

# CoScripter: Automating & Sharing How-To Knowledge in the Enterprise

Gilly Leshed<sup>1</sup>, Eben M. Haber<sup>2</sup>, Tara Matthews<sup>2</sup>, Tessa Lau<sup>2</sup>

<sup>1</sup>Information Science, Cornell University  
301 College Ave., Ithaca, NY 14850  
gl87@cornell.edu

<sup>2</sup>IBM Almaden Research Center  
650 Harry Rd., San Jose, CA 95120  
{ehaber, tmatthe, tessalau} @us.ibm.com

## ABSTRACT

Modern enterprises are replete with numerous online processes. Many must be performed frequently and are tedious, while others are done less frequently yet are complex or hard to remember. We present interviews with knowledge workers that reveal a need for mechanisms to automate the execution of and to share knowledge about these processes. In response, we have developed the CoScripter system (formerly Koala [11]), a collaborative scripting environment for recording, automating, and sharing web-based processes. We have deployed CoScripter within a large corporation for more than 10 months. Through usage log analysis and interviews with users, we show that CoScripter has addressed many user automation and sharing needs, to the extent that more than 50 employees have voluntarily incorporated it into their work practice. We also present ways people have used CoScripter and general issues for tools that support automation and sharing of how-to knowledge.

## ACM Classification Keywords

H.5.3 Group and organization interfaces: web-based interaction.

## Author Keywords

Automation, scripting, programming-by-demonstration, procedural knowledge, knowledge sharing, wiki, user study.

## INTRODUCTION

Employees in modern enterprises engage in a wide variety of web-based processes as part of their day-to-day work. Some of these processes must be performed frequently and are tedious (*e.g.*, conference room reservations). Others are less frequent but can be hard to perform because they require intricate domain knowledge (*e.g.*, expense reimbursement, procurement), are difficult to locate (*e.g.*, selecting insurance beneficiaries), or are too complex to remember. Knowledge of these processes is distributed throughout

an enterprise, leading employees to spend excessive effort not only in learning processes, but also in completing them.

Automation could help users complete web-based processes more quickly. However, most people do not have the necessary skills for developing custom automation. Further, it is unlikely that businesses will spend the resources to develop automated processes for all tasks due to the sheer number of processes and their frequently-changing nature [3,13].

Workplace social networks can help with the creation and distribution of how-to knowledge [15]. However, often employees may not know whom to ask and even if they know, it burdens experienced members of a community.

Our solution to these problems is CoScripter (formerly Koala [11]), a tool that enables end-users to create and share scripts to perform web-based processes. CoScripter consists of two components: (1) a browser plug-in that allows users to record and play back actions in a web browser, storing these actions as scripts of human-readable text, and (2) a wiki that serves as a central repository, where users can share, edit, rate, and search for scripts. CoScripter has been deployed within a large corporation since November 2006, and is regularly used by more than 50 employees at present.

By recording user actions, CoScripter captures how-to knowledge, helping materialize tacit knowledge (*i.e.*, practical knowledge that is in people's heads) in an explicit form. This enables a continuous dialogue between tacit and explicit knowledge, and as such builds the organization's body of knowledge [17], by translating these types of knowledge back and forth and transmitting them across the organization. Furthermore, by sharing this knowledge across communities of practice, CoScripter supports the transfer of best practices within an organization. As such, it has the potential to solve what Szulanski calls "internal stickiness," that is, the difficulty in transferring practical knowledge [22].

Our work brings together three main contributions. First, we present results from an interview study that explores how people practice, learn, and share their procedural knowledge in the enterprise. Second, we present results from an extended deployment of an end-user programming system in a large organization. Finally, we discuss a number of general issues that arose in the deployment that must

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI 2008, April 5–10, 2008, Florence, Italy.

Copyright 2008 ACM 978-1-60558-011-1/08/04...\$5.00.

be addressed by any similar system to succeed in a large company setting.

## RELATED WORK

Here we present prior work related to our three main contributions: user studies of end-user programming systems to which our second interview study adds new findings; empirical studies of organizational knowledge sharing, upon which our first interview study builds; and tools for automation and sharing, like CoScripter.

### User Studies of End-User Programming

Studies of end-user programming have a rich history. Nardi discovered collaboration practices that arose around spreadsheet and CAD tools [16], Rosson *et al.* examined end-user web development, showing that most tools are not up to the needs of non-programmers [19], and, Dorn and Guzdial report on the informal learning of programming skills by graphic designers [7]. We add to this body of knowledge, studying the use of an explicitly distributed and collaborative end-user programming tool (CoScripter) and describing the different communities and sharing practices that arose from its use.

### Empirical Studies of Organizational Knowledge Sharing

Knowledge Management considers how organizations represent, create, and disseminate knowledge to improve their competitiveness in the business world [5,18,20]. Empirical studies are common (see a review in [2]), as are studies of internal knowledge transfer [22] that search for causes of “information stickiness.”

Past studies help us to understand practices of sharing how-to knowledge in organizations. Edwards and Mahling [8] present a set of practices for sharing procedural knowledge in law offices, but their analysis was not based on an empirical method. Other research has presented analyses based on field studies of general knowledge sharing practices and needs in employees’ work [1,10,14], factual, scientific knowledge sharing [4], and FAQs development and sharing [9]. Perhaps most related is a field study of service technicians and their practices in seeking information to solve problems [23]. However, it focused on a small community and on specific problem solving practices.

