Industry:
Hardware/Software

Organization:
Salion, Inc.

Description:
Salion is an enterprise software company dedicated exclusively to the critical business needs of suppliers. Salion's software helps suppliers sell more, and sell more profitably. By optimizing the Revenue Acquisition process – everything from the beginning of the sales cycle through to the end, including all the departments and all the people involved in it – Salion enables suppliers to target the best business, win more of the time, and deliver increased customer satisfaction.

Business Problem:
Salion needed to develop their innovative Revenue Acquisition solutions rapidly while ensuring that the solutions they built would be scalable, extensible and maintainable for the long run.

Rational Solution:
Rational RequisitePro, Rational Rose, Rational SoDA, Rational Unified Process

Key Benefits:
Increased software development speed by up to 400% by adopting a complete solution including the Rational Unified Process and Rational tools to support it

Consistently met release deadlines without sacrificing features or quality

Streamlined due diligence procedures for both customers and investors by demonstrating a solid, proven development process

Designed and developed highly scalable, extensible and maintainable software systems in less time

Rational. software

Salion™ Uses Rational Unified Process to Succeed Today, Prepare for Tomorrow

Virtually every software developer is familiar with Meskimen's Law, "There's never time to do it right, but there's always time to do it over." Even if they don't know it by name, most developers recognize the sentiment expressed by this tongue-in-cheek adage. But the development team at Salion, Inc. believes that Meskimen was a bit of an optimist. As developers at a startup, the Salion team knows there is little tolerance for "doing it over" — investors want to see sustained progress, and customers want high-quality software on time. To surpass the expectations of both, Salion needed to develop their innovative Revenue Acquisition solutions rapidly while ensuring that the solutions they built would be scalable, extensible and maintainable for the long run. With no time to make mistakes, Salion adopted the Rational Unified Process® or RUP® and immediately started using Rational tools to support it. Since then they have not missed a single release deadline and each release has included every bit of functionality planned for it. According to Ross Buhrdorf, Salion's Vice President of Engineering, RUP and Rational® tools have enabled Salion's team to develop software up to 400% faster. This remarkable record of success has helped Salion quickly become a leader in Revenue Acquisition process optimization solutions.

With offices in Austin, Texas and Detroit, Michigan, Salion is an enterprise software company dedicated exclusively to the critical business needs of suppliers. Salion's software helps suppliers sell more, and sell more profitably. By optimizing the Revenue Acquisition process – everything from the beginning of the sales cycle through to the end, including all the departments and all the people involved in it – Salion enables suppliers to target the best business, win more of the time, and deliver increased customer satisfaction.

Salion's leading process management, data analysis and collaborative communications technologies come with a commitment to both rapid implementation and high return on investment.

Buhrdorf and Senior Architect Dale Churchett have been with Salion from its earliest days. They came to Salion with an understanding of the impact that rapid growth can have on software systems and the need for a comprehensive solution including both tools and process. Buhrdorf explains, "At our last company we delivered more than 30 products during my four years there. There was mass chaos in the beginning. We started out at 54 thousand page views a day on our Web applications and ended up with 45 million page views a day. It was massive growth. When I came to Salion, Dale was the first engineer I brought over. We both have a strong software background; we've both been through startups. After seeing all the confusion at the place we came from, we started looking at best practices and tools that were mature and were continuing to evolve. Our philosophy from day one is that we have to move fast, we have to move quickly, we really don't have time to make easy mistakes – easy mistakes are mistakes in process. Software development process is something that smart people have thought about and documented. From that point of view, it was easy to see that we also needed tools to support the process. We evaluated the options and chose Rational and the Rational Unified Process because the Rational Unified Process is part of a suite – it is integrated. It is a complete solution."

Buhrdorf knows that Salion is not a typical startup. Every startup is eager to show progress as quickly as possible. But, while many startups try to save time by skipping over process altogether, Salion has saved

IBM®

"We've hit all of our dates, with all of the functionality. It has been a real success. We've been able to adapt to changing requirements. Using RUP paid off immediately."

time and improved their products consistently by adopting the Rational Unified Process early on. Buhrdorf continues, "We are a startup. This kind of thinking is not typical of a startup. I think that is because startups have this belief that they have to start writing code on day one. The truth of the matter is that process is something that should be simple. If you're going to scale, and you're really going to make changes quickly and be agile, then your process has to be solid to support that. We have been very effective. We're on our twelfth development iteration for our product; we've hit all of our dates, with all of the functionality. It has been a real success. We've been able to adapt to changing requirements. Using RUP paid off immediately. When investors perform due diligence on us for funding we can return to our process and demonstrate that we really have it wired with respect to development. So in addition to the cost savings just in implementation RUP also returns dollars that way."

