Changing the corporate IT development model: Tapping the power of grassroots computing

L. Cherbakov
A. Bravery
B. D. Goodman
A. Pandya
J. Baggett

The recent rise of grassroots computing among both professional programmers and knowledge workers highlights an alternative approach to software development in the enterprise: Situational applications are created rapidly by teams or individuals who best understand the business need, but without the overhead and formality of traditional information technology (IT) methods. Corporate IT will be increasingly challenged to facilitate the development, integration, and management of both situational and enterprise applications. In this paper, we describe the emerging prevalence of situational application development and the changing role of IT. We also describe the experience at IBM in building, deploying, and managing the IBM Situational Applications Environment that enables employees to take responsibility for some of their own solutions. Finally, we discuss ways in which the situational application development paradigm may evolve in coming years to benefit enterprises, the demands that it will put on IT departments, and possible ways to address these challenges.

INTRODUCTION

The corporate information technology (IT) approach to solution development has been dominated by concerns for performance, availability, and security. Budget realities have limited corporate-sponsored projects to those with the highest impact, leaving many needs unfilled. Furthermore, many commercial software applications and homegrown IT solutions "... tend to be badly designed, badly made, incomprehensible and obsolete ..."[1] Long development cycles often result in applications that are unable to support evolved business needs. End-user efforts to address these gaps outside of the realm of corporate IT have been viewed, at best, with ambivalence.

The recent rise of Web-based ad hoc computing among both professional programmers and business professionals brings into the spotlight a software-development approach that diverges from tradi-

tional IT methods: Teams or individuals who best understand their business problem rapidly create informal solutions to solve it. Not burdened by the overhead and formality of traditional IT methods, these casual developers focus on fast, good-enough results that can be refined later, if needed. Applications developed in this manner may not be ideal. They may be slow or deliver only a subset of possible functions; yet, they provide immediate relief for a given situation. These *situational applications* (applications written to address particular situations at hand) are often short-lived or perpetually improved. This development approach—in combination with increased software-oriented thinking, the growing popularity of server-side scripting, and new Web technologies such as AJAX (Asynchronous JavaScript** and XML)—is forcing a reevaluation of corporate enterprise software-development models.

This new breed of applications, often developed by nonprofessional programmers in an iterative and collaborative way, shortens the traditional development process of edit, compile, test, and run. Situational applications are seldom developed from scratch; rather, they are assembled from existing building blocks (or *consumables*, as they are referred to here). They are often used by a relatively small number of users (less than 50, according to a 2005 IBM-sponsored market research study on the growth of situational applications and the new market for ad hoc development). Developers expect improved productivity and functionality from their situational applications, and they expect to greatly shorten the time from the identification of a need to using a productive application that fills it. These solutions can potentially solve immediate business challenges in a cost-effective way, capture a part of IT that directly impacts knowledge workers,[2] and address areas that were previously unaffordable or of low priority to the IT department. Application builders also report higher satisfaction with their jobs and a sense of being in control. The previously mentioned IBM-sponsored market research shows that users of situational applications feel that they are of core importance. More than half of situational-application users view them as mission critical and rate them as very important to the success of their everyday activities, their department, and their company. Moreover, this view is shared by the corporate hierarchy all the way up to company executives.

The way workers view their workplace is changing, especially as the new generation, *millennials*,[3] are starting to join the workforce. These new employees have different expectations, skills, and values.[3–5] After all, they are the first generation to grow up with IT as an inseparable part of their environment. Because they are used to customizing and individualizing everything—from phone ring tones to their Facebook**[5] spaces—when they move into a workspace, they translate these experiences into wanting to select their own tools, customize their environment, and take responsibility for automating many necessary activities.

By contrast, IT department managers—who have justifiable concerns with reliability and availability of corporate systems, data privacy, and security and who are faced with decreasing budgets—often tend to be conservative in their adoption of new technologies and agile development methods. As a result, corporate IT is often seen as unable to support the business and can be perceived as a hindrance to rather than an enabler of innovation.[6] During the last 30 years, while languages, platforms, and tools have changed significantly, IT solution-development processes have changed very little.

Understanding and taking advantage of the latest changes in Web computing has the potential of significantly improving the effectiveness of corporate computing. These changes include shifts in both technology and usage patterns, collectively referred to as *Web 2.0*, a term coined by Tim O'Reilly.[7] As we will show, using Web development in enterprise computing has the potential to fundamentally transform the role of the IT department from solution developer to solution enabler,[8] a change that corporate IT must make to remain relevant.

The remainder of this paper is organized as follows. In the section "Emergence of situational applications," we provide the context to recent changes that signify a renewed approach to application development. In the section "IBM Situational Applications Environment," we describe our experience building an environment to support a situational application-based approach (sometimes referred to as *community-based computing)*, the challenges that we faced, and the issues that we addressed during its construction. In the section "Changing role of corporate IT," we examine changes already taking place in the enterprise and others that are likely to

happen. We conclude the paper with a brief summary.

EMERGENCE OF SITUATIONAL APPLICATIONS

Evidence of end-user computing (including performing software engineering and development) goes back as early as the late 1970s,[9] with advances in the last 10 years making it increasingly easier for users to develop their own solutions. IBM-sponsored ad hoc development market research, described later in this section, and that of others[10] has revealed that development of applications by amateur programmers (i.e., employees who are not paid to program) is widespread. In 2006, approximately 12 million professionals identified themselves as programming in the workplace. Contrast that with the fact that there are only an estimated three million professional programmers (i.e., employees who program for a living).[10]

Both professional and casual programmers are engaged in some ad hoc application development characterized by the lack of formal engagement around a solution. They disregard formal requirements-gathering, architectural documents, and design specifications; instead, they focus on addressing immediate needs in the fastest possible way. IBM research shows that between 42 and 68 percent of IT employees and 12 percent of business employees have automated a business function, process, or activity in their department outside of a formal IT development project.

IBM research reveals that ad hoc application development activity tends to extend throughout the company (***Figure 1***). Although spreadsheet-based applications remain a prevailing choice for ad hoc programming, our research shows that Web development is rapidly gaining popularity. In the remainder of this paper, we focus on the subset of ad hoc applications developed using Web technologies and refer to them as *situational applications*.

Paradigm changes in Web development

Typical Web development requires a variety of skills at several layers, from the browser at the front end, to specialized middleware (e.g., IBM WebSphere* Application Server or IBM WebSphere Portal), to back-end database systems where the application-programming skills required are beyond the ability of a nonprogrammer. Languages such as Hop[11] and environments like Marmite[12] and IBM Sash Weblications[13] were introduced to simplify Web development. Continued work on rich client technologies demonstrates that the principles of lightweight development based on simpler Web technologies and skills have a value beyond simply easing application building.[14] Interactions with the Web platform are increasingly more compelling than even rich desktop applications. Web workflows can be collapsed into a single-screen experience, thereby gaining the benefit of desktop applications but with the simplicity and flexibility offered by Web development.

AJAX provides easy access to Web-based data and rich user-interface controls. The combination of AJAX and the REST (Representational State Transfer) architectural style of Web services offers an accessible pallet to assemble highly interactive browser-based applications. The Uniform Resource Identifiers (URIs) used by REST to identify Web resources allow equal accessibility to those resources from browsers, mobile devices, and server applications, and they link from e-mails and bookmarks, making this programming style very appealing to a wide range of Web developers.

The availability of a large number of simple application programming interfaces (APIs) and the enablement of AJAX-style Web components (e.g., Yahoo!** Developer Network[15] design patterns and programmable Web APIs[16]) have contributed to the upsurge in popularity of this development style with both professional and amateur programmers. Even older Web technologies such as JavaScript are reinvigorated.