We add to the body of empirical work exploring organizational knowledge sharing by focusing on practices that could be improved by both automating processes and sharing them. We follow Suchman’s call to observe the actual work involved in accomplishing procedural tasks [21].

### Tools for Supporting Automation and Sharing

CoScripter is a tool that brings together known ideas in a novel combination: (1) it allows end-user automation of procedures through recording and scripting, and (2) it stores scripts on a shared central wiki.

The programming-by-demonstration (PBD) approach used in CoScripter has a very long history [6]. Other browser recording and playback tools exist (*e.g.*, iMacros (<http://www.iopus.com/imacros/>), but none have used a human-readable recording format as in [12] together with a collaborative script repository.

Wikis have been growing in popularity as collaborative information repositories, but have seldom been used as a programming environment. There are some wiki implementations that support programming as a means of adding dynamic content (*e.g.*, Swiki (<http://wiki.squeak.org/swiki/>), QEDWiki (<http://services.alphaworks.ibm.com/qedwiki/>), but CoScripter is unique in supporting scripts whose execution accomplishes some task outside of the wiki. Other environments for sharing knowledge about procedural tasks (*e.g.*, wikihow (<http://www.wikihow.com/>), howtoforge (<http://howtoforge.com/>), and howtopedia (<http://howtopedia.org/>)) typically consist of offline procedures that must be created and executed manually, unlike CoScripter’s recordable and runnable scripts.

This unique combination of features, namely, PBD-based recording of human-readable scripts and a wiki-based repository, is exceptionally powerful. It allows any employee to record scripts, share them with others, or search and learn from the central repository of human-readable scripts.

## INTERVIEW STUDY 1: EXISTING NEEDS

We performed an initial study, interviewing employees in a large corporation, to understand what business processes were important in their jobs, and to identify existing practices and needs for learning and sharing these processes.

### Participants

We recruited 18 employees of the large corporation in which CoScripter was deployed. Since our goal was to understand business process use and sharing, we contacted employees who were familiar with the organization’s business processes and for whom procedures were a substantial part of their everyday work. Our participants had worked at the corporation for an average of 19.4 years (ranging from a few weeks to 31 years, with a median of 24 years). Twelve participants were female. Seven participants served as assistants to managers, either administrative or technical; 6 held technological positions such as engineers and system administrators; 3 were managers; and 2 held human resource positions. The technology inclination ranged from engineers and system administrators on the high end to administrative assistants on the low end. All but two worked at the same site within the organization.

### Method

We met our participants in their offices. Our interviews were semi-structured; we asked participants about their daily jobs, directing them to discuss processes they do both frequently and infrequently. We prompted participants to

demonstrate how they carry out these procedures, and probed to determine how they obtained the knowledge to perform them and their practices for sharing them. Sessions lasted approximately one hour and were video-recorded with participants' permission (only one participant declined to be recorded).

## Results

We analyzed data collected in the study by carefully examining the materials – video-recordings, their transcripts, and field notes. We coded the materials, marking points and themes that referred to our exploratory research goals: (1) common, important business processes, and (2) practices and needs for learning and sharing processes.

Note that with our study method, we did not examine the full spectrum of the interviewees' practices, but only those that they chose to talk about and demonstrate. Nonetheless, for some of the findings we present quantified results based on the coded data. In the rest of the paper, wherever a reference is made to a specific participant, they are identified by a code comprised of two letters and a number.

### *What Processes Do Participants Do?*

We asked our participants to describe business processes they perform both frequently and infrequently. The tasks they described included web-based processes, other online processes (e.g., processes involving company-specific tools, email, and calendars), and non-computer-based processes. Given that CoScripter is limited to a Firefox web browser, we focused on the details of web-based tasks, but it is clear that many business processes take place outside a browser.

We found that there was a significant amount of overlap in the processes participants followed. Seventeen participants discussed processes that were mentioned by at least one other participant. Further, we found that a core set of processes was performed by many participants. For example, 14 participants described their interactions with the online expense reimbursement system. Some other frequently mentioned processes include making travel arrangements in an online travel reservation system, searching for and reserving conference rooms, and purchasing items in the procurement system. *These findings show that there exists a common base of processes that many employees are responsible for performing.*

Despite a common base of processes, we observed considerable personal variation, both within a single process and across the processes participants performed. A common

cause for variation within a single process was that the exact input values to online tools were often different for each person or situation. For example, DS1 typically travels to a specific destination, whereas LH2 flies to many different destinations. We observed variations like these for all participants. Secondly, there were a number of processes that were used by only a small number of people. Eleven participants used web-based processes not mentioned by others. For instance, TD1, a human resource professional, was the only participant to mention using an online system for calculating salary base payments. *These findings suggest that any process automation solution would need to enable personalization for each employee's particular needs.*

Our participants referred to their processes using various qualities, including familiarity, complexity, frequency, involvement, etc. Some of these qualities, such as complexity, were dependent on the task. For example, purchasing items on the company's procurement system was a challenging task for most participants. Other qualities, such as familiarity and frequency, varied with the user. For example, when demonstrating the use of the online travel system, we saw CP1, a frequent user, going smoothly through steps which DM1 struggled with and could not remember well. We observed that tasks that were *frequent* or *hard-to-remember* for a user may be particularly amenable to automation.

