As Computer Users Grow More Savvy: Experiences with a Multimedia Tool

David Bargeron and Jonathan Grudin
Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399 USA
+1 425 703 7526
{davemb, jgrudin}@microsoft.com

ABSTRACT
Software that can be widely used with little or no modification has advantages for producers and consumers. But it is likely to require some adjustment by users. We describe more than three years of studies of a conceptually simple multimedia annotation tool, which we expected to be widely useful without modification. We encountered a surprising range of specific requirements, including context- and content-specific needs that forced us to shift from an application focus to a platform focus. Instead of a single general-purpose tool, we ended up producing a toolkit to support asynchronous group interaction, with which many task-specific applications were built. This has far-reaching implications. For application designers, software developers, and researchers. As computer users become more aware of the versatility and flexibility of software, general purpose shrinkwrap software may fade away.

Keywords
Software platforms, requirements, multimedia, annotation

INTRODUCTION
A single, widely-used application and interface has obvious benefits to its developer, and consumers benefit as well. Amortized design and development costs lead to lower prices. A uniform interface can be more carefully designed, as can training, maintenance, and upgrades. Help guides, books, and other support can more easily be provided in a scalable manner.

But there are limits to how generic an interface can be before it fails to meet context-specific requirements, and the contexts of computer use are expanding. Pull-down menus that worked well for the original 8" Mac will be unacceptable for wall-sized displays. A simple search feature that is useful in a word processor is insufficient for a database application or the web. Generic spreadsheet functionality is insufficient for vertical markets such as insurance and accounting. At some point differences in context force differences in the underlying software despite the added expense of adaptation.

Consider these trends:

1. Software is undertaking to do more for people. The closer it engages with our activities, the more it encounters differences in how we prefer to work.
2. Computer users know that software is malleable. As expectations rise, willingness to adapt behavior to the software may decrease. Usability has slowly emerged as an evaluation factor in the press.
3. Features that support communication and collaboration are increasing. Supporting groups and organizations introduces additional requirements [7].

These trends tilt the balance away from "generic" interfaces toward task-specific support. They encourage platforms on which specialized user interfaces can easily be fashioned. The third trend has been discussed less than the others but provides useful insights.

Groupware development was eagerly embraced in the 1980s and 1990s, but apart from email, few products designed explicitly to support groups were commercial successes. Consider the example of electronic meeting rooms: After decades of research, several products appeared in the early 1990s (e.g., IBM TeamFocus and Ventana GroupSystems). None did well. Several years later Microsoft NetMeeting was released for distributed meetings. It fared somewhat better, as has GMD's BSCW for asynchronous groups. Being free undoubtedly helped. But NetMeeting has been discontinued and BSCW remains in limited use after many years.

Despite the poor record of the 1990's, the flow of products in which software represents and supports groups continues unabated: Groove, eRooms, Sharepoint, scores of workflow

management systems, and so on. But until we understand better why so few products survive, the prognosis for these remains uncertain at best.

A notable exception to this bleak account is Lotus Notes, a commercial group support product that is widely used. A key characteristic distinguishing Notes is that it is an application development environment, not an application. Non-trivial development work is needed for each group using Notes, resulting in an application tailored to the group's particular requirements.

The case described in this paper suggests that in the decade since Notes appeared, the trends listed above have brought us to a point where the same platform-type approach may be needed even for a much more specialized product. We describe three years of experiences with a multimedia annotation tool that was designed to support collaboration around archived video, primarily in educational settings. The tool is easy to understand, and was intuitively appealing to many people, and we receive frequent requests for the software based on published accounts of specific studies. Yet our experience indicates that the software would not be adopted without major modifications before each deployment.

The key message is that very experienced researchers and developers do not realize how strong this effect is. Not only did we fail to see the reluctance of different groups to adopt our initial uniform application, but educational technology researchers at Illinois, Colorado, MIT, and elsewhere who requested the system underestimated these phenomena as well. It is important that they be better understood.

Following a brief discussion of related work in the next section, we describe our initial multimedia system. We then review a series of lab studies and field deployments, focusing on different interfaces that resulted. Finally, we discuss the implications of these results for designers of platforms and applications.