Many mashups (applications comprised of services and functions remixed to create a new context) embed a map into a Web page, where various actions drive the map to plot objects of interest, such as people, structures, or geographic locations. As far back as the mid-1990s, exposing geographic data to end-user developers was a popular activity.[17] The more recent variety illustrates the tipping point, where AJAX widget components enable rampant reuse. (A *widget* is a third-party item that can be embedded in a Web page.) The mapping mashup has become the prototypical situational application, primarily because of its simplicity and because it is a powerful paradigm for information

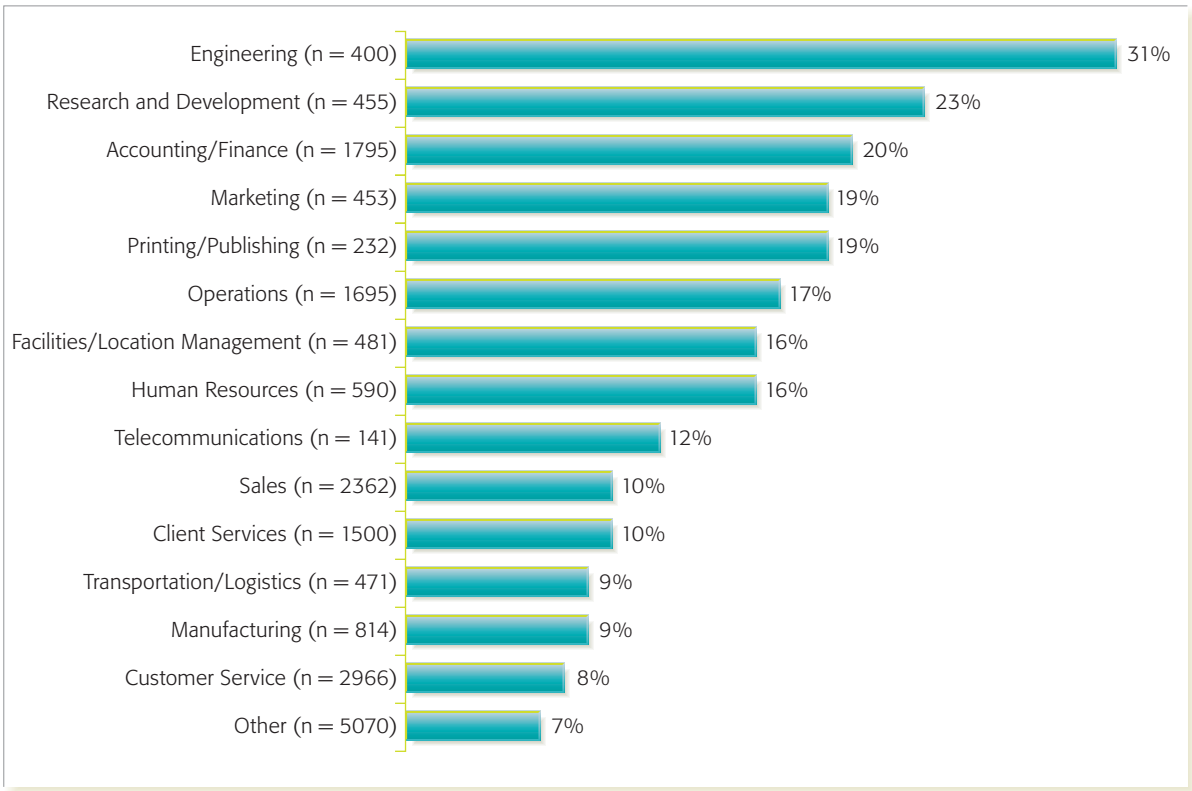

Figure 1
Percentage of employees who conducted ad hoc development during a 12-month period (IBM-sponsored ad hoc development market research)

organization. Combining services through simple interfaces and prebuilt components enables a nonprofessional developer to become an *assembler*—someone who understands the business problem and is comfortable with Web technology, but needs simpler concepts to assemble powerful solutions.

The rise of situational applications cannot be attributed to technological changes alone. Computer literacy is growing and, while the range of skills varies widely, the tooling is evolving to enable more users to build applications. Recent work to make Web development accessible to casual programmers includes assembly-level tooling that enables users to create composite applications out of components, even if they have little technical knowledge of the underlying capabilities. Examples of such platforms include QEDWiki (quick and easily done wiki),[18,19] ADIEU (Ad Hoc Development and Integration Tool for End Users),[20] and more informally, wiki platforms such as SnipSnap,[21] which enable a high degree of extension and customization. As tool design matures, even professionals who are uncomfortable with current Web technologies will be able to participate in their own solution design.

Social software and worker expectations

The introduction of social software (for example, blogs, wikis, activity management, tagging, and bookmarking) is contributing to the proliferation of situational applications. Social software offers data and widget services that enable other applications to offer capabilities that are hosted remotely in a new context. For example, IBM Dogear,[22] an enterprise social bookmarking solution, offers REST-style interfaces to data and provides a user experience that is notably like AJAX, where much of the user experience occurs on the same page without changing contexts. For example, when exploring related content and people, the user is able to toggle between these views without reloading the entire Web page. Third-party applications make use of

data stored in Dogear through the REST-style interfaces, and widget components further externalize reusable user interaction and visual design.

The evolution of tooling, skills, and usage patterns contribute to why community-based computing is seen as a high-potential opportunity. Enabling this development paradigm offers the opportunity to simplify IT to the point where the gatherer of requirements, the solution owner, and the developer are one and the same person, thereby ensuring that the solution delivered meets the immediate business need.

IBM ad hoc development market research

To better understand the market composed of nontraditional programmers performing ad hoc software development, IBM conducted a multiphase primary market research project. The objectives of this research can be summarized as follows:

1. Quantitatively profile the current ad hoc development activities and needs among different audiences.
2. Identify which ad hoc development tools are currently being used and assess the level of user satisfaction with these tools.
3. Gauge the relative market opportunity within those audiences.
4. Understand specific activities, preferences, and related factors in ad hoc development.
5. Determine and compare interest levels in an ad hoc development among the various audiences.

Participants were screened based on two criteria. First, professional developers who spent more than 50 percent of their time on formal application development projects using sophisticated development programming languages and tools, such as C++, C#, Java**, and advanced integrated development environments were screened out. Second, all participants were required to have conducted an ad hoc development activity within the previous 12 months. This activity was defined to participants as occuring when a person automates or facilitates a particular business function, process, or activity by producing a software application that can be described by these characteristics:

- *Often incorporates other existing software*—In addition to any added capability, this new application can modify, enhance, customize, or extend an existing application, or include and combine parts or components from multiple existing applications.
- *Occurs under the radar*—Usually not recognized outside of a department or business unit as a formal project; seldom has a specific project budget or tracked timeline (as do larger, more recognized IT projects); tends to be performed and managed in a relatively unstructured manner.
- *Built for the situation at hand*—Built to solve an immediate, specific business problem, with little concern over whether the application will fit or work in different situations, organizations, environments, or systems, and without features that might allow it to adapt or adjust for more long-lived usage across multiple situations. Could even be thought of as disposible or replaceable.
- *Developed in the most efficient, quick-and-dirty manner possible*—Does not use rigorous and structured steps of formal development methods meant to reduce errors, maximize efficiency and performance, extend the life, or expand usage through future changes.
- *Can be performed by people without extensive, sophisticated computer skills*—Business professionals, analysts, and other IT staff often are engaged in ad hoc development. Requires business knowledge of the task at hand, but not very specialized programming knowledge or extensive IT skills.
- *Developed using tools and components that do not require significant IT knowledge*—Unlike advanced programming tools used to build an application from scratch, ad hoc development employs more basic tools, such as macros, wizards, forms, templates, visual construction, and the like. It usually makes use of preexisting software components, such as spreadsheets, database programs, report generators, or vertical business programs already in use.

A total of 790 Web-based interviews were completed with three separate target audiences:

- *IT (excluding professional programmers)*—250 interviews with IT managers/directors/staff
- *Business partners or solution providers*—250 interviews with solutions providers or partners who have performed ad hoc development activities for customers
- *Line-of-business power users*—290 interviews with non-IT but computer-savvy line-of-business power users

A complete mix of industries and line-of-business functions and departments were surveyed, with each group being large enough to evaluate as a subsegment. Government agencies were excluded.

In addition to those completing the interviews, over 25,000 successful contacts were made across the three audiences in this research. Regardless of ultimate qualification, as long as a respondent had an appropriate job function and reponsibilty, the following incidence information was collected before interview termination:

a. Percent who understand the ad hoc development definition as provided
b. Percent who say that ad hoc development is ever conducted by anyone in their company
c. Percent who say they have conducted ad hoc development personally in the last 12 months

The interview asked responders a series of over 60 questions, grouped into seven categories:

1. Frequency and scope of their ad hoc development activities
2. Specific application and activity areas in which they have conducted, or plan to conduct, ad hoc development
3. Types of people, including themselves, who are involved in the ad hoc application development in their organization, and their various roles
4. Business value, top business benefits, and the reasons that drive the decision to conduct ad hoc development
5. Level of encouragement or discouragement they receive from other parties (e.g., team leaders, IT, and clients) in terms of conducting ad hoc activities, and types of benefits or barriers encountered
6. Perceived importance of ad hoc application development by the developers and by others in their organization, including various levels of management
7. The tools and mechanisms used in ad hoc development, and the level of satisfaction and desired features

The survey results were used to make conclusions about the future marketplace, trends and opportunities, mechanisms for targeting and selling to this market, and market size assessment.

