Multimedia Annotation: An Unsuccessful Tool Becomes a Successful Framework

Jonathan GRUDIN and David BARGERON
Microsoft Research,
One Microsoft Way, Redmond, Washington, USA

Abstract. We designed and deployed a conceptually simple multi-media annotation tool called the Microsoft Research Annotation System. We expected it to be widely useful in education settings with little or no modification. In a series of deployments over three years we encountered a surprising range of specific interface requirements. We ultimately shifted from an application focus to a platform focus, from a single general-purpose tool to a toolkit to support asynchronous group interaction. Software that can be widely used with little or no modification has advantages for producers and consumers, but as computer users become more aware of the flexibility of software, general purpose shrinkwrap software may fade away. We discuss implications for designers, developers, and users. In addition, our experiences publishing our results pointed to a tendency to favor initial descriptions of success and reject deeper analysis of failure, a practice that is likely to inhibit progress in systems research.

Introduction

Our initial project goal was to design, develop, and deploy a single, conceptually simple multimedia annotation tool that could be used without modification by a range of users. The users we wished to support included college students and professors, students and instructors in corporate education classes, and usability engineers. Each trial of the system by our target user groups was encouraging—most people reporting liking the system, we uncovered problems that could be fixed and features that could be added, and we published papers that generated interest. Nevertheless, after three years of studies, we concluded that one generic tool was not acceptable. Each use led to new task-specific requirements. Users had high expectations; they knew software is flexible and expected it to be tailored to their context. The application was abandoned; in terms of our original goals, the project failed. Fortunately, we were in a position to apply the lessons in developing a multimedia framework and toolkit, a project that has succeeded and might some day be applied to support asynchronous learning.

This paper is intended to contribute at three levels of generality. It reviews the studies of our system built to support asynchronous distance learning, describing in particular the different interfaces that we developed as a result of our experiences. Our experience with methods and interfaces could help people engaged in similar or related efforts. More generally, we surmised that the unexpected challenges we encountered, which led us to change our research focus are due to changes in the nature of system users. Application builders may in general find it more useful to focus on toolkits than generic applications. Finally, our experiences publishing our results over the years revealed a systematic bias in the reporting of systems research. Short-term, optimistic studies are favored. Long-term reports that include discussion of failures as well as successes are discouraged. The result is that a field does not learn from experience. The same mistakes will be repeated.

1. Initial System Design

1.1 Related Work

Our system, the Microsoft Research Annotation System (MRAS), shares features with other systems such as Classroom 2000 [1]. Our focus here is primarily on the application interface rather than system features. More detailed descriptions of the issues that guide the functionality of multimedia annotation systems, and reviews of related systems, can be found in [2-5]. Our experiences raised issues discussed in work on application tailoring and customization. In the human computer interaction literature, this has focused on enabling "end-users" to tailor their environments [6-9], although studies show they rarely bother [10]. In contrast to these research efforts, we do not go so far as to propose highly customizable interfaces aimed at end users. Rather, we suggest that designers shift their thinking toward more generic platforms on which domain experts (possibly including software developers) can fashion task-specific interfaces.

Extensible and tailorable systems have constituted a major topic in the software engineering literature [11-12]. 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.

1.2 Initial System Design

One designer and two developers were involved through the duration of the project. Two contract developers and two student interns participated for periods ranging from 3 to 12 months, and some scripting was done by one more developer at one study site (MIT). Over the years of the project, "users" participated in several ways, as informants helping us to identify requirements and as actual users of the system. As described in more detail below, these included volunteers in experimental studies, students and lecturers in university and corporate training classes, and usability engineers..

The project began in the spring of 1998. The series of design and deployment efforts discussed below continued through 2001. Experimental use of the system in classrooms has continued.

We begin by examining our initial system design goals, then describe the architecture and original user interface features of the MRAS system. As preface, we present a scenario to illustrate the use of our multimedia annotation system as we originally envisaged it.

1.3 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 point in the audio-video stream.

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.

1.4 Initial System Design Goals

This scenario illustrates some of our initial design goals for MRAS. 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.

We felt that a single, well-designed user interface could meet these goals and support a variety of collaboration scenarios. Thus, we did not plan any interface specialization capability. We used lab studies to identify problems and possibilities and iterate on this design.

1.5 Original System and User Interface

We built the original MRAS 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.