RELATED WORK

This paper describes a series of interfaces to an asynchronous multimedia annotation system. The system shares features with other systems such as Classroom 2000 [1]. Our focus here is only incidentally on system features. More detailed descriptions of issues guiding the functionality of multimedia annotation systems and reviews of related systems can be found in [2], [3], and [8].

This paper leads into a discussion of application tailoring and customization. In the HCI literature, this discussion focuses on enabling "end-users" to tailor their environments [4] [5] [6] [10], although studies show they rarely bother [e.g., 9]. In contrast, we do *not* propose highly customizable interfaces aimed at end users. Rather, we propose that designers shift their thinking toward more generic platforms on which domain experts (not necessarily software developers) can fashion task-specific interfaces.

Extensible and tailorable systems have constituted a major topic in the software engineering literature [11]. Although the goals of extensibility in software engineering differ from our goals in designing a multimedia annotation system, much can be learned from software engineering in the area of designing generally useful platforms.

MULTIMEDIA ANNOTATION SYSTEM

In this section we examine our initial system design goals, and we describe the architecture and original user interface features of the system. As preface, we present a scenario to illustrate the use of our multimedia annotation system as we originally envisaged it.

Scenario

A student logs in to watch a class lecture in the evening from her home computer. Through her web browser she receives the audio and video of the lecturer, the associated slides that flip in synchrony with the video, and notes associated with the slides. In addition to typical VCR-like navigation features for the lecture video, there is a table of contents of slide titles, and with a click she can "seek" or jump the presentation to the appropriate slide and audio-video point.

The student also sees questions and comments entered by classmates who watched the lecture before her, as well as responses from other students, teaching assistants, and the lecturer. These questions are linked to the lecture content. As she watches a lecture, questions asked during that portion of the lecture are automatically highlighted or "tracked." The content of a question appears in a preview window; if one piques her interest she can jump the presentation to it. As she is watching, she sees a question that nobody has answered. She types a response, which is automatically registered with the system and displayed with the question. The person who posed the question is notified of the reply by email.

Later, the student has a question. She selects the "ask question" button, then types a subject header and her question. Afraid that the question may sound uninformed, she makes it anonymous. In addition, she enters the email address of a friend, who may be able to answer it before the TA gets to it. When she saves the question, it is added to a pre-existing shared "discussion" collection, "tied" to the current point in the lecture, and automatically emailed to the TA alias and to her friend.

A TA browsing through his email sees the question arrive and opens the message. The email includes the text of the question along with a URL pointer to the point in the lecture where the question was asked. It also contains enough meta information for a reply to be added to the annotation database, making it visible to students who later watch the lecture.

The student can similarly record personal notes and participate in small-group discussions, also linked to the lecture. These are added into different collections, with permissions set appropriately.

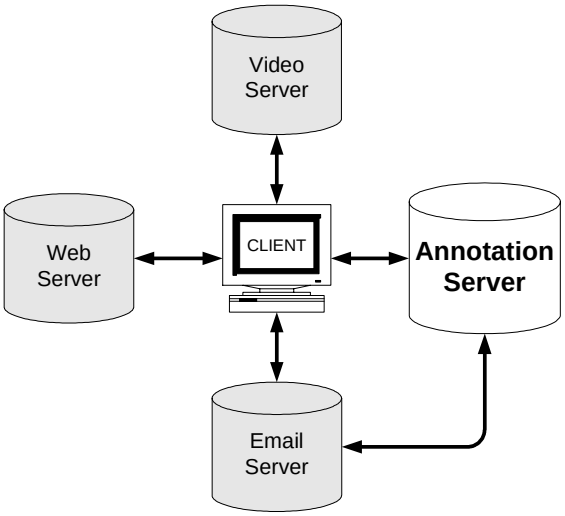


Figure 1. The Annotation Server fits into a standard multimedia network architecture.