Several findings from this market research are referenced throughout this paper. In addition to those mentioned specifically, the findings also helped define the need for and scope of the IBM Situational Applications Environment, described in the next section.

THE IBM SITUATIONAL APPLICATIONS ENVIRONMENT

In the future, situational applications may become more challenging to IT development methods and place new demands on the enterprise IT environment. This could put corporate IT in the position of managing enterprise applications while trying to determine how to best facilitate development, deployment, and management of situational applications. Community-based development within the enterprise may significantly increase heterogeneity in the environment and introduce more complexity into monitoring, event analysis, root-cause detection, patch management, and other systems management tasks. Conversely, development based on situational applications can present opportunities to encourage innovation at departmental and individual levels and, at the same time, improve the productivity of knowledge workers.

Situational applications can enable workers to react quickly to changing needs with just-in-time solutions that are a better fit to some business problems. In addition, by embracing this development paradigm, IT enables the automation of business areas that were not affordable or were considered too narrow a niche before—a phenomenon sometimes called *the long tail*, a term first coined and popularized by Chris Anderson.[23]

To accelerate the adoption of situational applications in IBM and to test the potential benefits of community-based development in the enterprise, the office of the chief information officer (CIO) established an initiative called the Situational Applications Environment (SAE). Envisioned as a living-laboratory experiment to observe and harvest best practices, SAE is enabling an increasing number of employees to benefit from the use, creation, and sharing of situational applications.

SAE scope

The IBM intranet contains an enormous wealth of information, services, and community spaces covering all aspects of the business from research,

product information, and marketing materials to business operations, personnel data, and social events. Some users are repurposing this information to meet their particular business needs, increasingly using Web technologies and the growing number of available internal and external services. SAE was conceived to recognize, encourage, and build a community around the activity of constructing situational applications so that solutions and best practices could be shared throughout IBM. As part of this initiative, corporate IT assumed the role of solution enabler by providing the tools and data services required by this community. SAE facilities built around three focus areas—consumables, tools and utilities, and community—are discussed in the next sections.

Consumables

A *consumable* is a building block used in application construction. It can be a service with a recognizable API called to obtain data, a code snippet that can be incorporated into a server-side script, or a JavaScript fragment to enhance a Web page. Without a substantial collection of consumables, the ability to build new applications is severely restricted. On the other hand, if there are a large number of consumables but they are difficult to find and understand, the adoption of situational application development will be inhibited.

Tools and utilities

Situational applications are built by combining consumables to create new capabilities, usually with some mediating logic and user-interface components. In this process, inevitably some integration code is required, and the resulting entity then needs a place in which it can be deployed. Included in tools and utilities are a Web presence for raising awareness of new situational applications and consumables, catalogs for locating and describing these assets, mashup makers for assembling applications, and lightweight hosting facilities for running applications and consumables once they are built.

Community

Several community aspects are important to the adoption of situational applications:

- *Collaboration*—The primary community of people who need a solution work on the application together, sharing it and improving it.
- *Wide communication*—As new applications and consumables are created, they are more likely to be exploited and reused if their existence is advertised widely outside of the primary community that created them.
- *Feedback*—Interested parties can comment on, suggest improvements for, or even share their original work adaptations.

SAE architecture

The requirements derived from consumables, tools and utilities, and community, as described above, shaped SAE and its architecture.

SAE Web site

The SAE Web site (***Figure 2***) is designed as a central hub for situational applications in IBM. The home page presents the latest and most popular applications and consumables, news items, latest forum threads, and guidance for developers. Other pages focus on available facilities; recommended processes for building, hosting and advertising applications and consumables; and help in the form of frequently asked questions.

SAE catalog

The SAE catalog (***Figure 3***) stores details of applications and consumables entered by an interested party, usually the owner. The details include minimal categorization augmented with tagging. Further tags can be added by the community, and both the entry and tags can be rated based on popularity and relevance. This user-generated taxonomy, or *folksonomy* as it is becoming known, is then used to filter entries along with more traditional keyword search techniques. Users can comment on entries and share their own usage examples.

The collaborative feedback loop implemented in the catalog design traces its roots to the open source movement. The feedback between the community and the developer is direct, allowing issues to be identified, acknowledged, and resolved, and resolutions praised. Community members take pride in their contributions, which can consist of writing the software, proposing new features, and identifying or fixing bugs. Everyone is solving the common problem so even the smallest recognition (a comment or a forum post) feeds the cycle of participation. Users can subscribe to most of the catalog information so that updates and other

Figure 2
SAE Web-site home page

activities can be streamed to interested users in a push rather than a pull manner.

Figure 4 shows how the catalog asset details are delivered to the SAE Web site by means of cached data feed. Users can navigate either directly to the application in which they are interested or to an entry in the catalog, where they can learn how to use APIs, provide their comments, or rate an asset. The catalog can be used to record details of assets hosted within or outside SAE and for any external services and applications that the community might find of particular interest.

Hosting

There are two SAE hosting offerings to meet different user requirements. The first type is a lightweight virtual hosting environment akin to a simple internal Internet service-provider offering. It includes a Web server, server-side scripting capability, and data storage. Users can upload code and other artifacts and make small configuration changes that affect only their virtual host. They are not allowed to make changes to system-wide configurations or to have root access. This option is more than adequate for most application developers who do not perform system administration tasks or complex system configuration or manage specialized middleware and back-end software.

The second type of hosted offering provides the higher-level tools that allow users to create content and to function with little or no coding. These