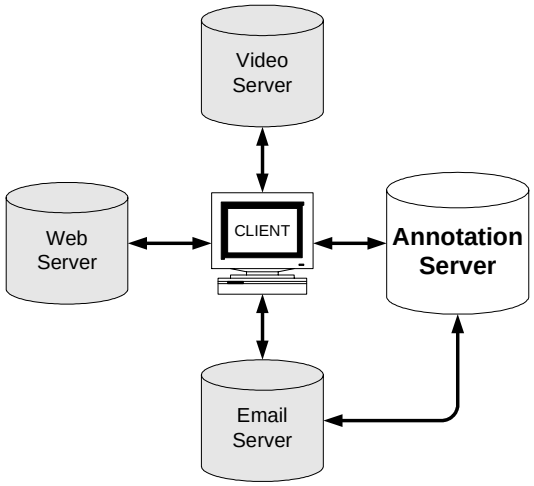


Figure 1. The MRAS Annotation Server fits into a standard multimedia network architecture.

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.

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.

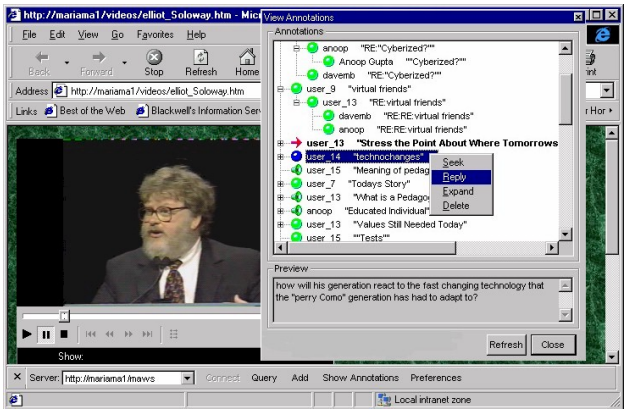
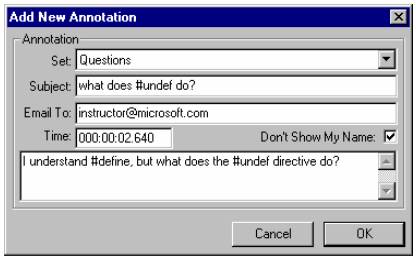
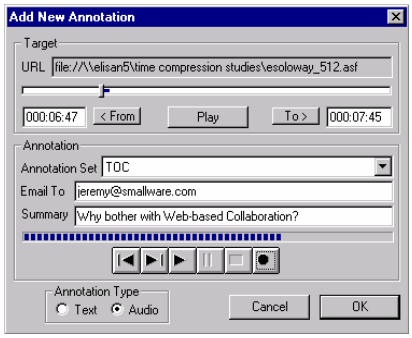


Figure 2. The first interface. An annotation window appears over a browser window in which a video plays.



A: text annotation.



B: voice annotation.

Figure 3. Adding a new annotation.

2. Laboratory Studies and Initial Interface Improvements

We conducted laboratory studies of MRAS use, 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 MRAS 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. For more details of the system, lab study, and the field study that follows, see [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.

Figure 4. Resulting browser-based interface.

3. Field Studies 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 MRAS 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.

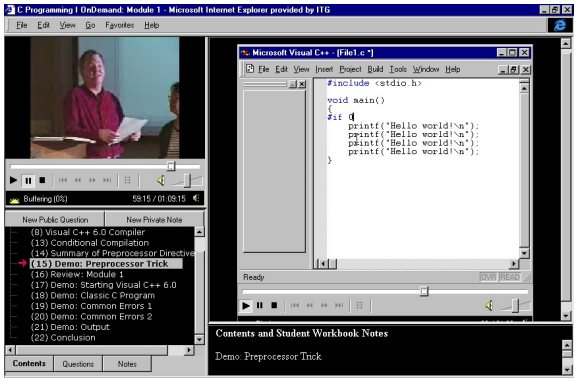


Figure 5. Final interface for 'C' class.

3.1 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 MRAS 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 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 comments or sharing comments with the class (e.g., via annotations on the lecture videos). This observation 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.

3.2 Follow-up Laboratory Studies

We used MRAS'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 [4]. 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 [5]. Questions appearing at designated points in the video are potentially useful for self-assessment, grading, or monitoring progress. After responding, students could be given a link to the appropriate place in the lecture to review the material.