Initial System Design Goals

This scenario helps illustrate some of our initial design goals. In particular, we wanted:

- A general-purpose user interface to support the kind of activity presented in the scenario above, across a wide variety of web pages containing embedded video (college course and corporate training web pages, news websites like CNN, and software usability study pages, to name a few).
- Fine-grained organization and access control structures to support structured sharing among groups of users. We wanted to be able to collect annotations into sets and control who could add annotations to the set and who could see annotations in the set. With this simple mechanism, we could create a shared discussion set for a college class, a personal notebook set for each user, a table of contents set for each video file, and so on.
- Close integration with email so that annotations could be sent out via email and replies could be cast as annotations by the annotation server. Email is widely used and well suited for asynchronous collaboration, and with close integration a single conversation can span both mediums.
- Anchoring and display of annotations "in-context" of multimedia content just like notes in the margin of a book, so that we could tie annotations to particular points or ranges along a media timeline.
- Annotations stored external to the annotated content (e.g., the audio-video file) in a separate store. This is critical as it allows third parties to add annotations without having write access to the content. For example, students should not be able to modify an original lecture.

At the outset, we believed a single, well-designed user interface could meet these goals and support a wide variety of collaboration scenarios. Thus, we did not initially plan any interface specialization capability. We used lab studies to identify problems and possibilities and iterate on this design. But each field study described below identified new design requirements specific to the group using it, and without which the acceptability of the system was in doubt. This called into question how useful a single "generic" interface could be, and we ultimately moved to a platform approach similar to Lotus Notes. This has significant implications for its eventual deployment.

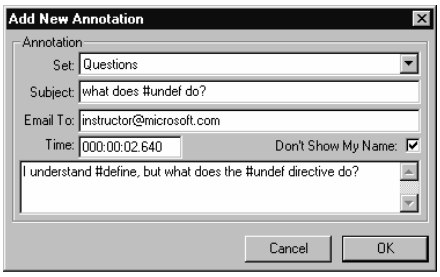
Original System and User Interface

We built the original system in 1998 to support annotation of multimedia content that appears anywhere on the web. When a user accesses a web page containing video, the browser contacts the web server to get the HTML page and the video server to get the video content. Annotations associated with the video on the web page can be retrieved by the client from the annotation server.

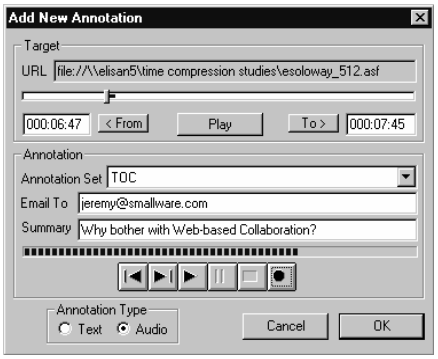
Figure 1 shows the interaction of these networked components. The annotation server communicates with the annotation client via HTTP. Annotations are keyed on the URL of the media with which they are associated. The annotation server communicates with email servers via SMTP, and can send and receive annotations in email.

Figure 2 shows the UI of our initial system. The user views a web page with embedded video in a standard web browser. There may be other synchronized content that accompanies the video, such as slides, but our initial system ignored all such content. Annotations made previously on the embedded video appear in a separate window (to the right of the video in Figure 2). Indented annotations are replies. An arrow indicates the annotation linked to the spot closest to the current position of the video, and the highlighted annotation has been selected by the user. The preview window shows the text of the selected annotation, and if none is selected it will show the text of the nearest annotation. Various controls appear at the bottom of the browser display and in a menu summoned with a mouse click.

Figure 2. The first interface. An annotation window appears over a browser window in which a video plays.



A: text annotation.



B: voice annotation.

Figure 3: Adding a new annotation.

When replying to an annotation or adding a new one, a viewer has a choice of making a text or voice annotation. Figure 3 shows the respective dialogue boxes.

LABORATORY STUDIES AND GENERAL INTERFACE IMPROVEMENTS

We conducted laboratory studies of the use of the system, detailed in [2]. We examined the use of text and voice annotations, contrasted paper and pencil annotation-taking with the use of the system, analyzed the effects of reading others' annotations on sustaining discussion among a group watching a lecture video asynchronously, and got feedback on many aspects of the interface. These studies led to substantial interface changes as well as some evolution of the features.

