

Evolution of the Electronic Calendar: Introducing Social Calendaring

Chris Lord
University of Southampton
School of Electronics and Computer Science
University Road
Southampton, SO17 1BJ
cil103@ecs.soton.ac.uk

ABSTRACT

With the advent of Web 2.0 technology, traditionally solitary applications are becoming more socially aware. An important feature of modern applications is their ability to make a user aware of what other users are doing. This can be seen in the popularity of new social networking and resource sharing web-sites. In this paper, it is suggested how features that support social translucency can be incorporated into electronic calendar applications. A brief history of electronic calendars is followed by an analysis of modern calendaring solutions, and finally a framework of current technologies is suggested to bring social awareness to the electronic calendar.

1. INTRODUCTION

Since the mid-80's, various products have aimed at augmenting the basic paper calendar using electronics. These devices have evolved over time and transformed the process of scheduling, by easing the process of recording and providing the ability to share events. It is argued by Dieberger[3], and Erickson and Kellogg[4], that there is value in making online systems support social navigation and awareness. In this paper, ideas that would allow online interaction and social features in electronic calendars are explored. Many other forms of data storage have followed the 'social' route (e.g. Web bookmarks, photo albums, personal diaries) and it is proposed that by integrating this new aspect, new opportunities will be opened up to all users of such a system. In this paper, the history of electronic calendars (both hardware and software) is discussed and a framework of current technologies is devised to support a socially-aware calendar application.

2. HISTORY

This section is intended to provide a brief overview of the history of the electronic calendar, as it pertains to the idea of social calendaring. As such, there is a focus on schedulers

and groupware applications, as opposed to detailed coverage of the many calculators and personal digital assistants (PDAs) that feature calendar functions.

The patent for the invention of the electronic calendar was granted to Alfred B. Levine in 1979 (U.S. Patent number 4162610). This patent describes a simple interface that, among other things, would allow a user to store a record of future events on a central memory, with the ability to recall, amend and delete these records, using a keyboard for input. This patent was filed in 1975, predating both PDAs and the availability of such features in portable electronic devices. At the time it was granted, portable devices (such as the Lexicon LK-3000) were just starting to incorporate such features[6].

Electronic calendars started to become important (and arguably, useful) with the introduction of groupware software. Groupware software is intended to enhance group-working experiences; to aid people to communicate and work together in collaborative environments[2]. The first groupware application came in the form of a WordPerfect extension in 1986 that went under the name of WordPerfect Library[12]. This allowed for shared contact management and calendaring between users, and would later become Novell GroupWise.

The success of groupware applications almost certainly influenced the design and functionality of PDAs, which tend to include similar functionality with less of a group-emphasis. Such PDA software is often collectively called Personal Information Management (PIM) software. The term PDA was invented by Apple CEO John Sculley in 1992 and was used to describe their new device, the Apple MessagePad (commonly referred to as the 'Newton', after the name of the OS it ran)[9]. The capabilities of electronic calendars were furthered on PDAs such as the Palm and PocketPC series, but the focus has always been on a single-user. Most PDAs offered the ability to synchronise with a desktop-based groupware application and the ability to transfer events between other supported devices.

The state of electronic calendars today is in flux, as new web-based services and hybrid devices are introduced. Most mobile phones today, for example, include calendaring capabilities, and also quite common is the ability to synchronise with an online storage. This allows a person to have a central

storage for their calendar and access anywhere there's Internet connectivity. With the advent of Web 2.0 technology, 'mash-ups' of different technologies and social translucency are becoming increasingly common. It is almost inevitable that electronic calendaring will evolve into a new form in the near future.

3. EXISTING TECHNOLOGY

Four existing services are detailed that provide a starting point for a fully-featured social calendar. The study of these existing technologies will help determine the viability of a social calendar, and also influence what features a solution would include. A brief overview of the iCalendar specification[1] is also included to provide some context for these summaries. With the exclusion of the aforementioned specification and SyncML, these services all embody Web 2.0 features, such as tagging, 'hackability' and rich, AJAX interfaces[10].

