

Innovation in Extremis: Evolving an Application for The Critical Work of Email and Information Management

Victoria Bellotti[†], Nicolas Ducheneaut^{‡‡}, Mark Howard[†], Christine Neuwirth*, Ian Smith[†]

[†]Xerox Palo Alto Research
Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
+1 650 812 4000
bellotti; duchen; mahoward;
iansmith@parc.xerox.com

^{‡‡}School of Information
Management & Systems
Berkeley 102 South Hall
University of California
Berkeley, CA 94720-4600
+1 510 642 1464
nicolas@sims.berkeley.edu

* Department of English and
Human-Computer
Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
+1 412 268 8702
cmn@cs.cmu.edu

ABSTRACT

We describe our experience of trying to develop a novel application that transforms information management (both coordination-based and personal) from stand-alone resources into resources deeply embedded in email. We explored two models for accomplishing this goal; these were to embed these resources in the email channel and to embed them in the client. Our exploration of the first model was intensive, in-depth and ultimately unsuccessful in large part due to our design process. We adopted Extreme Programming (XP) as a means to explore our second model more efficiently. This paper describes our motivations and experiences while exploring our first model before XP and then the advantages and disadvantages of turning to XP in the exploration of our second model.

Keywords

XP, Extreme Programming, Personal Information Management, email, workflow, ThinkDoc, TaskMaster

INTRODUCTION

In a recent paper (Bellotti & Smith 2000) we reported on an iterative fieldwork-based design effort that led us towards the design of an information management (IM) tool where the IM resources are embedded within an email client. In the two years since then we have been exploring ways to commercialize the idea of embedding IM features inside email. This process has been highly challenging for a number of reasons:

- Email has become an overloaded tool supporting far more than just messaging. It requires sophisticated integration with other resources such as document management and event management [7, 8, 9].

- Email users are heavily invested in their existing tool. They are thus unlikely to adopt an entirely new tool that requires them to move their legacy email archives and learn new software without some top-down organizational imperative (such as organizational edicts or moving to a new company with different supported software).
- Refining prototypes based on realistic evaluation is hard due to the commitment required from users to trust their critical email activities to a non-stable prototype.
- Designers of email clients for commercial markets face the obstacle of increasing market dominance by a few major players against whom competition is difficult.

This paper overviews our efforts to move from a research prototype to a commercial application that respects the notion of IM as an embedded activity, taking place in the email channel.

The structure of the paper is as follows:

First we briefly review the research philosophy behind our design effort, first introduced in [3] and evolving through [7, 8 and 9].

Secondly, we describe two very different models that embody our thinking about how to embed IM in email. These models have been developed with respect to the challenge of developing commercial software in a marketplace already dominated by established players.

Thirdly, we review the difficult process we went through relating extensive fieldwork to intensive design as we explored the first model. Despite a great deal of effort being devoted to both fieldwork and design, we found that we had not been successful at turning fieldwork into a resource that could strongly *motivate* design decisions, *prioritize* the most important features, and be used to *review the results* of engineering on a rapid turnaround basis.

Finally, we describe how we have attempted to bridge these gulfs between fieldwork and design during our exploration of the second model using the Extreme Programming XP approach [2, 10, 11] to improve the design process. We conclude with a discussion of the advantages and

disadvantages of XP as an approach to tie fieldwork and design together.

STUDIES OF INFORMATION MANAGEMENT ACTIVITIES EMBEDDED IN EMAIL

Information Management can mean many things, but what we are interested in here is closer to personal information management. This is the ordering of information through categorization, placement, or embellishment in a manner that makes it easier to retrieve when it is needed [1, 12, 14]. It may also involve a great deal of information related to coordination and collaboration (collaborative information management). It may involve resources such as piles, collections, file hierarchies, notes, to-do lists, calendars, contact managers and so on. We will use the term IM throughout this paper to refer to this personal, collaborative and coordinating information management.

We conducted a series of field studies, which will be referred to throughout this paper, looking at information management IM outside of and in email, as well as other aspects of the use of email. For clarity all the studies listed below, although studies 4-8 took place after the design activities described later got under way:

1. **Preliminary pilot IM** interview study to establish research questions (4 interviews).
2. **In-depth in situ IM** interview study to explore developing ideas (8 interviews).
3. **In-breadth IM** interview study to confirm hypotheses (24 interviews).
4. **Requirements** interview study small office/home office (SOHO) employees to establish email IM system requirements (28 interviews).
5. **Focus group** for feedback about preliminary design ideas and identify further requirements (12 people).
6. **Study of habits, IM and collaboration in email in 3 organizations** (28 interviews).
7. **Rapid field customer requirements assessment** for a possible customer for a leads-tracking application.
8. **System features feedback interviews** for reactions to and suggestions for features we were planning to implement in our technology (10 interviews).
9. **Survey of Microsoft Outlook™** feature adoption and understanding by users (64 respondents).

Information Management Embedded in Email

Our first three studies, previously reported in [3] focused on IM on- and offline and within and independent of email. These established that:

- IM is a ubiquitous activity on- and offline that takes place, using resources that are to hand, and likely to be to hand again, at the time the information is needed.
- Online the most convenient and ubiquitous resource seems to be email, which is on all the time and so is appropriated as a critical IM tool [13, 15, 16]. In particular, it is co-opted by users for many IM functions, such as to-dos (by marking-up or re-sending oneself

messages) and contact management (by sorting by name and filtering).

- Email is overloaded, providing inadequate support for certain tasks it is routinely used to accomplish.

Based on this research, a prototype system, Raton Laveur, was developed, which embodied the philosophy of embedded IM inside an email client [3]. Key innovations included the following ideas:

- Email was organized along with other kinds of items such as scanned documents and notes appearing as first class citizens (rather than simply attachments) in a sorted list viewer.
- Content was primarily organized on a search-based rather than filed basis. Users could create, informal collections where a search hit on one member would retrieve an entire collection, allowing email to be grouped with other documents for which it provided a context.
- Notes could be attached to any item and to-do and done items were highlighted and searchable as a category.

Informal evaluation and use of Raton Laveur led us to believe it held some promising ideas for a commercial product. However, we quickly recognized that, in a highly competitive marketplace, we would need to distinguish ourselves from the dominant players who could simply imitate many of our existing ideas and out compete us. We therefore began looking for a niche that was relatively open for the development of innovations that could be protected as intellectual property. There seemed to be little support in existing solutions for management of tasks within email in a collaborative setting so we began here by looking at existing practices of email users.

Coordination Activities Embedded in Email

A further in situ interview study was conducted across three organizations; our own technology research lab (300 employees; 10 interviews), a large start-up (150 employees; 12 interviews) and a small, 6-person, design consulting firm (6 interviews). This study focused in particular on email as a resource for collaborative IM. The findings [reported in 7, 8, and 9] included the following.

