Industry:
Hardware/Software

Organization:
Ceridian

Description:
Ceridian is one of the top national human resources outsourcing companies in the U.S., offering a suite of innovatively managed business solutions for HRMS, payroll, tax filing, application outsourcing, time and attendance, benefits administration and employee effectiveness services.

Business Problem:
Ceridian was developing an enterprise application that integrated numerous new and existing systems. They needed effective tools for modeling, requirements management, defect tracking, and testing.

Rational Solution:
Rational ClearQuest, Rational RequisitePro, Rational Purify, Rational Rose, Rational Suite TestStudio

Key Benefits:
Completed a large, enterprise-wide development project – that included replacing five legacy systems, connecting several other major systems, and consolidating numerous legacy databases into a single, comprehensive customer database — ahead of schedule and under budget

Automatically generated nearly 750,000 lines of code – including VB, C++, SQL and XML/XSLT code – from visual models, representing 90% of all the code constructed for one project

Improved communication and minimized misunderstandings across a distributed development team through better defect tracking and use of a common modeling language

Rational and Ceridian

Ceridian Brings Enterprise Project in Under Budget and Ahead of Schedule with Rational Tools

Late, cancelled, over-budget. More often than not, one or more of these labels will ultimately apply to a large software development project. Anyone who has worked on such a project knows how difficult it can be to complete it on time. Numerous studies have found that the percentage of projects that are delivered on schedule is disappointingly low. The percentage of projects delivered ahead of schedule and under budget is, of course, much lower still. Yet, that is just what a small team of developers, testers and analysts from Ceridian did by using Rational® Rose® for modeling, Rational® RequisitePro® for requirements management, Rational® ClearQuest® for defect tracking, and Rational Suite® TestStudio® for testing. The project, named ResponsePlus.net, is one of the largest and most successful development efforts that Ceridian has ever undertaken. Requiring almost two full years to complete, the project included replacing five legacy systems, connecting several other major systems, and consolidating numerous legacy databases into a single, comprehensive customer database.

Ceridian is a leader in managed business solutions for human resources and employee effectiveness services. The ResponsePlus.net team was focused on Ceridian's payroll and human resources solutions. In addition to building internal applications like ResponsePlus.net, Ceridian provides a full suite of solutions for all phases of employment, from recruitment and applicant screening, to payroll processing, tax filing and compliance services, human resource management systems, employee self-service, time and labor management, and employee effectiveness solutions, to benefits administration and retirement plan services.

The Challenge

Bob Hughes, program manager of the ResponsePlus.net project for Ceridian, explains what motivated the project and what it entailed, "Ceridian is a distributed company — we're in more than two dozen different major metropolitan areas around the country. Each one of those offices has historically operated as an individual business. Our charter was to consolidate several of the databases together, replace several legacy systems, and create a new application that combined multiple systems together into one. It was really an almost complete redesign."

Because of the wide range of legacy systems and databases involved, the development team faced the additional challenge of mastering an equally wide range of technologies. Hughes notes, "Our group had a tremendous amount of learning to do. They had to deal with mainframes, Web development, SQL Server, XML, XSLT (Extensible Style Language Transformation). They also had to know COM+, Windows 2000, UNIX, message queuing, and all of the mainframe protocols that we needed for what we had to do." When all of the requirements and all of the required technologies were taken into consideration, it was clear to Hughes that the project was going to be a challenge. "We think anyone would call this an enterprise project. It cost several million dollars; it involved many distributed systems. If you look at the statistics, the track records for completing large projects like this on time are not very good."

Getting Started

When the project began, the ResponsePlus.net team started with a blank slate. Working from the broad project objectives, it was up to the team to determine what tech-

"We had a lot of expectations because of how big this project was. We knew we had to have automated ways of helping us manage different aspects of the project. We had experience with Rational RequisitePro and Rational Rose from projects past, so we recommended that we use those tools for this project. That's really where we began — selecting Rose as a design tool — but we got much more out of it as we moved on."

Andrew Prymak
Team Leader for
Development Group, Ceridian

nologies, tools and approach to use, and then to define more detailed project requirements. Hughes remembers, "The only thing that was somewhat a given is that all of our desktops are running Microsoft Windows. Other than that, whether this was going to be a Java application, a Microsoft application, whether we were going to use any modeling tools and do any sort of object-oriented or object-based development was basically given to the project teams."

Development teams throughout Ceridian have been using Rational tools for several years, and their success was a big factor in the team's decision to use those same tools on the ResponsePlus.net project. Ceridian developers have used Rational RequisitePro, a powerful, easy-to-use and integrated requirements management tool, for almost five years and Rational Rose, the award-winning model-driven development tool, for four years. Andrew Prymak, Team Lead for the development group and a principal designer of ResponsePlus.net explains, "We had a lot of expectations because of how big this project was. We knew we had to have automated ways of helping us manage different aspects of the project. We had experience with Rational RequisitePro and Rational Rose from projects past, so we recommended that we use those tools for this project. We knew that using Rose would be a good way to do our design. It would be one place to house our documentation. That's really where we began — selecting Rose as a design tool — but we got much more out of it as we moved on."