3.1 iCalendar

Internet Calendaring and Scheduling Core Object Specification (iCalendar, hereafter referred to as 'iCal'), RFC2445, is a specification intended to provide a common format for the exchange of calendaring information over the Internet[1]. iCal is well supported by most modern calendar applications, allowing exchange of calendar information between different clients. The iCal specification defines how common attributes of events are stored, such as duration, location, repetition and category. In addition to this, it also specifies how to store miscellaneous, extra data that is not explicitly defined in the specification (see section 4.8.8.1 of RFC2445[1]). As long as clients don't discard these extra data fields, this enables custom data to travel back and forth between different clients of differing capabilities. Due to its common use in PDAs and groupware applications, it has become the de-facto standard for calendaring software.

3.2 Google Calendar

Google Calendar¹ provides a basic, iCal-supporting online calendar. The strength and potential of the Google Calendar is that it supports open standards for data sharing and import, and that it has a public, documented developer API, allowing third party software to interact with it. Unfortunately, interfacing directly with Google Calendar is a complicated affair if not using one of the supported languages. In addition to this, only high-level languages are supported. This makes an optimal, native Google Calendar interface more difficult to write than it could be, lowering its portability potential. Furthermore, the interface returns events in Google's custom GData format, as opposed to iCal components, requiring a translation process before interfacing with most other calendars. Google Calendar includes social elements in the form of the ability to search for public calendars and events to import into your personal calendar. Along with this is the ability to invite other users to events on your calendar. As events can hold location information, it is possible to search for events in particular locations, as well as by date. There is no offered integration with external services.

¹<http://calendar.google.com/>

3.3 Mobical / SyncML

Rather than providing an online calendar, Mobical² provides a free, SyncML synchronisation server. This allows any calendar client or back-end with SyncML support to synchronise their local calendar with a persistent online storage. SyncML synchronises iCal components, making it easy to integrate with most common calendars. An increasing number of devices and applications support synchronisation via SyncML. While it is not an ideal solution, SyncML synchronisation does allow a single calendar to be shared across multiple devices. Of course, it is important for a users' calendar to persist independent of the device used to access it. SyncML provides a basic method for non-web calendar storages to be accessed independent of location and client. This is important if specialist devices are to have access. Due to the multitude of different screen sizes and CPU capabilities, allowing a native client to be used provides an advantage over restricting access to a special-purpose interface. Mobical advertises their service for backing up and restoring PIM data, as opposed to a way of keeping multiple devices synchronised.

3.4 30 Boxes

Much like Google Calendar, 30 Boxes³ also provides a basic online calendar. As well as offering iCal export, it offers the ability to share your calendar via a variety of other methods. 30 Boxes goes further than Google Calendar, in that it has provisions for more advanced sharing and integration with other Web 2.0 services (e.g. photo, video and journal sharing services). 30 boxes has the ability to share events with any other 30 boxes users on your friends list, but does not seem to have any ability to search all public events by any criteria. To add events from other peoples' calendars, a user must know their e-mail address and manually navigate to their particular calendar. 30 boxes provides a public, documented API for third party software, allowing applications to add and modify all details specific to a particular user. It would quite easily be possible to extend 30 Boxes using this API, but its relative infancy and lack of popularity make it a poor choice.

3.5 Upcoming

Upcoming⁴ provides many of the same features as Google Calendar and 30 Boxes, but presents them in a wholly different and more socially-aware fashion. Upcoming discards the traditional paper-calendar style display for an 'Agenda' style display. Upcoming could be described as an event-manager rather than a calendar or scheduler. Further perpetuating its image as an event manager rather than a traditional calendar, it even allows export of events directly to Yahoo! Calendar, Google Calendar and 30 Boxes. If an event is public, or created by a user on the friends list, users are able to mark whether they are attending or watching the event. Upcoming lets users see how many users are attending an event and uses this number to create a 'Top Events' list, that it presents on the home page. Upcoming does not allow an event to be entered without a location, and all locations must be registered in the central locations database.

²<http://www.mobical.net/>

³<http://30boxes.com/>

⁴<http://upcoming.yahoo.com/>

Like the other online calendars covered, Upcoming provides a documented, public API for developers.