As a startup, Salion depends on funding from investors during their early stages. Buhrdorf believes that Rational has been a big help in this crucial aspect of the business. "We were able to show our investors that our process was based on RUP and that all modifications were documented. Dale would present the UML models where they could see everything mapped from the business model through to the product models. And, when we'd go out and perform technical due diligence for our customers, we would lay out all the process documentation and it absolutely became a no-brainer that we had a mature development model," Buhrdorf explains.

A Quick Review and A Quick Start

The Rational Unified Process is a software engineering process that enhances team productivity and delivers software best practices via guidelines, templates, and tool guidance for all critical software development activities. From a management perspective, the software lifecycle of the Rational Unified Process consists of four sequential phases, each concluded by a major milestone. The phases are Inception, Elaboration, Construction and Transition, which correspond roughly to requirements analysis, design, implementation and release, respectively.

Churchett was surprised how quickly Salion progressed through each phase iteration, and by how easy it was to adopt the Rational Unified Process. "The amount of time it took to Transition to the first baseline – and get a version of the product that everyone wanted — was pretty amazing. We basically did staffing, performed due diligence, co-location, investigation, product infrastructure – all the things you need to do up front. At the same time, we developed the product and we did it all in an amazingly short timeframe," Churchett reports.

Buhrdorf adds, "In eight months we had our first usable product out. If we would have had to figure out how the product managers work with engineers, how the architects work with the component developers, how the UI team works together, and so on, that would have held us up. We did not have to deal with that. We just said this is the way we are doing it. Here are the tools we're using. If you have any questions, it is all documented."

Buhrdorf explains that with the Rational Unified Process in place, the project began moving rapidly right from the start. "We went through Inception and Elaboration in the first few months and we did a baseline in the third month. We did our second baseline the next month along with a Transition to deliver product. Our iterations have been typically about 30 days. We had this philosophy based on our experience that you have to Transition as soon as possible so you can work out the kinks. We transitioned an internal customer first and deployed at our hosting facility. Then every month since, when we do another iteration on Construction we automatically do a Transition. We configured the Rational Unified Process to combine our Construction and Transition phases into a single iteration."

Scaling Simply Means Getting Bigger Hardware

Spending time up front to address architectural risk and scalability is a key concept in the Rational Unified Process. For Salion this step was critical to their success. At their previous company, Buhrdorf and Churchett saw first-hand how hard it was to try and scale systems that were not well-architected. Buhrdorf notes, "The big thing – which I think is a failure in

most startups – is that when we deployed at Transition, our software was fully distributed. The entire scaling strategy was done, and everything was deployed on its target platform. There were no short cuts taken in that Transition. We knew we had to build a solution that we could scale through hardware and not through refactoring software." Churchett adds, "Our decisions were driven by our experience with scaling to large systems. We saw what happened if you didn't consider scalability from day one. We had already seen how difficult it could be to port from SQL Server to Oracle, and we didn't want to go down that route. That meant we had to have the smallest Sun servers you could get originally. But now scaling is really just a case of getting a bigger piece of hardware – we're not going to have to do a port."

A Small Tweak for a Useful Technique

The Rational Unified Process is designed to be flexible. Users are free to adapt and configure in any way that works best for them. For example, in the Rational Unified Process, inspection is typically performed before the Construction phase is complete. Inspection is a formal evaluation technique in which artifacts – for example, models, documents, and software — are examined by a person or group other than the originator, to detect faults, violations of development standards, and other problems. The team at Salion has found that, for them, inspection is more useful after Construction, and they have modified the Rational Unified Process to reflect that. Buhrdorf explains, "We don't do code reviews before the Construction release. Instead we do inspection after the release, and then we apply refactoring to the next construction release. We found that this is far more useful for us because the person that did the development can go back and say 'I wish I could do this differently now that we've done it.' "

Refactoring is a technique that is used to reorganize and simplify code by eliminating redundancy. Some processes recommend refactoring continuously throughout development. Salion decided it was better to build the system, get it in use, take time to think about it, and then inspect it. Churchett found that one of the best ways to inspect code is to reverse-engineer it using Rational Rose® and look at the dependencies in the visual models. Churchett notes that the visual models make it easier to recognize the opportunities for refactoring, "When I reverse-engineer the code, I know where sub-optimal implementations exist because obvious errors jump right out at you."