***Frequent processes.*** Participants talked about 26 different processes they performed frequently, considering them tedious and time-consuming. For example, JC1 said: “[I] pay my stupid networking bill through procurement, and it's the same values every month, nothing ever changes.” *Automation of frequent processes could benefit users by speeding up the execution of the task.*

***Hard-to-remember processes.*** At least 8 participants mentioned processes they found hard to remember. We observed two factors that affected procedure recall: its complexity and its frequency (though this alignment was not absolute). In general, tasks completed infrequently, or that had complex steps, were often considered hard-to-remember. For example, a user of the procurement system said, “It's not so straightforward, so I always have to contact my assistant who contacts someone in finance to tell me these are the right codes that you should be using.” Alternatively, although AB1 frequently needed to order business cards for new employees, it involved filling out multiple data fields she found hard to remember. *Automation could benefit users of hard-to-remember tasks by relieving the need to memorize their steps.*

### How Do Participants Share Knowledge?

An important goal of our interviews was to develop an understanding of the sharing practices that exist for procedural knowledge. Thus, we asked participants how they learned to perform important processes, how they captured their procedural knowledge, and how and with whom they shared their own procedural knowledge.

**Learning.** Participants listed a variety of ways by which they learned procedures, most of them listing more than one approach. Figure 1 shows the different ways people learned procedures. Note that the categories are not mutually exclusive. Rather, the boundaries between contacting an expert, a colleague, and a mentor were often blurred. For example, KRI needed help with a particular system and mentioned contacting her colleague from next door, who was also an expert with the system. Participants often said they would start with one learning approach and move to another if they were unsuccessful. Interestingly, although each participant had developed a network of contacts from which to learn, they still use trial-and-error as a primary way of getting through procedures: 13 out of 18 participants mentioned this approach for obtaining how-to knowledge. *This finding indicates that learning new procedures can be difficult, and people largely rely on trial-and-error.*

For maintaining the acquired knowledge, 15 out of 18 participants kept or consulted private and/or public repositories for maintaining their knowledge. For the private repositories, participants kept bookmarks in their browsers as pointers for procedures, text files with “copy-paste” chunks of instructions on their computer desktops, emails with lists of instructions, as well as physical binders, folders, and corkboards with printouts of how-to instructions. Figure 2 shows sample personal repositories. *Users create their own repositories to remember how to perform tasks.*

One participant noted an important problem with respect to capturing procedural knowledge: “Writing instructions can be pretty tedious. So, if you could automatically record a screen movie or something, that would make it easier to [capture] some of the instructions. It would be easy to get screenshots instead of having to type the stories.” *This feedback indicates that an automatic procedure recording mechanism would ease the burden of capturing how-to knowledge (for personal or public use).*

**Sharing.** Eleven participants reported that they maintain public repositories of processes and “best practices” they considered useful for their community of practice or department. These repositories were commonly shared as databases accessible through the corporate email client. Four participants noted that although their repository was publicly open for posting and reading, they were the sole users of it. DS2 said: “I

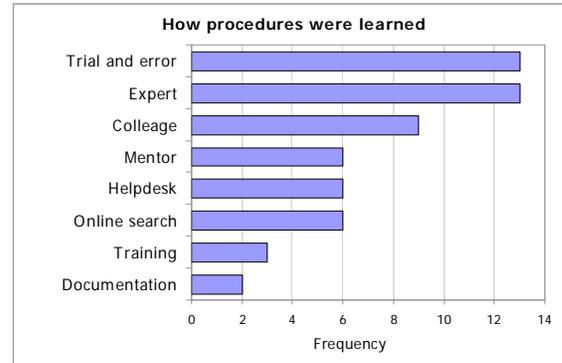


Figure 1: Frequency of ways participants learned new processes.

developed this, and I sent a link to everyone. People still come to me, and so I tell them: well you know it is posted, but let me tell you.” Using public repositories suggests that participants sought an open forum for others to find and share their how-to knowledge. However, knowledge was distributed across multiple single-authored repositories, which may have made it harder for learners to find. In fact, seven participants reported that they had resorted to more proactive sharing methods, such as sending colleagues useful procedures via email.

Fifteen participants said they serve as sources of information for many others (this number is high due to our focus on recruiting senior members of the organization or people with information sharing as a job function). These participants earned their position as knowledge sharers by serving as mentors for other employees; by assuming this role as part of their position (e.g., administrative assistants, HR professionals, etc.); and by simply being known in their community as experts for various processes.

People in need find experts in a variety of ways. One participant said, “I’m usually one of the people who gets called and [contacted via IM] and emailed because I’ve been at it for too long.” An experienced administrative assistant was listed as an expert for the administrative assistants’ community in the company’s directory: “They will contact you if they have questions regarding, let’s say, archiving. Because I’m on the list, they’ll call me.”

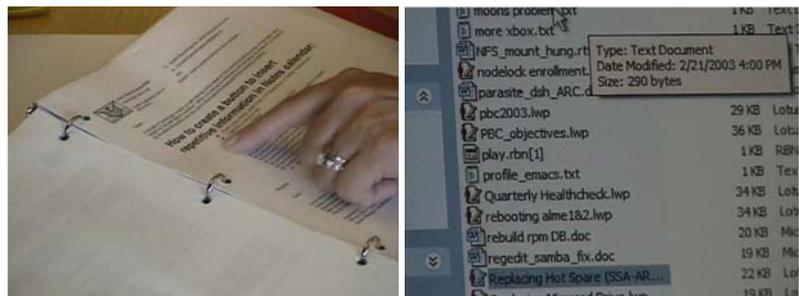


Figure 2: Ways participants maintain their procedural knowledge: one administrative assistant’s physical binder with printouts of emails with how-to instructions (left); a computer desktop folder named ‘Cookbook’ with useful procedures and scripts of a system administrator (right).