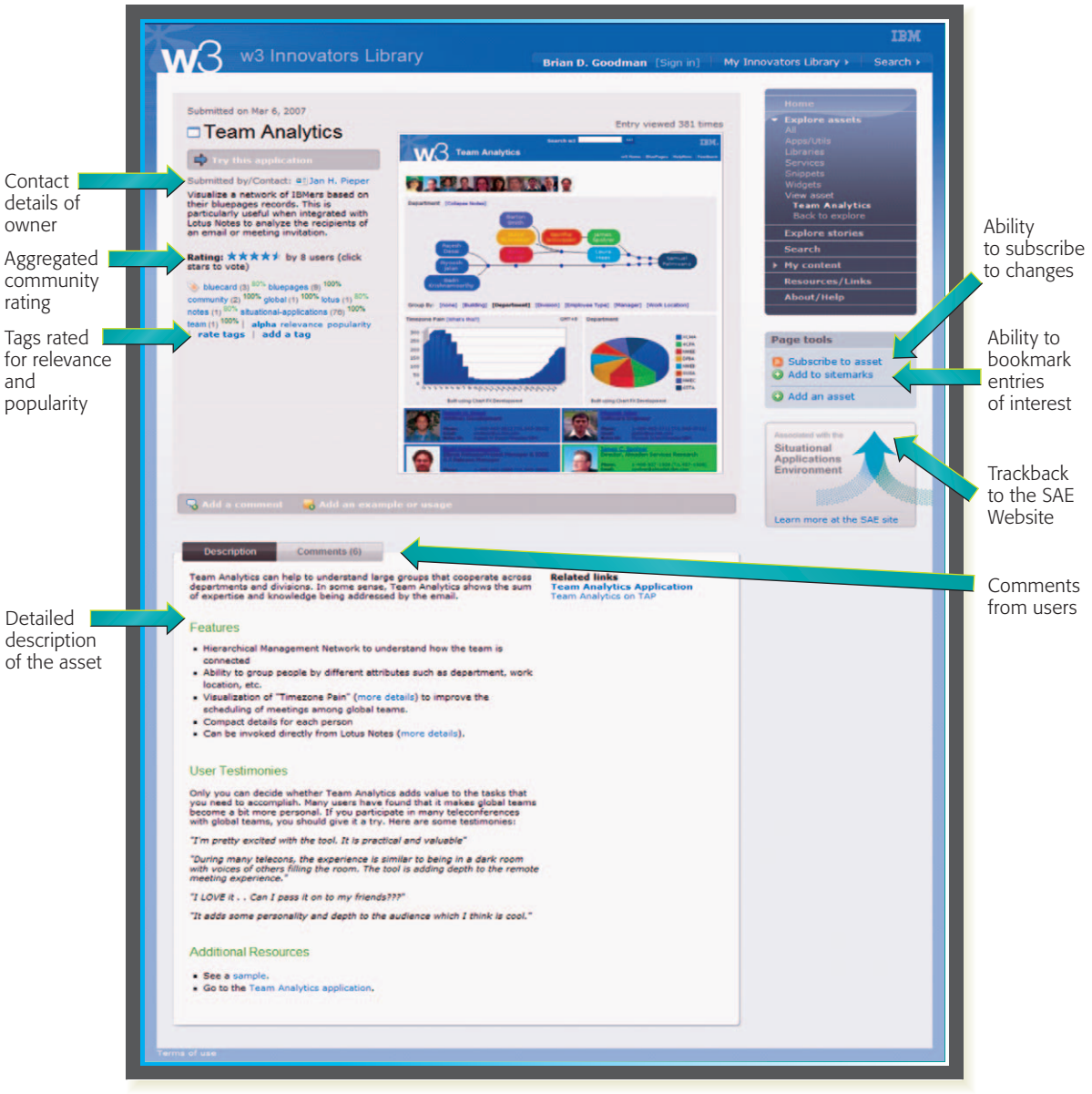

Figure 3
SAE-catalog asset details

environments, or *mashup makers* as they are becoming known, are built on the idea of wikis. They are community-edited Web sites in which a high-level markup language, or preferably a set of sophisticated graphical tools, are used to create pages of content with application function. Users build pages from a palette of components, test, and then share the resulting application without infrastructure considerations. The host platform manages the complete cycle of assemble, run, and share, exploiting rich client-side components to give the user an integrated development experience. Several mashup makers are emerging. Two of the more mature examples, ADIEU[20] and QEDWiki,[19] are briefly described later in the section "Mashup makers".

Seeding the environment

The mashup culture has taken off largely due to the availability of easy-to-use and compelling APIs from sources such as Google, Yahoo! Inc., and Amazon.com, Inc. Seeding SAE with a compelling set of

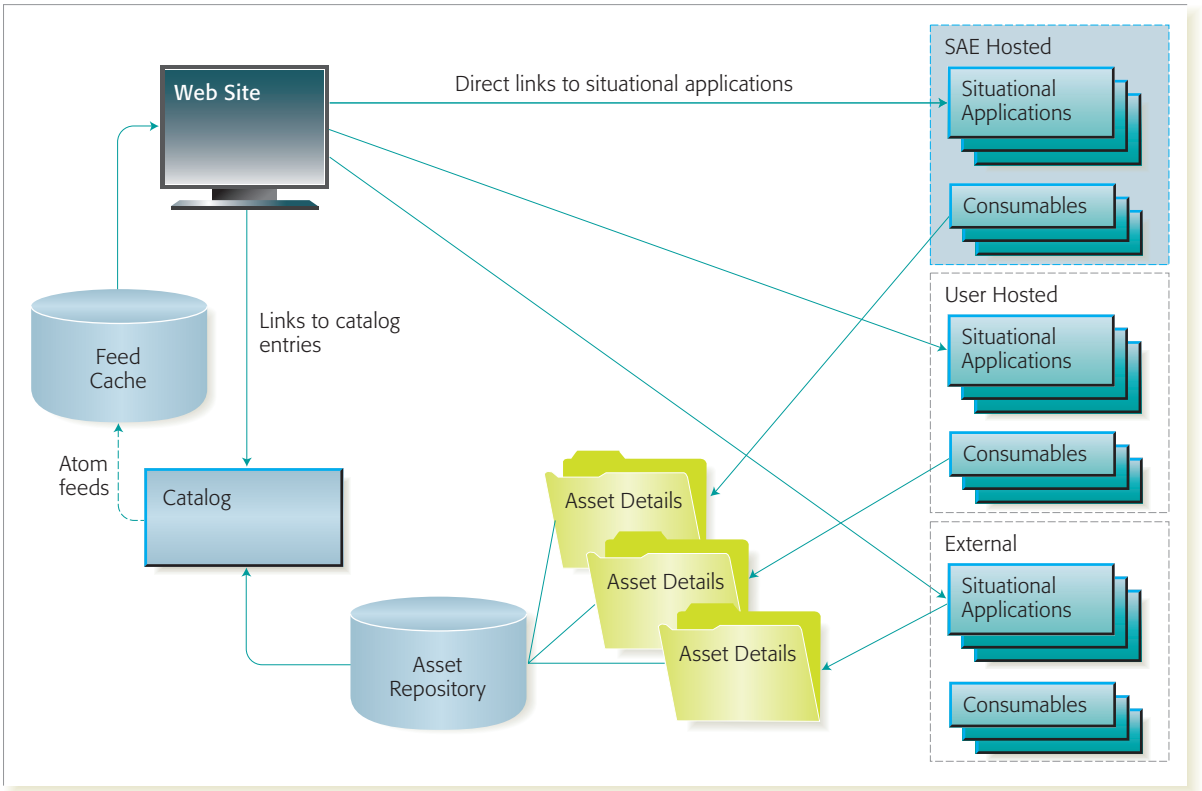

Figure 4
SAE architecture

example applications and consumables was seen as critical to stimulating a similar culture within IBM. Three applications and 27 consumables were included in the initial release; within the first two months, the community had helped swell these numbers to 28 applications and some 60 consumables, with the numbers rising to 137 applications and more than 100 consumables by the end of the seventh month. In the next sections, we describe several assets that have been proven popular based on user feedback and usage statistics.

IBM Travel Maps

The IBM Travel Maps application combines the IBM-recommended hotel list with information about IBM locations, rental car locations, airports, restaurants, and other points of interest on a navigable map, creating a convenient trip-planning aid for business travelers. Although an IBM business partner provides the IBM Online Travel Reservations (OTR) self-service system for planning and booking corporate travel, the OTR currently does not include convenient location-mapping features or information about local points of interest, as is often found on popular travel sites. Travel Maps offers these missing features.

The original Travel Maps demonstration was developed by two researchers for an internal mashup competition. Built in several weeks (approximately 80 hours of development time) with JavaScript and the PHP scripting language, the application was largely populated with Web-page scraped data (data extracted from the display output of another program), held in a static database, and limited to the United States only. Although both researchers had strong technical backgrounds with expert-level skills in IBM DB2* and Extensible Markup Language (XML), they had intermediate-level PHP skills, and they both were beginners in Hypertext Markup Language (HTML) and JavaScript.

Having won the competition, this original demonstration quickly became popular and was used widely within the IBM Web developer community. Users began to provide feedback to the authors,

including ideas to improve the application. Given that IBM owns much of the data behind Travel Maps, an obvious first improvement step was to provide more direct data access by using more robust methods. The office of the CIO negotiated with the IBM Real Estate Operations and the IBM Travel departments for the required access to enterprise data stored in DB2. With access given, during the next several weeks the enterprise business information (EBI) framework was built to make the data accessible through REST-style services.

External Web-page scraping algorithms were improved and error tracking was added to mitigate against problems caused by changes to the underlying Web pages. Data scraped from external sources included car rental locations and points of interest, such as restaurants and shopping centers, from travel Web sites. The Atom[24] data-feed format was used to make gathered data more easily consumable. A database cache was built to hold data from these feeds for runtime use. As the underlying data was nonvolatile and resource-intensive to retrieve, a refresh agent was added to update the database periodically, aiding performance without impacting results. The final change of note was a worldwide geocoding service that expanded the coverage beyond North America. (*Geocoding* is the process of assigning geographic identifiers [e.g., codes or geographic coordinates expressed as latitude-longitude] to map features and other data records, such as street addresses.) The application was incrementally enhanced by three developers over 10 weeks. Although the team had expertise in HTML and JavaScript, they had only intermediate XML skills and no skills in PHP.

These improvements have been augmented as a result of user feedback that helped identify bad data. The enhanced version became so popular that the IBM Real Estate Operations management group added it to the official intranet travel site only three weeks after the SAE launch.

Virtual Team Locator

The Virtual Team Locator application was created to assist sales-team communication by visually locating an account team for a specific client and instantly determining who was available. On a navigable map the application combines current employee location and instant messaging status with internal directory information and the sales organization list of client executives and representatives. The user can choose to send an instant message to logged-in colleagues or an e-mail to those who are disconnected.

The original application was built by one developer in QEDWiki in less than 40 hours over two weeks, demonstrating the design concept with a static data extract. The next version was built by another developer in less than 40 hours over four weeks. An XML, JavaScript Object Notation (JSON), and HTML expert, this developer had no QEDWiki experience but some experience with the SnipSnap HTML-enabled wiki, which he chose as an assembly tool.