As shown in Figure 4, we modified the tool's interface so that it could be embedded in a standard web page and easily configured to fit the "look and feel" of the rest of the page. In the web page displayed in Figure 4, tabs at the bottom of the annotation frame in the lower left allowed the user to select one of three annotation sets: Contents (a list of slide titles), Questions (for viewing prior public annotations), and Notes (for viewing personal notes one has taken). At the top of this frame were buttons for adding to the public discussion or personal notes.

Regarding functions, the lab study revealed an unanticipated lack of interest in voice annotations. We therefore added the ability to configure which annotation media types (text, audio, or web urls) were available to users. Voice annotations were found to be less useful for subsequent viewers since they cannot be previewed as the video rolls. More significantly, though, they cannot be edited and polished the way text can.

People chose to pause the video when adding text annotations, so we made it possible to configure the annotation client to pause the video automatically when an annotation is being added. Curiously, pausing a video to make notes on it more than doubled study participants' viewing time, however all participants reported preferring it to taking notes while the video was playing.

In general, the design of the annotation software evolved to accommodate more flexible construction of task-specific interfaces. More details of the system, lab study, and the field study that follows, can be found in [3].

Following this iterative design process we had a robust prototype and considered deployment sites. We settled on two domains: The education context covered in the scenario we presented earlier; and support for analysis and dissemination of results from software usability studies, which are routinely videotaped.

FIELD STUDY IN 'C' LANGUAGE COURSES

To conduct this study, we observed and videotaped a C programming course taught by our internal education group and attended by employees. We then used the digitized video and slides to conduct two on-demand versions of the course. Students signed up for the courses in the usual way, aware that these offerings would involve an experimental system. They met face to face at the beginning and end of the course and used our multimedia annotation interface to view the lectures and interact in the interim.

Early in our observations we realized that programming language classes make particularly heavy use of online demos that are not picked up adequately by a single video camera focused on the instructor. This motivated a system modification: demos were captured after lectures were taped, and links to the demos were added as annotations that would execute appropriately as the lecture video was viewed. This modification can be seen in Figure 5.

Results

Students in the two on-demand series of classes were generally very positive, citing the convenience of watching on-demand. Instructors had fewer time demands, however they missed direct contact with students. Class interaction was at a level close to that of the live class.

Student and instructor comments pointed out some general and specific aspects of using the system for the C programming class. At the general level, the students benefited from clarification questions asked in the live class, which they saw online. A number noted that they asked fewer questions than they might have because their questions were already answered, either on the video or with the system. Good questions and the replies could of course be left in place for subsequent classes.

One student complained about a detail in the interface, saying "I have questions about C, I don't want to 'Discuss' C." Our choice of the term "Discussion" for the public

Figure 4. Browser-based interface following lab studies.

annotation set might be appropriate for seminar-style classes, but was apparently not for this one. We subsequently changed this (Figure 5).

More seriously, the flexibility of asynchronous viewing led some students to procrastinate, to others' detriment as well as their own, since last-minute viewing is not conducive to creating and sharing comments with the class (e.g. via annotations on the lecture videos). This led us to extend the system to include features that are modeled on approaches to discourage procrastination in some live classes: group exercises and quizzes.

Follow-up laboratory studies

We used our system's built-in annotation set mechanism to create annotation sets for small-group projects. Annotations in a small-group set were shared by the few students assigned to work together if they could not meet face to face. The group's product was reported using the class-wide annotation set. In this case, 'Questions' was no longer an appropriate label for the shared class-wide annotation set, and 'Discussion' was restored.

The system and interface was then used in a laboratory study to gauge the effectiveness of the group project approach [8]. It was found to be successful, and the study also generated a strong demand for a feature that was not included: The ability to easily copy or link an annotation from one set to another.

We also extended the interface to include quizzes linked to the video via annotations. Questions appeared at designated points in the video and were potentially useful for self-assessment, grading, or monitoring progress. After responding students could be given a link to the appropriate spot in the lecture to review the question topic. We conducted a laboratory study to explore student reactions and such issues as whether people prefer a question to stop the video or to scroll by in the preview window, and found that preferences vary. In conclusion, the field study led to the discovery of a range of interface issues and the identification of additional task-specific features: means for incorporating demos for certain kinds of classes, interface terminology dependencies for different classes, support for group projects and assessment tools of different kinds. Class content, instructor style, and student style created different user interface demands.