How Much is Email Used?

Our respondents receive about 42 messages a day on average and send about 17, with roughly one in ten containing an attachment. They are heavily invested in email, using it throughout the day, and the longer they have used it, the truer this is.

When they are not reading or sending email, they are often working on objects they exchange through email, referring to the context of the message bodies for information to guide their work on these attachments.

How Does Email Support Collaboration and Coordination?

Email is widely used by the great majority of our informants for organizing and scheduling meetings, and exchanging files [8]. Document exchange is often linked to meetings, and frequently occurs before and after them.

The majority of our respondents document activity via email. It is also used by most to assign responsibilities to others, and to come to decisions with others.

Tracking Complex Work in Email

Email easily supports semi- and asynchronous communication about and coordination of work on objects through references (or attachments) that can be followed up on at the recipient's convenience. Since email often needs to be referred to later, it is usually organized (using folders and less frequently filters) and archived for easier access. Our interviewees have anything from an inbox and an outbox, to more than 400 folders, with a mean of 91 folders and a median of 27 folders.

Email conversations can become extremely complex. At any one time, the average user in our sample was involved in three extended conversations that were 6 or more messages long with 3 or more participants (lasting from a day to several weeks).

Our data, like that of [16], suggest that managers, entrepreneurs and more experienced users receive and send more email than others. Thus, based on this and the rest of our findings, we strongly believe that increasing pressure due to increased responsibilities is likely to be strongly correlated with more complex conversations, more coordination work and increased overloading of email as an IM tool, tracking these multiple threads of activity.

These *email 'multitaskers'* constitute our *prime target-user group* for IM support in their email because they are the ones who are the most overloaded and need the most support for personal and coordination related IM.

Given all of the above, we have come to think of email as more of a *habitat* than an application [7], since it tends to be running continuously on people's machines, is consulted continually through the day (often every time a new message arrives) and is overloaded in terms of the work it is being used to support and manage. This conclusion provided us with the inspiration to look for ways in which existing email tools could be improved and extended to better handle the IM and coordination tasks that they were so obviously being appropriated for.

The rest of the paper documents our exploration of two alternative models for embedding IM in email. One involves embedding resources to support IM *in the channel* and the other involves embedding them *in the client*.

TWO MODELS FOR EMBEDDING IM IN EMAIL

In this section we describe the two models for embedding IM in email that we explored. In both cases the challenge is to find opportunities for applications that go beyond the scope of existing solutions in the email software market. Our approach has not been to design an idealized application that covers the majority of email users' needs (many of which are already being met in the marketplace). Rather we seek a significant (sub-)market whose needs are not being met.

Model 1: Embedding in the Channel

Rationale

When we talked to knowledge workers (mainly in the US and the UK) at the end of 1999, we ascertained that there was a range of email solutions in use. Even though a few large corporations dominated the email marketplace, enough people were using different clients that professionals working outside of large corporations were frequently collaborating with people using a different email and IM infrastructure. We felt that this would be particularly true of the small-office/home-office (SOHO) market and, later, the cross-organizational customer relationship management (CRM) market, where companies distribute products through resellers. We believed that people in these markets might benefit from a "man-in-the-middle" Internet resource for IM and task management across incompatible software infrastructures.

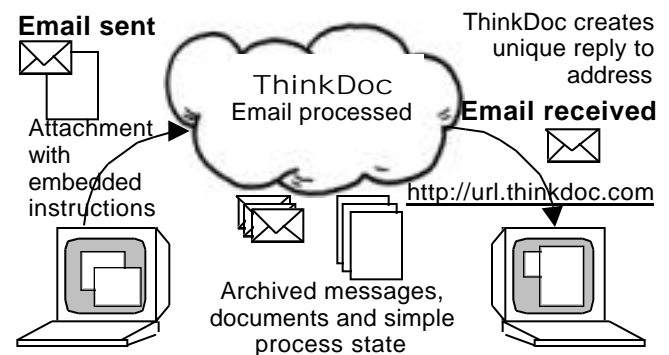


Figure 1. Schematic diagram of our mechanism for embedding IM in the channel. ThinkDoc server on the web intercepts and archives messages, documents and process status accessed via URLs and 'mail to' links.

Realization: Man-in-the-Middle

Our solution, ThinkDoc, is a server-based service interposed between email senders and recipients in the email channel; i.e., the network or Internet. Figure 1 is a schematic depiction of how the ThinkDoc mechanism can work. The basic concept is that ThinkDoc intercepts email from a subscriber (using a variety of possible re-addressing mechanisms) and invokes a service call that is embedded in the message (perhaps in an attachment). Recipients of the email receive a message with a reply-to address contains information to identify the respondent and the original message. Messages and attachments passing through ThinkDoc can be processed in various ways based on information embedded in messages, perhaps stripping out an attachment and replacing it with a URL, as shown in figure 1, or perhaps creating an archived copy of the email, or sending a notification to someone. The following are examples of services that are possible:

- Users can send all their email through ThinkDoc and use a ThinkDoc proxy email address for all contact with other users. In this mode, all their email comes through the service. ThinkDoc keeps archives of their email.

- Important emails and replies can be logged on the server providing a disinterested and trusted record of communications.
- Attached documents can be replaced with a link. Senders might update the document on the ThinkDoc server. Thus recipients of documents can always be sure to access the most recent version of the document.
- Users can forward messages to ThinkDoc to create new contact records for the sender perhaps returning a simple alias for future use.
- Users can forward messages to a ThinkDoc address to create a reminder that sends the message back in a specified amount of time (e.g., in 7 days).

Model 2: Embedding in the Client

Rationale

A later conception of embedding IM in email came about in response to a growing realization that Microsoft Outlook has rapidly begun to dominate the email marketplace. It is bundled with a popular version of Microsoft Office™ and has become the standard in many large corporations.

Outlook already supports many IM features, but a survey conducted at our lab into knowledge of and use of email tool features by 64 respondents suggested that even technologically sophisticated users do not use many of Outlook's features. Indeed many users were still relying on basic features available in much simpler clients. We felt that if certain features were easier to use and better integrated with email, users would be better served. We also felt that it would be a fool's errand to try to build an alternative email client in a market dominated by Outlook, and expect users to give up their investment in their current tool. This is particularly problematic since it is difficult to extract message archives from Outlook so that they can be read by other clients.

