

A Special Report From InformationWeek and Dr. Dobb's



# App Dev That Delivers

**F**rom design to delivery, software development isn't what it used to be. What's changed, more than anything, are the development process and the developers themselves—they're woven into the business fabric more than ever before. The cliché of the solitary coder is dead.

Driving this transformation is the need for relentless focus on the business results that software should produce, forcing developers to understand nuances of the health care or manufacturing or finance worlds in which they work. It's blurring the roles of analyst and developer.

Developers, and the executives leading development teams, still must be masters of their craft. In this economy, they're under more pressure than ever to turn out reliable software on time and under budget, and that shapes the processes and tools app dev leaders choose. Rapid delivery, for instance, is one key trend. That puts a premium on automating tasks such as builds and testing, freeing developers to focus on great code that quickly meets business goals. Coding as a team sport is another trend, spanning product managers to CIOs to developers and testers around the world. That's driving tools for collaborative development and increased visibility into the app dev process.

For this report, InformationWeek and Dr. Dobb's teamed to chronicle how leaders in application development are meeting those demands. With articles by app dev leaders themselves and from InformationWeek writers, we spotlight companies tackling a wide range of challenges across industries, from a world-leading cancer center building its own e-records system to a news organization taking delivery mobile.

—Jonathan Erickson, Dr. Dobb's editor in chief

## MOBILE DEVELOPMENT

### Plug-In Cuts AP's Time To Market

**O**n a given day, more than half of the world's population sees news from The Associated Press in print, broadcast, and online. With people everywhere using mobile devices as their first screen for news, we saw the opportunity to extend our presence by launching *apnews.com*, a multimedia mobile news portal.

Nokia's Web Runtime plug-in provided a solution, extending the Web browser by supporting widgets—small, task-specific, standalone Web applications that let us deliver im-

mediate access to hyperlocalized news from AP and more than 1,000 local content providers through an icon on the device's home screen. Widgets provide Internet content without a browser.

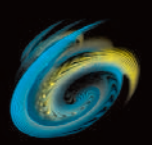
The AP Mobile widget we built is a free, ad-supported application preloaded onto the Nokia N97 smartphone. AP selected the N97 because of its touch-screen functionality and multimedia capabilities, letting us deliver news stories, photography, and video.

There were two main benefits in

using the Web Runtime plug-in to build the widget: It freed us from having to learn new development tools in new development environments, and it let us plug existing HTML, JavaScript, and Ajax code into the Nokia project to create mobile applications.

We were able to move from concept to application in just five weeks—that includes testing, debugging, packaging, and deploying our widget.

—Jeff Litvack, AP's general manager of mobile and emerging products



## PRODUCTIVITY

### Société Générale Cuts Costs Amid Complexity

**M**y team maintains two large applications for Société Générale, a Euro zone financial services company with 80,000 employees. One performs management operations on securities, while the other helps administer mutual funds. Also, we're responsible for a J2EE and Oracle PL/SQL project for managing mutual fund services in Europe. With ever more on our plate, I set out to cut development costs and increase productivity.

The risks in maintaining finance apps have sizable business consequences. That's one reason I turned to a management platform, Cast, with automated application intelligence to control the quality of critical apps and increase productivity.

By reading, analyzing, and semantically understanding code—our architecture includes 4 million lines of code—the platform gives me visibility and control. Bottom line, we've improved productivity 25% when integrating new resources, because of simpler application architecture and better documentation that mean a much easier learning curve for new developers. Also, we've cut in half the number of hot patches, since developers have a much better understanding of the impact their changes have on other areas of the application.

—Jean-René Calais, Société Générale's application development manager

## APPLICATION LIFE CYCLE

### UPS Works To Get Better, One Step At A Time

**W**hen UPS sets out to improve its software development processes, it doesn't assemble a squad of "application life-cycle management" gurus. Lately, it has been taking people out of their day jobs and letting them figure out how to fix a piece of the process.

That approach does two things. One, it makes sure people are solving real-world problems. "Yesterday they were running a test organization, or an organization of business analysts," says Mark Hilbush, UPS's VP of IS. Two, it puts skin in the game. Says Hilbush, "They know they'll have to go back to their teams and consume it."