We conducted a laboratory study to explore student reactions and to determine whether people prefer a question to stop the video or to scroll by in the preview window. We 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.

4. Multimedia Annotation Use by Usability Engineers

We explored MRAS use with usability engineers (UEs) who prepare presentations including video clips in reporting to 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 MRAS 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 MRAS "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.

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

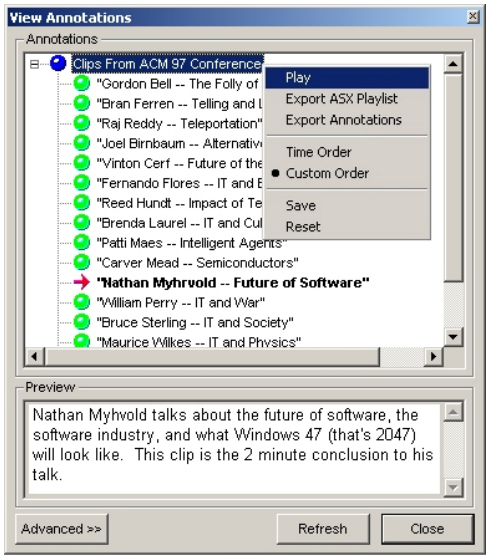


Figure 6. Playlists: A major conceptual change.

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 unwilling to adapt to the annotation system—or felt their teams were unwilling—even when we managed the process of digitizing the videos.

They wanted the multimedia annotation functionality to be embedded in the documents or slides that they distributed via email.

This led to the development of a prototype interface that did just that. Figure 7 is an example of an MRAS 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.

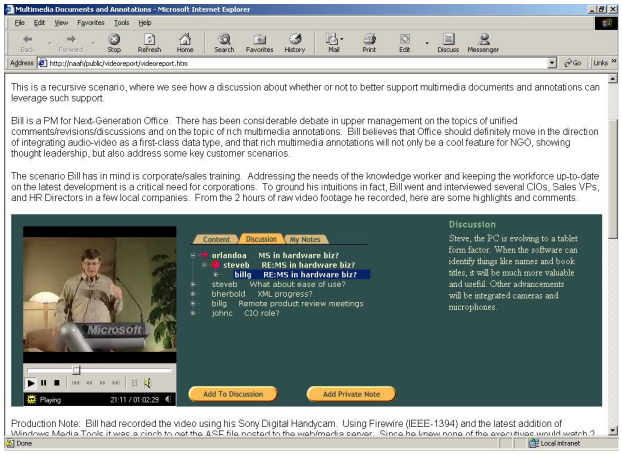


Figure 7. Annotation system embedded in a document.

5. 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 MRAS interface required additional modification. Figure 8 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.

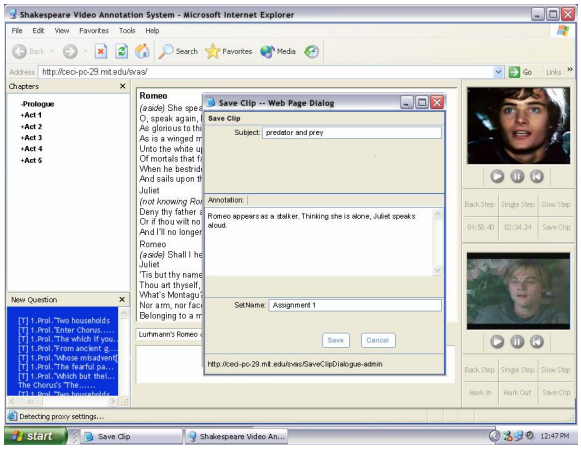


Figure 8. The Shakespeare interface: Understanding drama by comparing performances.

6. Shifting to a Platform

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

Were 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?

All indications pointed to the latter. There were commonalities across the task domains we studied, but there were enough differences 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. A more promising direction was to fashion a framework and generic annotation platform on which task-specific user interfaces could be fashioned with only a modicum of programming skill.

There were general lessons derived from our experiences. In particular, we distilled requirements for a generic multimedia annotation platform to support diverse asynchronous collaboration scenarios:

- Thorough support for common activities. The most common annotation functions—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.

Based on these requirements, we developed a more powerful annotation platform with a more flexible interface toolkit. Due to the accumulation of changes, MRAS was more centered on video and less of a general annotation tool than intended. The same evolution had occurred with the back end—the database and communication protocol had become specialized to support the applications we targeted. We could see how to truly generalize the architecture.

