently developed web application based on this standard data model of an Event, whose function was to manage web-based calendars and event information contained in these calendars.

Figure 1 depicts the Event model as a UML class diagram. We will discuss many of the analysis and design considerations that led us this final model. It is admittedly a very complex model, much of the detail of which will not be visible in the .pdf version of this document. However we have included this diagram anyway to convey a sense of the complexity of the final model.

Figure 1. UML class diagram of the Event model



When formulating the Event model, the modeling team followed a methodology taught by the second author at the School of Information Management & Systems (SIMS) at UC Berkeley, called "Document

Engineering." The goal of Document Engineering is to create "robust and re-usable models of information exchanges and their business processes suitable for deployment in service oriented architectures, such as web services."[DocEng] This case study will describe this methodology, explain the goals of each step in the process, and show how it was applied to the real-world Event modeling problem. It will then describe the resultant application developed by the Berkeley Calendar Network team, as well as the benefits of creating a web application with a XML schema-encoded data model.

Overview of the UC Berkeley Calendar Network Project

Calendars at UC Berkeley

Due to the decentralized nature of computing on the University of California at Berkeley campus, different schools, departments, and campus organizations often create applications in an independent and ad-hoc fashion. The lack of campus-wide guidelines and standards for designing and building applications make it difficult for developers to design for interoperability and reuse. Consequently, the Berkeley campus is inundated with applications serving a similar purpose and repurposing similar content but built with different technologies and based on different, and often incompatible, data models.

A striking example of this problem is illustrated by event calendars; at least 80 different calendars exist in the berkeley.edu domain. Each department independently creates a web-based calendar based on their own definition of an event, uses different forms for submitting these events, and different databases to store them. Consequently, there is no easy way to cross-post or share event information among calendars. This means that many calendars do not include events from other departments that would be of interest to their users, and there is no centralized place on the Web to go to find information on all events occurring on the Berkeley campus. Although the UC Berkeley gateway site (www.berkeley.edu) endeavors to be an aggregator of event information, they are unable to obtain a complete listing of all campus events. This is because to post to the gateway calendar, departments must either use a web-based “Add Event" form, or send an email request to the calendar administrator. Although many departments would welcome the chance to publicize their event on other calendars, they do not have the time, inclination, or resources to do this for each and every event. To make matters worse, if a department would like to share their events with additional calendars, they must repeat this process with every calendar.

The current system is problematic in many ways. The process of re-entering event information wastes time, is inherently error prone and makes it difficult to maintain the integrity of data located in multiple locations. Additionally, replicating event information increases data storage costs and can increase overall complexity. Finally, incompatible data models limit the amount and type of event information that can be shared or repurposed. These issues hinder the creation and consumption of web-based event information on the UC Berkeley campus.

The UC Berkeley Calendar Network Project

The UC Berkeley Calendar Network project was created to address these problems. The project began in the summer of 2003 when the first author, who was then a graduate student at UC Berkeley's School of Information Management & Systems (SIMS) and a Graduate Student Researcher at the Center for Document Engineering (CDE), undertook a collaborative effort with UC Berkeley staff members to develop a standard data model of an Event. The goal of this group was to create an Event data model that was flexible and scalable enough to accommodate the requirements of most calendars on the Berkeley campus. This Event model was then used as the basis for the design of an event management system, the UC Berkeley Calendar Network[UCBCN]. The goal of the Calendar Network project was to improve the process of sharing event information on the UC Berkeley campus. It was the SIMS master’s thesis of the Berkeley Calendar Network team, which consisted of the first author, Nadine Fiebrich, Myra Liu, and Zhanna Shamis.

The architecture of the system is depicted in Figure 2. The main components include:

- A standard data model of an Event, encoded in an XML schema
- A centralized repository of event information, based on the Event data model
- A Calendar Management Tool
  - Allows users to manage their events in the repository
  - Helps users customize a visually compelling dynamic web-based calendar
- XML document interfaces based on the Event model for external calendars to send event information to, and extract information from the central repository

Figure 2. UC Berkeley Calendar Network System Architecture diagram

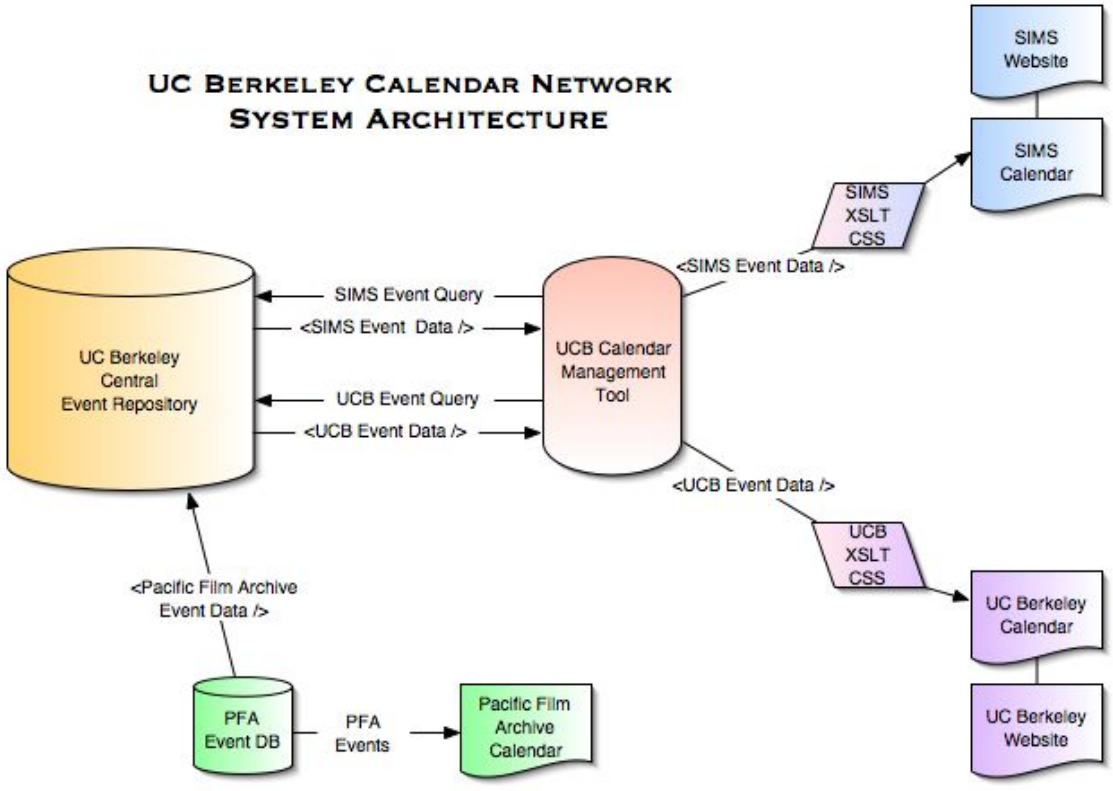


Figure 2 depicts two different ways of interacting with the central repository. Calendars may access the repository through the Calendar Management tool, using either the default calendar, a customized version of the default calendar created by modifying the Cascading Style Sheet (CSS), or a completely new calendar created by modifying the default XSL transform. Alternatively, calendars may use web services to send an XML document to the central repository to upload their information, or request a download of event information from the repository, which is returned in an XML document.

Development Constraints

Working in the University Environment

The development of this system was strongly constrained by the unique environment in which it would operate: the university. Most universities, including the University of California, have a decentralized administration structure designed to allow intellectual freedom and encourage innovation. This means that campus departments and organizations often have the authority to decide internally how they would like to do business. At UC Berkeley there are very few formal guidelines about how websites or web calendars should be set up, and few resources, such as centralized databases, which can be shared among the various campus departments and organizations. Because there are few standards set from above, most of these organizations create their applications and websites in isolation with little to no thought about how they might work together. In order to deal with this unique environment where any community-wide standards are usually adopted in a grass-roots fashion, the Event modeling team and Berkeley Calendar Network team viewed all work on this project as part of a larger marketing process. The groups involved as many campus calendar owners as possible in all phases of the project, including needs assessment, the Event modeling process, and the design and usability testing of the web calendar and Calendar Management Tool. We believe that integrating the input of all campus calendar owners into the development process was and will continue to be a critical step towards encouraging adoption of this system.