UPS treats application life-cycle management as something it needs to improve constantly, just like any step in its core package delivery business. Among the initiatives the company



has under way is one to create better requirements across its software-development projects. Get requirements and the architecture design wrong at the start of a project, says CIO Dave Barnes, and "as you go through the stages of application development, you just keep paying a price."

One way UPS is getting requirements right from the start is with visualization software from iRise, to the point that analysts must justify why they wouldn't do a simulation for any project involving a user interface. Three people are critical to getting requirements right up front, Hilbush says: the business analyst, the IT architect, and the business unit representative.

Another way is to make sure that

## OPEN SOURCE

### Why Reinvent? InfoPrint Integrates Free Code

**W**ith five operating systems to support and people spread across the United States, Europe, and Japan, it's a challenge to economically manage software development while meeting tight schedules. Developing all our code in-house certainly isn't practical. That's why InfoPrint has turned to a multisource development model, under which we use open source components where appropriate and integrate them with in-house code across our software projects.

Using open source is just one aspect that makes this component-based development approach work at InfoPrint, an IBM-Ricoh joint venture focused on

commercial printing. We also use agile techniques, for both integration with legacy platforms and creating new applications. By using appropriate open source components integrated with in-house code across our projects, we cut costs and work faster.

One warning for companies implementing this development strategy: It's critical that we know the origins of our code in the component approach, so that we don't leave the company open to unrecognized licensing, and so that we can ensure consistency. We use Black Duck Software's automatic code-scanning tools to look at every component for licenses, versioning, and other potential problems. We view the scanning process as a way to achieve standardization in code reuse.

—Mike Munger, InfoPrint's manager of component development



trio spends time on the most valuable requirements—the ones closest to UPS's business, like modeling a step in the package-sorting process to see how code could improve it, or understanding how a rep uses software when handling a customer complaint. To stay focused on the business-specific problems, UPS is working to standardize areas where it doesn't get advantage. So if a new mission-critical app must hit a certain performance level, the server, database, and configurations should follow the same blueprint that past critical apps used. "It lets my developers and architects focus on the business problems and not on how to design the next best high-availability infrastructure," Hilbush says.

In testing and quality assurance, UPS is creating a consistent process across all application teams—the same tools, methodologies, and best practices. The challenge is mostly cultural, getting people to change what they're doing pretty well today, Hilbush says. It ties into UPS's broader effort to design app dev best practices that can work across the various development methodologies—from waterfall to iterative to agile—that UPS uses. UPS doesn't want to be "religious" about any methodology, and instead wants to have teams considering the right approach for any given project.

Hilbush warns, however, against letting any of these efforts to improve application life-cycle processes turn into "long-running, amorphous activities." Instead, he says, treat them exactly like a project to develop a piece of code—provide a clear scope, the right people, a set time frame, and a measurable goal.

—Chris Murphy

(cjmurphy@techweb.com)

#### CONFIGURATION MANAGEMENT

### Nikon Teaches SCM New Tricks

At Nikon, we do what we call forensic source code analysis—review historical code changes to track how software evolves—using software configuration management tools. That approach lets developers learn from one another's coding techniques, comments, and changes.

But we're finding that SCM tools (we use Perforce) help in areas beyond conventional version tracking. As an international company, Nikon uses distributed development to increase productivity and parallel development to let us to work on multiple projects without collisions.

By letting developers work in these modes, and share and communicate about project resources, SCM tools improve transparency among teams. They help in code review sessions, build and release automation, and code reuse. And we've used SCM to develop a number of custom tools for automating processes, including automating daily build-and-smoke testing, delivery of documentation and schedules to our development wiki, and elements of project management.

By thinking of SCM as a medium for developers to share resources in a well-structured fashion, it's become a core part of our infrastructure, and we continually find new ways to use it. —Wilken Rivera, Nikon's software infrastructure administrator



#### AGILE DEVELOPMENT

### Idaho Makes Progress Despite Legacy Apps

The IT team at the Idaho Department of Health and Welfare knew its mainframe-based green-screen apps for processing applications for food stamps and other aid weren't ideal. The 22-year-old legacy system processed batches of eligibility applications overnight, and if an input error led to a rejection, the corrected application had to wait for the next batch process. It could take days or weeks to learn why someone wasn't getting benefits.