We designed a generic annotation database to store annotation linking information for any kind of annotation and an extensibility architecture that could support annotation content of any media type and annotation anchors on any media type. We kept the underlying RPC over HTTP communication transport (of which MRAS had been an early exemplar). We refined the communication protocol to support a simple API for querying annotations, creating new ones, deleting existing ones, caching annotations "offline" on the client, and synching annotations when the client was back online. And we created a client runtime and object model that served to insulate client applications from all of the details of the system (storage, network transport, etc.). All clients had to do was write to our API to get flexible annotation support.

The resulting annotation framework has made an impact internally and is likely to influence future products. Perhaps distance education applications will be among them.

7. Discussion

7.1 As Users Grow More Savvy: From Applications to Tools

We explored a multimedia system for distance education that we expected to be widely used without significant modification. After years of lab tests and field trials we changed direction, providing a toolkit or application development environment. We were fortunate to be in a company that develops tools as well as applications. Had our business been distance education this would have to be termed a failure.

Should we have anticipated this outcome? We were an experienced team, experienced in designing and assessing multimedia systems and experienced in the domain of education, two of us having been full-time tenured faculty at major universities months before the project began. The challenges were not anticipated by our numerous partners in these experiments. Quite the contrary, they were enthusiastic based on initial encounter with MRAS. Also, our published studies led to numerous requests for the software ‘as is’ from computer scientists focused on educational technology at major universities. No one suggested that it might not be accepted by users.

Software platforms and customization, the fact that one size does not fit all, are not new. But most products are marketed with the expectation that they will be useful to many people with little customization, and some “shrinkwrap” products have done spectacularly well. Products such as Word and Excel can be decked out with domain-specific templates, scripts, and add-ons, but they are also acquired and used with minimal customization by millions of people. A single, widely-used application and interface benefits the 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 are more easily provided.

We were confident that our lecture annotation system would be intuitive and useful enough to be used in myriad lecture courses with instructors and students willing to make minor adjustments to their practices as needed. The people we encountered were not willing to.

Our impression is that the world is changing. As users become more savvy, platforms that can be customized may do better than task-specific interfaces. Finer-grained support that includes general communication and collaboration features may be sought in many applications. Shifting from a product to platform focus also has implications for deployment. If this is a general trend, it deserves careful consideration.

A generic multimedia annotation system for asynchronous distance education may not be feasible, but an application that relies on 'end-user programming' or extensive personal customization is not a good bet either. 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 and teaching styles. Third parties are needed, whether consultants, vendors specializing in particular kinds of classes, or academic IT support staff.

As the contexts of computer use expand, there are limits to how broadly an interface can be deployed before it fails to meet context-specific requirements. Pull-down menus that worked well for the original eight-inch Mac display are 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 expense of adaptation.

Consider these trends:

- 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.
- 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.
- Features that support communication and collaboration are increasing. A need to support groups and organizations introduces additional requirements [13].

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 electronic meeting rooms: After decades of research, several products appeared in the early 1990s, but none fared well. Several years later Microsoft NetMeeting was released for distributed meetings. It fared somewhat better, as did GMD's BSCW for asynchronous groups. Being free undoubtedly helped these, but neither thrived. Nor did myriad workflow management products.

An interesting exception to a bleak account of groupware in the 1990s is Lotus Notes. Notes was an application development environment, not an application. Development work is needed for each use, resulting in an application tailored to a group's particular requirements. Doing this is not trivial, but Notes prospered in the pre-Web era.

7.2 Reporting Research Results

As we conducted the studies described here, we wrote them up and published them in four conferences [2-5]. Each of these papers reflected the optimism that we felt. We had

experimented with the system and discovered ways to improve it, which we were implementing. Participants in the studies were on the whole very positive about the system. Most of them had volunteered to use it. Many expressed eagerness to use it again. These success stories, reported soon after the data had been collected, were relatively easy to publish.

Over time, though, we noticed that much of our optimism was unfounded. With one exception, instructors who used our system, although they had been relatively positive when we asked them for candid assessments, did not ask to use the system again. This same pattern was evident in other prototype systems we were working on during those same years.