Different Levels of User Technical Expertise

The modeling team's survey of campus calendars during the Event modeling process revealed that calendars on the UC Berkeley campus exhibit a wide range of technical complexity. Some calendars are connected to sophisticated ticketing and payment systems and have event data which is very specialized to their domain. Other calendars are static HTML pages which are simply lists of very simple events. There are also many organizations on campus that do not currently have calendars, often due to a lack of available resources to create them. Our system had to meet the needs of as many of these different types of users as possible.

Event Data Modeling

Genesis of the Project

This idea for this project was generated in the Document Engineering course taught by the second author at SIMS in the spring of 2003. Students were asked to create a model of a calendar event by analyzing the data found in 6 UC Berkeley campus calendars. During the summer of 2003 as part of a Center for Document Engineering (CDE)[CDE] effort, a modeling team comprised of the first author and several campus calendar owners and administrators was formed to continue the Event modeling process. The second author was an advisor to this group. Although the team's efforts were focused primarily on the calendar domain, the overall goal was to create a model of a campus event that could be used in the type of calendaring application we envisioned as well as in other campus applications that might need to use the concept of an event. We believed this concept could apply to everything from a meeting of a class to a lunch date to a performance or sporting event. The group also believed that figuring out how to describe an event for the purposes of a calendar would allow us to capture most, if not all, of the essential descriptors of any type of campus event, whether it was routinely represented on a calendar or not. The intent was to design a model for an Event that would become part of the Berkeley Academic Business Language (BABL), a set of data models and associated XML schemas developed by SIMS students for the domain of university education.

Document Engineering an Event

Throughout the Event modeling phase of this project the modeling team followed an approach called Document Engineering. On the CDE website, Document Engineering is defined as "a new discipline for specifying, designing, and implementing the electronic documents that request or provide interfaces to business processes via web-based services."[CDE]Document Engineering is a way to analyze informa-

tion from diverse sources and merge it to create a single, unified data model. A Document Engineer has an "artifact-focused view of modeling,"[DocEng] and begins by analyzing existing documents from businesses or entities that need to communicate with each other. This data is augmented by gathering of information from other sources of requirements, such as the people who create or use the documents. A data model for the problem space is then created, which results in a set of reusable components that are usually expressed as XML schemas. This data model may then be used to build documents that all these entities can use to communicate. The use of this common data model allows different groups to exchange information in a loosely-coupled manner, while still ensuring that all entities know exactly what the information means. Thus Document Engineering may be thought of as "a 'document-centric' version of"…"the classical" analyze - design - refine - implement methodology."[DesDocTWS]

Context & Business Process Analysis

The Document Engineering Approach

The Document Engineering process often begins with a top-down analysis of the context, or strategic business objectives of the organization or organizations involved, as well as business processes to ensure that the Document Engineer understands the strategy driving the applications that use the data he is trying to model.

The Event Modeling Process

In the case of the event calendar domain, the contextual analysis was fairly straightforward. A web calendar is generally a marketing tool whose main purpose is to publicize events, either within a community or to the general public. Through interviews with campus calendar owners, which occurred concurrently with the "Document Analysis" data-modeling phase (see section below), we determined that this was an accurate description of the purpose of web calendars at UC Berkeley. Some calendars had different requirements regarding with whom they share the data, but the purpose for these calendars was essentially the same.

After analyzing the business processes of several calendars via interviews with 15 different calendar owners, the team determined that most web-based event calendars were a forms and workflow type of application where event information is entered into a database and displayed based on a certain set of rules. In most cases the calendars on the UC Berkeley campus were very simple, and the main rule was that event information was displayed based simply on the date range selected. This range was sometimes selected by the user, or, in the case of static webpages, the calendar administrator. The calendar administrator would of course have other rules based on departmental policies they would follow in deciding which events to post on their calendar.

Some of the more advanced calendars allowed users to query a database of events using additional search criteria beyond the date, sort event information in some fashion, or included ticketing and payment systems. Email distribution lists which operated behind the scenes and various features of event management systems currently in use were other important process details which helped the modeling team formulate the event model.

Document Analysis

The Document Engineering Approach

The Document Engineering approach emphasizes the analysis of existing physical models in a domain as a starting point for the creation of a new model. The objective of the Document Analysis phase, which is a core focus of Document Engineering, is "to create a conceptual perspective that encompasses all the information requirements within the required context of use."[DocEng] The use of existing physical models ensures that a new model will be comprehensive, as well as usable by all users in the domain. During this phase, a list of existing document models in the domain, called the Document Inventory, are analyzed and a representative sample of relevant documents are chosen for closer analysis. These documents may include "document guidelines and standards, sample document instances, Web pages, and other information sources to harvest all potentially meaningful information components and

the constraints that govern their values, arrangement and use."[DocEng]

The Event Modeling Process

Research On Existing Data Models

An important tenet of Document Engineering is "encourage reuse." This means that any Document Engineering effort should include the researching of any standards or work done by other groups in the chosen domain. Widely-accepted existing standards or models which already provide accurate representation of the context being analyzed should be reused wherever possible.

The modeling team began this process by researching other data models of calendar events. While we located some models that at first glance appeared extremely relevant, after closer analysis found that they were only partly appropriate for our requirements.

iCalendar

We first looked at iCalendar, the Internet Calendaring and Scheduling Object Core Specification (RFC 2445). This document describes the open standard created by the Internet Engineering Task Force in 1998, which is characterised by the authors as “a common format for openly exchanging calendaring and scheduling information across the Internet."[iCalendar]The group quickly realized that although we would need to understand this pervasive standard in order to make sure it could be mapped to our model, it was geared more towards personal calendaring systems like Microsoft Outlook or Apple’s iCal, not a public event calendar created for the purpose of publicizing events on the web. The iCalendar specification details the way that everything from events to to-do’s to journal entries and free-busy time on a personal calendar should be handled, but does not address the way event details such as speakers, sponsors, and admission information would be organized.

However, the problem of recurring events is one that both personal calendaring systems and public event calendars share, and the iCalendar specification solved that problem very elegantly. Thus the recurrence model in the final Event schema is based directly on the iCalendar recurrence model. Because this specification was somewhat difficult to decipher, we also reviewed the iCalendar DTD Document (xCal). This was a helpful reference which gave a quick overview of the specification by showing what a DTD for iCalendar would look like. [xCal]

SKICal

The modeling team then looked at the Structured Knowledge Initiative Calendar, or SKICal, which was created as an internet draft in July 2001. The authors of this document had a goal similar to ours: “to improve the information infrastructure concerned with public events (concerts, sports competitions, conferences etc.)."[SKICalSite] SKICal is an extension of iCalendar whose goal is to "make it more useful for managing public events outside of the business world (sports, culture, etc.)," the domain for which iCalendar was originally intended.[SKICal-iCal] The specification states that "SkiCal expands the traditional property-set of name, address, telephone number and business category, adding structured information about people, places, things, activities and the conditions and terms for interaction with resources. SkiCal provides a structure for information about dates and times, directions, rules and recommendations for participation, pricing and reservation schemes, access information for those with special needs, ownership and responsibilities and promotional material." Their concept of an event is based on the concepts of '"TimeSpenders' seeking information about available resources ("SkiSources") and 'Publishers' wishing to make that information public."[SKICal]

Although SKICal was closer in principle to what we were looking for, we did not find this system entirely intuitive or feel that it mapped closely enough to our domain of university education. It seemed that SKICal offered elements that were not defined specifically enough for campus calendar administrators to understand how to use them in our system, and the modeling team believed we could organize them more effectively.