MULTIMEDIA ANNOTATION USE BY USABILITY ENGINEERS

We explored the use of the system with usability engineers (UEs) who supported several product groups. After taping participants in studies, UEs typically review and take notes on the videos, laboriously identify and excerpt segments illustrating key points, and disseminate observations in meetings (where they show the video highlights) and documents (where they do not). We expected that the annotation system would allow them to annotate digitized videos as they review them, providing others with links to relevant portions. Viewers could choose to view material before or after a chosen highlight if need be (which they could not do when the highlights were excerpted).

We quickly discovered that this activity represents a conceptual shift from lecture viewing. The shift could be described as going from a timeline-centric point of view to an annotation-centric perspective. The assumption with a lecture is that viewers generally watch from beginning to end, though they can use annotations to jump from point to point. Everything is organized around the video timeline.

Usability engineers, once they have reviewed the tapes and annotated segments of interest, need to collect the segments for presentation together. For example, a usability engineer may annotate three different regions (in different video files) showing examples of users misunderstanding the same menu label. They then need to play the video segments to which their annotations correspond one after another. Thus, rather than watching a single video and its associated annotations, they need to watch a set of annotations and the video segments they annotate.

This led to the development of the "playlist" feature. A playlist is a sequence of video annotations, possibly from different target videos, which can be played sequentially: when one segment ends, the next will be played.

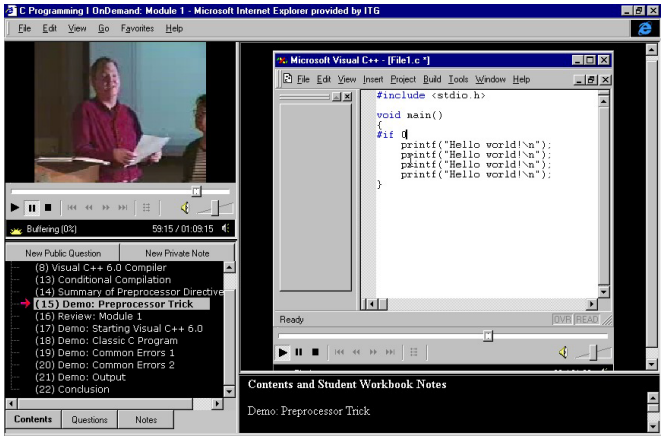


Figure 5: C class interface: "Questions" replace "Discussion," among other changes.

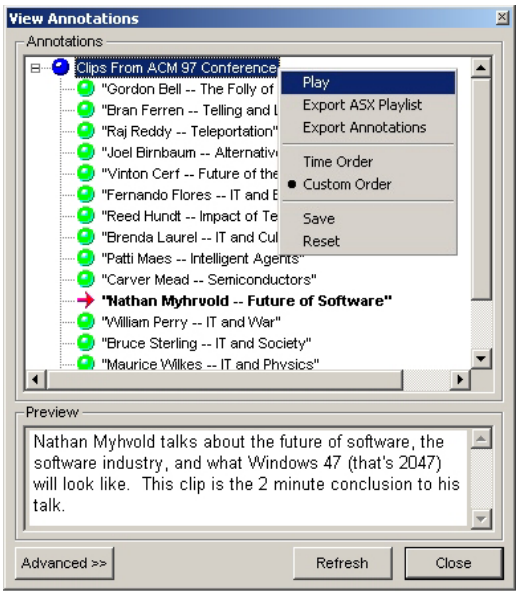


Figure 6. Playlists: A major conceptual change.

Our first interface handled playlists in a straightforward manner: right-clicking brings up a menu item that provides access to playlists (Figure 6).