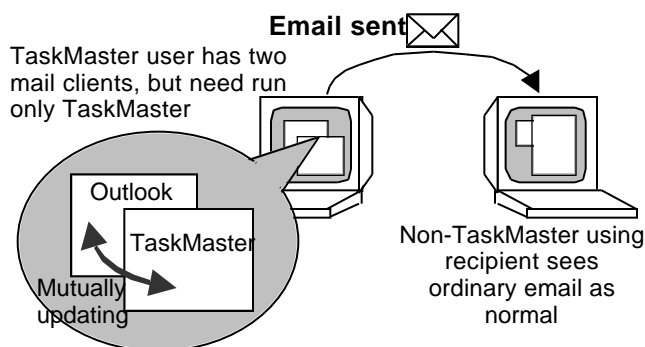


Figure 2. Schematic of the IM embedding in the email client mechanism. TaskMaster mirrors Outlook content adding advanced support for task and information management.

Realization: An Additional Task-Centric Mail Client

The basic solution was to build a client that leverages and works *with* the Outlook client on an individual user's

machine. Outlook remains the mail archive with its folders and other features, however the new client, TaskMaster, supports another kind of view onto email.

The new view provides a number of IM resources that are designed to reflect the findings from our email collaboration and coordination research; that users need these resources to be deeply embedded in the flow of email. Some of these resources are already supplied by Outlook (although they are not necessarily easy to discover), and others are innovations to support IM. Many of these features parallel features in ThinkDoc.

The important issue is that, while the end-user services are similar, the underlying model differs drastically from our previous model, since it relies on code running on a user's machine rather than on a central server. There are other examples of this IM strategy already in existence (e.g. Caelo's Nelson™ and My Reality—a shareware tool) that also run with Outlook.

Figure 2 shows a schematic representation of the architecture for TaskMaster. TaskMaster can import old messages from Outlook, but once installed, it keeps on updating its content from Outlook's incoming messages. However, items that are deleted in one remain in the other, unless they are also deleted there.

THE ROCKY ROAD: EXPLORING THE MODELS

In this section we now describe the process that led us from one model of support for IM in email to the other. In fact, our turning to the second model only came about after some rather depressing discoveries about the first model.

The first model was explored using user-centered methods similar to [4], including fieldwork to uncover target user practices and requirements, scenarios of use and grounded brainstorms to explore design ideas, rough look-and-feel prototypes to walk through usability and informal user evaluations of design ideas. And yet we were blindsided by various critical issues that eventually led us to move to a second model instead.

IM in the Channel

We began looking for a technological niche that might support our business aims by offering some unmet needs in a significant market. Our key advantage with our IM-in-the-channel technology was the ability to support people who do not share an infrastructure with a simple easy-to-adopt solution. This led us to begin by exploring the small-office/home-office or SOHO market and its requirements.

SOHO: Multiple Scenarios for Collaborative Information Management Solutions

A series of informal discussions and 28 formal interviews were conducted with people who work from small or home businesses to gather SOHO requirements for IM in email. These interviews established that coordination issues, especially between organizations, rank high amongst the problems experienced by these workers constantly trying to manage a variety of threads of collaborative activity in their email. Many of these people fit our *email multitasker target-user profile*. We then invited a dozen customer representatives to attend a focus group meeting where we

presented a series of scenarios elaborating our basic idea of IM in the email channel to them. These scenarios addressed both personal and collaborative IM. They included:

- Sending a document for review, tracking the state of responses and sending notifications when updates occur.
- Finding archived copies of disputed communications and binding agreements.
- Creating informal schedule and progress representations on the fly.
- Adding new contacts to a database by forwarding a mail message from a new sender to the server, which automatically mails that sender to fill in details.

We asked for our focus group participants to give us feedback about the basic idea and each of our scenarios and to provide input and suggestions as to what features and solutions most useful to them.

Key findings from this information gathering exercise were:

- There were many intriguing possibilities that appealed to these participants as means to multitask, manage complicated exchanges of documents and to keep track of the state of collaborations.
- Unfortunately, the entire man-in-the-middle concept was complex and difficult for people to grasp.
- There was also no single, simple winning solution that this mechanism could address, with no strong differentiation in the resulting service, compared with other offerings from various web-hosted services.

While some of the feedback from this exercise was encouraging, we did not discover a major unmet customer requirement in the SOHO market that would be easy for us to deliver. Given that this market for our technology has never been a highly profitable one for technology products, we decided that this made it too high-risk to explore further with no compelling killer application opportunity.

B2B: A Leads Tracking Solution for Value Added Resellers

At this time in the development process, the latest fad for the Internet was the application service provider (ASP) model for delivering solutions to meet the needs of businesses in search of cheaper solutions for certain kinds of application (e.g., at this time, Oracle, SAP, PeopleSoft, and IBM began looking into providing ASP solutions for enterprise resource planning—ERP).

We wondered if there was an opportunity in this more lucrative space for collaborative IM and workflow tracking technology that could mediate between organizations with incompatible infrastructures. Thus, in this phase of our project, the search for a business model for our technology, rather than implications from fieldwork, began to drive the agenda. We identified a company with a specific problem that seemed perfectly matched to our mechanism.

This company (we'll call it Xco) is a large business software vendor that sold its products through value added resellers (VARs). The problem described to us was that Xco could not keep track of what was happening to the customer leads generated by its marketing department

(leads are details verified by Xco's marketing staff of customers who have enquired about and seem likely to buy one of their products), Hundreds of leads would be distributed amongst many VARs, but they often did not seem to translate into sales. Xco needed to be more certain as to whether the VARs were really pursuing these leads or allowing them to slip and thus not following up with all-important sales.

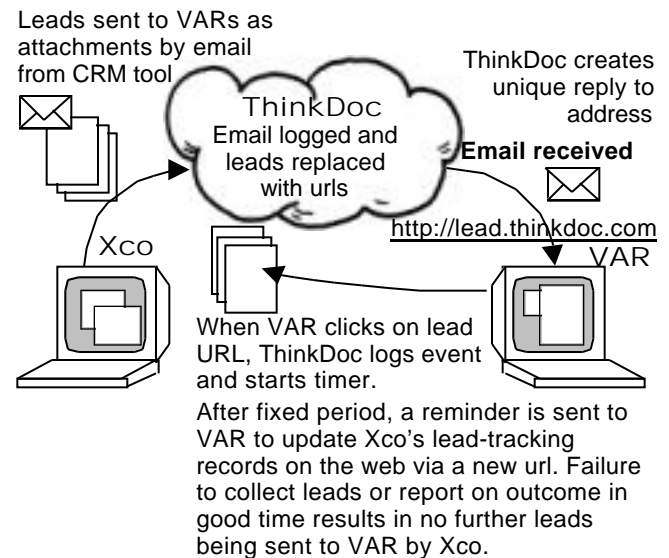


Figure 3. The ThinkDoc solution to Xco's leads tracking problem. Xco sells products through VARs with whom it shares no customer relationship management software that allows it to track how leads are followed up on.