These results give examples of employees who seek to share their how-to knowledge within the company and ways in which learners find the knowledge. Nonetheless, *our results show that sharing is time-consuming for experts and shared repositories are seldom used by learners, suggesting that sharing could be bolstered by new mechanisms for distributing procedural knowledge.*

### Discussion

Study 1 has shown that there is a need for tools that support the automation and sharing of how-to knowledge in the enterprise. We observed a core set of processes used by many participants, as well as less-common processes. Processes that participants used frequently were considered routine and tedious, whereas others were considered hard-to-remember. Automating such procedures could accelerate frequent procedures and overcome the problem of recalling hard-to-remember tasks.

Our data also suggest that existing solutions do not adequately support the needs of people learning new processes. Despite rich repositories and social ties with experts, mentors, and colleagues, people habitually apply trial-and-error in learning how to perform their tasks. New mechanisms are needed for collecting procedural knowledge to help people find and learn from it.

We also found that people who were sources of how-to knowledge needed better ways for capturing and sharing their knowledge. These people were overloaded by writing lengthy instructions, maintaining repositories with “best practices,” and responding to requests from others. Also, distribution of their knowledge was restricted due to limited use of repositories and bounds on the time they could spend helping others. As such, automating the creation and sharing of instructions could assist experts in providing their knowledge to their community and colleagues.

We now turn to describing CoScripter, a system we developed to support the needs uncovered in our interviews.

### COSCRIPTER: COLLABORATIVE SCRIPTING

CoScripter is a system that enables users to record, play back, edit, and share web-based tasks. An earlier paper describes in more detail its mechanisms for recording, playback, sloppy interpretation, and script personalization [11]. We summarize these mechanisms below.

CoScripter includes a plug-in for the Firefox browser, which appears as a browser sidebar (Figure 3). This plug-in can record a user’s actions in the browser and play them back at a later time. User actions are recorded as pseudo-natural-language instructions such as “go to <http://portal.acm.org>” and “click the Search button.” Because these instructions are purely textual, and do not require a precise syntax, CoScripter scripts can be read and modified by people as easily as reading and editing text.

CoScripter can also parse these instructions, and programmatically execute them in order to automatically perform the steps in a script. Scripts may be executed all at once using a “run” mode, which enables people to automate frequent or rote tasks. Scripts may also be executed one step at a time, which is useful for learning how to perform complex or unfamiliar tasks.

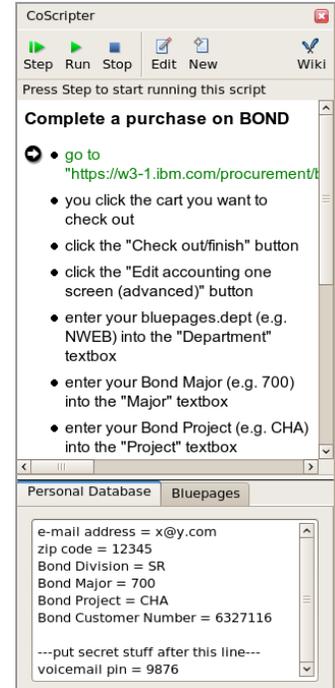
Some steps require the user to take an action before continuing, such as entering their password. These steps are recognized by the presence of the word “you” in the instruction. When encountering such a step, CoScripter will pause and wait for the user to follow the instruction before continuing. This “mixed-initiative” style of interaction enables users to benefit from partial automation.

Scripts can also be personalized through the use of a “personal database” of name/value pairs initially seeded from the corporate employee directory (bottom part of Figure 3). During recording, if a value in the personal database is entered into a form, the instruction will generalize the step to refer to the named variable instead. During playback, values from the user’s personal database will be substituted into the script at the appropriate point. This feature facilitates script re-use by making scripts more general, only personalizing the script at run-time.

Scripts are, by default, stored publicly on the CoScripter wiki to enable sharing between users, though users may mark scripts as private to prohibit others’ access. Users can also browse, tag, rate, edit, and run any public script directly from the wiki. Thus the wiki serves as a central repository of “how-to” knowledge. Experts can share their knowledge by recording their actions and publishing the results on the wiki, and learners can easily find it there.

### STUDY 2: REAL-WORLD USAGE

The first interview study explored needs and practices of sharing how-to information. In a second study, we examined how well CoScripter supports these needs and practices through analysis of usage logs and interviews with regular users. The purpose of these studies was threefold:



**Figure 3:** CoScripter sidebar (appears on left side of Firefox browser) with a script and data in the personal database.

(1) determine how well CoScripter supported the user needs discovered in Study 1, (2) learn how users had adapted CoScripter to their needs, and (3) uncover outstanding problems to guide future CoScripter development.

### **Log Analysis: Recorded User Activity**

CoScripter has been available within a large corporation since November 2006. Usage logs provide a broad overview of how CoScripter has been used in one company, while the interviews in the following section provide more in-depth examples of use. In this section we present an initial analysis of quantitative usage, with a content analysis left to future work. The data reported here excludes the activities of CoScripter developers.

#### **Script Usage Patterns**

Users are able to view scripts on the CoScripter wiki anonymously; registration is only required to create, modify, or run scripts. Of the 1200 users who registered, 601 went on to try out the system. A smaller subset of those became regular users, either recently or in the past.