A REST service was created to extract sales account-team information stored in a relational database. The IBM Lotus* Sametime* 7.5 SOAP-based location awareness service provided employee current-location and instant-messaging status (e.g., *available, away*, and *do not disturb*) for an employee. The developer quickly discovered that if an employee was not logged in or had decided to make the location private, the service returned incomplete data. To mitigate the situation, he developed a proxy REST composite service returning an employee default work location extracted through the API of the IBM BluePages (the corporate internal directory) if the Sametime location could not be determined (***Figure 5*** shows the Virtual Team Locator architecture).

This application made information created by the sales organization available for use throughout IBM. It has also demonstrated how SOAP and REST services can be combined in a single mashup.

Bluecard widget

In the ever-changing social landscape of increasingly mobile business, the need to recognize fellow employees is important. The IBM corporate directory, BluePages, enables employees to easily manage, find, and connect with colleagues. All of the rich metadata associated with an employee, from basic contact information to skills, current projects, and patent activity, can be accessed through a series of REST-style services, facilitating reuse and remixability. It is the most highly trafficked application in IBM, serving millions of requests per day by employees looking for contact information and expertise. Bluecard, one of the first SAE widgets,

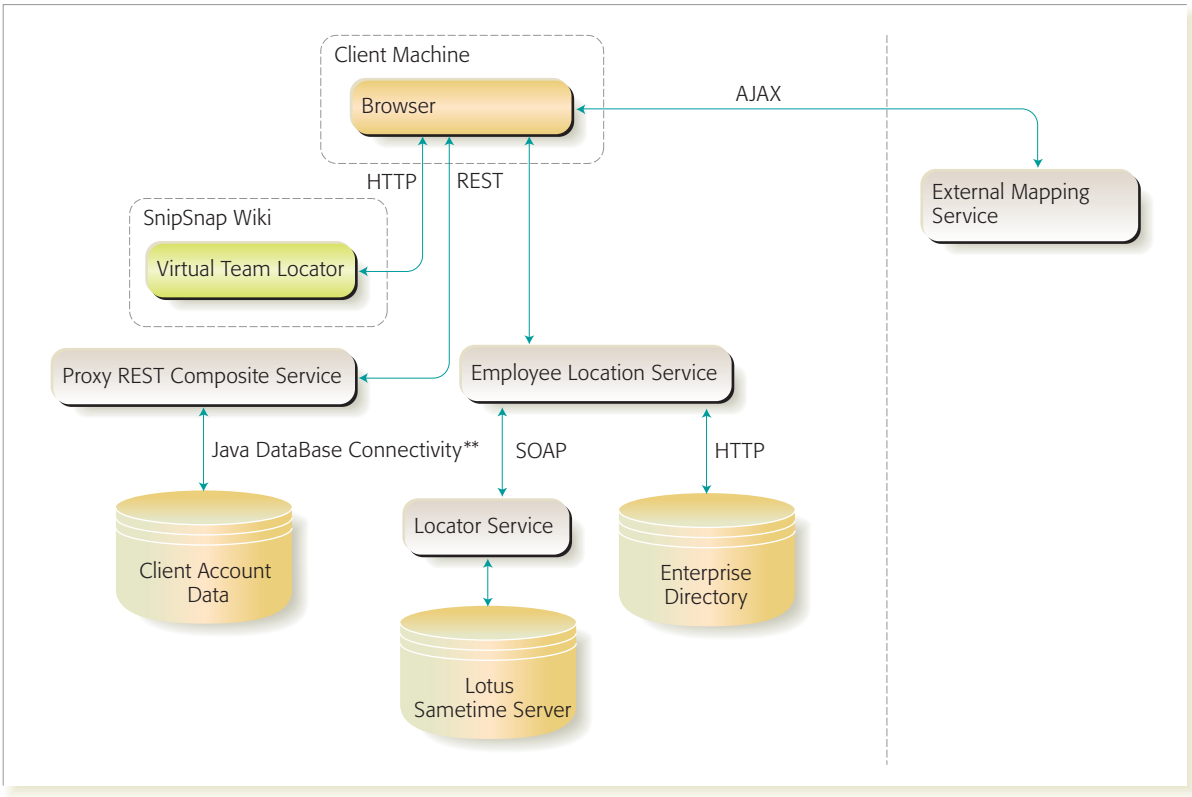

Figure 5
Virtual Team Locator architecture

takes advantage of this popularity by automatically providing contextual employee information.

In fewer than five lines of JavaScript, Web site owners can cut and paste this functionality into their solution. It automatically scans the HTML document and binds to e-mail addresses it recognizes or specially tagged page areas. When a user hovers over one of these bindings (e.g., an e-mail address) a nonintrusive overlay prompt indicates that a Bluecard is available. Clicking on that overlay produces a business card like that shown on the right in ***Figure 6***.

Bluecard is a prototypical example of the corporate IT role as a producer of high-quality reusable components that require both greater skill levels and greater access to corporate data and systems. In addition, corporate IT has the opportunity to create components that establish a common look, feel, and interaction pattern.

Tooling

There are three tool categories that can be found in SAE: frameworks, ad hoc tools, and mashup makers.

Frameworks

Frameworks are used to facilitate the construction of services, typically those that provide access to corporate data. The IBM CIO office built Nova Services and the EBI framework. Nova uses a simple Java Platform, Enterprise Edition (J2EE**) pattern to allow developers to define and host new REST-style Web services. The EBI framework takes data in relational databases and uses Structured Query Language (SQL) to construct REST services that can return data in XML, Atom, or JSON format. External mashup frameworks, such as Yahoo! Pipes**[25] and Kapow Technologies openkapow,[26] help users create new data services and functions with interactive construction tools that can be learned in a few hours.

Ad hoc tools

Ad hoc tools developed by the community perform functions such as mediations and data translations or interpretation of different postal address formats. They can be used at runtime or in the process of building an application or a consumable. One such tool, registered in SAE, allows a user to discover the

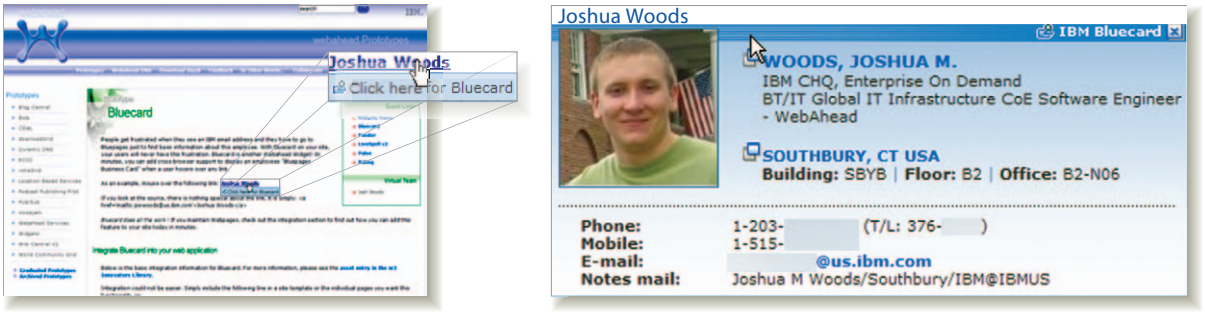

Figure 6
Bluecard: (A) hover in a Web page, and (B) view

latitude and longitude of any place on the globe. This can then be used to help improve geocoded information that may have been obtained automatically from an address lookup service. We expect the use and number of these tools to grow as the community matures and the culture of sharing components becomes more ingrained.

Mashup makers

Mashup makers aim to engage nondevelopers in the construction of applications. They provide visual page editors for assembling components to build useful function, at the same time minimizing, and in some cases eliminating, the need to write code. IBM has developed such environments and continues its work in this area. Two of the more mature examples are QEDWiki and ADIEU.

QEDWiki is a mashup builder that can be used to create a Web application by assembling a collection of widgets on a page, wiring them together to define the application behavior, and then possibly sharing the mashup with others. Mashup enablers (programmers) populate QEDWiki with a palette of widgets that provide application domain functions or information-specific functions. To create a widget requires knowledge about QEDWiki widget specifications. A new widget can be loaded into the environment to extend native capabilities. Utility widgets can be configured to receive and make use of Really Simple Syndication (RSS) or Atom feeds, access databases, or produce visual controls, such as tables, notebooks, and maps.

