IEEE pervasive COMPUTING
MOBILE AND UBIQUITOUS SYSTEMS
www.computer.org/pervasive

Using Mobile Phones for Secure, Distributed Document Processing in the Developing World

Tapan S. Parikh

Vol. 4, No. 2
April–June 2005

Using Mobile Phones for Secure, Distributed Document Processing in the Developing World

Paper plays an essential role in many information ecologies in the developing world, but it can be inefficient and inflexible. The CAM document-processing framework exploits smart mobile phones' utility, usability, and growing ubiquity to link paper with modern information tools.

Tapan S. Parikh
University of Washington

We've recently seen growing interest in the *digital divide*—the division between people who use modern digital tools and information services for personal or professional purposes and those who don't. This has given rise to the optimistic notion that if digitally disenfranchised people in rural areas of the developing world can adopt information technologies in a sustainable way, they can achieve many development objectives. Termed *Information and Communication Technologies (ICT) for Development*, this vision carries a broad mandate—allowing billions of people access to services as important and varied as health care, education, and financial and governmental services. While initial implementations have far to go before they claim any definitive successes, this idea has great intellectual and moral appeal. Our project team has worked on several projects that sought to make this vision of low-cost rural information services a reality.

Even as currently posed, rural and pervasive computing share many infrastructural, interface, and application-level themes. The basic problem is to introduce computing seamlessly into a physical, natural world populated by nonexpert users who can't, don't, or would rather not spend their time in front of computers. These users aren't knowledge workers, don't use modern information technology as part of their daily lives, and would like maximal utility from computing devices with minimal engagement. At a systems level, both rural and pervasive computing face environmental constraints in power, network connectivity, and cost-bearing capabilities. Moreover, applications in both domains must build value chains around technological innovations that can have a real, beneficial impact on people's lives.

The smart phone is our pervasive information appliance of choice. Its communication facilities offer immediate utility to any user, urban or rural, rich or poor, which lubricates its introduction to any context. The numeric keypad's familiarity and long history make it comfortable for billions of users. Solid state memory, extended battery life, and a compact, rugged form are all great design choices for the village environment.

