

Community Event Calendar System: Project Synopsis

Steve Talbot, University of Cambridge

June 2004

Cambridge is an excellent example of a high-technology city located in an otherwise fairly rural and sparsely-populated area. The surrounding villages house many of the city's workers and some small, technology-based businesses. This generates considerable demand for broadband Internet services. However, current ADSL¹ technology is not economically viable with only a few customers, so telephone companies have not installed it in the exchanges serving the villages. Furthermore, most villages are not served by a cable Internet provider.

A viable solution is a community broadband network, a local area network (LAN) connecting the homes and businesses in a village. The LAN is connected to the Internet via a high bandwidth link to Cambridge. Studies show that, given typical patterns of Internet use, this is sufficient to support several dozen simultaneous users with connection speeds comparable to domestic ADSL [3].

If a significant proportion of the homes in a village have “always on” Internet access, a range of Internet-based community applications become technically feasible. The idea for such an application—a web-based event calendar and reminder system—came to villager John Harris when he forgot to put out his recycling rubbish and wished he had been reminded to do so. He envisaged a web site allowing members of the village community to share information about forthcoming events, ranging from refuse collections to social activities, with a mechanism for automatically reminding them about important events.

The aim of the community event calendar project was to translate this concept into a working web application for use on a community broadband network in Bottisham, a

¹ADSL (Asymmetric Digital Subscriber Line) is a method of transferring data at high speed over existing telephone lines, commonly used in domestic broadband installations.

village near Cambridge. The work undertaken covered all areas of the software engineering process—planning, requirements engineering, research, specification, design, implementation, testing and maintenance—and has culminated in the production of a free community calendar system, codenamed “Comendar” (an abbreviation of *community calendar*).

The primary requirement of the system was to store and publicise details of community events. To encourage people to use the web site regularly, each registered user also has their own personal calendar (Figure 1), which they can use to manage their time.

To prevent people from forgetting about important appointments and tasks, automated reminders are provided via e-mail (Figure 2). Considerable time was spent researching the characteristics that make a reminder effective [4, 5, 6, 9] and it is believed that this research will set the software apart from many of its competitor products.

Since it was envisaged that many of the system’s users would be computer novices, ease of use was a necessity. The software process involved prototyping of key aspects of the interface, evolutionary development and substantial acceptance testing. At each stage, feedback was sought from users of varying levels of computer literacy. This was combined with research and brainstorming to iteratively improve the design. Clear on-screen instructions are provided and, for the benefit of users who might struggle without further assistance, an on-line user manual has been built into the system (Figure 3).

Security is a serious consideration for web applications. The community calendar has unusual security requirements because its users will probably know one another in person, whereas most Internet communities have fairly anonymous and geographically diverse users. The need for moderation is less, but perhaps the risk of misuse of administrative powers to achieve censorship is greater. A community security model based on peer approval was developed to reflect this.

Events added to the community calendar are categorised by means of “interest groups”, which are groups of users with a common interest. These allow users to find information they are interested in quickly, without being bombarded with irrelevant material. Many of the system’s security rules are based around membership of these groups.

The system was split into two strands for specification, design and development, as illustrated by Figure 4. The underlying “engine”, which is responsible for storage and manipulation of data, was tackled by the waterfall method of software engineering; the user interface was developed by the evolutionary method.

During the specification phase, it was discovered that repeating events is an extremely complicated subject, requiring considerable thought. It was necessary to reach a com-