ADIEU users can develop Web services and Web applications without knowing any specific programming language. They develop applications using collections of *cards*. A card acts like a single-function application in a form-based, desktop-like environment. The data fields in cards can be used like cells in a spreadsheet and can contain either data or an expression that determines the data at runtime. Cards can also run other cards, a capability that provides the basic flow control necessary for programming concepts such as decision branching, sequences, and loops. Specific card types produce the Web pages needed for the application and drive the calls to external services.

Early observations

At the time of this writing, SAE was in its eighth month of use. Some important usage patterns, benefits, and challenges have already been detected. The insights gained through these informal observations help us plan future SAE improvements and determine the next areas of interest to be explored.

Access to data and data ownership

A fundamental requirement for a successful situational-application ecosystem is access to data, preferably through a simple, standardized interface provided and sanctioned by data owners. In the case of external services, this may be provided on a fee basis, in which case it is important to negotiate contracts with providers who recognize the nature of situational applications.

For example, a consumable that accesses an external geocoding service might be included in many applications and perhaps called hundreds or thousands of times in the development process by a nonprogrammer as a means of testing through use. This could result in exponential increases in transaction rates. For external services with

per-transaction fees, this could result in huge, uncontrolled costs. Usage contracts based on set limits or unlimited access for a fixed or capped fee can avoid this cost.

A quick way to acquire data is by scraping existing Web pages (Dapper[27] and offerings by Kapow Technologies are examples of scraping tools). Data scraping can be done without the knowledge of the data owner, who otherwise might object to the way the data is used or the impact on site performance. This, in turn, can lead to the data owner adopting spoiling tactics to make the scraping process more error-prone or impossible. The data owner may also take legal action for copyright breach or brand image damage if the information reuse is in conflict with the owner's business interests or if it creates objectionable associations with information or parties. Scraping Web pages might be a quick way to obtain internal data when there is no time to negotiate access. External page scraping should be used sparingly and with consideration to potential consequences for both the data owner and the application builder.

Data quality

When data access is opened for use in situational applications, sometimes fields that have traditionally been unimportant or even hidden are made available. Users can often assume levels of completeness and consistency that in reality do not exist. A classic example is address information that has been collected by many applications and made use of in freeform style. A developer who may want to use that data for assigning an address programmatically will find that the inconsistent address information styles now need to be handled with complex code.

This problem can be mitigated by providing information to users on the quality of the data they are seeing and by informing data owners about the new ways that their data is being used and the data quality issues that have arisen. Often data owners are willing to tidy up data when problems are identified. This realization has prompted the addition of features into the EBI framework to facilitate the reporting of data issues discovered by users. This addition completed the user-developer feedback loop by also including the data owner in the process. We observed improvements in data quality in enterprise data sources because of this active user participation.

Data interpretation and provenance

When applications are tightly coupled to data sources or are using Web services with agreed-upon data schemas, the data meaning is usually well-specified and its provenance is easily traced. With situational applications this is not the case.

Data from an unfamiliar domain can easily be misinterpreted by the developer who is eager to get a service up and running as fast as possible. The layering of consumables and their output can result in the logic and assumptions masking the true meaning of the data. For this reason, it is important to advise users at the outset that they should assume a lower level of accuracy than what they usually expect from formally developed applications. Community rating and feedback can help others understand what to expect from a service and possibly initiate an improvement process. As situational applications move into the realm of business applications, it will become even more important for data provenance and quality to be clearly indicated if costly errors and breaches of legal responsibilities are to be avoided.

User expectations

Expressions such as "quick and dirty," "just good enough," and "the perpetual beta" are the mantras of situational application developers. Does this encourage sloppy programming practices? It depends on your point of view.

A developer who was able to get a particular narrow scenario that met a specific business need up and running in record time views this effort as a success and perhaps as a demonstration of technical skill. The very first user outside of the targeted team will, in all likelihood, stray away from the narrow scenario and, unsurprisingly, encounter problems. The SAE usage patterns show that developers who are interested in having their applications adopted by a larger community can accelerate the process by clearly communicating which scenarios are addressed by their application.

The willingness of users to offer feedback to the developer, who may be unaware of problems or alternative uses, directly contributes to the adoption of the application and can further its improvement.

The SAE catalog supports this kind of dialog, along with rating systems that encourage a general evolution toward more successful and higher quality work. Over time, problems caused by poor coding practices should be sifted and eliminated by the community.

Our observations show that Web developers and those who are accustomed to the social aspects of Web 2.0 are more tolerant of bugs and poor performance than business users. This might continue to be a barrier to the early adoption of situational applications by business users until this new culture becomes ingrained and all parties understand the limitations as well as the strengths.

Performance

Users who are often tolerant of functional errors in prototypical applications, accepting, for example, that it is "just good enough," may be less readily accepting of poor performance (e.g., a trivial task taking a relatively long time to complete). In traditional application development, extensive resources are used to ensure good performance by building well-designed end-to-end architectures. With situational applications, an application is built quickly, often with components that are not under the developer's control. For example, the only way to acquire data might be through an inefficient or complex query that spans several data sources, including Web services, multiple databases, and, through scraping techniques, Web pages. There are few opportunities to optimize data gathering in this scenario. Because data used by situational applications is often not volatile, caching and occasionally refreshing it might be a good strategy. This cache can then be made accessible through an API to improve performance.

To reduce the load on often lightweight server platforms, some builders exploit client-side scripting capabilities with AJAX. In addition to improved performance, this also provides a rich user experience that previously was reserved for applications installed and run on the client. A skilled AJAX developer is able to exploit the multithreaded nature of the technology to prevent modal behavior (i.e., behavior where a user must complete one interaction before any further interactions are allowed). It is characterized by dialog boxes that require the user to click a button to make a choice before he or she is allowed to perform any other actions. However, it remains to be seen whether nonprogrammers can realistically achieve such user interface sophistication, as it is a nontrivial challenge to assemble visual components that represent the flow of events.

Hosting issues

Providing cost-effective lightweight hosting for situational applications raises some complex issues. Hosting platforms must be standardized and eliminate as much manual intervention as possible through automation. They need to offer resilient partitioning and containment because situational applications by their very nature are more prone to errors and to the effects of poor coding than traditional applications.

Some developers take advantage of the latest functions in the new releases of languages and tools, whereas the hosting platforms need to remain with earlier stable versions. By reducing the number and frequency of platform migrations, the chances of adversely affecting the hosted applications is also reduced.

Many professional programmers are accustomed to having root access to a machine, but such access cannot be offered in a hosted environment where services are shared and changes to the configuration would affect other users. Developers must consider the benefits of the hosted environment and determine when using that environment serves better than managing their own infrastructure.

Adoption among different user types

Since launching SAE, we have observed enthusiasm for situational applications from both business and technical users. Response to some applications has resulted in their use shaping the next generation of existing corporate systems. For example, the Travel Maps application discussed earlier is now being used to help shape the future requirements for the corporate OTR system.

Lines of business are showing increasing interest in SAE as an opportunity to meet business challenges faster, especially by automating mostly manual processes or serving areas with only a few users for whom there are no planned IT projects. We have already seen encouraging results when, for example, a situational application used by less than 30 intellectual property analysts resulted in productivity

gains in excess of 50 percent. Another application used by the same small team reduced one of their processes from 2.5 hours to 4 minutes. Much of the focus of the next phase of SAE will be on areas of high business value like this; thereby showing the direct business impact of situational applications on the enterprise.

CHANGING ROLE OF CORPORATE IT

The CEO study[6] done by IBM Global Business Services showed that companies with better business-IT integration deliver significantly better financial results. CEOs articulated the necessity to go beyond simple alignment and completely close the business-IT gap. Based on observations from our SAE initiative, we reason that capitalizing on technological potential and unleashing worker creativity offers a high likelihood of closing that gap.

Four major changes occurring in enterprises are creating the need for the CIO and the IT department to redefine their roles fundamentally, cultivating an entrepreneurial atmosphere by enabling workers to share in the responsibility for their workplaces and the variety of applications they use. These changes, together with the recommended IT adoption actions, are described below and summarized in *Table 1*.

1. *Changing enterprise boundaries*—When asked which sources their companies relied on most for innovative ideas, CEOs ranked employees (41 percent), business partners (38 percent), and customers (36 percent) at the top of the list.[6] This means that two of the three most significant sources of innovative ideas lie outside enterprise boundaries. Companies that outperform the averages financially are much bolder in their reliance on external sources for new ideas. Companies with higher revenue growth reported using outside sources significantly more than those with slower growth.