The CAM framework

We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/05/$20.00 © 2005 IEEE

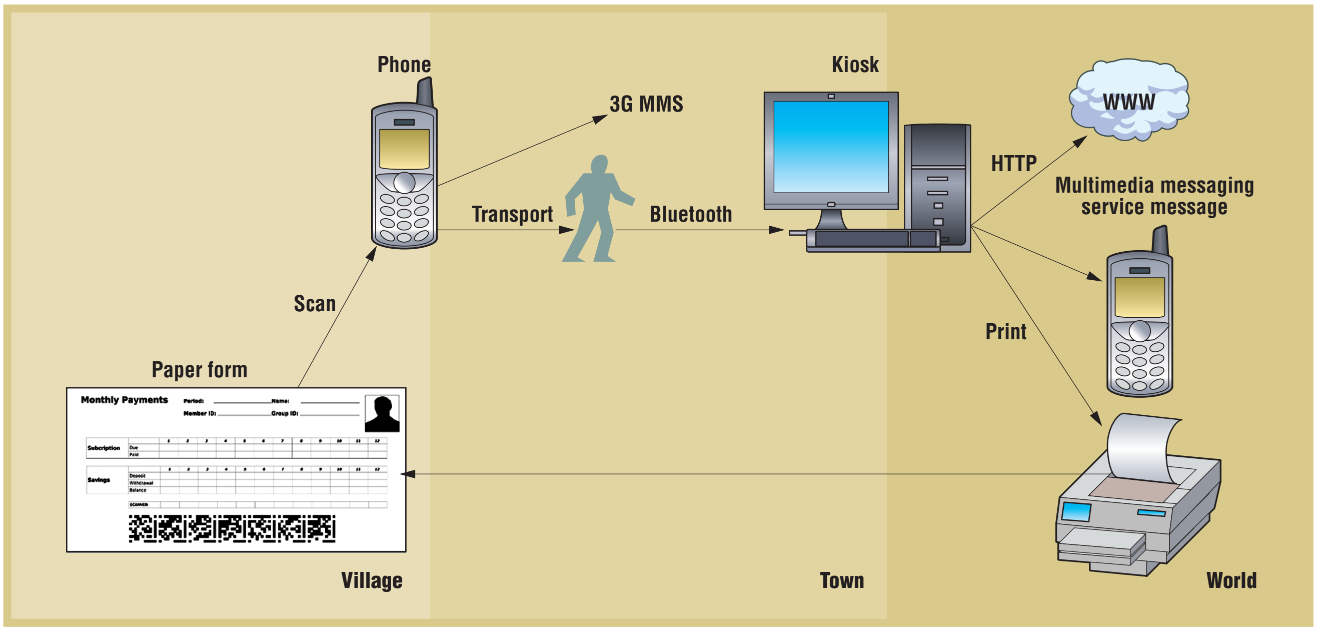


Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure.

The CAM framework comprises four components. Users initially interact with *CamForms*, which are augmented paper applications. The *CamShell* scripting language is used to embed processing instructions in these forms, which the *CamBrowser* mobile phone application scans and processes. CamBrowser connects to a local or remote *CamServer*, which receives and organizes CamForm submissions and supports additional data entry and document processing.

CamForm applications

In designing CAM services, we found it convenient to refer to sets of augmented paper documents as CamForm applications. Each document can represent a distinct query or data entry action, one step in an action sequence (as in the case of a multipage form), or a previous action's result. Taken together, CamForms give a view of the application's current state and provide various interaction options with a distant server, not unlike a traditional client.

Paper has several advantages. Having daily lives filled with physical tasks, rural users are much more comfortable with artifacts they can handle and touch. Paper affords a physical representation of information that can be communally viewed, edited, and stored. In some cases, illiterate users have even learned to interpret data from paper forms. For the time being, it's unlikely that any affordable digital interface will be suitable for this kind of use.

However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information.

CamShell

We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document.

We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've

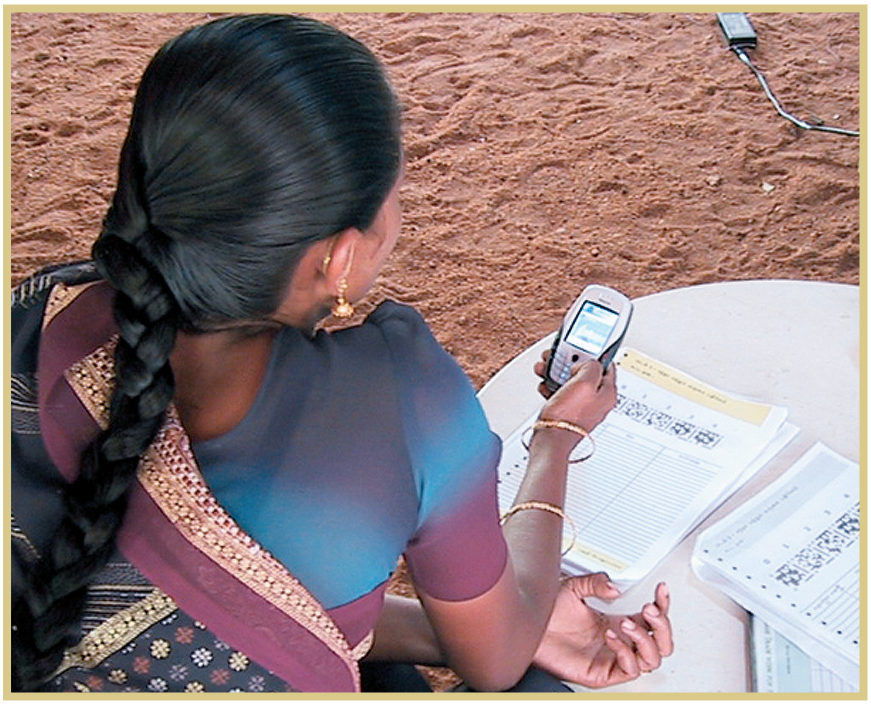


Figure 2. A prospective user entering CamForm data using the mobile phone's alphanumeric keypad.

designed a compact but powerful programming language that's sufficient for many uses.

Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. The sequence number is useful for ensuring that the phone executes each of the instructions in the proper order and completes all instructions before the program's end. The main instruction classes are

- assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents.
- dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user.
- control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table.
- application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message.
- multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired.
- networking and security instructions, which provide for secure communications with CamServer.

CamShell programs are composable—that is, if you have a multipage form or different sections of a program on separate documents, you can scan them in sequence or overlay them as long as the sequence numbers match up. This lets you compose several physical artifacts and documents together for a single request.

CamBrowser

The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop.

We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream servers.

The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content.

In the case of a URL, the phone can operate in one of three network modes:

Integrating Digital Media into Paper-Based Information Ecologies

During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology.

Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility.

Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world.

REFERENCES

1. T. Parikh et al., "Design Studies for a Financial Management System for Micro-Credit Groups in Rural India," *Proc. 2003 ACM Conf. Universal Usability* (CUU 03), ACM Press, 2003, pp. 15–22.

2. A. Sellen and R. Harper, *The Myth of the Paperless Office,* 1st ed., MIT Press, 2002.

3. W. Mackay, "Is Paper Safer? The Role of Paper Flight Strips in Air Traffic Control," *ACM Trans. Computer-Human Interaction*, vol. 6, no. 4, 2000, pp. 311–340.

4. W. Johnson et al., "Bridging the Paper and Electronic Worlds: The Paper User Interface," *Proc. SIGCHI Conf. Human Factors in Computing Systems*, ACM Press, 1993, pp. 507–512.

5. F. Guimbretière, "Paper Augmented Digital Documents," *Proc. 16th Ann. ACM Symp. User Interface Software and Technology* (UIST 03), ACM Press, 2003, pp. 51–60.

6. "IFP Success Stories," IBM, www2.clearlake.ibm.com/GOV/ifp/success_stories.html.

7. J. Rekimoto and Y. Ayatsuka, "Cybercode: Designing Augmented Reality Environments with Visual Tags," *Proc. Designing Augmented Reality Environments* (DARE 2000), ACM Press, 2000, pp. 1–10.

8. D.L. Hecht, "Printed Embedded Data Graphical User Interfaces," *Computer*, vol. 34, no. 3, 2001, pp. 47–55.

9. M. Rohs and B. Gfeller, "Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects," *Advances in Pervasive Computing,* A. Ferscha, H. Hoertner, and G. Kotsis, eds., Austrian Computing Soc. (OCG), 2004, pp. 265–271.

- online, in which the phone executes HTTP requests over an Internet connection.
- offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection.
- proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests.

These network modes let users choose where and how to incur communications costs. If they live in an area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location.

Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input.

CamServer

The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing.

We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or recognition tasks

Figure 3. Self-help group members reviewing notes in rural Tamil Nadu, India.

to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry.

Security

The CAM architecture's distributed nature makes security an important concern. CamShell provides instructions for performing encryption, digital signing, or both (*signcryption*). These instructions include digital keys and can be embedded either directly in CamForms or within special security tokens, called *CamTokens*. When the CamBrowser processes these instructions, the software automatically signs and/or encrypts the collected data before transmission. CAM application developers can use these instructions to create secure, signed communications channels between a CamForm client and the CamServer.

Usability concerns in maintaining system security have recently received attention in the literature.[2] Remembered tokens such as passwords and PINs have come under fire as being inflexible and insecure. Conversely, humans have a long history of trust practices built around exchanging physical tokens (for example, money). Physical keys as implemented in CamShell might be more usable in rural areas because

- they carry more information. Most rural users are unlikely to remember long or unfamiliar combinations of letters, and their remembered passwords would be vulnerable to cracks. Physical tokens can carry much more data and would be easier to replace if compromised.
- users can enter them securely. Rural users are unlikely to have the capacity or unfettered access to enter a conventional password or PIN without others observing it.
- users can explicitly share and distribute them. Physical tokens can have an arbitrary nonzero duplication cost, so users can distribute, lend, or share them with fixed risk.