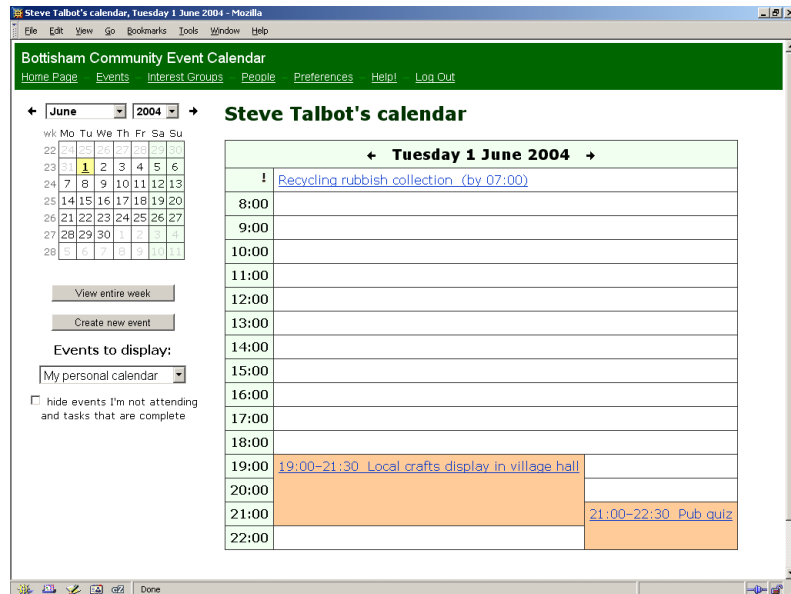


Figure 1: The day view of a user's personal calendar

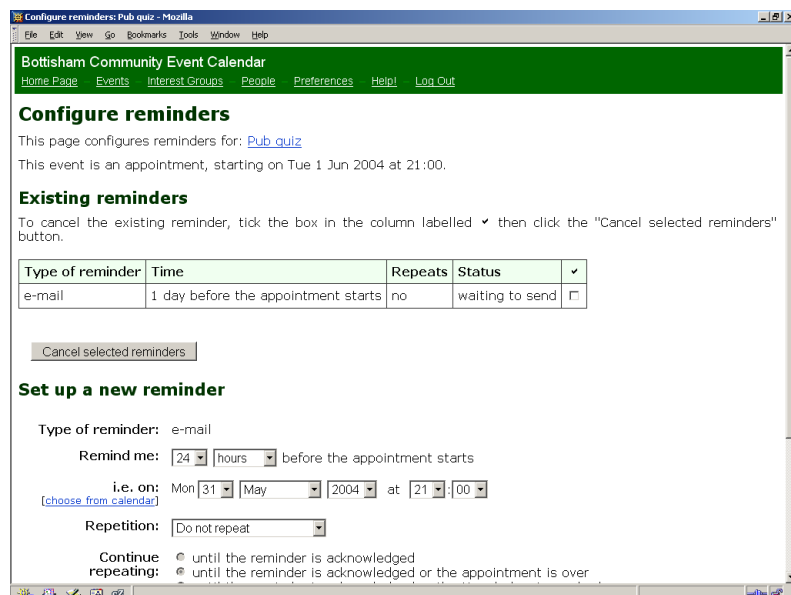
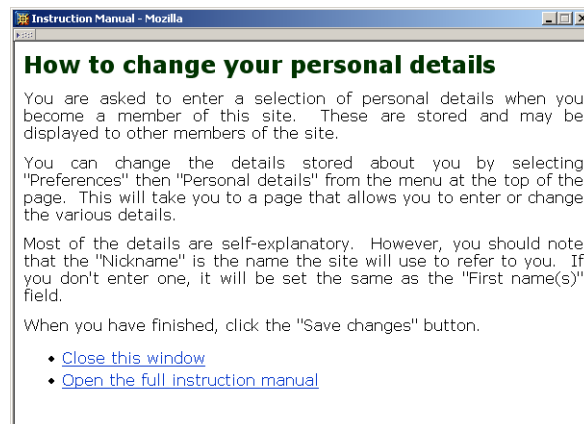
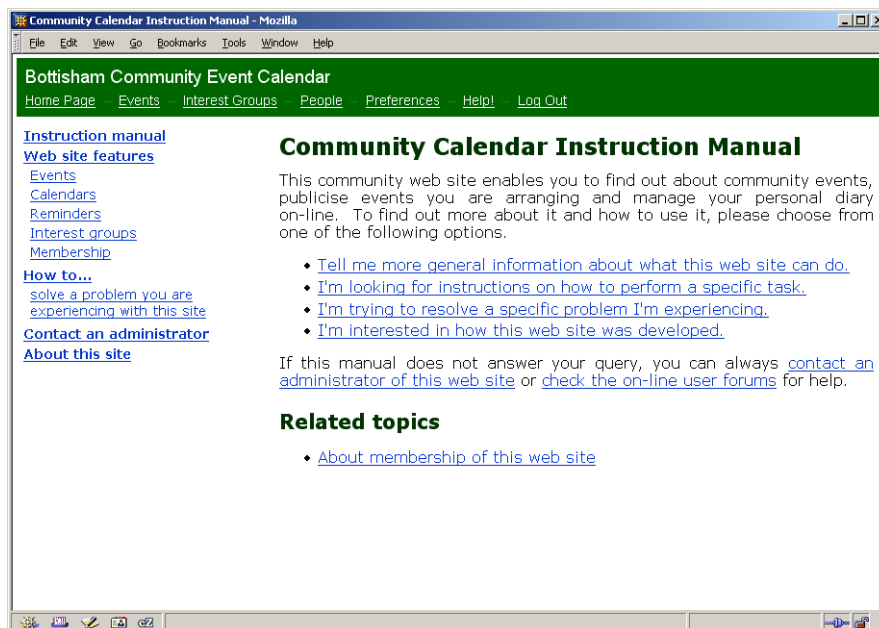