We spent a few days at Xco and visited some of its VARs to interview marketing employees and observe how leads are handled. We rapidly developed a basic understanding of the tracking problem and proposed a specific solution based on our technology to allow the company to monitor the uptake of leads by VARs and force them to report back on progress before they could obtain further leads.

The solution, implemented through our man-in-the-middle email mechanism is depicted in figure 3. We presented this solution to Xco staff who found it compelling and strongly encouraged us to develop it further.

Now that we had a viable product requirement specification and a real customer we felt confident that we could develop a useful solution, which might form the basis for a strong business model. This business would be an ASP for leads tracking with IM, notifications and automatic status report generation capabilities all delivered in the email channel.

Competitive Analysis: Validating and Driving Design with Scenarios

Throughout the process of searching for a business opportunity for our ASP technology we needed to establish whether potential customers might see ThinkDoc as a clearly superior solution to problems, including, but not limited to the leads tracking problem of Xco. We explored

the space of competing solutions that were on offer to deal with these problems and selected a number of viable alternatives that met the following criteria, which we believed were selling points for our solution:

- **Infrastructure independent:** does not involve a single specialized software infrastructure across organizations (i.e., uses common resources such as email and the web).
- **Low cost:** Does not involve large initial investment and extensive maintenance.
- **Simple:** Does not require extensive training and customization.

We found the following alternatives (competitors) that potential customers might consider as solutions to various IM problems and the leads tracking problem:

- **Plain email:** use of any mail client would be included in this category.
- **ACT!:** A low-end CRM solution that can be used to manage contacts and track the state of communications with them. New versions of this tool integrate with Outlook to support message tracking.
- **i-Drive:** A shared repository on the Internet accessed via the web.
- **Eroom/HotOffice:** Two roughly equivalent virtual collaborative spaces that provide resources such as task lists and shared email folders.
- **Zaplets:** HTML collaboration resources, such as voting, scheduling etc. embedded in email messages. Users can click buttons in their email to send response to messages direct to central server on the Internet.
- **Done/Rememberit.com:** Two reminder services that send email notifications.
- **Marketforce/Partnerware/Upshot/Asera:** Four web-enabled CRM solutions. They provide an HTML front-end for web client interaction mainly, although email is often used for reminders and notifications.

Systematic Scenario-Based Side-By-Side Comparison

Our approach here was to develop a number of scenarios that illustrated our system's strengths and then to extract the key features that these scenarios demanded. These features were entered into a large spreadsheet in the first column, one below another. Across the top, we put headers for each of the competitors listed above. The last four CRM solutions were handled in a separate spreadsheet that was dedicated entirely to the leads tracking problem.

This process was an extended one that both reflected and drove our assessments in our competitive analysis and, while this process may be familiar to those in the business community, it is worth outlining in some detail for the benefit of the design community:

- A scenario would be created by one or more members of the team based on fieldwork, past experience or group discussions. It would usually be written up and distributed to the team for comment.

- We would then meet to discuss what critical requirements the scenario demanded from a solution to support it and enter these into the leftmost column of our spreadsheet. These would be things like "System sends automatic reminders," or "Can import legacy data."
- We would try to fill in each cell below each competitor at the top of each column with a description of their solution for the requirement. If we didn't think the competitor had a solution, the cell might be left blank (if in doubt) or we would enter "no" if we were sure.
- In many cases we had imperfect knowledge of our competition. This meant that some cells, which we thought represented potentially important differentiating features for our technology drove further research into the capabilities of our competition.
- As time progressed we would learn about interesting and useful features from one or more of our competitors that would cause new rows to be added to our spreadsheet. The empty cells in our own ThinkDoc column became new design requirements.

Depressing Realization 1

The ultimate outcome of our competitive analysis was to convince us that, despite the fact that our technology was novel, intriguing and useful from a technical and feature-based competitive standpoint, we could not find a decisive enough advantage over more web-centric solution vendors for us to get a business off the ground *from scratch*. These competitors were already too well established in our target market for CRM solutions and, regardless of our innovations, it would take us, as market latecomers, far too long to develop a customer-ready solution to a level where it could out-compete those already in existence.

After a number of explorations of different ways of implementing our IM in-the-email-channel model, we still had not overcome the business feasibility problems above.

Engineering and Evaluation

At the same time as the competitive analysis exercises, we prototyped the software that would support either an IM application or, ultimately, the leads tracking application. We built many system components for ThinkDoc, using Enterprise Java Beans. These included:

- SMTP gateway for capturing email traffic
- Mail parsing system for analyzing message content
- Storage for contents of received messages and state
- Security system for managing trusted network domains
- Interface for administrators to set up user accounts

Although many look-and-feel mock-ups of the interface were produced and reviewed, the actual UI to the working ThinkDoc machinery was delayed until very late in the day, after these and other infrastructure components were developed and stabilized. It proved particularly difficult to hook up the web interface to the complex code (for users to interact with archived content and work status information on our server) so we eventually compromised on a look-and-feel that was greatly scaled down from our original

vision. With these delays, we were not able to evaluate the working system's usability for a long time (see figure 4). The UI we did evaluate was far short of our ideal because we had invested so much less effort in it than in getting the infrastructure to deliver the ThinkDoc services reliably.

What we found when we did evaluate the UI was that, apart from having many usability bugs, the user experience was far short of where we would like it to be in terms of speed and reliability (for example, clicking on a link in a message to view present task tracking status, required the user to wait for a web browser window to open). A great deal more innovation and engineering effort would be required to meet busy multitasking users' performance expectations.

Depressing realization 2

While the infrastructure components were being built, a further system features feedback study was undertaken. We interviewed various lab members to get reactions to the features the system would deliver. While they were positive about many of our design ideas, there was no strong demand for computationally augmented email support for IM; *"That seems useful, but I probably wouldn't need it, because I don't do a lot of multitasking,"* was the sort of response we heard all too often. It turns out that our colleagues, mainly researchers setting their own agenda, have few coordination requirements (such as reminders, deadlines, to-dos and so on). The main candidates for evaluating our technology's features should be people with high levels of multitasking in their email. These people are not common in our lab; perhaps only our managers and some support staff qualify.

We now realized that we were unlikely to find a representative target user set for trying out the ThinkDoc prototype for two reasons. Firstly, for intellectual property reasons, our technology could not be revealed to outsiders and there were few email multitaskers in our own lab. Secondly, the UI was the least well-developed part of the code and would require a great deal of tolerance from users. Our target users, critically dependent on a fast and reliable tool to coordinate their work, were the least likely to be able to maintain this level of tolerance. Indeed our own manager deserves great credit for using our prototype for several weeks despite the inconvenience.