The notion of uniting physical and information security isn't novel. Magnetic stripe cards, smart cards, and radiofrequency identification tags (RFIDs) all take this approach. CamTokens stake a middle ground between these alternatives and the remembered token schemes popular on the Internet. While they're not as hard to duplicate as other physical approaches, CamTokens would require an explicit and protracted action to forge or steal in a village. In rural areas, a printer or photocopier might be hours to days away.

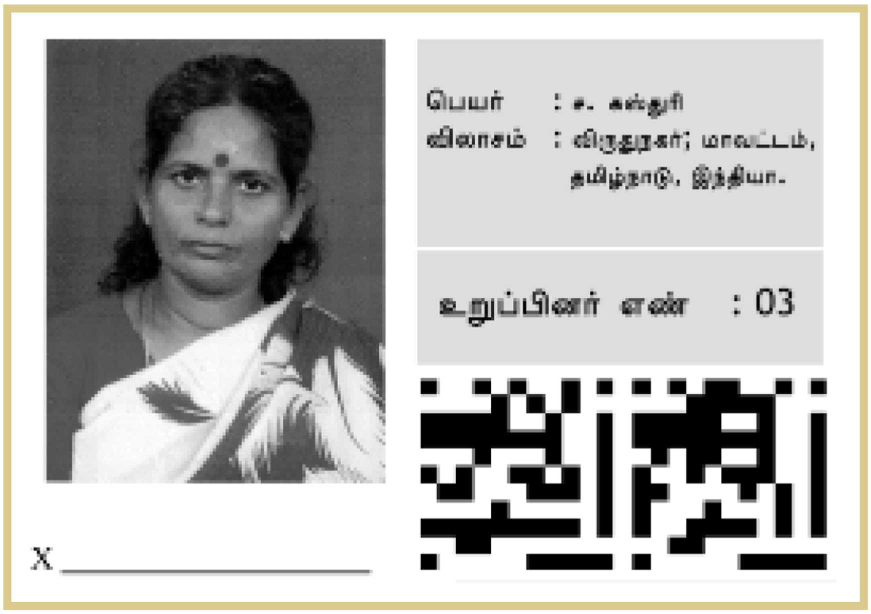
At the same time, CamTokens' production cost isn't out of reach for the community. This lets the community create and issue CamTokens, in contrast to smart cards and RFIDs, which require external manufacture and management. Anne Adams and Martina Angela Sasse have shown that most secure information isn't kept private, but is shared within a circle of people who have some temporal trust relationship.[3] This is even more true in the family-based but tumultuous environment of a rural village. With CamTokens, village users can easily experiment with different security schemes.

We're working on encoding public, private, and shared one-time keys using CamTokens. The CAM framework uses these to create PGP-like secure communication channels between independent parties or to provide one-time secure channels between a user and a service provider. Because we can exchange security tokens physically through readily available "back-channels," we avoid using a centralized public-key infrastructure that plagues current public-key implementations.

Trust

Trust was another important consideration in the CAM user interface's design. In most cases, CamForm users won't have direct access to a mobile phone and will therefore have to rely on local phone-equipped agents to capture and transmit their data. Because of the mobile phone screen and keypad's limited size, it's difficult for these users to ensure that the information entered in the CamBrowser matches the values they've provided on the form. We're experimenting with several ways to address this difficult issue.

First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content.



பெயர் : ச. கஸ்தூரி
விலாசம் : விருதுநகர்; மாவட்டம்,
தமிழ்நாடு, இந்தியா.

உறுப்பினர் எண் : 03

X

Figure 4. A CamID used in the SHG-Checkbook application.

Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered.

Because we've found audio to be one of the most important factors driving CAM interaction, we expect the last method to most effectively and immediately reassure users that their data has been entered and transmitted correctly. However, system implementers can choose any of the methods outlined above to ensure user trust in their application.