Figure 2: The user interface to set up reminders for an event



(a) Pop-up help window



(b) Contents of the full instruction manual

Figure 3: The on-line instruction manual

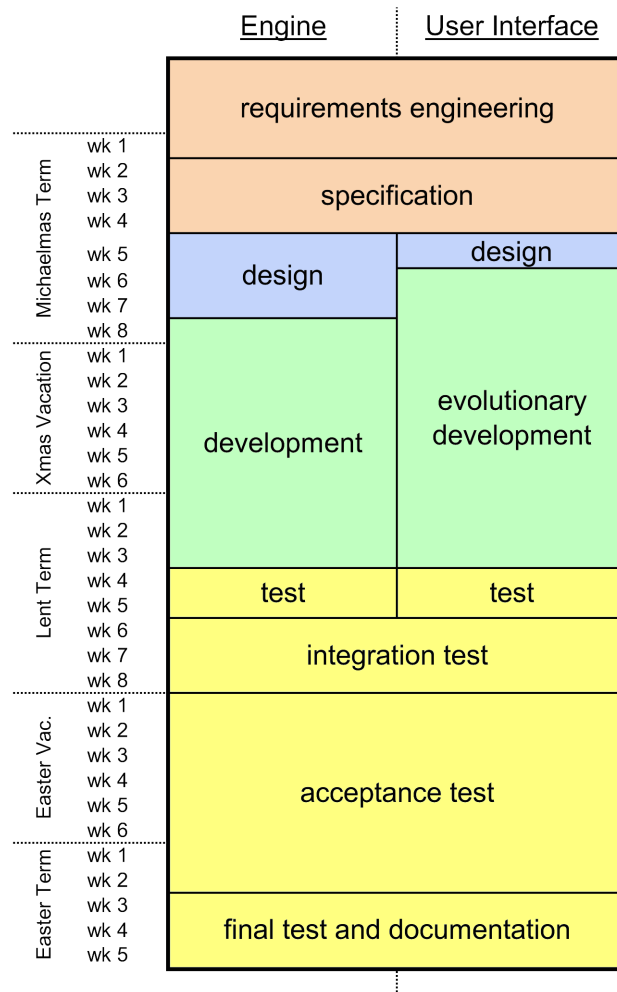


Figure 4: Division of the time available for the project

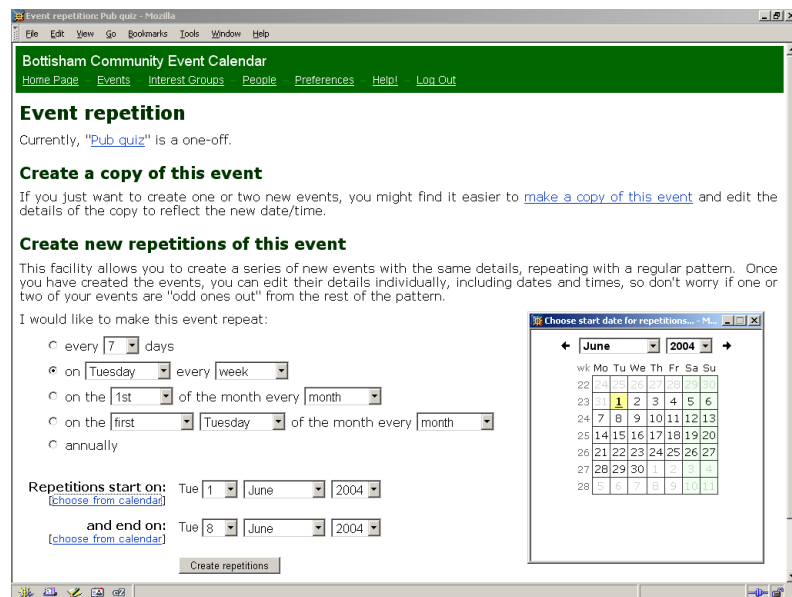


Figure 5: The user interface to set up event repetition

promise between the range of repetition patterns possible and the complexity of the user interface. The chosen design balances flexibility with ease of use by allowing the user to apply several different, simple repetition patterns simultaneously (Figure 5). The user can also make individual copies of events, thus accounting for events that recur with no pattern to their dates.