For instance, there is an element called "Persons" whose purpose is "to facilitate for TimeSpenders the discovery of SkiSources relating to specific people - living or dead."[SKICal] "Persons" can also have a

descriptor called "SKIRole." This SKIRole could be anything from "Performer" to "Conductor" to "Creator," and seemed to encompass both people who were participants in the events, as well as people who were the topic of discussion at or had contributed something to the event. However, we felt that these things should be clearly separated. Additionally SKIRoles such as "Present" were allowed for names like "The Queen of Spain." This did not seem like an accurate description of a role in relation to the event; it was more of a role in relation to the world. In terms of organization, in our system the modeling team had already hypothesized that the important roles would be things like Sponsors, Participants, and Contacts. We believed that using those more specific labels would be more intuitive to calendar administrators and help them more easily understand what information they should collect, make it easier to query for specific information (e.g. "Who is participating in/sponsoring the event"), and give us the ability to specify what the required pieces of information were for a particular event (e.g. an event must have a sponsor).

There were other elements in SKICal which did not seem to make sense for our domain. For instance, there were categories like "Thinks" which was supposed to help people locate events "relating to specific concepts, topics, themes and areas of interest," "Things" which described events "relating to objects, things, specific goods and products, collectors items, various commodities and so forth," and "Actions" which related to "things happening, processes, movements, etc."[SKICal] We felt that these categories unnecessarily complicated what campus calendars usually called an event "Keyword" or "Event Type" (e.g. lecture, performance, meeting), and thus would not be useful to us. However, one area of the specification we did find particularly useful was the "How" section, which described different types of information necessary admission to events. It helped us make sure we weren't missing anything important when developing the elements we ended up calling "Entry Conditions."

Selection of Calendars for Evaluation

The modeling team began this calendar selection portion of the Document Analysis phase by creating a list of the different types of calendars at UC Berkeley that we knew about from personal experience. These included:

- Academic Departments
- Academic Colleges/Schools
- Research Centers
- Libraries
- Performance (e.g. musical, dance)
- Museums
- Athletics
- Personal calendaring systems

The group then discussed whether we should broaden our project scope and model calendars outside of the university domain, including calendars with different purposes and distributed via different media, such as via newspapers like the Washington Post. In the end we decided that because our goal was to create an Event model for the UC Berkeley campus, because there were scores of calendars we could potentially analyze in this domain alone, and because it seemed that other types of calendars we could think of were similar to the ones found on campus, we would limit ourselves to analyzing UC Berkeley campus calendars only.

When selecting the actual calendars to analyze, it was suggested by the second author that the modeling team evaluate about a dozen calendars and select the calendars using the following criteria:

- Does it have a large user base?
- Does it have something distinctive about its type of content?
- Are its users friendly to our efforts?

Selecting big, important calendars whose owners were friendly to our efforts *was* important because the success of the system would be based on convincing current calendar owners to stop using their own calendar and switch to ours. If we had analyzed a particular calendar during the modeling process, it would be more likely that our Event model would meet that calendar's needs. We believed it would be difficult to convince some calendar owners to switch to our system in such a decentralized university campus, where everyone creates their own calendar which is very specific to their own needs. In order to gain the critical mass necessary to make the project a success, we would need to create a system that convinced both those with simple calendars whose needs we could easily meet as well as owners of large, important calendars to convert to our new system.

However, in the end the group decided that despite the public relations value of selecting certain important calendars to analyze for our model, our first priority was to ensure that we had an accurate model. We could only do this by analyzing a comprehensive and representative sample of calendars. For this reason, we decided to select calendars primarily based on the distinctiveness of their content, and use information on the size of the user base or an assessment of the "friendliness" of the calendar owners only as a tiebreaker if we had similar calendars with the same type of content. Nonetheless, we were reminded that we would have to these types of political and marketing concerns in mind throughout the Document Engineering process.

To help us choose calendars to analyze, one of the members of the modeling team created a master list of campus calendars that we labeled as "stakeholders" in the project. A stakeholder to us could be any calendar on the UC Berkeley campus (as they were all potential clients of our system), but this list did not cover every calendar on campus. Our goal was to create a list that covered all important calendars we should not forget when deciding which ones to model, as well as a representative sampling of all other types of calendars we could think of. We used this list not only to select calendars, but also to determine whether we had missed certain categories of campus calendars. After reviewing our initial list, we determined that we should add the following types of calendars:

- Administrative Departments
- Student Groups

As often happens in projects, the modeling team began to experience significant scope creep and had to stop and remind ourselves of the original goals for the project in order to decide where to set limits on the calendars we should include in our model. In one instance, a group member suggested that we consider modeling not only events, but also the "tasks" which appeared on personal calendaring systems. It was hypothesized that a task might be simply an event without a specific date or time, or even an event that was to take place at some point in the future, but did not have a set day or time. It was also pointed out that these tasks might be related to holidays, which had a specific date but not a specific time.

In the end because our domain was event calendars and not personal calendars, and because we had limited time and resources, we decided to limit the scope to events which were actual "happenings," usually having at least date and usually a time, as opposed to tasks, or "to-do's." We chose to limit our scope in this fashion to ensure that we could complete the project in a reasonable length of time without spending time modeling things which would not be necessary for the system we were envisioning. Determining the scope a modeling project is often a difficult task, and although some might content that a "pure" model would not have any view towards the eventual implementation, we found it was necessary to set these limits in order to keep our discussions focused.

Another way we tried to focus our efforts was to use the 80-20 rule. We decided that the goal in creating

the campus Event model would not necessarily be to enumerate every single data element of every campus calendar and create a model that had every single element in it. Indeed, our goal was to encourage calendars to converge toward a standard model of an event, so we decided to only include data elements that were useful to about 80% of the calendars. Calendars that had domain-specific information that fell in the other 20% could extend our model and store their additional information in their own repository.

We began the modeling process by putting together a list of calendars to analyze and assigned them to various group members, who reported back on what they found at subsequent meetings. The modeling team decided that we'd follow the "law of diminishing returns," and would continue to analyze calendars in this fashion until we stopped finding new candidate elements in new calendars. This rule required us to review almost twice as many calendars as we had originally expected, 23 in all. They included:

1. UC Berkeley Calendar [http://www.berkeley.edu/calendar] - Main calendar for the university
2. Letters & Science [http://ls.berkeley.edu/events/] - Academic College/School
3. Haas School of Business [http://www.haas.berkeley.edu/calendar/] - Academic College/School
4. Men's Basketball [http://calbears.collegesports.com/sports/m-baskbl/sched/cal-m-baskbl-sched.html] - Athletics
5. Academic Calendar [http://opa.vcbf.berkeley.edu/AcademicCalendar/calendardisp.cfm?terms=current] - Calendar of critical academic dates for the university
6. PDA [http://dream.berkeley.edu/CDE-Events/PDAScreens.html] - Personal Calendaring System
7. International Area Studies [http://ias.berkeley.edu/calendar/] - Academic Department
8. Math Department [http://ls.berkeley.edu/dept/math/calendaring/] - Academic Department
9. Men's Football (Intercollegiate sports) [http://calbears.collegesports.com/default/cal-default.html] - Athletics
10. CalFit [http://calbears.berkeley.edu/calaerobics/aerclasses.asp] - Athletics
11. College of Engineering [http://www.coe.berkeley.edu/events/index.html] - Academic College/School
12. UC Berkeley Library [http://www.lib.berkeley.edu/news_events/exhibits/] - Library
13. Bancroft Library [http://bancroft.berkeley.edu/events/] - Library
14. Capital Projects [http://www.cp.berkeley.edu/TempAccessInterruption.html] - Administrative Department
15. University Health Services [http://www.uhs.berkeley.edu/home/news/calendar.shtml] - Administrative Department
16. Center for Document Engineering [http://cde.berkeley.edu/events/] - Research Center
17. California Biodiversity Center [http://cbc.berkeley.edu/thisweek.html] - Research Center
18. Cal Performances [http://www.calperfs.berkeley.edu/presents/season/2003/calendar_of_events/index.html] - Performance
19. Lawrence Hall of Science [http://www.lawrencehallofscience.org/pubprogs/] - Museum

20. Music Department Noon Concerts [http://music.berkeley.edu/noon.html] - Performance
21. SUPERB [http://superb.berkeley.edu/calendar.html] - Student Group/Performance
22. Berkeley Art Museum & Pacific Film Archive [http://www.bampfa.berkeley.edu/calendar/index.html] - Museum
23. CalAgenda [http://calagenda.berkeley.edu/] - Personal Calendaring System

Component Analysis

The Document Engineering Approach

In Document Engineering, Harvesting Data Elements is the first step of the Component Analysis phase. Harvesting is the isolation of individual semantic components from the data elements in existing documents. These data elements are put in a "Table of Candidate Content Components" and given tentative names. It may make sense to change these names as elements are collected, however, in order to ensure that each element name is "semantically unambiguous within their context of use."[DocEng] As data elements are harvested from existing documents, it is important to determine what the real data elements are, and not to be fooled by structure or presentation information. For instance, on a website an element called "Event Description" may contains information on the event's topic, the cost of the event, and who the event sponsor is. However, while they may look like they are all part of an aggregate called "Event Description," in reality these elements may be being collected separately. Additionally, even if these elements aren't actually being collected separately, in order to create models with the most reusable components, we may find that it makes sense to separate them in the data model we eventually create.