We undertook to write this more nuanced account of our experiences, emphasizing that it had not turned out as anticipated, that we failed to reach our initial goal. Someone reading our earlier papers could be misled by the optimism and apparent success. We felt that this could be very useful. But an earlier version of this chapter was rejected by three conferences before finally being published at HICSS-37 [14]. Reviewers typically argued that we should have anticipated that our original effort would fail. But the eager requests for our system from educators at major universities show that it was not obvious.

The scientific ideal is that if studies are well-designed, we learn from any outcome. It is well known, however, that it is difficult to publish "negative results." Yet if we do not, other researchers will follow down the same path. This is true of systems research. If only one in five efforts succeeds and we only report the success, others will probably repeat the futile four. Even worse, if we report apparent successes immediately, before time allows a more sober assessment, we create overly rosy views that will lure future researchers into dead ends.

At a recent high-level series of presentations, one of us congratulated a speaker on the successful project he had described. He replied that he actually considered it something of a failure, since they had tried about ten things and only one worked. He could not mention the others publicly, he said. It would not look good if he told funding agencies that most of what he had done did not work. Of course, this meant others were very likely to repeat his mistakes.

Multimedia support for collaboration is one of many examples of this phenomenon. The literature includes dozens of apparently successful and optimistic studies of prototype systems conducted in laboratories that long ago abandoned their efforts. Few reflective post-mortem accounts were written. So the reports glisten seductively like the water on the surface of a prehistoric tar pit, concealing signs of danger and evidence of the violent thrashing about not long before.

8. Concluding Remarks

We describe three years of experiences with the Microsoft Research Annotation System (MRAS), 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. It seemed well within the range of software that is designed and marketed for broad use with limited customization or some progressive disclosure of features. Yet our experience was that the software would not be adopted without major modifications for each deployment.

A decade ago, our tool may have worked as intended: Users might have adjusted as necessary. Software users are now more sophisticated and demand support at a more detailed level. Their willingness to adapt to all-purpose interfaces is diminished.

To support specialized activities carried out by increasingly demanding customers, turn-key systems or shrinkwrap software is less viable and potential partners in development must be considered from the outset. Designers should also plan field studies to identify the range of interfaces and components that will be needed.

Experienced designers and developers may not realize how strong this effect is. We hope that this chapter will save those working on problems of comparable complexity the years it took us to fully understand this. We also hope that this chapter sets an example by reporting failures as well as successes. We learn from each.

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, *Proc. Multimedia '96,* 187-198, 1996.

[2] Bargeron, D., Gupta, A., Grudin, J., and Sanocki, E. Annotations for streaming video on the Web: System design and usage studies. *Proc. Eighth World Wide Web Conference,* 61-75, 1999.

[3] Bargeron, D., Gupta, A., Grudin, J., Sanocki, E. & Li, F. Asynchronous collaboration around multimedia and its application to on-demand training. *Proc. HICSS-34,* CD-ROM, 10 pages, 2001.

[4] LeeTiernan, S. and Grudin, J. Fostering engagement in asynchronous learning through collaborative multimedia annotation. *Proc. INTERACT 2001,* 472-479, 2001.

[5] LeeTiernan, S. and Grudin, J. Supporting engagement in asynchronous education. *CHI 2003 Extended Abstracts,* 888-889, 2003.

[6] Dourish, P. Developing a reflective model of collaborative systems. *ACM Transactions on Computer-Human Interaction, 2,* 1, 40-63, 1995.

[7] Eisenberg, M. & Fischer, G. Programmable design environments: Integrating end-user programming with domain-oriented assistance. *Proc. CHI'94,* 431-437, 1994.

[8] Fischer, G. & Girgensohn, A. End-user modifiability in design environments. Proc. CHI'90, 183-191, 1990.

[19] Mørch, A. 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, 1997.

[10] Mackay, W. Users and customizable software: A co-adaptive phenomenon. Ph.D. thesis, Sloan School of Management, MIT, 1990.

[11] Reiss, S.P. Connecting tools using message passing in the field environment. *IEEE Software,* 57-66, July 1990.

[12] Teitelman, W. & Masinter, L. The Interlisp programming environment. *Computer, 14,* 4, 25-34, 1981.

[13] Grudin, J. Groupware and social dynamics: Eight challenges for developers. *Comm. ACM, 37,* 1, 92-105, 1994.

[14] Bargerson, D. and Grudin, J. As users grow more savvy: Experiences with a multimedia tool. *Proc. HICSS-37,* CD-ROM, 10 pages.