A late-in-the-day analysis of Outlook as a growing competitor to our approach revealed that many IM features possible with our in-the-channel model were also supported here including to-do's, tasks, reminders, threading (to track the state of an email conversation), read receipts, and so on. While not obvious, with a lot of tweaking and customization, we felt that Outlook could be configured by very expert systems support staff to provide a lot of the same benefits to end users that our approach might offer.

While ThinkDoc had the advantage of client independence, it suffered from a trade-off of requiring access to a server for IM resources. Outlook stores IM resources locally, reducing the time taken to get to them. We began to believe that for the growing number of Outlook users at least, the trade off of using ThinkDoc might not be worth the costs, especially

if future versions of Outlook improved on the existing features. This conclusion was discouraging.

Post Mortem of ThinkDoc

So what went wrong in pursuing our first model for embedding IM resources in email? We believe that, no matter how we had approached the problem, we would have arrived at the same conclusions sooner or later. These are:

- Putting IM resources in the email channel, no matter how we implemented it, would not allow us as a market latecomer to out-compete established web-based solutions in the more profitable market that we had targeted.
- While people generally liked the idea of better support for IM in their email, this was not a critical requirement for the majority of those who work in our own lab where we would be forced to test our non-disclosed software. For the minority for whom IM support is a key requirement, our existing prototype with its non-ideal UI was very heavyweight and awkward to try out.
- Outlook, by now a far more dominant player in the email market than when we began this project (recall that we saw several major players back in 1999), could be configured (albeit only by an expert) to provide many of the kinds of resources we are interested in already.

We had devoted around 5 person years to exploring this space (aside from the fieldwork) and coming to these conclusions. We felt that we could have done much better with our process. On the other hand, we had two very real accomplishments as a result:

- Our fieldwork helped us understand how email is used in depth-in, and in a way perhaps few other groups do.
- Our competitive analyses of various solutions out there, including the rapidly emerging main email contender, Outlook, now meant that we knew a lot more about what we were up against in terms of competition.

A Process Predicament With All the Best Intentions

Figure 4 depicts a simplified view of our process during the exploration of the IM in-the-email-channel model.

All of the work on the first model for getting IM support into email took place at the same time that enormous effort was being devoted to our fieldwork. Concurrent with this phase in the design we ran four separate studies to find out about user practices, requirements and to gather feedback for our emerging design ideas.

Fieldworkers diligently conducted these studies and channeled the findings into scenarios, reports, presentations and brainstorming exercises. Meanwhile, there was a great deal of intensive effort towards 10,000 lines of code devoted to the infrastructure required to process messages, automated responses, generate reports and so on. Competitive analysis helped us to understand what our competitors were capable of and where we needed to support additional requirements. Intensive brainstorming led to even more innovative technological mechanisms for IM in the email channel (e.g., documents and URLs that invoked computation embedded in messages).

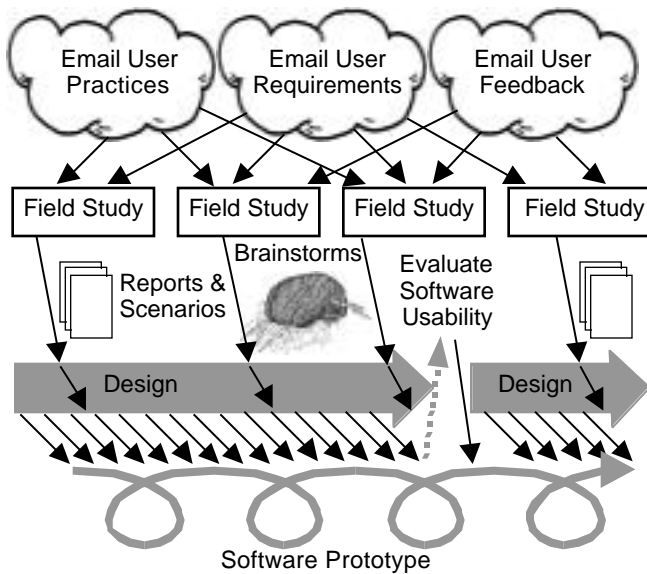


Figure 4. Schematic representation of our design process during exploration of Model 1: ThinkDoc IM in-the-email-channel. Findings from multiple fieldstudies were fed by fieldworkers to engineering in the form of reports scenarios and scenario-driven brainstorming. Engineering used insights to influence design. Software occasionally reviewed by all (dashed grey line) for usability.

However, despite our efforts, including many techniques used in accepted user-centered approaches such as Contextual Design [4], it appears that a fundamental problem in our process kept emerging, which was:

Fieldwork inspired engineering repeatedly led to innovations that delivered no significant end-user advantages over what was already supported.

Figure 4 illustrates this problem with fewer arrows connecting fieldwork to engineering than there are connecting data to fieldwork or engineering to the prototype. Further, the opportunity to review the utility of the working system came very late in the day.

At this point it was clearly time to radically rethink our approach. We needed:

- A technological means to improve support for IM that would minimally disrupt extremely overloaded email users, who are precisely the ones we want to help. This meant doing better than ThinkDoc in terms of user experience of IM in email.
- A methodology that would ensure that our fieldwork was more rapidly and systematically translated into design innovations that would then be evaluated in terms of what we knew about user practices and competing solutions, so that we would not invest so much effort in building technology that had no marketable advantages.

At one of our project reviews we finally decided to take an entirely different approach, which is described below.

IM in the Client

Towards the end of our efforts on our IM in-the-email-channel approach, as mentioned above, we conducted a late-

in-the-day competitive analysis of Outlook, which, while best suited to larger organizations with IT support for the mail server side, seems to be rapidly increasing in its popularity. This analysis revealed many features that we had overlooked for supporting IM beyond pure messaging. We realized that we had overlooked many of these features because we had not observed anyone using them in our fieldwork. We wondered whether (a year after our last investigation of email practices generally) they were in use yet at our lab. So we instigated yet another study. This was an online survey of 64 email users that enquired whether they used Outlook and if knew of and used certain features such as “Follow up,” “Tasks,” “Notes,” “Threading,” “Categories” and so on.

The results from 33 Outlook users suggested that even our technologically savvy lab members often did not use the more sophisticated features of Outlook (e.g., only just over half use the Tasks feature and far less than half use Categories to organize their mail). Our less tech savvy lab members (a small minority) were especially unlikely to use these features or even to know about them. We feel that these users are more likely to be representative of the majority of the population of email users than the technology experts in our lab.