Upcoming cannot be considered a calendar, however, as it does not support many common calendar functions, such as recurring events or customised categories. The implication of this is that it cannot be integrated with a standard iCal-based calendar in a generic fashion, nor can it be used as a general-purpose organiser. Upcoming expands on the social potential of standard calendars, while neglecting the advanced organisational features that make them useful for other day-to-day purposes.

3.6 Evaluation

The purpose of this paper is to investigate how to make a standard electronic calendar socially aware and translucent. Awareness implies that the calendar has some idea about its user, something that seems to be lacking from the online calendar services covered above. Upcoming goes some way to providing features that would enable an amount of awareness, but focuses solely on planning and attending events. A common theme in many Web 2.0 applications is categorisation, or tagging. As discovered by Vlkel et al[13], adding semantic data to stored information allows more specific searches to be made and informational relations to be derived programmatically. With regards to calendars, tagging would allow recommendation of similar events, providing the system with a level of awareness. Social translucency suggests the ability to partially see what other users are doing[4]. Upcoming is based around the idea of social translucency, and it is suggested that its features of searching and tracking of attendees should be mimicked. To differentiate from previous attempts, it is proposed that it is important to seamlessly integrate socially-oriented features with a traditional calendar interface. To be used effectively, this system should be able to be used from a multitude of existing systems, and provide backwards compatibility where custom extensions are unsupported. The approach taken will be centered around the use of a native client and a local data store, allowing for increased privacy and offline functionality.

4. PROPOSED FRAMEWORK

As this system is intended to work with previously established systems, it is assumed that a basic iCal-based local calendar is available as a base to work upon. Desktop groupware applications, such as Microsoft's Outlook and GNOME's Evolution commonly have an API with which third parties can query event data. The evolution-data-server, for example, is even based on Berkley's DB, allowing for fast searching of events. To support pseudo-semantic query, it is proposed that an extra field be used on events, 'X-SOCIAL-TAGS', which will contain space-delimited tags, related to the event. For an event to be included in queries, this field, along with the location field must be present. Events lacking either of these fields will be excluded from queries. With the guarantee of these two fields, it is possible to search for, for example, 'music in Southampton'. Due to the ambiguity of tags, however, results may not always be what the user searched for. As it is important for a user to be able to maintain privacy, a field representing an event's privacy level must also be introduced. This could be represented in a 'X-SOCIAL-PUBLIC' field, where the field's presence, regardless of content, indicates that an event

should be public. To track popularity, a numerical field, 'X-SOCIAL-ATTENDEES' could be added. To assist in this tracking, it would be necessary to know the originator of said event, so one further field, 'X-SOCIAL-ORIGINATOR' is proposed.

Events could then be searched for, based on their location and descriptive terms. Using a location database would lower the possibility of mismatches when searching for events in a particular place, as it would eliminate mistakes from typos and misspellings. As an alternative to tagging, queries could be enhanced using Resource Description Framework (RDF). RDF allows semantic meaning to be added to arbitrary resources[8], via the use of so called 'triples'. Triples consist of three parts; The subject, the predicate and the object. For example, the statement '*United Kingdom has the abbreviation of UK*' has a subject (*United Kingdom*), a predicate (*has the abbreviation of*) and an object (*UK*). Utilising RDF would provide true semantic meaning to events, and using an RDF query language (such as SPARQL), would allow for semantic queries, such as '*All live-music events in the Southampton area*'. The disadvantage of RDF, versus tagging, is that it is a more complex task to provide an intuitive interface for editing RDF relationships and queries.

The act of automating queries to form recommendations could be handled by agents. Making recommendation an external process reinforces the modularisation of the system, and isolates the recommendation process, making it simpler to improve and upgrade as technologies progress. The described framework can support both information filtering (IF) and collaborative filtering (CF) agents. An IF agent in the context of this framework would make recommendations by, for example, calculating the frequency of particular tags and locations and searching for similar events. A CF agent would work in a similar way, by again calculating the frequency of tags and locations, but would make recommendations based on the public events of users with similar distributions. Taking the work of Good et al[5] into account, it would be advantageous to somehow combine these two methods, as it is proven that the combination of the two is more effective than either alone. Location of events can be used not only to determine where a user is, or is likely to be, at a particular point in time, but can also be used to determine how far a particular user is willing to travel. This should be the first, and most important, criteria when recommending events. The second criteria will be tag frequency, and can use both IF and CF methods for recommendation. An initial set of viable events can be built using pure CF methods. This would be done by identifying other users in similar locations, with similar most-used tags and collecting their public events. These recommended events can then be strengthened using IF method, by promoting those events with strong correlations between their tags and the users most-used tags. It is important not to weight the IF method so highly as to nullify the CF method.