We define *active users* as people who have run scripts at least five times with CoScripter, used it for a week or more, and used it within the past two months. Fifty-four users (9% of 601 users) are active users. These users, on average, created 2.1 scripts; ran 5.4 distinct scripts; ran scripts 28.6 times total; and ran a script once every 4.5 days. While 9% may seem to be a relatively low fraction, we are impressed by the fact that 54 people have voluntarily adopted CoScripter, and derive enough value from it to make it a part of their work practices.

We define *past users* as those who were active users in the past, but have not used CoScripter in the past two months. This category consists of 43 users (7%). Finally, we define *experimenters* as those who tried CoScripter without becoming active users, of which we have 504 users (84%).

The logs suggest that individual users are automating frequent tasks, with 23 scripts run 10 or more times by single users at a moderate interval (ranging from every day to twice per month).

#### **Collaborating over Scripts**

One of the goals of CoScripter is to support sharing of how-to knowledge. The logs imply that sharing is relatively common: 24% of 307 user-created scripts were run by two or more different users, and 5% were run by six or more users. People often run scripts created by others: 465 (78%) of the user population ran scripts they did not create, running 2.3 scripts created by others on average. There is also evidence that users are sharing knowledge of infrequent processes: we found 16 scripts that automate known business processes within our company (*e.g.*, updating emergency contact info), that were run once or twice each by more than ten different users.

In addition to the ability to run and edit others' scripts, CoScripter supports four other collaborative features: editing others' scripts, end-user rating of scripts, tagging of scripts, and free-form comments added to scripts. We found surprisingly little use of these collaborative features: fewer than 10% of the scripts were edited by others, rated, tagged, or commented on. Further research is needed to determine whether and how these features can be made more valuable in a business context.

### **Email Survey of Lapsed Users**

To learn why employees stopped using CoScripter, we sent a short email survey to all the *experimenters* and *past users*, noting their lack of recent use and asking for reasons that CoScripter might not have met their needs. Thirty people replied, and 23 gave one or more reasons related to CoScripter. Of the topics mentioned, ten people described *reliability* problems where CoScripter did not work consistently, or did not handle particular web page features (*e.g.*, popup windows and image buttons); five people said their tasks required *advanced features* not supported in CoScripter (most commonly requested were parameters, iteration, and branching); three people reported problems *coordinating mixed initiative scripts*, where the user and CoScripter alternate actions; and two had *privacy* concerns (*i.e.*, they did not like scripts being public by default). Finally, seven people reported that their jobs did not involve enough tasks suitable for automation using CoScripter.

### **Interview Study 2: CoScripter Usage**

In a second set of interviews, we explored the actual usage of CoScripter by talking with users who had made CoScripter part of their work practices.

#### **Participants**

Based on usage log analysis, we chose people who had used one or more scripts at least 30 times. We also selected a few people who exhibited interesting behavior (*e.g.*, editing other peoples' scripts or sharing a script with others). We contacted 14 people who met these usage criteria; 8 agreed to an interview.

Seven interviewees were active CoScripter users, one had used the tool for five months and stopped 3½ months before the interview. Participants had used CoScripter for an average of roughly 4 months (minimum of 1, maximum of 9). They had discovered the tool either via email from a co-worker or on an internal website promoting experimental tools for early adopters. Seven participants were male, and they worked in 8 sites across 4 countries, with an average of 10 years tenure at the company. We interviewed 4 managers, 1 communications manager, 1 IT specialist, 1 administrative services representative, and 1 technical assistant to a manager / software engineer. Overall, our participants were technology savvy, and five of them had created a macro or scripts before CoScripter. However, only two of them claimed software development expertise.

**Table 1:** Description of routine tasks automated by CoScripter.

User	Scripted Task	Sharing	Frequency Run
JL1	Send a page via an online paging tool.	Private	M-F: 6 runs / day Sa-Su: 3 runs / day
PV1	Login to two technical support voicemail inboxes and check for new messages (2 scripts).	Public	Many runs / day
AM1	Fill in a form to register service packs for clients (form data is the same for each client).	Private	20-30 runs / day
CR1	Change call forwarding and phone status messages (9 scripts).	Public	Several runs / day

### Method

We conducted all but one of our interviews over the phone, due to geographically dispersed participants, using Net-Meeting to view the participant's Firefox browser. Interviews lasted between 30 and 60 minutes and were audio-recorded with participants' permission in all cases but one.

We conducted a semi-structured interview based on questions that were designed to gather data about how participants used CoScripter, why they used it, and what problems they had using the tool. In addition to the predetermined questions, we probed additional topics that arose during the interview, such as script privacy and usability. We also asked participants to run and edit their scripts so we could see how they interacted with CoScripter.

### Results

Interview Study 1 established user needs for automating frequent or hard-to-remember tasks, and sharing how-to knowledge. Study 2 explored how well CoScripter meets these user needs and areas where it falls short.

#### Automating Frequent or Hard-to-Remember Tasks

Four participants described CoScripter as very useful for automating one or more frequent, routine tasks. Each person had different tasks to automate, highlighting various benefits of CoScripter as an automation tool. Table 1 lists the most frequent routine tasks that were automated.

Those four subjects described instances where CoScripter saved them time and reduced their effort. For example, CR1 said, "Two benefits: one, save me time – click on a button and it happens – two, I wouldn't have to worry about remembering what the address is of the messaging system here." PV1 also appreciated reduced effort to check voicemail inboxes, "I set up [CoScripter] to with one click get onto the message center." AM1 used his service pack registration script for a similar reason, saying it was "really attractive not to have to [enter the details] for every single service pack I had to register." JL1 runs his script many times during non-business hours. He would not be able to do this without some script (unless, as he said, "I didn't want to sleep").