A custom software solution satisfying the user requirements was implemented using server-side PHP² scripts and a MySQL³ database. These were complemented by client-side scripts written in JavaScript⁴, which enhance the web pages that form the user interface. Attention was given to the production of maintainable code and code was reused where possible.

After consideration of the differences between PHP4 and PHP5 [7], the server-side code was written in PHP4 with an upgrade to PHP5 in mind. It seems this choice was wise. At the time of writing, the latest stable release of PHP is 4.3.6; PHP5 release candidate 2 is available, but is “not recommended for mission-critical use”.

The system makes use of the object-oriented features of PHP to implement a modular

²PHP is a widely-used, open source scripting language that can be embedded into HTML [2]. <http://www.php.net/>

³MySQL is an open source, fast and robust database server [8]. <http://www.mysql.com/>

⁴JavaScript is a scripting language that can interact with HTML pages.

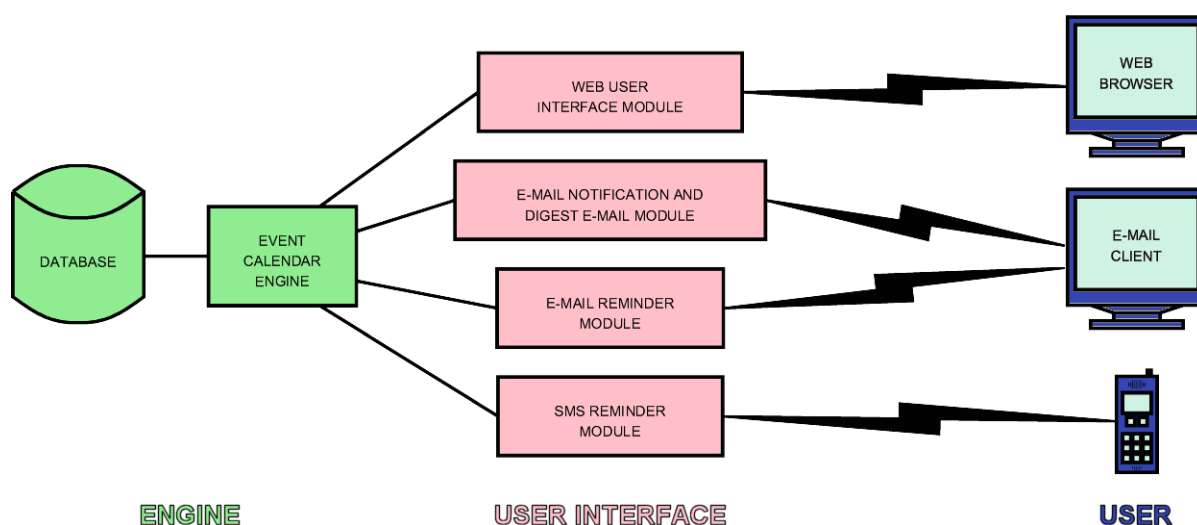


Figure 6: The relationship between the modules of the system

architecture. Developers can create modules to add new functionality without modifying any of the existing code. New reminder mechanisms, such as SMS text messages, could be added in this way.

Figure 6 illustrates the relationship between the modules. The engine module provides a library of functions, which the user interface modules can make use of in presenting the calendar system to the user. For example, the web user interface module generates the web pages the user sees when they visit the site and the e-mail reminder module is responsible for sending the reminder e-mails that a user has requested. Each module must implement a defined interface and must register itself with the engine module when it loads, thereby allowing the engine to communicate with any of the other modules at any time.

The database structure was designed to make efficient use of storage while also allowing common data retrieval tasks to be executed quickly. For example, an event search can be performed by issuing a single query to the database, avoiding the need for significant processing of the data before it is displayed. This makes the system efficient.

The required appearance of key pages of the user interface was determined by prototyping, then the prototypes were scrapped and re-written as PHP scripts. Hence, the evolution of these pages' appearance and logic was effectively separated into two stages. Similarly, the logic behind a page, which determines what information should be displayed, was separated as far as possible from the code that outputs the HTML⁵, which determines

⁵HTML (HyperText Markup Language) is the coding language used to create web pages.

the page's appearance. It is believed that these techniques helped to reduce the number of bugs introduced during development.

A particular challenge faced in the course of this project was the need to work with various different languages. The PHP scripts dynamically generate HTML pages, SQL⁶ queries and occasionally some JavaScript to interact with the user's web browser and the database. Additionally, there is some static HTML, cascading style sheets—which define aspects of the pages' appearance such as colour schemes, fonts and spacing—and JavaScript function libraries—which enhance the user interface. These all needed to be seamlessly integrated to produce the single application that users experience.