But a multiyear, multimillion-dollar overhaul was considered too risky. Previous attempts to replace the legacy system never got past user disagreements on requirements.

"We needed an agile approach, and that went against the grain" of how things were done in the past, says Randy Ashton, senior project manager at the department. In 2006, it settled on the Scrum method of agile development, using project tools from Danube, which lets developers start building before they have every requirement.

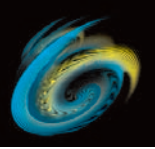
There were areas everyone agreed should improve, such as cutting delays in food stamp eligibility. Using Scrum principles, the division's IT team developed in two months a proof-of-concept Web app that gave an unofficial eligibility determination in real time. If the application included a clerical error, it could be caught before being sent for overnight batch processing.

Idaho's still chipping away at the legacy replacement project. But with its Scrum approach, it's delivering quick-hit apps that bring program improvements along the way.

—Marianne Kolbasuk McGee

(mcmgee@techweb.com)





## CUSTOM DEVELOPMENT

### Cancer Center Builds E-Records From Scratch

An extraordinary thing has happened at M.D. Anderson Cancer Center, one of the premier treatment and research facilities. What began as a skunk-works project about 10 years ago is now a custom-coded electronic patient record system accessed by up to 10,000 people at a time working on the cutting edge of cancer care.

It's not supposed to happen this way anymore. We buy packaged software, or at the very least we write gigantic requirements documents and ship them to outsourcers. But as the need for an electronic health data system became clear at M.D. Anderson, the IT team became convinced the software had to be built to fit the center's unique mission and how its clinicians and researchers work, says CIO Lynn Vogel.

Led by elite academic and treatment centers such as M.D. Anderson, which is affiliated with the University of Texas,

"the world of research and the world of clinical care are coming increasingly together," Vogel says. Buying packaged software for electronic records would have only helped with clinical care, Vogel says, and it wouldn't have been fine-tuned to the nuances the center brings to cancer care. Vogel and his team believed M.D. Anderson could spend about the same money developing its own software as buying it, and get exactly what it needed. Vogel credits the center's executives with seeing the opportunity. "It's a risky business," Vogel admits. "Executive management typically doesn't like to take risks with IT."

But it worked. The custom ClinicStation software has been used for about three years as the way clinicians access electronic patient data, accessing close to 50 data sources. The center has added integrated modules called ResearchStation to manage data used in research projects, and TissueStation,



Vogel's team builds instead of buys

which tracks the thousands of tissue samples stored around the facility and the research it's tied to, with links to the related patient's records.

Along the way, M.D. Anderson had to grow up as a software development organization. In the early years, the project that grew to become ClinicStation was informal, used by a small core for very specific needs, and people could get features added just by talking with Chuck Sutor, one of the champions of the project through its early years who continues as director of electronic records development and support.

"So you ended up with a set of very useful, very visionary products that

## REQUIREMENTS

### Delphi Engineers Use Single Repository

Requirements management discipline isn't just for managing code. At auto component maker Delphi, the strategy that developers use to get software right is the same one used to manage all the requirements an automaker gives the company to build a particular piece of a vehicle.

Delphi has created a single repository for all requirements of a given component, helping Delphi's 1,500 software, electrical, and other engineers discuss and comply with requirements, even when those people are spread around the world, including its 22 software development sites. The repository, based on IBM's Doors software, also is used for search—for example, to trace details about where within the software architecture and code a particular required function shows up.

"It's a fairly complex scheme to track requirements, where they were implemented, tested," says Cory Wentz,



Delphi's lead engineer on the project, and this system "provides the backbone for this." For a typical vehicle, Delphi can get about 300 different electronic documents, each with 20 to 30 pages, and the requirements Delphi needs to work may be spread throughout many places in the documents. Besides having search, a central repository lets Delphi engineers compare original specs against changes the customer made along the way.