These participants, none of whom have significant software development experience, also demonstrate an important

benefit of CoScripter: it *lowered the barrier* for automation, since it required no programming skills.

Though most participants interacted with CoScripter via the sidebar or the wiki, JL1 and PV1 invoked CoScripter in unexpected ways. JL1 used the Windows Task Scheduler to automatically run his script periodically in the background. Thus, after creating his script, JL1 had no contact with the CoScripter user interface. PV1 created a bookmark to automatically run each of his two scripts, and added them to his Firefox Bookmarks Toolbar. To run the scripts, he simply clicked on the bookmarks – within a few seconds he could check both voicemail inboxes.

In addition to automating frequent tasks, CoScripter can act as a memory aid for hard-to-remember tasks. For example, DG1 created scripts for two new processes he had to do for his job. Both scripts used an online tool to create and send reports related to customer care. Creating the report involved a number of complicated steps, and CoScripter made them easier to remember:

*"[CoScripter] meant I didn't have to remember each step. There were probably six or seven distinct steps when you have to choose drop-downs or check-boxes. It meant I didn't have to remember what to do on each step. Another benefit is I had to choose eight different checkboxes from a list of about forty. To always scan the list and find those eight that I need was a big pain, whereas [CoScripter] finds them in a second. It was very useful for that."*

After using CoScripter to execute these scripts repeatedly using the step-by-step play mode (at least 28 times for one script, 9 times for the other) for five months, DG1 stopped using them. He explained: "It got to the point that I memorized the script, so I stopped using it." This highlights an interesting use case for CoScripter: helping a user memorize a process that they want to eventually perform on their own.

Participant LH1 found CoScripter useful in supporting hard-to-remember tasks, by searching the wiki and finding scripts others had already recorded: "I found the voicemail one, it's really useful because I received a long email instruction how to check my voicemail. It was too long so I didn't read it and after a while I had several voicemails and then I found the [CoScripter] script. It's really useful."

**Automation Limitations and Issues.** Despite generally positive feedback, participants cited two main issues that limited their use of CoScripter as an automation tool, both of which were named by lapsed users who were surveyed by email: *reliability problems* and a need for more *advanced features*.

Four out of eight participants noted experiencing problems with the reliability and robustness of CoScripter's automation capability. All of these participants reported problems related to CoScripter misinterpreting instructions. Misinterpretation was a result of CoScripter's human-readable

scripting system, which does not enforce any particular syntax, but instead does best-effort interpretation of arbitrary text. For example, one user reported running a script that incorrectly clicked the wrong links and navigated to unpredictable pages (e.g., an instruction to “click the Health link,” when interpreted on a page that does not have a link labeled “Health,” would instead click a random link on the page. This results from CoScripter’s sloppy programming approach as described in [11]). Another user reported that his script could not find a textbox named in one of the instructions and paused execution. Misinterpretation problems were exacerbated by the dynamic and unpredictable nature of the web, where links and interface elements could be changed or removed.

Five users reported wanting more advanced programming language features, to properly automate their tasks. In particular, participants expressed a need for automatic script restart after mixed-initiative user input, iteration, script debugging help, and conditionals.

#### **Sharing How-To Knowledge**

Our interviews revealed some of the different ways CoScripter has been used to share how-to knowledge. These included teaching tasks to others, promoting information sources and teaching people how to access them, and learning how to use CoScripter itself by examining or repurposing other peoples’ scripts.

Participants used CoScripter to teach other people how to complete tasks, but in very different ways. The first person, LH1, recorded scripts to teach her manager how to do several tasks (e.g., creating a blog). LH1 then emailed her manager a link to the CoScripter script, and the manager used CoScripter to complete the task. The second person, DG1, managed support representatives and frequently sent them how-to information about various topics, from generating reports online to using an online vacation planner. DG1 used CoScripter to record how to do the tasks, and then copied and pasted CoScripter’s textual step-by-step instructions into emails to his colleagues:

*“When I want to give people instructions on how to do something on the Internet, it was great for doing that so I didn’t have to write them out. I found it very useful for that, creating clear instructions on what to do. I sent the text, but I never sent the link so that they could just run it. I never actually sent a script to someone because I didn’t know anyone who used the tool. I could have asked people to install it, but I never did. I tend to work with others who are afraid of new technology. I used that a number of times. For that it did exactly what I needed.”*

DG1’s use of CoScripter highlights a benefit of its human-readable approach to scripting. The scripts recorded by CoScripter are clearly readable, since DG1’s colleagues were able to use them as textual instructions.

Another participant, MW1, a communications manager, used CoScripter to promote information that is relevant to employees in a large department and to teach them how to access that information. He created one script for adding a news feed to employees’ custom intranet profile and a second script to make that feed accessible from their intranet homepage. He promoted these scripts to 7000 employees by posting a notice on a departmental homepage and he plans to send links to the scripts in an upcoming email. Before using CoScripter, MW1 said he did not have a good method for sharing this how-to information with a wide audience. With such a large audience, however, correctly generalizing the script was a challenge; MW1 said he spent two to three hours getting the scripts to work for others. Still, MW1 was so pleased that he evangelized the tool to one of his colleagues, who has since used CoScripter to share similar how-to knowledge with a department of 5000 employees.

Participants used CoScripter for a third sharing purpose: three people talked about using other peoples’ scripts to learn how to create their own scripts. For example, PV1 told us that his first bug-free script was created by duplicating and then editing someone else’s script.