At the same time, a high-level qualitative *condensation exercise* was conducted on the fieldwork findings to prioritize and extract the most important email-related problems of email multitaskers such as managers and coordinators. The top 6 email IM problems that we felt we had established through our many studies are as follows:

- Keeping track of lots of concurrent actions; (a) My to-do list and (b) Outstanding actions (things other people are doing for me).
- Marking things as important or outstanding.
- Managing activity extending over time or tracking threads.
- Managing deadlines and reminders.
- Event-based ordering of documents and discussions (meetings are a salient conceptual organizing principle).
- Collation of related items including email and documents (e.g., multiple responses to an email survey).

In principle, Outlook offers resources to handle each of these problems, but our analysis suggested that users are not making use of many of these resources, perhaps because it is too hard to discover and use them. Over the period of our project we interviewed many managers who are Outlook users who seemed to be complaining that they were still experiencing these problems, so we knew that Outlook was letting them down somehow.

Given the growing popularity of Outlook we decided that it might be strategically advantageous to explore the option of embedding IM resources in a client-based resource, rather than on a server where they would take time to access. We explicitly decided to trade client independence for speed and began designing a tool, TaskMaster, that worked *with* Outlook, bringing the best of its many underlying

capabilities to the forefront and adding features customized for IM embedded in email. However, as mentioned above, we now realized that a major flaw in our previous design process had been *to use fieldwork to inspire design, but with no mechanism for funneling fieldwork findings into the design process on a clearly motivated, systematic and critically selective basis*. We needed a methodology that would force us all (fieldworkers and engineers) to adopt new habits to promote better communication; we picked Extreme Programming XP as a promising option.

A Brief Primer of XP

In XP there are a small number of simple principles that define the methodology, making it easy to understand and adopt [2, 10, 11].

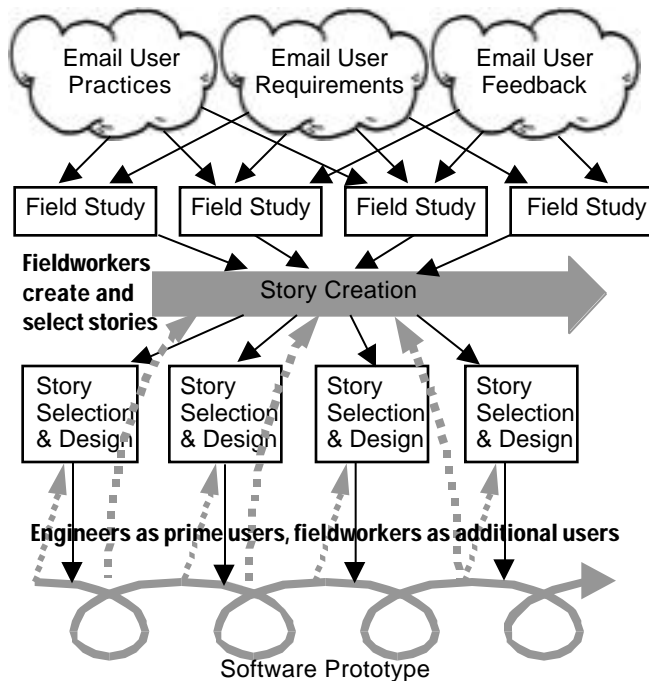


Figure 5. Schematic representation of our design process during exploration of Model 2: TaskMaster IM in-the-email-client. Findings from multiple fieldstudies are fed into stories by fieldworkers, which are used to drive and constrain design. Everyone, but especially engineers, uses prototype. Experience of use (dashed gray lines) leads to new stories and influences story selection process.

Risk Reduction: Engineers use a simple procedure for improving accuracy of engineering time estimates by assigning effort points to design objectives. The design-review cycle is kept very short, in our case to about two weeks, so that objectives are parceled into batches of an estimated two-week's worth of work. If estimates prove inaccurate then the number of allowed points per two-weeks is modified.

Customer is King: Customer representatives, in our case fieldworkers, mediate between engineers and the mass of

possible requirements for the software. Nothing is built that doesn't go directly towards customer requirements. For example, in our case the very first thing that was engineered for TaskMaster was a window that displayed a list of messages. Also there is no effort invested in any software infrastructure that eventually turns out to be unnecessary. For example, we are using Access as a database until we really can show that we need a faster (but more engineering intensive) database solution.

Scenario-Driven: Customer representatives specify short stories; lots and lots of them, throughout the development process, which are scenarios, each written on its own card, that describe how the system is going to work and what features it will have. These may be supported with sketches (in our case sometimes scribbles on a whiteboard since team members' offices are very close to one another).

Usable Prototype: Iteration takes place around a continuously usable prototype that can be quickly evaluated by users. This is accomplished by defining and running automatic tests of system behavior, such as "when the application starts up, all old messages in the database will reappear in the main window, and new items will be updated from Outlook." These tests are run throughout the design process to make sure everything keeps on working.

By adopting this approach and sticking to it, we have now established a much more direct and systematic line of communication between engineering and fieldwork in our project. We detail some of the advantages we have experienced on our project below.

Advantages of the XP Approach in Our Project

- **Limiting Unquestioned Assumptions:** Before using XP, fieldworkers would present findings, and engineers would then go ahead and design some features, but there was little attempt by fieldworkers to scrutinize these features and their implicit assumptions and to question their utility or usability (see figure 4). Now fieldworkers play the role of customer representatives, which entails specifying the features themselves, and engineers, who have not studied user practices do not make unfounded assumptions about what people want. Engineers usually take the line that they will not try to override the customer representative's requirements, even if they feel they are wrong (of course being this dispassionate is not always easy for us and heated debates do often break out). Ultimately the *customer representatives own the responsibility* for making sure the design does the right things. Of course, either side may be right or wrong in actuality (sometimes even users themselves ask for things that won't really work); the point is to minimize the usual design bias that arises from engineers responding to instincts about what is usable for them and fieldworkers failing to scrutinize the design for such assumptions.
- **Enforcing Clarity:** During the development of ThinkDoc, if a feature was suggested, it was not specified and scrutinized to any great level of detail. This left plenty of scope for usability problems that only emerged late in the day when email and web user interface elements were developed for the ThinkDoc

infrastructure. In XP, customer representatives define interaction stories. They then write tests to run to make sure each story is still supported. This generates detailed requirements and immediately reveals hidden ambiguities or misunderstandings that could lead to serious usability problems. For example, our fieldworkers often define stories that multiple implementations might satisfy (e.g., the window must display the incoming email messages in a list view). The ambiguity is only uncovered when they try to write a test to make sure the story is supported (we didn't say how many messages and what to display if the window isn't big enough).