As previously mentioned, the proposed system is decentralised. That is, clients will communicate in a peer-to-peer fashion, with the primary data storage being local. This simplifies layering this system over an existing system, as data storage is not affected. Due to the nature of the queries to be performed, using a peer-to-peer protocol based on a

distributed hash-table, such as Kademlia[7], is not feasible. Such a network is more suited to searching for unique items, as opposed to collections of items based on varying criteria. Instead, it is proposed that a peer-to-peer protocol similar to that of Gnutella⁵[11] is used. The Gnutella network has no central server, or tracker of connected clients. To become part of the network, a client must know the IP of an existing member of the network. When connecting to the network, your presence is broadcast to your known client-list, who broadcast the message to their known client-list, up to an amount of hops, before stopping. Receiving clients then ‘back-propagate’ a reply, allowing a newly connecting client to quickly populate its known client-list. Queries are performed in the same way, with the result message including enough details to establish a direct connection with the responding client and transfer the resulting data. As the Gnutella network is designed for file searches for clients that are connected for long periods of time, some modifications may be necessary. It is suggested that the locally maintained client list should be based on the same criteria that the CF agent works on, i.e., known clients should be ranked on their geographical distance to the local client. This would reduce the time recommendation queries take, as the most likely clients to respond with viable recommendation events will be asked first. As it is unlikely that any interface would stay open for a long amount of time, the connection and recommendation agent should be run as a daemon/service, to increase the amount of time a user will stay connected. Results can be collected constantly and refined over a period of time, to increase accuracy and quantity of results. To track popularity, when a user subscribes to an event, the originator of the event (found by checking the ‘X-SOCIAL-ORIGINATOR’ field) is notified. If the originator does not respond, the daemon can retry periodically. This requires an amount of bandwidth from the organisers of very large events, however, it is safe to assume that such organisers would also have sufficiently wide connections to handle the demand.

Accessing a particular calendar from an arbitrary location is complicated by the peer-to-peer nature of the proposed communications system. It cannot be expected of every user to install their own CalDAV setup, or indeed to have a server to install such a setup on. So, it is suggested that the ability to access a single calendar from multiple locations be accomplished via synchronisation. For most situations, existing synchronisation infrastructure can be used. For example, PDAs often come with synchronisation software that ties into popular Groupware applications. To synchronise between different desktop computers, SyncML is recommended, for which several free services exist (including the previously covered Mobicall). In the case that neither of these solutions are appropriate, there exist synchronisation frameworks (e.g. OpenSync⁶) that may be suitable. Indeed, using OpenSync, synchronisation between the local calendar and an online calendar, such as Google Calendar, could be performed. This would allow a user to use Google’s web interface, where a native interface is inaccessible, or when synchronisation is not possible. It would be important in this case for such synchronisation to be automated, as caus-

⁵<http://www.gnutella.com/>

⁶<http://www.opensync.org/>

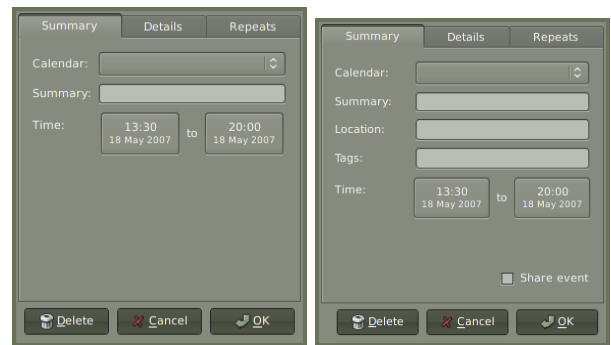


Figure 1: Illustrating the modifications necessary for event editing

ing the user to manually initiate the process highlights a flaw of the peer-to-peer paradigm.