Before any design could begin, the team needed a much clearer picture of the project's requirements. For ResponsePlus.net, the team decided that creating and evolving use cases in Rational RequisitePro would be an optimal way to track requirements. Prymak continues, "We also wanted to manage our use cases. Since we had experience with Rational RequisitePro and there are so many requirements for this huge project we decided to use this document repository for all of our use cases. It worked out very well."

The Code Generation Revelation

Soon after the team began building the application models for ResponsePlus.net in Rational Rose, they came to a realization that they would shave months of development time off the project. The lead developers saw that they could create not just their application models in Rational Rose but also their data models; and from these models they could automatically generate much of the source code and data tables for the entire project.

Prymak explains, "When we started out, we asked ourselves, 'Well, how are we going to do this? What kind of objects are we going to have?' We sat down and started to do object-oriented design the way that we had on past client-server projects. We created some sequence diagrams, and what we really fell upon was, 'Hey, it would be great if we did data modeling in Rational Rose too,' because our transfer method for passing data is XML-based.

We had to figure out the complex parent-child relationships of all this data that was going back and forth and we really needed to model it. For instance, a payroll deduction can have other data elements that are its children. It was a very hierarchical structure that could be many layers deep, and we wanted to create these layers as different reusable data objects. The best way to do this was to model these as different classes in Rational Rose and show those relationships and hierarchies. We started with what we logically wanted in the XML documents, put those in classes, and that helped us determine what database structure we wanted. Once we started doing this we said, 'Wow, we can do our database designing in Rose as well.'"

Tim Carroll, one of two System Architects for ResponsePlus.net, soon saw the potential for generating code from the extensive models that were being developed. Prymak continues, "That's when Tim stepped in and said, 'Well, if we have it all here in the model, why don't we generate the code?'"

Although Rational Rose supports forward-engineering code from visual models, Carroll opted to create custom components that used the Rational Rose Extensibility Interface to access the ResponsePlus.net models. The Rational Rose Extensibility Interface is comprised of the Rational Rose API (application programming interface) and Rational Rose

Scripting, which allows developers to use RoseApp object to automate manual functions within Rational Rose, create versions of Rational Rose that are specific to particular problem domain, and integrate Rational Rose with other software applications. Carroll decided to use the Rational Rose Extensibility Interface because it offered more flexibility and enabled him to create Visual Basic components that generated very specific and tailored code for each object. Carroll notes, “I chose to go ahead and write our own components using the Rational Rose Extensibility Interface. The extensibility interface is, I think, very good. It’s very well documented and it has everything in there. With the RoseApp object, I was able to go in and find out everything I needed to know about what each XML document that we were going to be generating needed to look like. I could find out what tables it related to, if I needed to be able to join tables together, what fields we needed to use for key values to update — all kinds of things like that.”

For some of the objects, 100% of the code was automatically generated from the model – requiring no hand coding at all. Once generated, all the team needed to do was compile it. And when the model changed, code could be re-generated immediately. Carroll notes, "We had one object which had over 100 different columns; keeping this object in sync would have been a nightmare. Any time somebody changed a column somewhere, we would have had to make sure we made that change exactly the same in the code and everywhere else. I don’t think you can even estimate how much time that saved."

Saving Time and Simplifying Maintenance

The ability to generate code automatically from the model yielded benefits immediately. Hughes reports, “In the first three months we generated code for 350 objects with four stored procedures each, like add, delete, update, and XML documents for the data access — all of it. And it was all consistent, all perfect names everywhere, all in sync. If you had a whole team doing it, things might be named incorrectly. Since we’re pulling it all from schemas there is no problems like that. And, it’s all generated in the same way.” There were other benefits as well, “Our approach forced standardization to the design, so everything that we did was consistent. Because the code was all being generated, we knew it was consistent and it was going to behave the same way, which would help with maintenance down the road. Even our stored procedures to retrieve the data were generated so the table joins were orderly and efficient. This virtually eliminated the possibility of getting database deadlocks — something that could potentially cripple an application or require extensive performance tuning. This was something our DBAs were very happy about. The benefits were many, but in terms of the amount of work and the time savings alone that we got from generating all the code — there’s no way we would have met our project dates at all if we had to hand code it.”

Prymak adds, “We’d go back to the model, make any changes necessary and regenerate. We’d regenerate these things several times, when any error was found. It forced us to follow best practices in terms of going back to design, making sure that everything was properly documented and fit together from a logical, business standpoint.”