- **Feedback for Refinement:** *In our development of ThinkDoc, we had no usable prototype for a long time, while a sophisticated infrastructure for handling the message processing on the server was developed.* Now having a constantly working prototype is a key social catalyst for design interaction. It affords demonstrations, which spur interest and suggestions for refinements. The customer representatives can also use it and provide feedback (which spurs even more, usually useful, debate about design refinements that reflect poor communication, see our previous point. As a real example here; Customer representative, "It doesn't have a scrollbar, how can I read the whole message?!!!" Engineer, "You didn't put that in your story.").
 - **Debating Usability and Utility:** *With no UI to the prototype until very late in the day with ThinkDoc, there was no way of knowing what it would really be like to use the system. Discussions about usability tended to take a back seat and were entirely hypothetical when they did occur.* XP focuses efforts on usable features before any infrastructure is ever built, and our prototype, as a result, is constantly in use. It must be admitted that our engineers are still the main users of the prototype, in our case, due to one of the fieldworkers being present less than half time, one of us being remote and one of us being a Mac user. However, as the prime users of the system, the engineers are now usually the first to spot certain usability problems (at least for their own uses and practices) as they have the most experience of dealing with their email using the system. Healthy discussion about usability issues and the utility of our design conception as a whole now regularly arises as a result of everyone's use of the system. Our personal experience is balanced against our own and others' research, which shows that email users' practices vary widely, so not all users will be likely to want or even tolerate the same features as some or all of our team do.
 - **Widening the Design Space:** *Without experience of use, all features proposed for ThinkDoc were derived from scenarios about things that we could imagine. But there was plenty of opportunity for oversights as far as what might be the most useful features.* So an advantage closely related to the one above with XP is that, since all of the team, and especially the engineers, have experience of using the prototype and stumble on a wide range of problems and ideas for possible improvements that only direct experience can give. Engineers frequently submit
- their own stories in response to this experience. This is not a violation of XP, simply an additional resource that can be taken advantage of by the method. Occasionally something comes up that is an important new requirement that only experience of intensive, long-term use, such as the engineers have, can reveal. For example the need to get temporarily inactive threads out of the way for a while inspired a feature that would reduce clutter on a temporary basis. On the other hand, it is frustrating as a more experienced user to see things that would make your life easier sometimes get passed over in the story selection process. It is even more frustrating when something gets into a story that actually makes things even less usable, as is occasionally the case.
- **Focusing the Scope:** *We used to have a lot of brainstorming meetings in which we came up with a great many interesting ideas, but we had no mechanism for feature prioritization.* With the discipline of XP forcing customer representatives to choose between a huge range of possible features and refinements at every iteration, where new story cards to implement are selected, only the most important stories from the customer perspective are chosen from the pile. It doesn't matter when the story was created, at any time, only the most important get chosen each time round. Some early stories have simply sat on the pile indefinitely and we have as many stories-in-waiting as implemented ones. The selected stories are chosen bearing in mind our experiences from our fieldwork. For example, knowing how invested people are in their existing email tool meant that, early on, we realized that users must have the ability to import legacy email from Outlook, but we left the less critical requirement for feedback when hitting a target for drag-and-drop until later.
 - **Engineers' Revenge?** *It has traditionally been the role of the customer to ask for the moon and the role of engineers to reply that the best that can be delivered is a balloon. This was typically the case with ThinkDoc with engineers constantly saying that there way only so much that could realistically be accomplished in a reasonable amount of time.* XP puts the decision of what to cut back in the court of the customer, through the selection of stories. Engineering says, "We'll do whatever you put on the cards" so the customer now must think about their strategic objectives, how those are related to the timeline of evaluations, deliverables and so on. Further, as unexpected engineering problems arise, these land back in the lap of the "king" who must decide which strategic path to pursue in the light of these changes. This means that an understanding of the limits of what can be accomplished is far better distributed amongst the stakeholders in the project than was previously the case.
 - **I Hear Your Point:** *We used to have a lot of arguments about the features that ThinkDoc should support.* Now, we still have a lot of arguments, but XP provides a mechanism for diffusing the contention around whether or not a feature is an important requirement. We can now acknowledge the feature suggestion, no matter who made it, turn it into a story and add it to our pile of

story cards. When we come to choose between stories to implement at our next iteration, it is much clearer what is really important when it is balanced against the many other outstanding features we have to choose from.

- **Real Data, Real Evaluation, Really Early:** *In ThinkDoc we didn't get experience with real data (actual email message handling) until late in the day.* With XP, the prototype we are building is being tested on real data from the outset. This is unusual for groupware, which is more often than not evaluated with artificial data and user tasks in a short-term study (a few hours), late in the design cycle. This testing-in-anger from early on in development can reveal the sorts of problems that remain obscure for pure look-and-feel early prototypes; such as sorting algorithms being too inefficient on slow machines or the inability to handle unusual, but necessary data objects (like Outlook event invitations).

Disadvantages of the XP Approach in Our Project

There are definitely some downsides to adopting XP, which, even though they do not outweigh the benefits in our project, should be listed.

- **A Very Ugly Baby:** XP is built upon the principle of a non-throw-away prototype, a common engineering practice in the real world, but counter to the wisdom of proponents of throw-away prototyping in the engineering community [e.g., 5]. For this reason there is less incentive (and less engineering and usability analysis resources) to produce a good-looking prototype purely for look-and-feel evaluation. It's not impossible, but one tends to focus one's resources on the existing prototype and refining the usability problems it has. Thus our early prototype has been a very ugly baby indeed. We have produced some more attractive UI in our stores, but these have not yet been implemented.
- **How simple is simple?** Part of the philosophy of XP is to build nothing that isn't in a chosen story. The idea is to have the *simplest implementation* that meets the specification. In practice this can lead to some problems deciding where to draw the line between what needs to be specified, versus what is just obvious design. In an earlier example we mentioned that at one point customer representatives complained that there was no scrollbar on the message content window. It was assumed that this was obvious and so it wasn't specified. Engineers simply didn't implement it until a complaint arose. Some debate arose and remains largely unresolved about how much detail needs to be explicitly described in stories.
- **You Need to Know Your Stuff:** Even for innovative design situations such as ours, XP does not recommend throw-away, look-and-feel prototypes that might be used for iterative user testing and refinement of the big ideas. If you are not going to produce such prototypes, as we have not, then you must have a good grip on customer requirements or fieldwork findings by the time you engage in XP in order to prioritize engineering effectively. At the very least, you need to have some very good data, and if fieldwork is still going on, you still need someone to mediate between busy fieldworkers and

busy engineers to prepare scenarios and performance tests, and to run the prototypes and do usability walk-throughs.