5. PROPOSED USER INTERFACE

For the best user experience, where possible, native clients should be modified to integrate the new features of the social calendar framework. It would be impossible to integrate into the various, closed groupware clients that are common on desktop computers, however, so it is suggested that a separate application be provided that provides an interface to all the new features provided by the framework, similarly to the Juggler system[3]. This separate application will be detailed after the discussion of necessary modifications to existing clients. To create such an interface, the new, user-facing features must be identified. They are considered as follows:

- Tags on events
- Public/Private marker on events
- Attendee count for public events
- Public event searching
- Public event recommendations

Along with these new features, emphasis must also be placed on the existing feature of event location. This can be achieved by putting the location field in a more prominent place, and alerting the user if it’s empty when trying to make an event public. For the first two new fields, modifications will be necessary to the event-editing interface. Figure 1 shows a modification of a current calendar application⁷ to support these new features. The attendee count would require a modification to the event viewing interface, as shown in figure 2.

Searching for events may be possible to integrate into the applications current search feature, depending on how that feature works, but it is more likely that a new interface would be necessary. A ‘Public Event Search’ option could be included near to wherever the current, local event search is situated. It is suggested that the public event search be

⁷Dates - <http://pimlico-project.org/dates.html>

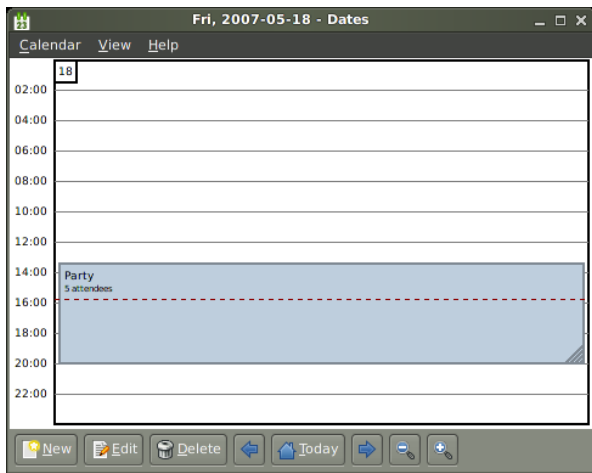


Figure 2: Showing the attendee count on an event

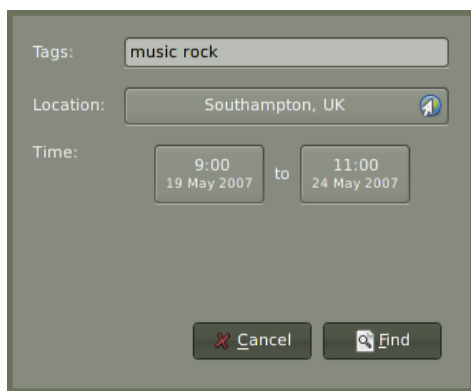


Figure 3: Initiating an event search

implemented via two dialogues. The first dialogue would allow the user to input search terms and initiate the search (see figure 3), the second dialogue would indicate search progress, display results and allow the user to import events into their calendar. As the queries are communicated over a peer-to-peer network, it is important that search results and progress are reported asynchronously. A prototype for such a dialogue is shown in figure 4.

Recommended events could be integrated in a number of ways, two of which will be detailed. The first is to show recommended events on the calendar, using some highlighting method to show that they are not events in the user's calendar. This view could be toggled on and off, and would allow for fast importing of recommended events, simply by activating them. Such an interface is illustrated in figure 5. The downside to this technique is that, depending on the amount of recommended events, it may make the user's calendar appear cluttered. This is easily remedied by imposing limits on the amount of recommended events for a particular time-space, but then the user may miss events they would otherwise be interested in. Another potential problem is that by seeing the recommended events in their calendar, a user may never actually import these events, meaning the popularity of the event would not increase. A second way