   Globalization is challenging the traditional definition of enterprise boundaries. Work is performed independent of time and place, or even of who does it. Collaborating teams are no longer defined by physical proximity. The need for new skills and ideas drives businesses to tap into communities outside of their companies. The Web-based community formed by InnoCentive, Inc., for example, matches top scientists from more than 170 countries with difficult research challenges faced by leading companies.[28] The line between a corporation and global communities is blurring.

   Much looser organizational structures require technologies that support virtual teams of specialists scattered around the globe who can collaborate independent of location, time zone, technology, or language. These technologies must also support the instant reconfiguration of such teams. Teams that are formed quickly must be able to define their work environment and address many of their own needs as they arise.

2. *Increasing pace of change*—The use of technology and the pace of technological development are accelerating. The evidence of this is abundant: from the number of Internet users worldwide passing the one billion mark[29] to the number of Web sites growing by 30.9 million in 2006 and shattering the record gain of 17.5 million sites in 2005.[30] Technologies that were unheard of several years ago are fast becoming part of the invisible fabric of everyday life, and the pace of change continues to accelerate. As Ray Kurzweil has put it: "Because we're doubling the rate of progress every decade, we'll see a century of progress—*at today's rate*—in only 25 calendar years."[31] Our expectations of technologies and how they should support us have also changed. We have come to expect new technological inventions to work autonomously and to be integrated seamlessly with devices that are already part of our personal and business environments.

3. *The rise of the knowledge worker*—Knowledge has become one of the most valuable enterprise resources. In North America, knowledge workers are estimated to outnumber all other workers by at least a four-to-one margin.[32] The knowledge worker's environment is dynamic and unpredictable. Their actions and decisions are often driven by unexpected events and exceptions to documented business processes. They have to deal with the growing amount of information and the growing complexity of relationships and regulations in the global economy. To succeed, knowledge workers have to respond instantly to continuous changes, often relying on their know-

Table 1 IT adaptation to changes occurring in the enterprise

	Changes occurring in the enterprise	How corporate IT must adopt
Enterprise boundaries	• Increasing reliance on external sources for innovation • Globalization • Work done by SMEs not employed by the enterprise	• Provide tools for easy collaboration outside enterprise boundaries • Enable quickly formed teams to define their own work environment and automation needs
Pace of change	• Deepened competition, escalating customer expectations, and unexpected market shifts • Accelerating pace of technological developments	• Muster tools and approaches to quickly identify, develop, and move forward innovative ideas • Immediately and seamlessly integrate ideas and new technologies into the enterprise
Workforce	• Rise in grassroots computing among both programmers and business employees • High overall computer literacy • Increased individualization of work environment • Knowledge of and easy access to externally available tools and APIs	• Actively cultivate an entrepreneurial atmosphere • Provide tools and services to enable workers to automate their own work environments • Open enterprise data sources and provide building blocks that can be quickly assembled into solutions • Negotiate licenses to provide access to external services • Implement lightweight goverance for situational applications

how, personal relationships, and unique understanding of their environment. This tacit knowledge is not easy articulated or codified.

On the other hand, the corporate approach to building automated systems, is based on documenting and freezing a relatively static set of end-user requirements. Unsurprisingly, these requirements capture the knowledge worker's environment at a particular instant, not what will be required when the system is deployed. Consequently, solutions are often obsolete by the time they are developed. Because a knowledge worker's environment is unique and ever-changing and cannot be captured in a finite enumerated list, the development and maintenance of specialized applications by the IT department is expensive. Traditional software development approaches produce far better results when applied to generally understood, anticipated, nonspecialized processes, such as recording financial transactions or generating payrolls. The environment of the knowledge worker requires the ability to create just-in-time solutions to address unique situations without waiting for the IT department.

4. *Millennials in the workplace*—Millennials who grew up surrounded by a digital world both at home and at school are starting to enter the workforce. This generation, largely shaped by the Internet, is poised to change our work environments dramatically.[3,4,8,33] Millennials have skills and expectations different from today's mainstream workforce. Unafraid of technology, accustomed to figuring things out on their own, and skilled at acquiring knowledge they may need but do not have, they are likely to create their own solutions instead of engaging or waiting for the IT department.

Instead of investing valuable resources trying to document and freeze requirements, the IT department is starting to improve worker effectiveness by providing better tools and services to enable workers to create their own specialized ad hoc applications. Business users increasingly pressure the IT department to expose functional interfaces of enterprise applications and to make corporate data sources—both structured (e.g., databases) and unstructured (e.g., e-mail)—more broadly available. These interfaces are being refactored into specialized consumables that can be quickly assembled into a situational application. The IT department can further assist the grassroots development by building and enabling more general-purpose services, such as authentication, geocoding, and mapping.

At the same time, the role of the IT department should include fostering an environment in which workers can share their solutions with others. This will encourage improvement of solutions through social collaboration and support their adoption to meet new and evolving business needs. The IT department will be accountable for simplifying and enabling collaboration across geographic areas, time zones, and corporate boundaries.

Although each situational application is relatively simple, the enterprise IT environment will grow in heterogeneity and complexity. To shield users from these intricacies, the IT department can employ lightweight governance to prevent them from inadvertently damaging their own solutions or the solutions of others and to prevent the accidental disclosing of protected assets or the violating of agreements with third-party asset providers.

SUMMARY AND CONCLUSIONS

In this paper we have discussed the growth of grassroots computing and presented our vision of the changing role of corporate IT as it deals with the challenges and capitalizes on the opportunities arising from these trends. Corporate IT will gradually move from being the exclusive provider of enterprise systems to an enabler and facilitator of solutions built by self-reliant employees. We argue that this change is a necessity. The health, competitive power, and even survival of an enterprise will largely depend on its ability to understand and harness the power of knowledge workers who are enabled to take responsibility for providing automatic solutions to meet many of their business needs.

CIOs have an important but challenging role to change the enterprise culture to one that encourages and embraces innovative thinking and individual self-sufficiency. IBM is actively removing existing technological and cultural barriers to support an entrepreneurial atmosphere. SAE was designed as an evolving experiment in enterprise-wide enablement and adoption of situational applications.

Although the initial SAE release addressed some challenges described in this paper, there is much left to explore and discover, such as access to enterprise data sources through automated role-based entitlement systems, data provenance, strengthening overall governance, especially in the area of externally produced consumables, and the evolution and introduction of mashup makers targeting business users. The increasing acceptance of SAE by IBM organizations, teams, and individuals and growing interest on the part of our clients inspire us with confidence that enterprises will benefit from these ideas and the SAE experiment described here.

ACKNOWLEDGMENTS

For their valuable contributions, we thank David Simmen and Mehmet Altinel for development of the original IBM Travel Maps, Josh Woods for development of the SAE catalog and several popular widgets, and Robi Brunner for development of the Nova Services framework. For their work on the EBI framework, we are indebted to Jamshid Vayghan, Steve Garfinkle, and their team. We also thank many IBM colleagues (too numerous to individually name) who either contributed to the development of SAE or shared their experiences and "lessons learned" with the situational-applications development team.

CITED REFERENCES AND NOTES

1. D. Gelernter, "The Second Coming–A Manifesto," *Edge Foundation, Inc.*, http://www.edge.org/3rd_culture/gelernter/gelernter_p1.html.
2. "Knowledge worker," a term coined by Peter Drucker in his 1959 book, *Landmarks of Tomorrow: A Report on the New "Post-Modern" World* (Transaction Publishers, New Brunswick, N.J.), is one who works primarily with information or one who develops and uses knowledge in the workplace.
3. The term "millennials," as used by Neil Howe and William Strauss, coauthors of *Millennials Rising: The Next Great Generation* (Vintage Books/Random House, New York, 2000), describes the generation of people born between the early 1980s and 2000. Other authors have coined different terms to describe approximately the same generation: Y generation, echo boomers, and NetGeneration.
4. C. Raines, *Connecting Generations*, Crisp Publications, Inc., Berkley, CA (2003).
5. Facebook (http://www.facebook.com/) is a social networking Web site, popular among college students.
6. *CEOs are Expanding the Innovation Horizon: Important Implications for CIOs*, IBM Global Business Services, http://www-03.ibm.com/industries/retail/doc/content/bin/IBM_CEO_Study.pdf.
7. T. O'Reilly, *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*,

O'Reilly Media, Inc., http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1.