In the end, almost 750,000 lines of code — including VB, C++, SQL and XML/XSLT code — were generated from the models in Rational Rose, representing 90% of all the code constructed for ResponsePlus.net. According to Carroll, it was Rational Rose’s well-documented extensibility interface that was key to the code generation, and in turn, to the success of the entire project. “Rational Rose’s extensibility was very good. I couldn’t ask for anything more than what Rational Rose had. It was very easy to use and I was able to get at everything — nothing was hidden. That’s what all made it possible. Without that we couldn’t have done generation.”

Improving Communication with Rational ClearQuest

Though the ResponsePlus.net development team was relatively small — approximately 16 developers, four analysts and three testers — they were not all in one location. With team members in Colorado, New Jersey, Georgia and Minnesota, effective communication was crucial to the project’s success. The team

"We had one object which had over 100 different columns; keeping this object in sync would have been a nightmare. Any time somebody changed a column somewhere, we would have had to make sure we made that change exactly the same in the code and everywhere else. I don’t think you can even estimate how much time that saved."

Tim Carrol
System Architect, Ceridian

relied on Rational Rose and Rational ClearQuest to keep communication channels open and to minimize misunderstandings.

Rational Rose unified the software development efforts of Ceridian's distributed team through modeling based on the Unified Modeling Language (UML), the standard notation for software architecture. And Rational ClearQuest facilitated improved communication by providing a comprehensive solution for managing all change requests, including enhancements, defects and other changes.

Hughes adds that Rational ClearQuest also simplified overall project management, "Rational ClearQuest improved communication as well as work flow — from developer back to analyst to verify it, to QA to test it and so on, until a change was concluded by putting into a build. Any developer at any of our sites could go in and see which ClearQuest items were assigned to them without having someone tell them what their workload was. Managing all those change requests for a distributed development team — knowing where each ClearQuest item was, whether it was in the hands of a developer, in QA, and when it was included in a build — was a key benefit. This project had over 125 builds in its last year and ClearQuest provided a complete history of each build's contents."

Bringing It All Together

One of the most important advantages the ResponsePlus.net team gained from using Rational solutions was the ability to link the project's requirements, design, test cases, and change requests together. The integration between various tools helped the team visualize the impact of changes, and also enabled better task planning to ensure that all requirements were addressed by both the design and by testing.

Carisa McCann, Team Lead for the Quality Assurance Group recalls, "We were able to establish traceability by setting up links between our use cases in Rational RequisitePro and the test cases. This was really great for us from a testing perspective. With relatively few testers trying to keep up with the pace of development, we didn't have a lot of time to spend trying to figure out what test cases needed to be updated due to design changes. These links were really helpful for us because they made it easy to see what had changed and automatically identified which test cases needed to be updated." The testing team used Rational TestManager to manage test assets like automated scripts and data files, to create data pools for automated regression testing with Rational Robot, and to link requirements with test cases. Along the way, developers also used another testing tool, Rational Purify, to quickly pinpoint memory leaks in C++ code. All three testing tools — Rational TestManager, Rational Robot and Rational Purify — are integral components of Rational Suite TestStudio, as are Rational ClearQuest and Rational RequisitePro.

Hughes agrees that the ability to trace requirements through development and testing was a substantial benefit, "Both Rational RequisitePro and Rational ClearQuest help to enforce best practices in managing design changes. In RequisitePro the requirements changes were documented, prioritized and linked to test cases for traceability, which allowed us to easily identify and analyze the impact those changes would have downstream."

Small Team, Big Results

A small group delivered a huge project ahead of schedule and well within budget. The challenges faced by the ResponsePlus.net team were complex, but the recipe for the team's success was not. They followed best practices of software engineering: effectively managing requirements and change, using component architectures, modeling extensively, testing early and often, and communicating clearly. And they used Rational tools designed and built to help teams follow these best practices.

Hughes concludes, "I think the group is quite proud of the fact that considering the size of the project and the magnitude of its impact to the business, it was a very small development group. Rational enabled this small group to complete a huge project ahead of schedule. Plus, we've established a platform to repeat this success as Ceridian moves forward."

About Rational Software Corporation:

Rational Software provides a software development platform that improves the speed, quality, and predictability of software projects. This integrated, full life-cycle solution combines software engineering best practices, market-leading tools, and professional services. Ninety-six of the Fortune 100 rely on Rational tools and services to build better software, faster. This open platform is extended by partners who provide more than 500 complementary products and services. Founded in 1981, Rational is one of the world's largest software companies, with revenues of $796.7 million in its twelve months ended September 30, 2001, and over 3,800 employees worldwide. Rational is a component of the Nasdaq-100 Index®. Additional information is available on the Internet at www.rational.com.

Additional Rational success stories and video clips are available at **www.rational.com/success**.

Rational Software
Dual Headquarters

18880 Homestead Road
Cupertino, CA 95014

20 Maguire Road
Lexington, MA 02421

Toll-free: (800) 728-1212
e-mail: info@rational.com
Web: www.rational.com

International Locations:
www.rational.com/worldwide

CS-573; 6/02. Subject to change without notice.