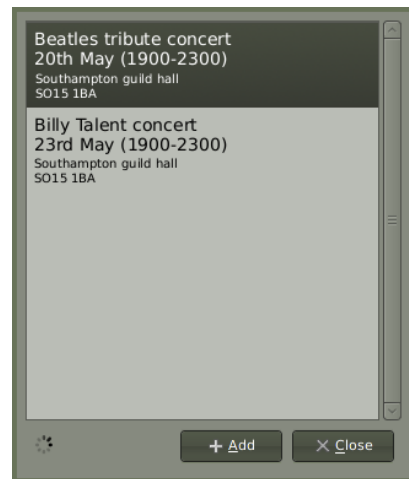


Figure 4: Viewing the results of a search

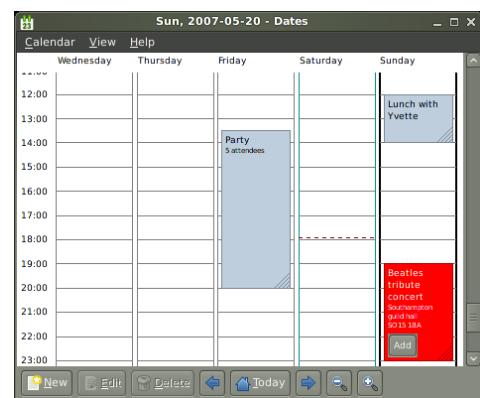


Figure 5: Recommended events embedded in main view

to view these events would be to integrate into an application's 'Agenda' view. Many calendar applications have a special view that shows a summary of the day's events. This is more common on PDA applications than desktop applications, but the concept exists in both. This view usually takes the form of a vertical, ordered list of the day's events. A second column could be added to this view to show recommended events, with shortcut buttons to quickly add events to the user's local calendar. Such an interface is illustrated in figure 6.

In the case that a native client cannot be modified, the external application can be used. This application would take a form similar to the agenda view in figure 6, with integration of search features and a facility to share and tag local events. Sharing an event would be a similar task to the searching of public events, as detailed above, except the final steps would differ. The initial search dialogue would be somewhat similar, with more advanced search options, given the store is local. The display of resulting events would be the same, with the 'Add' button replaced with a 'Share' button. When share is selected, a final, new dialogue would be displayed, with an interface to allow basic editing and tagging of that event before sharing. This interface could be similar

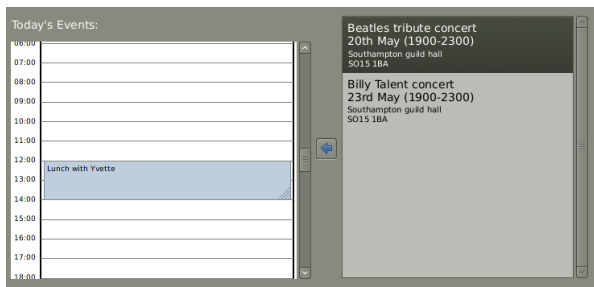


Figure 6: Recommended results, separated from an agenda view

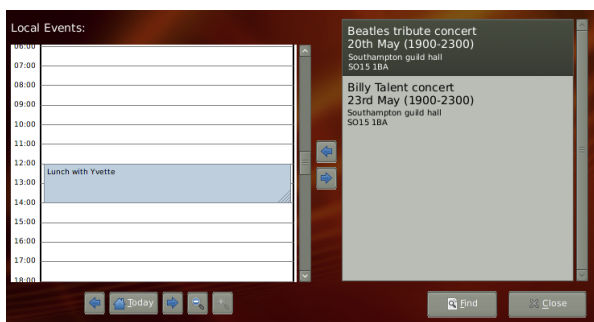


Figure 7: Prototype for external interface

to that shown in figure 1. To avoid making the experience too disjoint, it may be sensible to present this process in a 'Wizard' style, with 'Back' and 'Forward' buttons. A prototype for the initial interface can be seen in figure 7. Note its similarity to the agenda view, with the added features of search and sharing. To create this application, it is assumed that the main calendar application has some developer API for reading and writing events. Although it is possible this is not the case, it is unlikely, as such features are required for synchronisation plug-ins. When it is not available, it is possible that the application could modify the main calendar application's database directly.