The next step in Component Analysis is Consolidating Candidate Components. In this phase the Table of Candidate Content Components is consolidated to ensure that every element is semantically distinct and unique. This means that elements that are synonyms for other elements are given a single name, and elements that are homonyms are each given a distinct, unique name. This activity results in a "Consolidated Table of Content Components."

The Event Modeling Process

Harvesting Data Elements

The modeling team began this process by harvesting the data elements from each calendar that had been selected for analysis. The team found that the best way to determine which data elements a calendar was actually collecting was to look at their event data entry form, often known as an "Add Event" form. This included both the part exposed to the user via the interface, as well as the underlying HTML code. It was then helpful to look at the document instances to determine what the possible values for each data element were. Document instances in this case were the distinct events listed on each calendar. When certain fields were not exposed on the public web calendar interface, we made a note to ask the calendar administrator about the function of that element when we interviewed them. This task took on average about 2 hours per calendar.

After reviewing the data entry form and event listings of a particular calendar, each team member created a list in Excel of all data elements found in that calendar. We also formulated a short glossary of data elements, which was a list of data elements we knew were commonly found in event calendars. This helped us determine at the collection stage whether elements with different names in different calendars were actually the thing, and designated a standard name for the element. For instance, some calendars might list a "End Date" for an event, while others labeled it "End On."

It was difficult to know in advance how much information or metadata would be needed to understand and distinguish the candidate data elements. So we iterated a bit before we settled on collecting the following information about each data element:

Calendar	The calendar in which the data element was found.
Calendar Element Name	The name of the data element as it appeared in the calendar in which it was found.
Element Glossary Name	The name of the data element in our glossary.
Composite Name	We used this field to group the elements into categories that we intuitively understood (e.g. DateTime, Admission Restrictions, Publicity, Contact Info, Sponsor) for the purposes of work distribution among the team.
New to Glossary	This field was used to indicate that an element was new, and had not yet been added to our glossary.
Description	A description of the data element.
Data Type	This field indicated the datatype of the data element (e.g. string, integer, enumeration, date, time, boolean, URL).
Possible Value	An example value of the data element, which could be found in the calendar analyzed.
Default Value	If the data element had a default in the calendar, it was indicated here.
Code List Values	If the element offered choices in the form of an enumeration, the possible values were listed here.
Required	This field indicated whether the element was required in the calendar we were analyzing.
Min Occurs	If the data element has a minimum number of occurrences in the event (e.g. it is required, or must occur twice), this was indicated here.
Max Occurs	If the data element has a maximum number of occurrences in the event, this was indicated here.
# of Page Occurrences	This element was not used consistently, but indicated when data elements were more frequently found than others by counting the number of page occurrences in a particular document.
Name	Name of the person who had done the analysis
Needs Clarification	Indicated whether we needed to do further follow-up with a calendar owner to determine the true meaning of a data element.
Notes	Additional notes about the data element.

Initially the modeling team also tried to use the concept of composites, or aggregate and leaf elements, to help us capture information about data elements that should be grouped together. We collected data about whether an element was a parent (also known as an aggregate element), or a child (or a leaf element) as well as what its parent element was (if it was a child) or what its children elements were (if it was a parent). We found that although this made sense right away for the DateTime elements, it was difficult to figure out which of the other aggregate elements should be decomposed into subelements at the time we were harvesting them. For example, we were unsure if the "Name" of a "Sponsor" should be an aggregate of "First Name" and "Last Name." It would seem logical to do that for a person, but would not make sense if the sponsor was an academic department. As a result, we decided to postpone decisions about whether elements should be broken down into subelements for the modeling stage.

We also believed that to some extent how far we would decompose the data elements would depend on the functional requirements of the target application. For example, if we didn't foresee a requirement for users to search separately on "First Name" and "Last Name," we were not sure at this point that it made sense to break them down in the model.

The modeling team used the 80-20 rule make judgments on when not to model very domain-specific data, such as the data that was found in some sports and arts-related calendars. After some debate, we decided it was unnecessary to analyze the specialized data of these domains in detail. We made this decision because the substantial required effort to do this analysis would not result in enough of a benefit, since any data elements we found would definitely not be used in at least 80% of the calendars.

As the harvesting process progressed and we identified new candidate components, questions about what an event was continued to come up. Some of these questions and our answers to them included:

Are hours of operation events?	They may be permanently recurring events which should be flagged differently somehow.
Are occurrences that span long periods of time, such as art exhibits, events?	They seem to be a different type of event that should be flagged differently somehow.
How do we deal with conference events?	Set up a parent and child event structure where the conference is the parent event, which crosses all the days of the conference, and the sub-events that occur at the conference are child events.
Is an on-line class that doesn't have a specific date or time an event?	It may be a long-running event, but only if it has a specific start and end date.
What is the difference between an "Event URL" and a "More Info" link?	An "Event URL" links directly to a webpage or site devoted only to the event, and a "More Info" link can go to any page with information on or related to the event.
Should we store the rules used to generate recurring events (events that occur more than once following a pattern), or should the dates simply be calculated at the application level?	We may want to store the pattern in case it is important to someone using the model.

It was suggested by several people working on the development of the Universal Business Language (UBL) that the modeling team create a controlled vocabulary for every individual term we used to ensure that we knew what each term meant. For example, if we had determined that we would use the word "start" anytime we needed a term that meant "to begin or commence" something we would indicate that in a controlled vocabulary. Thus when naming elements the group would know that we should never use the word "begin;" we would always use "start." Additionally, if we had a definition in our controlled vocabulary for the word "date," we would always be able to determine what aggregate terms such as "start date" meant by combining the separate definitions of the two terms.