Salion uses Rational Rose – the world's leading model-driven development tool – to architect all components of their applications using a common notation, the Unified Modeling Language (UML). Buhrdorf explains, "When Dale goes back and inspects the code, he uses Rational Rose to create a list of package dependencies and inspects the interfaces. Then he makes a list of recommendations, which is far simpler and far more effective than the time-consuming process of doing pre-inspection." Churchett agrees, "With Rational Rose, I get visibility across the entire system. I basically live out of Rose these days."

Linking Requirements in Rational RequisitePro with Use Cases in Rational Rose

Rational Rose is just one of the tools that Salion uses along with the Rational Unified Process. They also use Rational RequisitePro® to capture and manage project requirements throughout the development lifecycle. "We use Rational RequisitePro heavily for getting input and requirements from our product marketing and following them through to implementation. The other tools we considered had no support for that because they were not well integrated with other development tools," Buhrdorf says.

The requirements actually are derived from use case diagrams that Churchett models using Rational Rose. With the industry's most comprehensive support for use cases, Rational Rose and Rational RequisitePro help Salion design and deploy software with the focus squarely on end user needs. Churchett is impressed with the integration between Rational Rose and Rational RequisitePro, "We do all our requirements capture in Rational RequisitePro. Typically, I'll create the use case models first in Rational Rose and then immediately create them in our Rational RequisitePro project. I literally just right-click and RequisitePro pops up. That is beautiful because Product Management can do a query in RequisitePro and see immediately how many use cases there

"With Rational Rose, I get visibility across the entire system. I basically live out of Rose these days."

"It's been great — we haven't had any issues with capacity planning. RequisitePro helps us develop solutions on time so we meet customer and business expectations."

"The QA staff is building system tests based on the use cases. They perform a query in Rational RequisitePro to obtain the use cases for a particular release, and then develop test cases from those use cases. The quality is awesome from that. We have zero quality issues. We just don't have them."

are in any release. Or we can use RequisitePro to go through and see the use cases that we either missed or that were deprecated."

At a higher level, Buhrdorf relies on Rational RequisitePro for scope management – tracking historical trends in use cases to improve planning and scheduling. He explains, "We also keep track of productivity via use cases. We can do forward scheduling by looking backwards. For example, we know from our past 12 iterations how many use cases were done per staff member. We use that to schedule future projects. If product management asks us to do 200 use cases in the next release and we've been doing 100, then we immediately know there is a problem. It's been great — we haven't had any issues with capacity planning. RequisitePro helps us develop solutions on time so we meet customer and business expectations."

Managing Complexity

For Churchett, Rational Rose has simply become indispensable. "I don't see how I could manage the complexity without Rational Rose. It has been a huge win. The thing that blew me away about Rose and UML originally, was just how simple it was. How easy it was to group packages of use cases together. If you do them right, they generally match the subsystems and components. If something isn't right, it just looks wrong and jumps out at you as wrong. As our software has gotten more complex, being able to share those models is saving a lot of time. The analysis and the domain modeling helped a great deal too. When I look at the API, I expect it to mirror the domain model, and if it doesn't I know sooner or later it is going to come back and bite us. So I show the developers the domain model, and ask them 'Does your code look like this?' and that saves us pain moving forward."

Churchett continues, "Managing the complexity is a big deal. Being able to just put a UML diagram in front of someone and just have them understand it is great. Right now, we've got 400 or 500 use case diagrams in the system. When we introduce new functionality, I need to know what the impact of that is and if we're duplicating existing functionality. I am also able to quickly identify use-case packages that can be reused in other subsystems or products. I couldn't do that trolling through code, and I don't know how you'd do it without Rose. I guess you'd end up making poor or uninformed decisions because you wouldn't see the system clearly. Without Rational Rose, incremental additions would've been a nightmare." Buhrdorf concludes, "This comes back to scalability. By design you want everything to go through your Senior Architect. He's got to be able to understand it all. There's no way he could understand it all without Rational Rose and Rational RequisitePro."