Certain devices will not be able to provide a modifiable native client, or an interface to the native calendar application, if there is one. Devices such as phones and public Internet kiosks would fall into this category. For these extreme circumstances, a web interface would be appropriate. Using the previously discussed synchronisation, this web interface could be any one of a number of third-party online calendars. There is also the possibility of synchronising with the device directly, at least in the case of phones. Without a specialised online web interface, however, it is unlikely that a user would be able to access many of the new features of this framework externally to a native client.

6. CONCLUSION

In this paper, a framework to add features that support social navigation and translucency to a standard calendar application has been discussed. Social navigation is supported by the search feature, and translucency is dealt with by the recommendation feature. Both these features allow for a user to make decisions based on the actions of other people,

this being the most important feature of a socially-aware computer system. This framework consisted of some extension fields to the standard iCal component, a peer-to-peer network to communicate events between clients, an agent to make event recommendations and an extended interface to access these new features. Where possible, this framework has been built on top of existing technologies (iCal, Gnutella, SyncML) to increase its feasibility and ease its implementation. It has also been designed so as to augment current calendaring framework, as opposed to displacing it.

As the initial design has been detailed in this paper, all that is left is to create a reference implementation. Such a work would make more sense as part of a network-oriented desktop project, rather than to be handled on its own. Although this paper has just dealt with calendaring, social awareness can be added to many types of common desktop application. Perhaps this could be implemented as part of a wider, PIM-data network. The framework detailed in this paper could be likened to the mesh network that is being developed as part of the One Laptop Per Child⁸ project and such a project would be well-suited as the testing-ground of this framework. This project may also be value to event organisers who wish to advertise their events to potential attendees in an unobtrusive fashion. By sharing their events through this system, and perhaps with a small amount of integration (e.g. being able to directly buy concert tickets from the calendar interface), it is possible that sites that sell event tickets could broaden their customer-base and thus increase sales.

Future work for this design would involve further thought into the pitfalls presented by the peer-to-peer nature of the network. Perhaps this design would be easier to implement as part of a server-client architecture. The main advantage of the network being peer-to-peer are resistance to failure, security and cost. These are countered by the disadvantages of sporadic information availability, efficiency and speed. It could be the case that as hosting becomes cheaper, the extra effort in implementing a robust peer-to-peer architecture is unwarranted. Certainly, providing an interface that is available from any terminal becomes easier when the data is in a central, always-available and easily accessible location. It will only be possible to discern this, however, through further work.

7. REFERENCES

- [1] F. Dawson and D. Stenerson. *Internet Calendaring and Scheduling Core Object Specification*, November 1998.
- [2] P. Dewan, P. Dourish, K. Ehrlich, C. Ellis, S. Greenberg, H. Ishii, C. Johnson, W. E. Mackay, A. Prakash, and M. Roseman. *Computer Supported Co-operative Work*. John Wiley and Sons, 1999.
- [3] A. Dieberger. Supporting social navigation on the world wide web. Technical report, 1997.
- [4] T. Erickson and W. A. Kellogg. Social translucence: An approach to designing systems that support social processes. In *ACM Transactions on Computer-Human Interaction*, pages 59–83. IBM T. J. Watson Research Center, March 2000.

⁸<http://www.laptop.org/>

- [5] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the sixteenth national conference on Artificial Intelligence*, pages 439–446, 1999.
- [6] E. Koblenz. The evolution of the pda: 1975-1995. May 2005.
- [7] P. Maymounkov and D. Mazires. Kademia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems: First International Workshop*, pages 53–65. Springer Berlin / Heidelberg, 2002.
- [8] E. Miller. An introduction to the resource description framework. *D-Lib Magazine*, May 1998.
- [9] L. M. Ni and P. Zheng. *Smart Phone and Next Generation Mobile Computing*. Morgan Kaufmann, 2006.
- [10] T. O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. September 2005.
- [11] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *First International Conference on Peer-to-peer Computing*, pages 99–110, 2001.
- [12] Various. Wordperfect. *Wikipedia, the free encyclopedia*, April 2007.
- [13] M. Vlk, M. Krtzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *Proceedings of the 15th international conference on World Wide Web*, pages 585–594, May 2006.