**Sharing Limitations and Issues.** Though some participants found CoScripter a valuable tool for sharing, these participants noted limitations of *critical mass* and others raised issues about the *sharing model*, *generalizability*, and *privacy*, which were barriers to sharing.

Sharing has been limited by the narrow user base (so far) within the enterprise. As CR1 said, critical mass affected his sharing (“I haven’t shared [this script] with anyone else. But there are other people I would share it with if they were CoScripter users.”), and his learning (“I think I would get a lot more value out of it if other people were using it. More sharing, more ideas. Yet most of the people I work with are not early adopters. These people wouldn’t recognize the difference between Firefox and Internet Explorer.”). DG1, who sent CoScripter-generated instructions to his coworkers via email, could have more easily shared the scripts if his coworkers had been CoScripter users.

We also saw problems when users misunderstood the wiki-style sharing model. For example, PV1 edited and modified another person’s script to use a different online tool. He did not realize that his edits would affect the other person’s script until later: “I started out by editing someone else’s script and messing them up. So I had to modify them so they were back to what they were before they were messed up, and then I made copies.” A second participant, PC1, had deleted all the contents of a script and did not realize this until the script was discussed in the interview: “That was an accident. I didn’t know that [I deleted it]... When I look at those scripts, I don’t realize that they are public and that I can blow them away. They come up [on the sidebar] and they look like examples.”

Easy and effective ways to generalize scripts so that many people can use them is essential to sharing. Though users are able to generalize CoScripter scripts, participants told us it is not yet an easy process. For example, MW1 created a script for 7000 people, but spent a few hours getting it to work correctly for others. Also, while CoScripter's personal database feature allows users to write scripts that substitute user-specified values at runtime, not all processes can be generalized using this mechanism. For example, PV1 copied another person's script for logging into voicemail in one country and modified it to login to voicemail in his country, since each country uses a different web application.

Finally, sharing was further limited when participants' were concerned about revealing private information. Three participants created private scripts to protect private information they had embedded in the scripts. AC1 said: "There is some private information here – my team liaison's telephone number, things like that. I don't know, I just felt like making it private. It's not really private, but I just didn't feel like making it public." Others used the personal database variables to store private information and made their scripts public. This privacy mechanism was important to them: "Without the personal variables, I would not be able to use the product [CoScripter]. I have all this confidential information like PIN numbers. It wouldn't be very wise to put them in the scripts." However, one participant was wary of the personal variables, "The issue I have with that is that I don't know where that is stored... If I knew the data was encrypted, yeah."

### Summary

These findings show that, while not perfect, CoScripter is beginning to overcome some of the barriers to sharing procedural knowledge uncovered in Study 1. First, CoScripter provides a single public repository of procedural knowledge that our interviewees used (*e.g.*, several participants used scripts created by other people). Second, CoScripter eliminates the tedious task of writing instructions (*e.g.*, DG1 used it to create textual instructions for non-CoScripter users). Third, CoScripter provides mechanisms to generalize instructions to a broad audience so that experts can record their knowledge once for use by many learners (*e.g.*, MW1 generalized his scripts to work for 7000 employees).

### GENERAL ISSUES AND FUTURE WORK

By helping users automate and share how-to knowledge in a company, CoScripter is a good starting point for supporting the issues uncovered in Study 1. User feedback, however, highlights several opportunities for improvement. While some of the feedback pointed out usability flaws in our particular implementation, a significant number of comments addressed more general issues related to PBD-based knowledge sharing systems. Participants raised two issues that highlight general automation challenges – reliability challenges and the need for advanced features – and four collaboration issues that will need to be addressed by

any knowledge sharing system – the sharing model, script generalization, privacy, and critical mass.

One of the most common complaints concerned the need for improved *reliability and robustness* of the system's automation capability. These errors affected both users relying on the system to automate repetitive tasks, and those relying on the system to teach them how to complete infrequent tasks. Without correct and consistent execution, users fail to gain trust in the system and adoption is limited.

Users also reported wanting more *advanced programming language features*, such as iteration, conditionals, and script debugging help, to properly automate their tasks. These requests illustrate a tradeoff between simplicity – allowing novice users to learn the system easily – and a more complex set of features to support the needs of advanced users. For example, running a script that has iterations or conditionals might be akin to using a debugger, which requires significant programming expertise. A challenge for CoScripter or any similar system will be to support the needs of advanced users while enabling simple script creation for the broader user base.

Our studies also raise several collaboration issues that must be addressed by any PBD-based knowledge sharing tool: *the sharing model, privacy, script generalization, and critical mass*. The wiki-style sharing model was confusing to some users. Users should be able to easily tell who can see a script and who will be affected by their edits, especially given the common base of processes being performed by many employees. A more understandable sharing model could also help address privacy concerns, as would a more fine-grained access control mechanism that enabled users to share scripts with an explicit list of authorized persons. Finer-grained privacy controls might encourage more users to share scripts with those who have a business need to view them. For generalization, we learned that personal database variables were a good start, but we are uncertain as to what degree this solution appropriately supports users with no familiarity of variables and other programming concepts. One way to better support generalization and personalization could be to enable users to record different versions of a script for use in different contexts, and automatically redirect potential users to the version targeted for their particular situation. Finally, a small user base limited further use of the system. We hope that solving all the issues above will lower the barriers to adoption and improve our chances of reaching critical mass.