It's critical work because if engineers misinterpret or overlook a requirement, or fail to incorporate new requirements that customers frequently add, Delphi must rework the design—which can burn a month or two, says Wentz, and add anywhere from tens to hundreds of thousands of dollars in new design costs, depending on the importance of the requirements. In an industry hit as hard as automotive, he says, "reducing engineering costs is critical for survival."

—Marianne Kolbasuk McGee



didn't have a terribly disciplined development strategy, documentation, and all those things that for creative, smart people are a pain in the neck," Vogel says.

One step in that maturation came when Microsoft ended support for Visual Basic 6, with which the software was written. The development team concluded it essentially had to rearchitect the project, and it had to admit that it wasn't up to that job. The center hired Microsoft specialist Avanade to rearchitect the project and to help the team move to a professional, documented development process.

That was a cultural shift for people used to the talk-to-Chuck-Suitor approach. Now there's a formal feature review process, an emphasis on efficiency and code reuse, and professional testing. Just executing a test to find if a new feature conflicts with existing ones can now take three days. "The real challenge was not unlike going from a startup to going to a commercial venture," Vogel says.

The development team has stuck to its reliance on doctors and researchers to drive development, however. "They're

the source for the good ideas for what the software should do," says Suitor. The team uses simulation software from iRise, for example, to show doctors new features before they're coded, and to communicate requirements to out-sourcers, which the center has increasingly relied upon in recent years.

The system's based on a service-oriented architecture, and this year brought the first major test of reusing the code. It involved giving read-only Web access of ClinicStation to referring physicians and patients. It meant reusing code for tasks like calling up test results data, but adapting the results to be presented as generic HTML.

Months into the project, hospital policy makers decided to put a seven-day delay before lab results showed up in the patient's view, so a doctor can discuss a result with a patient before the patient sees it. "That's where you're reusing, but also adding new variations at the same time," says Suitor. The team—and the code—were up to the task. The Web version was running in four months. —Chris Murphy

## AGILE DEVELOPMENT

### XDx Crafts A Platform To Track Clinical Tests

**X**Dx is a molecular diagnostics company on the cutting edge of genomics, focusing on discovery and development of noninvasive gene testing in the areas of transplant medicine and autoimmunity. We're one of the first to develop products from the Human Genome Project.

The IS team routinely had to build business applications to track and analyze tests for clinical studies that draw data from disparate sources. We needed a standard application. After evaluating the usual development platforms to build it on—.Net, PHP, and open source Java—we settled on OutSystems Agile Platform, our first big move to agile development.

We delivered our new Analysis Request Management System in six weeks—a company record. ARMS automates tracking of clinical trial samples and pathology requests for diverse studies, while managing all our studies' clinical data and sample workflows. We had three code sprints of a week or two each, plus a week of fine-tuning.

One of agile development's biggest advantages is the communication it demands between IT and business users. With agile, the most important features are developed first, followed by refinements. We built mock-ups, some with bare functionality, to get feedback and add value quickly. It engages business users in the development process—and ensures adoption of the applications.

—Jochen Scheel, XDx's director of software development, and Stefan Meier, IS associate director

## CODE REUSE

### Defense Department Tries Opening Up—A Bit

**F**or the Department of Defense, time has been a bottleneck in software development. Following government rules for procurement and testing, just getting the development environment in place for a project can stretch into months.

The DOD's IT group, the Defense Information Systems Agency, is trying to change that in part with its Forge.mil approach, modeled on open source code-sharing sites such as SourceForge. Forge.mil, though, will only hold approved open source code or code OK'd for sharing within the department. Forge.mil, based on CollabNet software, allows Net-centric collaborative development, and provides a destination for cross-program sharing and reuse of code. That should let developers get large projects developed, fielded, and tested faster. Launched last fall, it's now generally available for unclassified use, with 70 projects under way, for combat, business, and intelligence.

It's about more than delivering code "faster, better, cheaper," says Rob Vietmeyer, Forge.mil's project director. It's also about new ideas. The ability to tap into collaborative tools without a long acquisition process lets developers try "risky things" in development, he says. And if people don't like the capability, they can turn it off without a big loss of time or money. —Marianne Kolbasuk McGee