This interface quickly proved mismatched to the UE's task, however. Results are communicated in face-to-face review meetings, where this playlist feature would be a fine supplement to a slide presentation. However, comments from team members are collected verbally in such meetings, and the key feature of supporting asynchronous discussion is not useful. UEs circulate their findings via email also, and we thought that this is where our annotation system could be of use.

We found, however, that UEs were not willing to adapt to use the annotation system -- or felt their teams were not willing -- even when we managed the process of digitizing the videos. They wanted to have the multimedia annotation functionality embedded in the documents or slide presentations that they sent around in email.

This led to the development of a prototype interface that did just that. Figure 7 is an example of a video annotation interface embedded directly in a Word document. The example shown is not from a usability report, but it shows a new arrangement of features including no slide window and a larger preview window.

SHAKESPEARE COURSE USE AND INTERFACE

It was clear to us at this point in our research that, contrary to our initial conception, the classroom and usability requirements for multimedia annotation differ quite substantially. We focused on the classroom environment for the next experiment. Peter Donaldson, an MIT professor of Drama, was interested in using our annotation system for a class in which students compare filmed performances of Shakespeare's plays.

It became clear that to be acceptable in this class, the interface required additional modification. Figure 8 on the next page shows several major feature changes. To compare performances or aspects of performances, a second video window is added. Buttons beneath the video windows provide much finer-grained control of playback than previous interfaces, such as single-stepping forward or backward by frame.

One might consider this to be "gold-plating' the interface, but without these and other features, the system would not have been accepted. It might have been accepted ten years ago, but due to the software's flexibility and users' savvy in this case, there was significant demand to create a specialized interface.

The system was used in class projects during the fall semester, 2000. "In a couple of cases [the annotation system] got students in touch with the films in ways that don't happen with conventional essays," Donaldson reported [personal communication, Winter 2001].

Although this outcome is exciting, the experience has also proved to be a source of concern. It required considerable effort to develop the interface for the film class, and that interface is not likely to be useful for other classes. It may be useful for other *film* classes, but even that is not a foregone conclusion, since instructors' teaching styles and class format differ significantly.

PLATFORM REQUIREMENTS

Our initial hope, that a single multimedia annotation interface would sufficiently support asynchronous group collaboration and would find broad applicability, has so far not been borne out. Wherever we have looked, we have found new and often orthogonal, perhaps incompatible, interface requirements.

Are we involved in a process for defining a range of features that can eventually be brought into a single package that many users can adapt to their purposes? Or should we focus on building a platform or toolkit that others can use to design specific interfaces that will vary considerably based on domain and approach?

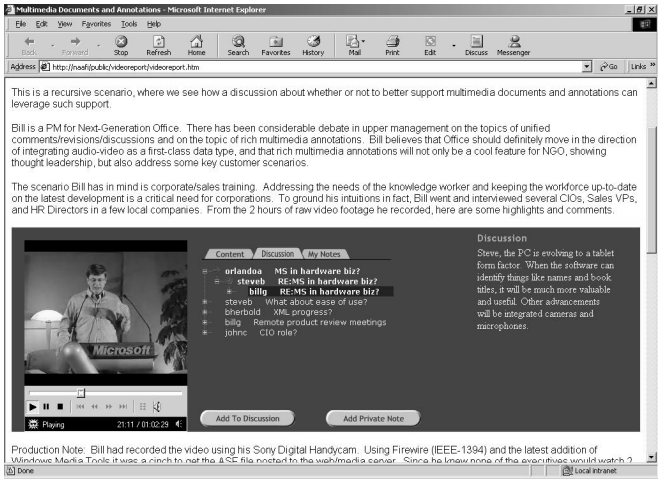


Figure 7. Annotation system embedded in a document.