Self-help group applications

Over the past few years, we've been designing and implementing information systems for self-help groups. SHGs are semiformal, community-owned financial cooperatives (see Figure 3). In India, they usually include 15 to 30 members, almost always women. Each member contributes a fixed amount of money at regularly scheduled monthly or weekly meetings. The group lends this community capital to members for education, health care, buying agricultural inputs, starting a business, and so on.

In India, once these SHGs have achieved maturity and accumulated a certain amount of capital, they're eligible for loans from commercial banks. Under a National Bank for Agriculture and Rural Development program, loans to SHGs are refinanced at preferred interest rates, creating a viable business opportunity for banks working in rural India.

We've created two CAM applications that we believe will help SHGs achieve independence and economy of scale, greatly expanding the reach of this fledgling community revolution.

SHG-Notebook

Local nonprofit nongovernmental organizations (NGOs) most commonly form and train SHGs. Government agencies, banks, international NGOs, and for-profit companies might also play a role. These organizations assume that although organizing and training SHGs has significant costs, the groups will eventually become independent, no longer requiring regular technical assistance.

However, SHGs have many detailed information management tasks. They must maintain accurate internal records, recording transactions and current balances for group members. Moreover, they must use this information effectively to collect outstanding payments and guard against fraud.

SHGs keep their records in physical notebooks and ledgers, which requires significant rigor in bookkeeping methods. Most SHG members in India are poor rural artisans, laborers, or farmers unable to manage these records' complexity. So, they permanently rely on local literate people and NGO staff to maintain their records and mediate disputes. NGOs must often vouch for the groups' records when approaching a bank for a loan. This dependence on NGOs limits the groups' independence and self-sustainability.

We're working on *SHG-Notebook* as a way for SHGs to digitally transcribe their records for bulk processing. SHG-Notebook captures the minimal amount of information required for SHGs' accounting and loan tracking. SHG members can take the SHG-Notebook to a kiosk, where a local phone-equipped agent can scan the records with the CamBrowser and upload them to the server. The group can request monthly statements and financial reports from the agent for a small fee. This creates a commercial service-based model for SHG data management.

Some organizations have attempted to capture data from SHG group meetings using PDAs.[4] Our system offers advantages in terms of the paper user interface's usability, the use of a mobile phone camera as the data-collection device, and the proposed service architecture's scalability and sustainability.

SHG-Checkbook

The high cost associated with money transport is another significant overhead in interacting with SHGs and other microfinance clients around the world. After microfinance group meetings, clients or NGO staff often must travel to and from bank branches with large amounts of cash to make group deposits and withdrawals. As microfinance groups usually meet according to a regular schedule, thieves can easily predict when such individuals might be traveling through a particular area.

In one case, the manager of a large

microfinance organization told us how one of his staff was murdered during such a robbery. In another case, an NGO had to equip its officers with private cars because public transit was considered unsafe. Moreover, while funds are in transit and financially idle, someone—in most cases, SHG members—must pay for the interest that should be accruing on that money.

Transacting in cash also increases the potential for fraud by SHG members and NGO staff. NGOs and other microfinance service providers usually can't offer their clients flexible savings accounts because it would be too difficult to track how much money should be collected each day. This leaves the door wide open for fraud that could take weeks to track down. Several of the organizations we've worked with have caught staff members underrepresenting client payments.

We're working on *SHG-Checkbook* as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee.

When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request.

SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity.

Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

Usability testing

We conducted three days of usability testing and design workshops with SHG members, community leaders, and supporting staff in Pulvoikarai, Tamil Nadu, in mid-January 2005. Although a detailed analysis of this data is pending, our initial impressions are remarkably positive.

The users we tested had never used a camera or a PC. Most had never used a mobile phone, although they'd seen others using one. We found that most of these users were using the CamBrowser with ease in five to 15 minutes. This was a remarkable improvement in the learning times we'd noted in our earlier research with a PC-based interface.[5] The accessible locus of interaction afforded by paper CamForms and the guided interaction driven by the audio-enabled CamBrowser made introducing the system almost seamless. Out of the 14 users with whom we conducted detailed usability tests, all found the system either easy or very easy to use.