8. J. Sapir, *Igniting the Phoenix: A New Vision for IT*, Xlibris Corporation, Philadelphia, PA (2004).
9. E. R. McLean, "End Users as Applications Developers," *Proceedings of the Application Development Symposium*, Monterey, CA (1979), pp. 49–55.
10. B. A. Myers, A. J. Ko, and M. M. Burnett, "Invited Research Overview: End-User Programming," *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada (2006), pp. 75–80.
11. M. Serrano, E. Gallesio, and F. Loitsch, "Hop, a Language for Programming the Web 2.0," *Proceedings of the 21st ACM SIGPLAN Conference on Object-Oriented Programming Languages, Systems, and Applications*, Portland, Oregon (2006), pp. 975–985.
12. J. Wong and J. Hong, "Marmite: End-user Programming for the Web," *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada (2006), pp. 1541–1546.
13. E. J. Lerner, "Sash Simplifies the Web," *Think Research*, IBM Corporation, http://domino.watson.ibm.com/comm/wwwr_thinkresearch.nsf/pages/200011_sash.html.
14. J. Ponzo, L. D. Hasson, J. George, G. Thomas, D. Gruber, R. Konuru, A. Purakayastha, R. D. Johnson, J. Colson, and R. A. Pollak, "On Demand Web-Client Technologies," *IBM Systems Journal* **43**, No. 2, pp. 297–315 (2004).
15. Yahoo! Developer Network: Design Pattern Library, Yahoo! Inc., http://developer.yahoo.com/ypatterns/.
16. programmableweb: Web 2.0 APIs and Mashups, ProgrammableWeb.com, http://www.programmableweb.com/.
17. C. Traynor and M. G. Williams, "A Study of End-User Programming for Geographic Information Systems," *Proceedings of the 7th Workshop on Empirical Studies of Programmers*, Alexandria, VA (1997), pp. 140–156.
18. M. LaMonica, *IBM Eyes Programming for the Masses*, CNET News.com, http://news.com.com/2100-1007_3-6065324.html.
19. QEDWiki, IBM alphaWorks Services, http://services.alphaworks.ibm.com/qedwiki/.
20. Ad Hoc Development and Integration Tool for End Users, IBM alphaWorks, http://www.alphaworks.ibm.com/tech/adieu.
21. SnipSnap, Fraunhofer Institute for Computer Architecture and Software Technology, http://www.first.fraunhofer.de/en/snipsnap.
22. D. R. Millen, J. Feinberg, and B. Kerr, "Dogear: Social Bookmarking in the Enterprise," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada (2006), pp. 111–120.
23. C. Anderson, *The Long Tail, Wired*, CondéNet Inc., http://www.wired.com/wired/archive/12.10/tail.html.
24. Atom Syndication Format specification, AtomEnabled Alliance, http://www.atomenabled.org/developers/syndication/atom-format-spec.php.
25. Pipes, Yahoo! Inc., http://pipes.yahoo.com/pipes/.
26. openkapow, Kapow Technologies, http://www.openkapow.com.
27. Dapper, http://www.dapper.net.
28. *Dr. Alpheus Bingham, InnoCentive's President and CEO, Receives Business Processes Award at the Economist's Fourth Annual Innovation Summit*, Press release, InnoCentive, Inc. (November 30, 2005), http://www.innocentive.com/about/press/20051130_DrWinAward.html.
29. Internet World Stats, Miniwatts Marketing Group, http://www.internetworldstats.com/stats.htm.
30. April 2007 Web Server Survey, Netcraft, Ltd., http://news.netcraft.com/archives/web_server_survey.html.
31. R. Kurzweil, *The Law of Accelerating Returns*, KurzweilAI.net (2001), http://www.kurzweilai.net/articles/art0134.html?printable=1.
32. S. Haag, M. Cummings, D. J. McCubbrey, A. Pinsonneault, and R. Donovan, *Management Information Systems For the Information Age*, Third Edition, McGraw-Hill Ryerson, Whitby, Ontario, Canada (2006).
33. D. Morello and B. Burton, "Future Worker 2015: Extreme Individualization," *Proceedings of the Gartner Symposium ITxpo*, Orlando, FL (2005).

Accepted for publication March 22, 2007.
Published online September 25, 2007.

Luba Cherbakov
IBM Corporation, IBM CIO Office, 7850 Langley Ridge Road, McLean, VA 22102 (lubacher@us.ibm.com). Ms. Cherbakov is an IBM Distinguished Engineer and a member of the IBM Academy of Technology. She leads the SAE initiative to bring together Web 2.0 technologies, available services, and enterprise data. In her previous assignments with IBM Global Services, Ms. Cherbakov designed architectures and delivered complex and first-of-a-kind solutions to a wide variety of customer industries. She is an author or contributor to the IBM Service-Oriented Modeling and Architecture (SOMA) method, reference architectures, the Architectural Description Standard, and grid computing assets. Ms. Cherbakov is a two-time recipient of the IBM Outstanding Technical Achievement award and a recipient of the IBM Corporate Award, the highest technical award for unique technical contributions of superior business value. A member of the IEEE Computer Society, the Society of Women Engineers, and the Association for Computing Machinery, she has an M.S. degree in computer science from George Washington University, with a major in software and systems and a minor in artificial intelligence and simulation.

Andrew Bravery
IBM Hursley Laboratories, Hursley Park, Winchester, Hampshire, SO21 2JN, United Kingdom (andrewjf_bravery@uk.ibm.com). Mr. Bravery is a senior IT architect on the IBM Software Group Architecture Board Incubator Projects team. He is on a rotational assignment with the IBM CIO Office as the chief architect of the Situational Applications Environment, for which he recently received an IBM Outstanding Technical Achievement award. Mr. Bravery has worked in emerging technology fields such as object-oriented programming, content management and portals, and multichannel architectures. His recent focus is on simplified programming models and Web 2.0 technologies and how they can be applied to development in the enterprise. Mr. Bravery is a member of the British Computer Society and has a B.S. degree with honors in physics from the University of Birmingham, United Kingdom.

Brian D. Goodman
IBM Corporation, IBM CIO Office, 150 Kettletown Road, Southbury, CT 06488 (bgoodman@us.ibm.com). Mr. Goodman is a senior software engineer and Certified IT Architect leading an innovation team responsible for developing emerging technology that enriches collaboration and productivity. Recent work involves creating social software and grassroots collaboration environments for the enterprise. He cofounded and was the principal architect directing technical enablement for the IBM Technology Adoption Program (TAP), which accelerates the process of identifying, developing, and moving innovation from the laboratory to internal applications and then, to customer implementation. Mr. Goodman's expertise is in the area of collaboration, innovation management, and high-performance Web applications. He has authored over 33 publications and holds 37 patent filings worldwide. He is a two-time recipient of the IBM Outstanding Technical Achievement award. He is a member of the IEEE Computer Society, the Association for Computing Machinery, and the Association of Open Group Enterprise Architects. Mr. Goodman earned a multidisciplinary B.A. degree in computer science, psychology, and graphic design from Hampshire College.

Aroop Pandya
IBM Corporation, IBM CIO Office, 2455 South Road, Poughkeepsie, NY 12601 (apandya@us.ibm.com). Mr. Pandya is a Certified IT Architect with chief architect responsibilities in the IBM CIO Office innovation team. Currently, he is spearheading the adoption of situational applications development by business teams. In the past, Mr. Pandya has led or created architectures for such projects as WorldJam, ThinkPlace®, and Lasso/Effective Meetings, which have become offerings through various business units. Mr. Pandya has received the Outstanding Technical Achievement Award and the Execute Now and Innovator Award. He has a Masters degree in computer science from Rensselaer Polytechnic Institute and a B.S. degree in computer science from Marist College.

John Baggett
IBM Corporation, Research Division, 11501 Burnet Road, Austin, TX 78758 (jbaggett@us.ibm.com). Mr. Baggett is an advanced strategy and planning marketing manager in the Market Insights organization. His broad experience in many technical roles supports his current coverage of the IBM Research portfolio across eight worldwide laboratories and all strategy areas. His focus for the past two years has been on programming models and tools as well as on distributed computing. Mr. Baggett holds B.S. and M.S. degrees in electrical and systems engineering from Oklahoma State University, completed executive business programs at the Graduate School of Government and Business Administration at George Washington University and at the Colgate Darden Graduate School of Business Administration at the University of Virginia, and holds an advanced certificate in marketing from the Chartered Institute of Marketing. ■