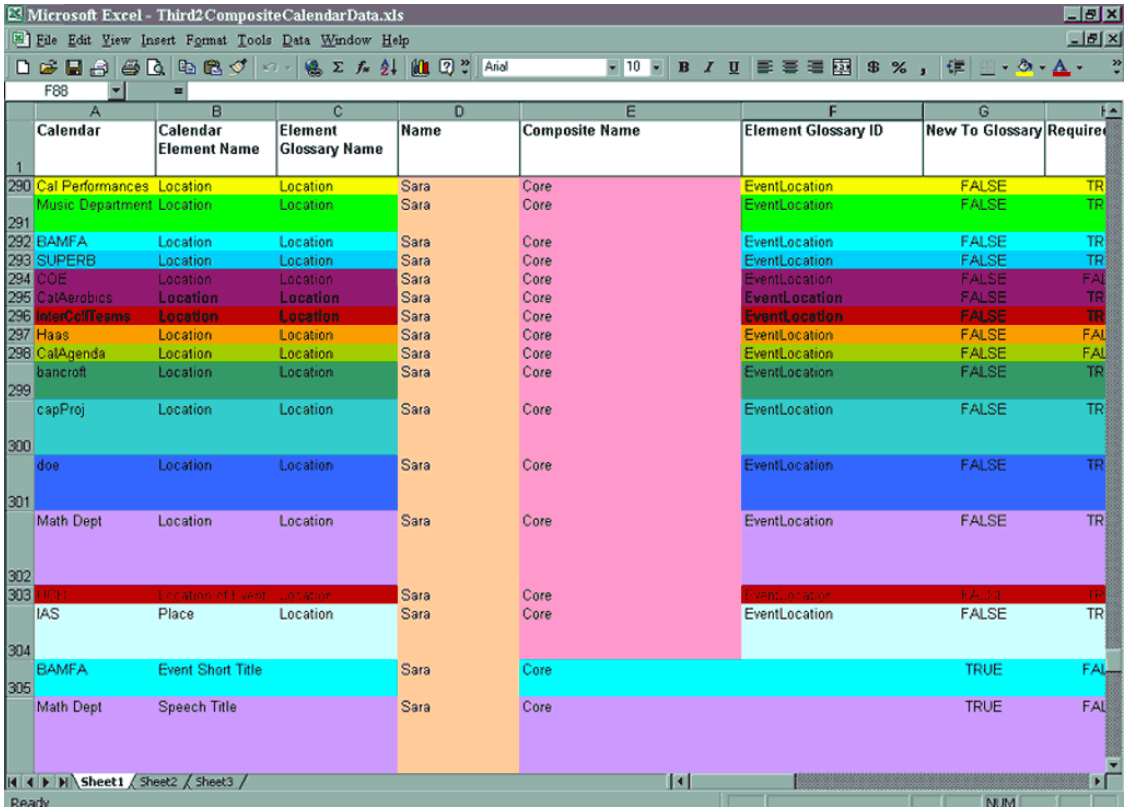
However, there were other terms, such as "time" or "contact," that were harder to define individually and have them make sense as part of aggregate terms. Additionally, the modeling team found that much of our terminology was already standardized within the documents we were analyzing within the calendar problem space, so there weren't often several commonly accepted choices between which we had to decide. In the end for our project we decided that it would be easier to just use the glossary we had created to determine what each term should be. It seemed that the effort required to create a controlled vocabulary was not warranted our domain.

Consolidating Candidate Components

After the modeling team had collected a total of almost 350 data elements from the 23 calendars we had selected, we needed to organize them to determine all the unique data elements. First, we merged all the individual spreadsheets into one master spreadsheet, which is referred to as the Table of Candidate Content Components. At this point we had data elements with over 150 different names. Next, we ensured

that all elements with had been assigned a glossary name. Elements that only appeared in one calendar were usually assigned a Glossary Name that was the same as their Calendar Element Name, as were elements that appeared in more than one calendar but had the same Calendar Element Name. During this process we ensured that elements that were synonyms (different name, same meaning) had the same Glossary Name, and elements that were homonyms (same name, different meaning) had different Glossary Names. We highlighted the elements in each calendar in a different color so that it would be easy to see at a glance where each element came from. Then we sorted the entire spreadsheet containing the data elements from all 23 calendars by Glossary Name. This spreadsheet then became our Consolidated Table of Content Components and contained close to 100 unique data elements.

Figure 3. Excel spreadsheet Consolidated Table of Content Components



The modeling team used this list of Consolidated Table of Content Components to determine what elements to include in the data model. Because we hasn't collected much "domain-specific" information (such as a model of sports scores or an art exhibit), at this point we reversed the 80-20 rule and included anything in the model that was included in at least 20% of the calendars. To some extent we made these decisions qualitatively as well; if we thought that although the element didn't appear in many of the calendars we analyzed it was an important component of a generalized Event, we included it.

Additionally, we considered the fact that it was possible in some of these cases that the elements had existed in multiple calendars, but we just hadn't disaggregated the elements enough to find it. For example, we included "Sponsor URL" even though it was only officially included in our spreadsheet once. Other infrequently occurring elements that we included were: "Participant Description" (which was called "Biography" in the two calendars where we found it), "Participant Role" (which was usually an implicitly-occurring element, even though it wasn't usually enumerated separately), "Sold Out," "Private" (which is very important to our system because it allows calendar owners to indicate whether or not they want to share an event with other calendars), "Supplemental Info" (about attending the event, not the event's

content), and "Reservations Recommended."

After the modeling team finished the consolidation process, we had a list of most of the data elements that we would use in the conceptual model. Later as we created the model, we would add some other elements that we thought were necessary for the domain but did not find in the calendars. The group also refined some elements that were not initially represented in as robust a manner as we would have liked. For example, we changed the structure of most of the elements relating to event repetition because we thought the calendars we analyzed didn't do an adequate job of representing repetition. As mentioned previously, the group modeled the repetition rules for the Event model directly from the iCalendar specification.

Component Assembly

The Document Engineering Approach

Component Assembly is the re-assembling of the consolidated list of semantically unique components into a structure of functionally dependent aggregates. This is called the conceptual model or Document Component Model, and is often encoded in a UML (Unified Modeling Language) Class Diagram. This model does not represent the structure of a single document, but rather "defines all potential document structures that might be required in our context of use."[DocEng] The creation of this model can sometimes be accomplished using a heuristic, informal design approach that groups data elements into aggregates that are obvious; the different data elements in a mailing address are a good example of things that obviously compose an aggregate. However, a more rigorous approach is the use of data normalization techniques, which are based on the concept of functional dependency, to determine aggregates. This approach says basically that an element is functionally dependent on another element if when the subelement changes, the aggregate element changes. For instance, if you have an "Address" element composed of "Street," "City," and "State," the aggregate element "Address" is considered to have changed if the "Street" changes.

The Event Modeling Process

Scope & Level of Modeling Granularity

As the modeling team started to work on the Document Component model, which we encoded as a UML Class Diagram, the question of what to include in the model as well as what level of granularity we should use came up again. Some of our advisors suggested that we should determine the level of granularity in a data model based on the requirements and business rules of the domain they are modeling. This meant that an element should be broken down into subelements if it was important in this domain to be able to determine the values of the subelements separately. For instance, would we ever need to know the exact second an event began or ended?

To answer these questions, the modeling team had to ask again what the scope of our efforts was. We initially had hoped to create a model of an Event that would serve both our intended application as well as any future needs the campus might have in terms of collecting information on events. However, the group realized that without significant additional analysis, we could not adequately predict all possible future applications of this Event model. Thus we again decided that we would for the most part restrict our modeling efforts to the domain of calendar events. At the same time, we would watch for elements that would make sense as part of our calendar Event model, but would significantly increase the robustness of the model if it were applied to other Event-type domains. For instance, it might make sense to eventually be able to use our Event model to describe class meetings of courses on campus.

Data Element Normalization

The modeling team began the creation of the conceptual model by taking all the elements in the Consolidated list of Content Components and determining which ones had functional dependencies on each other in order to create aggregates. For example, we created an element called "DateTime" which indicated the point (or points) in time that the event was occurring. It included all subelements relating to date

or time, such as "StartTime," "EndTime," "StartDate," and "EndDate." As mentioned previously, we often determined the level to which we should break down elements by trying to ascertain whether we would ever need to know their component parts separately in our domain. Additionally, if we could easily parse out the component parts if necessary, we generally did not break the elements down any further. For example, we decided not to break down "StartDate" into "Day," "Month," and "Year" because we could parse those components out of "StartDate" if necessary, and did not feel it would be helpful to add that additional level of complexity to the model.