A user's instinctive reaction is that any failure of a web application is the fault of the application developer. Unfortunately, some web browsers contain bugs, developers of different browsers interpret the various web standards slightly differently and some browsers—especially old versions—do not implement the standards completely. Coping with some browsers' buggy implementation of style sheets and dynamic HTML and ensuring that all pages were usable without JavaScript, style sheets or graphics was a further challenge that was overcome. “Graceful degradation” was achieved; in other words, the site remains usable in the absence of JavaScript and style sheet support. It is even fully usable in a text-only web browser.

Because of the usability requirement, testing has been a major focus of this project, as shown by Figure 4. Sixteen volunteers were recruited to assist with the various stages of testing, in three small teams:

- a group of highly computer-literate Cambridge students and recent graduates;
- a group of computer novices with no village connection; and
- a group of villagers from Bottisham.

Their bug reports and feature requests were used to iteratively improve the pre-release versions, both in terms of reliability and user interface functionality. During the course of testing, the system has been installed on servers running both Linux and Windows operating systems and its web pages have been successfully viewed with a variety of web browsers under several different operating systems.

Of course, not everything worked first time; the project suffered a few setbacks. For instance, the village server was not fully set up by the time the first version of the calendar

⁶SQL (Structured Query Language) is a standardised language for retrieving information from and updating a database.

was ready for testing. When the server was available, it was found to have an inappropriate installation of PHP to run the calendar software. However, sufficient time had been planned to account for unforeseen difficulties and the project remained on schedule.

Table 1 shows how the project evolved. The large size increase from version 0.1 to 0.2 is mostly due to the inclusion of the on-line instruction manual.

Table 1: Event calendar system release history

Version	Date	Files	Total size	Lines of code ^a	Comments ^b
0.1	9 Mar 2004	83	804 kB	12,071	2,948
0.2	8 Apr 2004	131	1,003 kB	14,125	3,221
0.3	22 Apr 2004	134	1,025 kB	14,334	3,264
0.4	6 May 2004	135	1,065 kB	14,717	3,355
0.5	17 May 2004	138	1,106 kB	15,270	3,461
1.0rc1	June 2004				

^a This measure excludes comments and blank lines.

^b This measure excludes the public licence header at the top of every file.

The finished community calendar system is now installed and in use in Bottisham, where it seems to have been well received. The source code of the project and accompanying documentation, including installation instructions and the full specification, has also been released under public licence via SourceForge⁷. It is hoped that the system will encourage and promote interaction between the villagers of Bottisham and that it will be useful to any other community that wishes to set up its own web calendar.

<http://comendar.sourceforge.net/>

⁷“SourceForge.net is the world’s largest Open Source software development website, with the largest repository of Open Source code and applications available on the Internet.” [1]

References

- [1] *SourceForge.net Welcome Page*. <http://sourceforge.net/>.
- [2] Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski, and Jouni Ahto. *PHP Manual*, chapter 1. April 2004. <http://www.php.net/docs.php>.
- [3] FlexWork. *Regional case study: Community broadband networks in the east of England*, January 2003. <http://www.flexwork.eu.com/members/regions/cambs.pdf>.
- [4] John E. Harris. External memory aids. In M. M. Gruneberg, P. E. Morris, and R. N. Sykes, editors, *Practical Aspects of Memory*. Academic Press, London, 1978.
- [5] J[ohn] E. Harris. Remembering to do things: a forgotten topic. In J[ohn] E. Harris and P. E. Morris, editors, *Everyday Memory, Actions and Absent-Mindedness*. Academic Press, London, 1984.
- [6] John E. Harris and Arnold J. Wilkins. Remembering to do things: a theoretical framework and an illustrative experiment. *Human Learning*, 1:123–136, 1982.
- [7] php.net. *Changes in PHP 5/Zend Engine II*, March 2004. <http://www.zend.com/php5/articles/engine2-php5-changes.php>.
- [8] Michael Widenius and David Axmark. *MySQL Reference Manual*, chapter 1. O'Reilly, June 2002. <http://dev.mysql.com/doc/mysql/en/Introduction.html>.
- [9] A[rnold] J. Wilkins and A. D. Baddeley. Remembering to recall in everyday life: an approach to absent-mindedness. In M. M. Gruneberg, P. E. Morris, and R. N. Sykes, editors, *Practical Aspects of Memory*. Academic Press, London, 1978.