We also identified some potential usability concerns. Users weren't comfortable with the phone's small buttons and stylized design. Process breakdowns that moved the focus away from the camera-driven CamBrowser interface took great effort to resolve, often requiring external intervention. We're exploring ways to address these issues, possibly through new hardware or software prototypes that are more specialized for CAM-based interaction.

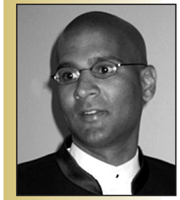
Pilot implementations

We're planning the first pilot implementations of SHG-Notebook and SHG-Checkbook for summer 2005. We'll conduct them with help from our partners ekgaon technologies, the Covenant Centre for Development, and the Mahakalasm SHG Federations. We've begun discussions with the local bank for linking these applications with its information systems and internal processes.

In the meantime, we've been working with rural communities, companies, and social-service organizations to identify more CAM applications. We're targeting a few application domains, including rural distribution tracking, agricultural scoring, government services, and personal communications. We expect to do at least one more pilot in one of these application areas by summer 2005.

Our experiences designing, implementing, and deploying information systems in rural India have shown that the CAM framework offers significant usability, accessibility, and scalability advantages over traditional information service architectures. Our system facilitates the management of large amounts of information across a distributed, infrastructure-poor landscape dotted with novice users. This is similar to many existing systems—even to the Internet as

the AUTHOR



Tapan S. Parikh is a PhD candidate in computer science at the University of Washington. His research is focused on the design, development, and implementation of accessible information services increasing economic opportunity and potential for expression in disconnected rural communities around the world. Earlier projects included the Hisaab project, which investigated user interface design for semiliterate rural users, and the Knownet-Grin project, a grassroots information network linking rural innovators and entrepreneurs. Tapan received his MS in computer science from the University of Washington. Contact him at Univ. of Washington, Dept. of Computer Science, Box 352350, Seattle, WA 98195-2350; tapan@cs.washington.edu.

a whole. As we've seen in the developed world, if this service is made available in an affordable, accessible way, there are almost unimaginable new markets just waiting to be created. We look forward to helping create them. P

ACKNOWLEDGMENTS

Several colleagues have provided valuable feedback and technical advice during the course of this project. I would specifically like to thank James Landay, Scott Klemmer, Ed Lazowska, David Notkin, Gaetano Borriello, Richard Anderson, and Ken Fishkin for their contributions. Kaushik Ghosh and Apala Lahiri Chavan of Human Factors India worked on earlier iterations of this research and are contributing to the CAM usability testing. ekgaon technologies and the Covenant Centre for Development in Madurai, India, have provided translations and are working on the pilot implementations. Last but certainly not least, I thank the women of the Mahakalasm SHG Federations for their continued partnership and patience.

REFERENCES

1. M. Rohs and B. Gfeller, "Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects," *Advances in Pervasive Computing*, A. Ferscha, H. Hoertner, and G. Kotsis, eds., Austrian Computing Soc. (OCG), 2004, pp. 265–271.
2. A. Whitten and J. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Proc. 8th USENIX Security Symp.*, USENIX, 1999, pp. 169–184.
3. A. Adams and M. Sasse, "Users Are Not the Enemy: Why Users Compromise Security Systems and How to Take Remedial Measures," *Comm. ACM*, vol. 42, no. 12, 1999, pp. 40–46.
4. C. Waterfield, "CGAP IT Innovation Series: Personal Digital Assistants (PDA)," Consultative Group to Aid the Poorest, www.cgap.org/docs/IT_pda.html.
5. T. Parikh et al., "Design Studies for a Financial Management System for Micro-Credit Groups in Rural India," *Proc. 2003 ACM Conf. Universal Usability* (CUU 03), ACM Press, 2003, pp. 15–22.