Finally, one important area for future work is to study the use of CoScripter outside the enterprise. While we conducted the studies in a very large organization with diverse employees and we believe their tasks are representative of knowledge workers as a whole, the results we have obtained may not be generalized to users of CoScripter outside the enterprise. CoScripter was made available to the public in August 2007 (see <http://services.alphaworks.ibm.com>).

com/coscripiter/), and more research on its use is needed to examine automation and sharing practices in this larger setting.

## CONCLUSIONS

Sharing and automation of how-to knowledge is an important need, particularly for enterprise workers who interact with corporate processes on a daily basis. In this paper, we reported on an initial user study that established user needs for automating frequent and hard-to-remember processes, and for sharing how-to knowledge with others. We presented CoScripter, a web-based tool for capturing, playing back, and sharing scripts to accomplish web-based tasks. Based on log data analysis and interviews with active users, we found that people have successfully used CoScripter to automate frequent and hard-to-remember tasks, and to share how-to knowledge with others. Moreover, our analysis reveals that a growing number of users are incorporating CoScripter into their work practices. By lowering the barrier to automation, CoScripter enables users to save time and effort capturing and reusing procedural knowledge, and to more easily share this knowledge with others.

Our email survey of lapsed users and interviews of active users also revealed several issues that are general to PBD-based knowledge sharing systems, pointing toward interesting future research. First, we learned that script reliability is critical. One future effort will be to achieve better reliability while preserving CoScripter's natural-language representation, which can be challenging in the ever-changing Web. Second, it is important to provide expert users with more advanced scripting features while maintaining a simple recording function and scripting language for novice users. Our studies also reveal outstanding issues for any tool that supports how-to automation and sharing: the sharing model needs to be apparent, better mechanisms for generalizing scripts are needed, privacy needs to be protected, and critical mass will need to be reached for knowledge sharing to reach its full potential. Future work will involve research, design, and development to address these issues and create solid automation and sharing tools for a lively community both within and outside the enterprise.

## ACKNOWLEDGMENTS

We thank the CoScripter development team for their support and all our interviewees for their time. Thanks to Jeremy Birnholtz, Eser Kandogan, Saeko Nomura and our anonymous reviewers for their insightful comments on earlier revisions of this paper.

## REFERENCES

1. Agostini, A., Albolino, S., De Michelis, G., De Paoli, F., and Dondi, R. Stimulating knowledge discovery and sharing. *Proceedings of GROUP'03*.
2. Alavi, M., & Leidner, D.E. (2001). Review: Knowledge management & knowledge management systems: Conceptual foundations & research issues. *MIS Quarterly*, 25(1), 107-137.
3. Anderson, C. (2006). *The long tail: why the future of business is selling less of more*. New York: Hyperion.
4. Birnholtz, J.P., & Bietz, M.J. Data at work: supporting sharing in science and engineering. *Proceedings of GROUP'03*.
5. Brown, J.S., & Duguid, P. (2001). Knowledge and organization: a social-practice perspective. *Organization Science*, 12(2), 198-213.
6. Cyper, A. (1993). *Watch What I Do: Programming by Demonstration*. Cambridge: MIT Press.
7. Dorn, B. and Guzdial, M. Graphic designers who program as informal computer science learners. *Proceedings of ICER'06*.
8. Edwards, D.L., & Mahling, D.E. Toward knowledge management systems in the legal domain. *Proceedings of GROUP'97*.
9. Halverson, C.A., Erickson, T., and Ackerman, M.S. Behind the helpdesk: evolution of knowledge management system in a large organization. *Proceedings of CSCW'04*.
10. Hoffmann, M., Loser, K., Walter, T., & Herrmann, T. A design process for embedding knowledge management in everyday work. *Proceedings of GROUP'99*.
11. Little, G., Lau, T.A., Cypher, A., Lin, J., Haber, E.M., & Kandogan, E. Koala: capture, share, automate, personalize business processes on the web. *Proceedings of CHI'07*.
12. Little, G., & Miller, R.C. Translating Keyword Commands into Executable Code. *Proceedings of UIST'06*.
13. Lochovsky, F.H., Woo, C.C., & Williams, L.J. A micro-organizational model for supporting knowledge migration. *Proceedings of SIGOIS'90*.
14. Millen, D.R., & Fontaine, M.A. Improving individual and organizational performance through communities of practice. *Proceedings of GROUP'03*.
15. Nahapiet, J., & Ghoshal, S. (1998). Social capital, intellectual capital, and the organizational advantage. *The Academy of Management Review*, 23(2), 242-266.
16. Nardi, B.A. (1993). *A small matter of programming: perspectives on end-user computing*. Cambridge: MIT Press.
17. Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1), 14-37.
18. Nonaka, I., and Takeuchi, H. (1995). *The knowledge-creating company: how Japanese companies create the dynamics of innovation*. New York: Oxford University Press.
19. Rosson, M.B., Ballin, J., & Rode, J. Who, what, and how: A survey of informal and professional web developers. *Proceedings of VL/HCC'05*.
20. Stewart, T.A. (1997). *Intellectual capital: the new wealth of organizations*. New York: Doubleday / Currency.
21. Suchman, L. (1983). Office procedure as practical action: Models of work and system design. *ACM Transactions on Office Information Systems*, 1(4), 320-328.
22. Szulanski, G. (1996). Exploring internal stickiness: impediments to the transfer of best practice within the firm. *Strategic Management Journal*, 17, 27-43.
23. Yamauchi, Y., Whalen, J., & Bobrow, D.G. Information use of service technicians in difficult cases. *Proceedings of CHI'03*.