Having current and comprehensive documentation is exceptionally helpful in managing complexity. Churchett uses Rational SoDA® to automate the generation and maintenance of thorough, up-to-date project documentation, reports, and vital project information. He notes, "I use Rational SoDA primarily for formal documents. For instance we recently completed our Build Process Customization Strategy – which allows us to support multiple customers from a single source code base – and we modeled it to business use cases. I just programmed a SoDA template to produce that. I also scripted a version to do a 4+1 views of the entire system which includes everything that I think should be in there – the business model, the logical domain. It's very concise. It is really the blueprint of the system and its all done in SoDA." The 4+1 model describes software architecture using concurrent views, including a logical view, a process view, a physical view and a development view. Buhrdorf interjects, "The 4+1 views are awesome. We can go to that any time and see exactly where we're at."

Building Scalable Teams, Improving Quality, Filtering New Hires

The Rational Unified Process and its supporting tools have provided Salion with a number of more subtle benefits, in addition to helping them rapidly deliver a superior application.

First, as Salion grows, the Rational Unified Process is helping everyone stay on the same page. With a development team of fewer than 20 people, Buhrdorf and Churchett have found that RUP can benefit relatively small teams, especially those with plans for

continued growth. Churchett notes that RUP is helping Salion build not just scalable systems, but also a scalable development team. "At one point, we modified our collaboration process to try to make it easier to work with Product Management because I was spending a lot of my time asking them questions on the functional specs. We changed RUP and combined some documents and deleted others. It worked with the initial team, but as the staff grew we had to relearn how to work with each other. It became very difficult to focus on what the deliverables were, what it meant to be in Inception," Churchett remembers.

Buhrdorf agrees that some RUP modifications were not worth the effort because RUP was better documented, "We've had the experience that there are side effects to RUP modifications that just don't scale as your team grows. Although we write down changes to our process, when we bring on new people not everyone reads it or understands it. Sometimes we find its better to just stick with RUP, as it is delivered from Rational, on the simple things because it is so well documented and there is a tool to support it. We get better scalability out of it. We're becoming very critical about side effects that can occur, and we are sticking to standard RUP for more things. We are eliminating risks as a result."

Quality Assurance has proven to be just as big a success. As part of the Rational Unified Process, quality assurance tests are started as early as possible in the development lifecycle. Buhrdorf continues, "While the construction phase is going on, the QA staff is building system tests based on the use cases. They perform a query in Rational RequisitePro to obtain the use cases for a particular release, and then develop test cases from those use cases. The quality is awesome from that. We have zero quality issues. We just don't have them."

Lastly, Churchett believes that the Rational Unified Process has improved Salion's ability to employ the best and the brightest. "RUP has also allowed us to hire some good people. We can filter out people very quickly when we show them what we're doing. It is easy to spot the developers that can't handle process and those that are interested in it at the initial interview. As soon as we start talking about unit tests and use cases, the candidates we ended up hiring showed an immediate and real interest in working that way."

The Big Benefits

Happy investors. Satisfied customers. Delivering fully functional products, with zero quality issues, on time, every time. What else is there? For Buhrdorf the most important benefit is predictability. "I think predictability is the compelling argument for any software process. The bottom-line is really predictable results, and the Rational tools and the Rational Unified Process deliver that. In addition to that, the secondary benefits for us are scalability, agility, and quality. We have the tools in place to support our process and our architecture. We take this for granted now. I don't know how other companies can do it without Rational. I believe that the Rational Unified Process made us four times as fast – a fourfold increase. There are so many other things that can go wrong we decided the process should be right. There is so much other risk in software development why would you want to assume another one?"

When Meskimen's Law stops working, and there really is no more time to do it over, there is an alternative. Buhrdorf concludes, "If it's going to be harder in the long run, then we'll take the time to do it right the first time." Call it the Salion Principle.

About Rational

Rational 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.

IBM Rational software
Dual Headquarters

18880 Homestead Road
Cupertino, CA 95014

20 Maguire Road
Lexington, MA 02421

Toll-free: (800) 728-1212
Web: www.ibm.com/rational

CS569A 5/03. Subject to change without notice.