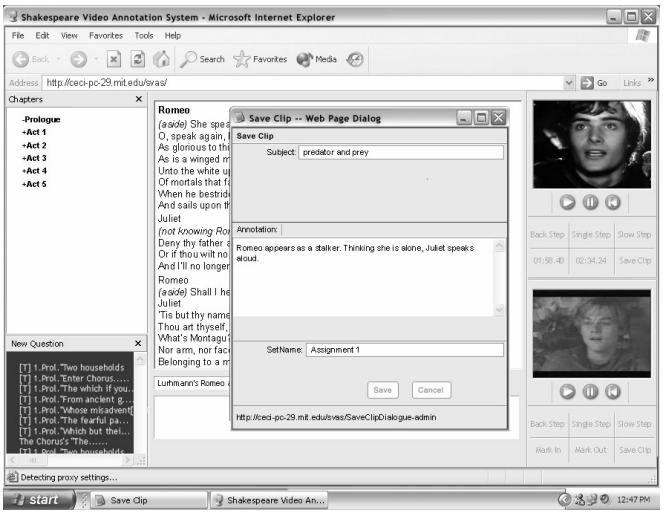


Figure 8. The Shakespeare interface: Understanding drama by comparing performances.

All indications point to the latter. There are clearly commonalities shared among the different task domains we have studied, however there are enough differences among them to require distinct interface features for each context. A general-purpose multimedia annotation interface that incorporates all features could be too complex to be useful in any context. The most prudent and fruitful direction to head in has been toward a generic annotation platform, on top of which task-specific user interfaces can easily be fashioned with a modicum of programming skill.

There are general lessons to be learned from our experience. First, we distilled several requirements for a generic multimedia annotation platform for support of a wide variety of asynchronous collaboration scenarios:

- *Thorough support for common activities*. The most common annotation functions -- such as creating, saving, retrieving, and deleting annotations -- should be the easiest to incorporate into an interface.
- *Extensibility and customizability* at both the interface and platform levels. For instance, designers should be able to extend an annotation's schema to accommodate task-specific features like voting, logging the number of times the annotation has been read, assigning an annotation "type" ("comment," "question," etc), or controlling annotation status ("open issue," "resolved," etc).
- *Storage flexibility*. Designers should be able to store annotations in a variety of configurations. For personal annotations, it may be important to store annotations in the video or audio file itself for portability purposes. For shared annotations on read-only media, annotations may be stored in a separate database. Storing annotations in one facility should not preclude transferring them to another.
- *Universal annotation support*. Ultimately, a general-purpose annotation platform should support annotating any media type with any other media type. Many of the problems encountered with annotations on video and audio apply to annotations on text, images, and complex composite presentations.
- *Interoperability* among task-specific interfaces. Annotations made in one interface based on the platform should be transferable to another interface based on the platform with minimal effort.

We are currently developing a more powerful annotation platform with a more flexible interface toolkit. More generally, these studies and field deployments have implications for designers of other systems, for third parties who will tailor such platforms for specific domains, and for the users of resulting applications.

GENERAL DISCUSSION

We set out to explore a multimedia system, drawing on field studies to enrich our understanding of what such systems should support and laboratory studies to refine the interaction design, and on interface features of what we anticipated would be a widely-used product. We were led to a different goal, providing a toolkit or application development environment.

Should we have anticipated this? If so, there is little point to this paper. But there is evidence that the world is changing. A single product and interface in an area this focused is still very intuitive, and until recently users might have been willing to adopt it and adjust to it 'as is'. Yet the users and groups we encountered were not.

Shifting from a product to platform focus has extensive implications for deployment, so if what we encountered reflects a general trend, it needs to be considered carefully across the software industry.

The challenges we encountered were of course not anticipated by our team, despite considerable experience designing multimedia systems. Nor were the challenges anticipated by our numerous partners in these experiments. Quite the contrary, they were enthusiastic about the system based on initial viewings. Finally, when we published specific studies, we received numerous requests for the software 'as is' by a broad range of people, including computer scientists focused on educational technology at several major universities. The fact that it might not be accepted by users was voiced by no one.

The process of analyzing general platform requirements from task-specific contexts and then driving them into a platform has been a crucial step in our work, because it makes quickly developing new task-specific interfaces easier. As users become more savvy and demand even more customization, providing platforms that can be easily specialized will become easier than providing one-off task-specialized interfaces.