The modeling team also tried to reuse elements that had already been defined by existing standards groups wherever possible. We decided to use the element that the Oasis Universal Business Language (UBL) Technical Committee[UBLTC] had defined for "Address" in their 1.0 Beta Release.[UBLReusable] We also looked at UBL's "PartyType"[UBLReusable] element as a possible way of defining both a person and an organization, but found that it contained elements that we did not need, such as "Party Tax Scheme" and "Language." We also considered using the "PersonType" defined by BABL in its "Roles" schema.[BABLRoles] However, found that it was, again, too specific to that domain, and included elements that we didn't need in our model, such as "Gender," "Marital Status," and "Ethnicity." We did, however, end up using the "Personal Name Type," which defined a the structure of a name for a person, from the BABL Common Components Type.[BABLCCT]

Strict Normalization vs "Core + Contexts"

As the modeling team went through Component Assembly process, we were trying to reconcile two approaches to creating a conceptual model that differ in their prescriptiveness and formality. One approach was to follow the strict normalization procedures described above, only creating aggregates that had clear functional dependencies. A schema could then be created by following relationships through the model in any direction, picking up components along the way. It is suggested that this more "bottom up" method not only allows an analyst to see patterns in the data that could define reusable aggregates (e.g. "Contact Info" includes "Email," "Address," "Phone Numbers," etc.), but also allows the analyst to create better models by understanding the actual semantics and business rules which define the relationships between components.

Another approach was to follow what the second author had defined in his SIMS Document Engineering course as the "Core and Context" methodology. In this system, a set of "Core" elements that comprise the absolute definition of an event would be chosen, and then additional "Contexts" would be created that could be added to the "Core" to create models that would be appropriate for different calendars. This is a more top-down and heuristic approach where the Document Engineer looks at the documents within the problem space and figures out what makes them unique as a way to determine aggregates.

For instance, we could have decided that every event must have a "Core" composed of Title, Description, and DateTime. We might even decide that one of these elements is optional, but that these things define the "Core" of what an event is. Then we would add contexts, which would define optional "add-ons" to the Event model, such as "Location," "Recurrence," "Sponsors," "Sports," "Arts," or "Attendance Restrictions." This was the approach used by the spring 2003 Document Engineering course that created the first Event models that were part of this project. An Event schema could then be constructed on a calendar-by-calendar basis by starting with the "Core" module and adding other "Context" modules as needed.

In our case, after creating a UML model following strict normalization procedures, the modeling team realized that we had a very granular model which numerous child elements and not many distinct groups of elements. For instance, we had elements such as "Refreshments" and "Supplemental Info," for which it might have been more logical to find an aggregate that encompassed them. It seemed that process of determining groupings based entirely on functional dependencies did not seem to work well in this domain. For example, when trying to determine what was functionally dependent on Event, it was argued that maybe nothing really was, because theoretically you could change the title as well as the description and be referring to the same event. This may be because there isn't a clearly defined model of an Event within the world; it is a somewhat different concept to different people. What something like an "Address" is composed of is much more well-defined. Additionally, we had scoped out domain-specific ("Context") elements such as those relating to "Sports" or "Arts," but we were not sure how we would fit

them into a strictly normalized model. Following a strict functional dependency methodology, many of these elements (e.g. "Opponent," "Score," "Media," "Artist") would end up being individual, disaggregated components, and there would be no easy way for campus calendar owners to "add the arts context" or the "sports context."

In the end the Event schema was developed using both normalization procedures and a more heuristic definition of a "Core" set of elements along with "Contexts." These "Contexts" are really aggregates that can be used to augment the "Core" model if necessary, depending on the needs of the calendar. It seems that normalization procedures are most valuable as you begin to build the model, allowing the analyst to determine business rules, relationships between the components, and opportunities for reuse. Later on, depending on the domain, it may make sense to create additional aggregates or groupings not necessarily based on functional dependencies to make the model more segmented and approachable to the users.

Document Assembly

The Document Engineering Approach

In this phase of the Document Engineering process, an actual Document Model is created from the Document Component Model. The Document Component Model represents all possible associations between the components it contains. A Document Assembly Model, however, is created by choosing a root component and "walking through" the associations of the Document Component Model to create a hierarchical document.

The Event Modeling Process

When creating a Document Component Model, or conceptual model, a Document Engineer may choose to enumerate all possible associations within the model. This could be represented by associations within the model that move in multiple directions (e.g. an "Event" has a "Location," "Locations" have "Events", and "DateTimes" may 'contain' "Events"). However, because it was clear to the modeling team from the beginning of the project that the Event model was our final deliverable, we did not model these types of multi-directional associations. In our Document Assembly Model, "Event" became the root element and other elements and aggregates became children of "Event."

Implementation

The Document Engineering Approach

The last phase the Document Engineering process is to represent the models created in some physical form so they can be used in applications. One way to do this is to encode the model in an XML Schema. Using a model that has been encoded in a physical form ensures that for software applications developed based on the model, "the rules about information and process that are captured by the model remain explicit or externalized from the software that enforces them."[DocEng]

The Event Modeling Process

The final step was to begin the "implementation" process by encoding this Document Assembly Model into an XML schema. The final version of the Event model can be found at http://dream.berkeley.edu/EventCalendar/Events.xsd.

UC Berkeley Calendar Network[UCBCN]

The Event model forms the basis for the development of all the tools the Berkeley Calendar Network team has created. We believe these tools will fulfill the needs of a majority of calendar owners on the UC Berkeley campus, and will ensure that these calendar owners are able to easily switch to the Berkeley Calendar Network system.

This final section will show some preliminary designs for Calendar Management Tool and web-based calendar. As these tools are central to the acceptance of the Event model, they have been the focus of most of our development efforts to date. We are currently developing a complete design and software implementation plan and expect to begin development soon. Further specifications for the Calendar Management Tool may be found in the Berkeley Calendar Network team's final report.[UCBCNFR]

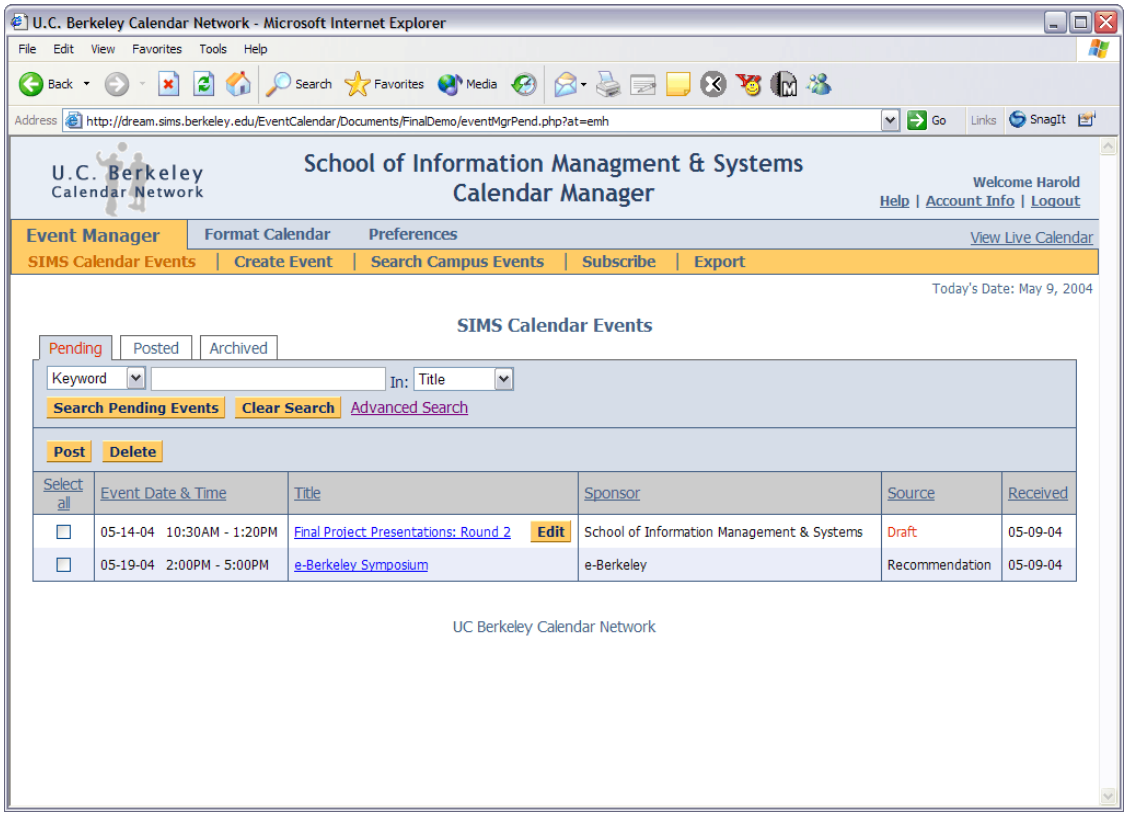
Calendar Management Tool

Event Manager

The Event Manager comprises the main set of functionality in the UC Berkeley Calendar Network Tool. It provides all of the functionality needed to manage events. This includes:

- Creating, editing or deleting events
- Posting or removing events from the calendar
- Searching for events on other calendars within the network
- Setting up subscriptions for particular types of events from other calendars within the network
- Exporting event data out of the system

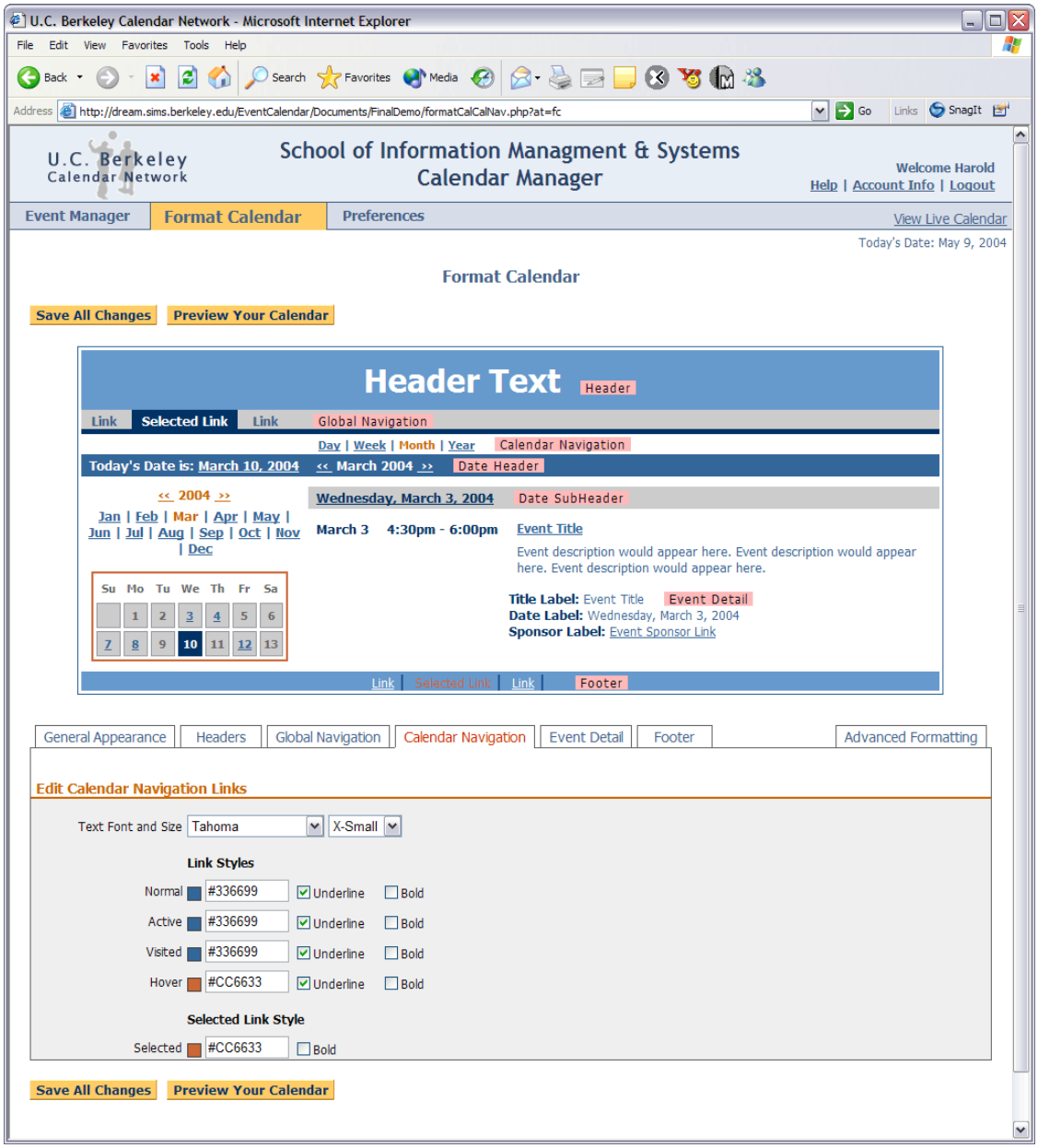
Figure 4. Event Manager



Format Calendar

The Format Calendar page allows calendar administrators to customize their calendar's 'look-and-feel' by modifying the settings in each of its five sections: General Appearance, Headers, Global Navigation, Calendar Navigation, Event Detail and Footer. These sections are displayed as separate tabs within the body of the page. The application then generates calendar views with a XSL transform and CSS file stored within the system. For more advanced users who want to have greater control over the appearance of their calendar beyond the customization settings of the application, they can choose to replace the default XSL transform and CSS files with their own in the Advanced Settings Tab.

Figure 5. Calendar Navigation



Web-based Calendar

This calendar is created by applying an XSL transform & CSS file to an XML document conforming to our Event schema. This is an important feature in the system as it allows for the separation of content from presentation. The XML document which is returned to the web browser will be created based on the type of information requested in the query to the repository. This document can then be styled with either the standard XSL transform and CSS file, or a customized XSL transform and/or CSS file which will allow calendar administrators to replicate their own website's "look and feel." This was an essential feature for many of the different campus calendars, as branding and integration with their current website is often a concern.