- **Expectations of Use:** The flip-side of having a running prototype constantly available is that engineers tend to expect that everyone should use the prototype to support the design. But some of us have slipped back to our old email tool and habits because under some circumstances, the prototype has simply been very slow and buggy. This has been an occasional source of conflict with both sides having clearly valid points of view; people have to use the software to help evaluate it, but they also have to get their email work done. There is no easy way around this tension here.
- **How can we Evaluate Our Prototype?** We are building a prototype that will ultimately be extremely difficult to evaluate. It is intended to provide support that will best benefit email users who are overloaded with multitasking. Other users will probably find it only mildly better or perhaps even worse than their existing email tool. However, it is those overloaded users who are the least willing to tolerate an unstable prototype to manage the email that they are already struggling with. We discuss our strategy for dealing with this in the future work section below.
- **Is the Customer Really King Here?** We are using a research prototype to explore some ideas about how to embed IM deeply within email, rather developing a standard application for a well-known domain. We have no precedent for some of the fundamental new ideas in our system that could give us any direct experience or information about what users think about them at this point. So we are committed to investing effort building working implementations of features that may not turn out to be useful at all for our email multitaskers. As far as we know, there is no way around this in these circumstances because we do not believe a look-and-feel prototype will really expose the benefits of features (such as the advantage of having reminders better integrated into one's email). Only long-term use of a working tool will prove us right or wrong.
- **Innovation Comes Late:** The opposite side of the coin of the last of the advantages we listed is that XP may work against extreme design innovation and exploration, particularly in its early stages. Just getting something up and running; a window with email in it, text panes with scroll bars, and so on was not a very impressive early reification of what is interesting about our design research effort, despite the fact that it worked with real email. It's not that innovation has been impossible for us; it's just that it had to be prioritized against having a working and usable prototype.

This last problem may be a self-imposed sentence for our research project because we have chosen to build something that can be evaluated through real use rather than in a constrained and controlled short term study, such as a task-based walk-through, or a short experimental session with an artificial set of email and other items. Consequently, since we have limited people and time resources, we have

striven for a convincing proof-of-concept prototype that really works with people's real email data as soon as possible, at the expense of focusing on innovative prototypes with highly experimental visualizations and functions that might only work as a less persuasive demo.

ONGOING AND FUTURE WORK

Throw-Away Users v Throw-Away Prototypes

At the present time we are ready to release our code to other members of staff in our lab. We are not planning to evaluate throw-away-prototypes, such as paper mock-ups or look-and-feel prototypes, since we feel that they will not really convey what is interesting about our technology to users (e.g., will our reminders work for users who are busy doing actual work). We need users to be able to experiment with real data over an extended period of time in order to discover if our design works. Our current prototype has been in use for long enough for us to be confident that other users will be able to use it to read their real email.

However, as mentioned above, we are aware that the people who are most representative of the ultimate target users of our system are the least likely to be able to tolerate a prototype that is in any way unreliable or difficult to use. For this reason we are planning a two-phase evaluation study. In the first phase we are asking researchers in our lab who are not extreme email multitaskers to try out the software to iron out usability kinks and provide us with feedback about how to improve the features. We call these, jokingly throw-away users, since they are not our intended target users and we intend to keep the prototype they use and simply improve the software after they have used it.

In the next phase, we will try out our software on a more demanding "realistic" user population. We are designing an in-depth qualitative and quantitative study to track their email practices both before they use the prototype and then while they use it, for comparison. From these users we will discover if our design efforts have been truly successful.

CONCLUSIONS

We explored two models for embedding information management in email. Our exploration of the first model, embedding IM in the channel, was unsuccessful even *with* a user-centered approach, largely because fieldwork inspired engineering repeatedly led to innovations that delivered no significant end-user advantages over what was already supported. For our second exploration of IM embedded in the client, we adopted XP as an approach to keep design closely tied to fieldwork findings and usage experience, with fieldworkers playing the part of customer representatives. This approach has been far more successful, giving us a usable prototype in a fraction of the time it took to explore the first model.

ACKNOWLEDGMENTS

To Karen Marcelo and Trevor Smith for their contributions to ThinkDoc code. To the many email users who took part in our various studies. And finally, to our managers, the Richards, Bruce and Burton for continued support of our research direction, even when we lost faith in it ourselves.

REFERENCES

1. Barreau, D.K. and Nardi, B. (1995). Finding and reminding: File organization from the desktop. *ACM SIGCHI Bulletin*, 27(3), 39-43.
2. Beck, K. (2000) *Extreme Programming Explained*. Reading, Massachusetts, Addison Wesley.
3. Bellotti, V., & Smith, I. (2000). Informing the design of an information management system with iterative fieldwork, *Conference proceedings on designing interactive systems: processes, practices, methods, and techniques* (pp. 227-237). August 17 - 19, 2000, Brooklyn, NY United States.
4. Beyer, H. & Holtzblatt K. (1997) *Contextual Design*. San Francisco: Morgan Kaufman.
5. Brooks, Fred (1975). *The Mythical Man-Month*. Reading, Massachusetts: Addison-Wesley.
6. DeSanctis, G., & Poole, M. S. (1994). Capturing the complexity in advanced technology use: adaptive structuration theory. *Organization science*, 5(2), 121-147.
7. Ducheneaut, N., & Bellotti, V. (2001). Email as habitat: an exploration of embedded personal information management. *Interactions*, 8(5), 30-38.
8. Ducheneaut, N., & Bellotti, V. (forthcoming a). Ceci n'est pas un objet? Talking about objects in email. *Journal of Human-Computer Interaction*, *Forthcoming*.
9. Ducheneaut, N., & Bellotti, V. (forthcoming b). *A study of email work activities in three organizations* (Working paper): Xerox PARC.
10. Extreme Programming. <http://www.extremeprogramming.org>
11. Jeffries, R., Anderson, A. & Hendrickson, C. (2001). *Extreme Programming Installed*. Reading, Massachusetts, Addison Wesley.
12. Lansdale, M. (1983). The psychology of personal information management. *Applied Ergonomics* 19, 55-66.
13. Mackay, W. E. (1988). More than just a communication system: diversity in the use of electronic mail, *Proceedings of the conference on computer-supported cooperative work*. September 26 - 28, 1988, Portland, OR USA.
14. Malone, T. W. (1983). How do people organize their desks? Implications for the design of office information systems. *ACM Transactions on Office Information Systems*, 1, 99-112.
15. Venolia, G. Dabbish, L., Cadiz, J. Gupta, A. (2001) *Supporting Email Workflow*, MSR-TR-2001-88.
16. Whittaker, S., & Sidner, C. (1996). Email overload: exploring personal information management of email, *Conference proceedings on Human factors in computing systems* (pp. 276-283). April 13 - 18, 1996, Vancouver Canada.