Software specialization is not new. Novel interfaces have long been built to applications such as Microsoft Word or

Excel to support vertical markets. But generic off-the-shelf Word and Excel were useful applications to many people 'out of the box.' Customized versions came later. It is not clear that a generic multimedia annotation system can be designed that will be widely appreciated 'as is.' A toolkit approach that supports a range of interfaces customized to different content and styles looks more promising here. Because the trends of more knowledgeable users desiring more fine-grained support that includes communication and collaboration support are general, many other applications are likely to show the same pattern.

This has implications for developers and vendors, and for customers and users. The former must focus on developing platforms or toolkits, and must consider who will use these to develop specific interfaces for each customer or group of users. The dream of 'end-user programming' or extensive personal customization is not safe to rely upon.

Consider the educational environment that we targeted. We could not construct interfaces for many courses ourselves. At the same time, course instructors could not be expected to recognize in advance what features and interfaces would work for their classes. Third parties are needed, whether consultants, vendors specializing in particular kinds of classes, or academic IT support staff.

CONCLUSION

Software users are becoming more sophisticated and demand support at a more detailed level. Their willingness to adapt to an all-purpose interface appears to be diminished, with significant implications for vendors.

We have described experiences with multimedia annotation systems designed to support classes and other groups. Despite favorable responses to our specialized interfaces, we came to appreciate the elusiveness of a single interface that would be widely useful. The technology is promising, but the path to realizing its uses is likely to be through development of a platform on which third parties can build. This did not seem appropriate for what initially appeared to be a simple application.

To support specialized activities carried out by increasingly demanding customers, potential partners in development must be considered from the outset. Turn-key systems or shrinkwrap software are rapidly losing viability. Designers of systems should take this into consideration and plan field studies as well as laboratory studies to identify the range of interfaces and components that will be needed, and partners who will work with specific individuals and groups.

We hope that this paper will save other researchers and developers who are working on problems of comparable complexity the years we spent reaching this view of the changes in our field.

ACKNOWLEDGMENTS

We thank Anoop Gupta, Scott LeeTiernan, Francis Li, Elizabeth Sanocki, Peter Donaldson, Belinda Yung, Marshall McClintock, Randy Hinrichs, David Aster, and others for their contributions.

REFERENCES

1. Abowd, G., Atkeson, C.G., Feinstein, A., Hmelo, C., Kooper, R., Long, S., Sawhney, N., and Tani, M. Teaching and Learning as Multimedia Authoring: The Classroom 2000 Project, Proceedings of Multimedia '96 (Boston, MA, Nov 1996), ACM Press, 187-198.
2. Bargeron, D., Gupta, A., Grudin, J. & Sanocki, E., 1999. Annotations for streaming video on the Web: system design and usage studies. *Proc. WWW8,* 61-75.
3. Bargeron, D., Gupta, A., Grudin, J., Sanocki, E. & Li, F., 2001. Asynchronous collaboration around multimedia and its application to on-demand training. *Proc. HICSS-34*, CD-ROM, 10 pages.
4. Dourish, P., 1995. Developing a reflective model of collaborative systems. *ACM Transactions on Computer-Human Interaction, 2,* 1, 40-63.
5. Eisenberg, M. & Fischer, G., 1994. Programmable design environments: Integrating end-user programming with domain-oriented assistance. *Proc. CHI'94,* 431-437.
6. Fischer, G. & Girgensohn, A., 1990. End-user modifiability in design environments. *Proc. CHI'90,* 183-191.
7. Grudin, J., 1994. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM, 37,* 1, 92-105.
8. LeeTiernan, S. and Grudin, J., 2001. Fostering engagement in asynchronous learning through collaborative multimedia annotation. *Proc. INTERACT 2001, 472-479.*
9. Mackay, W., 1990. Users and customizable software: A co-adaptive phenomenon. Ph.D. thesis, Sloan School of Management, MIT.
10. Mørch, A., 1997. Three levels of end-user tailoring: Customization, integration, and extension. In *Computers and Design in Context.* M. Kyng & L. Mathiassen (Eds.), pp. 51-76. MIT Press.
11. Reiss, S.P., 1990. Connecting tools using message passing in the field environment. *IEEE Software,* July, 57-66.
12. Teitelman, W. & Masinter, L., 1981. The Interlisp programming environment. *Computer, 14,* 4, 25-34.