Figure 6. Horizontal Navigation Grid View

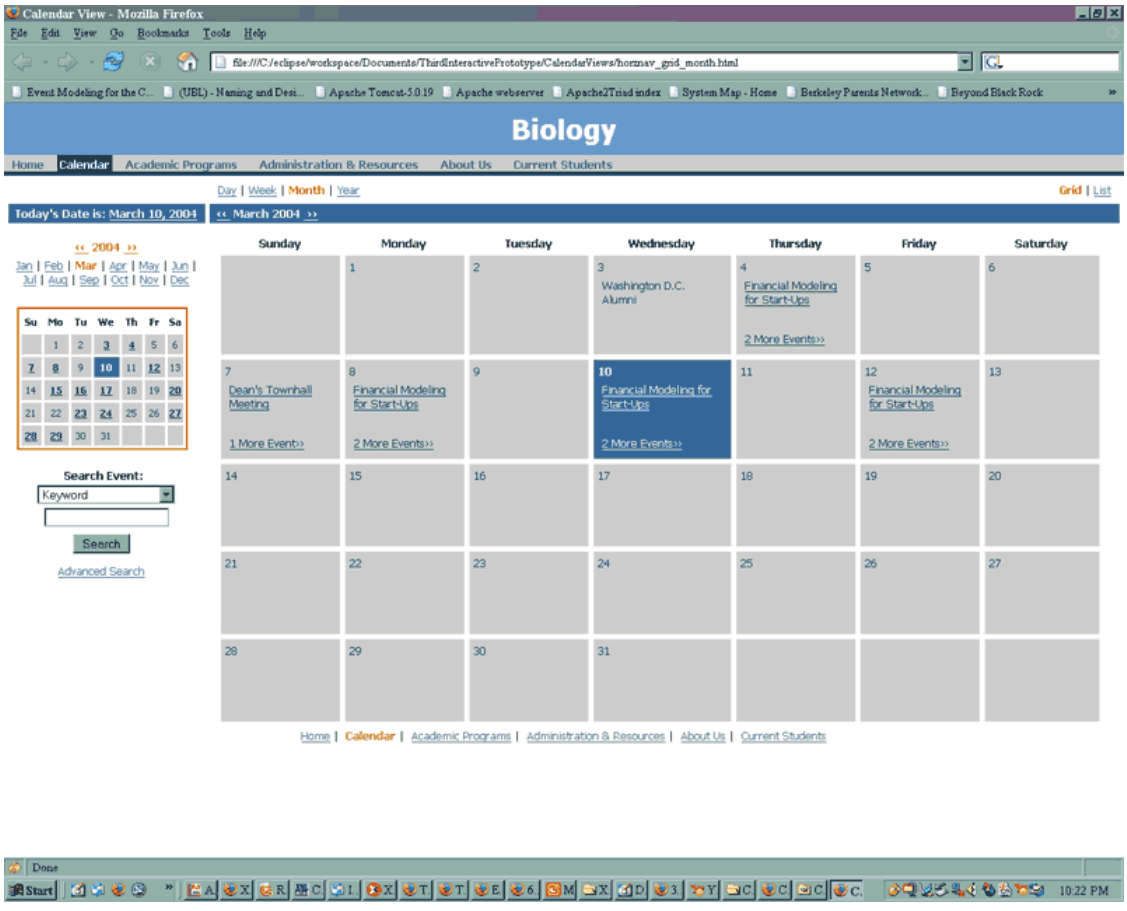
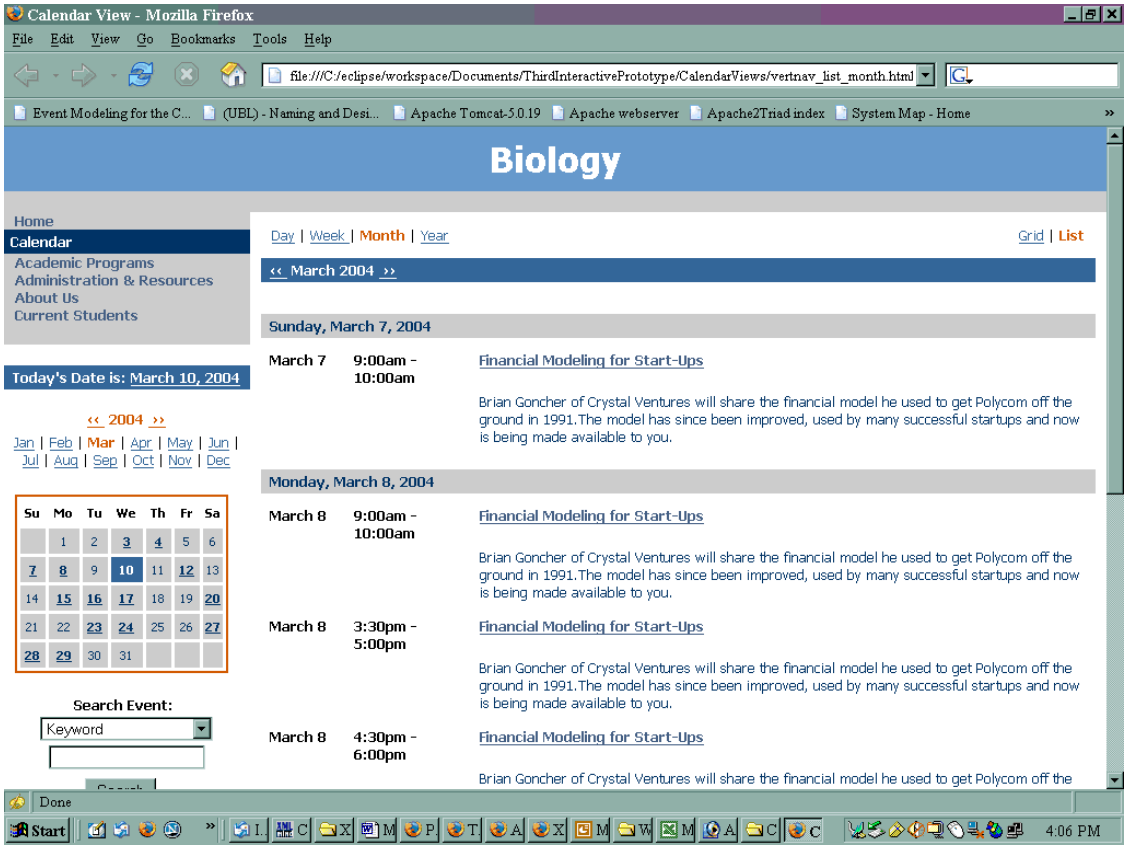


Figure 7. Vertical Navigation List View



Many thanks to the second author of Document Engineering, Tim McGrath, for his guidance on applying the Document Engineering methodology to this project; all members of UBL who advised the group on the Event modeling phase of the project, including Sue Probert, Dave Carlson, Lisa Seaburg, Jean McInerney, Chee-Kai Chin, and Stephen Green; Professor Marti Hearst for her help with the Berkeley Calendar Network interface and guidance on user-centered design principles; Nadine Fiebrich, Zhanna Shamis, and Myra Liu for their hard work on the development of the Berkeley Calendar Network (BCN); Carolyn Cracraft and Alex Milowski for their assistance with the BCN project; Jon Conhaim and Jeff Kahn of UC Berkeley for their strong support of the BCN project; Sara Leavitt, Kathleen Connors, Jeff McCullough, Sarah Jones, and Mimi Mugler for their diligent work on the Event model; and all the UCB calendar administrators who participated in our interviews and user testing.

Bibliography

[BABLCCT] *Berkeley Academic Business Language CommonComponentTypes.xsd.* Available at: http://dream.berkeley.edu/doc-eng/xml/xsd/babl/schemas/babl_0p02/CoreComponents/CommonComponentTypes.xsd

[BABLRoles] *Berkeley Academic Business Language Roles.xsd schema.* Available at: http://dream.berkeley.edu/doc-eng/xml/xsd/babl/schemas/babl_0p02/AggregateComponents/Roles/Roles.xsd

[CDE] *Center for Document Engineering.* Available at: http://cde.berkeley.edu/about/

[DesDocTWS] *Document Engineering: Designing Documents for Transactions and Web Services*, Glushko, R., XML 2003 Conference Presentation, 8 Dec 2003. Available at: ht-

tp://www.sims.berkeley.edu/~glushko/xml2003/slide37.htm

[DocEng] *Document Engineering*, Glushko, R., & McGrath, T., MIT Press, In press.

[iCalendar] *IETF RFC 2445: Internet Calendaring and Scheduling Core Object Specification (iCalendar)*, Dawson, F., & Stenerson, D., IETF, Nov 1998. Available at: http://www.ietf.org/rfc/rfc2445.txt

[SKICal] *SkiCal - An extension of iCalendar*, Internet-Draft, Fitzpatrick, G., Lanner, P., Hjelm, N., IETF, July 2001. Available at: http://skical.metamatrix.se/skical20010905.html

[SKICal-iCal] *SKICal and iCalendar.* Available at: http://skical.metamatrix.se/eng/icalendar_eng.html

[SKICalSite] *SKICal: Structured Knowledge Initiative Calendar.* Available at: http://skical.metamatrix.se/. [http://skical.metamatrix.se/]

[UBLReusable] *UBL-Reusable-1.0-beta.xsd.* Available at: http://www.oasis-open.org/committees/ubl/lcsc/UBLv1-beta/xsd/common/UBL-Reusable-1.0-beta.xsd

[UBLTC] *Oasis Universal Business Language Technical Committee.* Available at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl

[UCBCN] *UC Berkeley Calendar Network.* Website: http://dream.berkeley.edu/EventCalendar/

[UCBCNFR] *UC Berkeley Calendar Network*, Bloodworth, A., Fiebrich, N., Liu, M., Shamis, Z., 15 May 2004. Available at: http://dream.berkeley.edu/EventCalendar/Documents/FinalReport/Website/

[xCal] *iCalendar DTD Document (xCal)*, Internet-Draft, Dawson, F., Reddy, S., Royer, D., Plamondon, E., IETF, 15 Feb 2002. Available at: http://www.ietf.org/proceedings/02mar/I-D/draft-ietf-calsch-many-xcal-01.txt [http://www.ietf.org/proceedings/02mar/I-D/draft-ietf-calsch-many-xcal-01.txt]