IBM Rational Unified Process and IBM Rational Rose Enterprise Help PMI Manage Outsourced Development Projects

Overview

- ***The Challenge***
  The PMI Group, Inc. decided to replace its legacy mainframe system with a more flexible, extendable and maintainable J2EE-based system. The company needed an effective way to manage an outsourced development effort that spanned multiple countries and comprised three major development initiatives — each with two separate development projects. Specifically, PMI needed to define standards, convince the outsourcing vendor to follow those standards, and establish clear lines of communication between the two organizations.

- ***The Solution***
  PMI adopted IBM® Rational Unified Process® to provide a consistent, proven methodology for the entire development effort. In addition, project standardization using IBM Rational Rose® and the Unified Modeling Language (UML) helped the entire distributed team — including PMI business analysts and as offshore development groups — clearly communicate designs and specifications.

- ***The Benefit***
  More than sixteen months into the project, PMI is consistently reaching milestones on schedule and the projects are appreciably under budget. Rational Rose Unified Process and Rational Rose have contributed to improvements in productivity, quality, consistency, predictability and risk management throughout the project.

The PMI Group, Inc. is an international provider of credit enhancement products that promotes homeownership and facilitates mortgage transactions in the capital markets. PMI offers residential mortgage insurance and credit enhancement products, lender services and financial guaranty reinsurance. Through its subsidiaries, PMI is one of the largest private mortgage insurers in the United States, Australia, New Zealand, and the European Union, as well as the largest mortgage guaranty reinsurer in Hong Kong.

Recently, PMI found that its existing legacy mainframe system was beginning to limit the company's ability to respond quickly to evolving business needs. Chris Hovey, Associate Vice President of Systems Development and Integration at PMI, explains, "PMI has begun a multi-year effort to reengineer and replace its legacy mainframe systems with a contemporary, modular J2EE™-based system. We had achieved the limits of what was possible, from a standpoint of functionality, with our mainframe implementation. When the mainframe system was first built, it was designed to support a mono-line mortgage insurance portfolio. Since then, our business has become more complex and PMI has successfully evolved into a multi-line international provider of credit risk enhancement. However, our legacy mainframe applications could not support these new business operations. Therefore, to enable the firm's continued growth, PMI decided to rebuild these key systems."

Aside from the business needs, there were technical challenges associated with the mainframe system as well. "As is the case with many legacy systems, our systems had grown organically over a long period of time and were not very well documented. The systems group found that maintaining them was becoming more cumbersome and burdensome. We wanted to architect and engineer the new systems with a component-based, service-oriented framework using modern tools and modern approaches to create a more robust solution," says Hovey.

PMI began a development initiative, called pmiCentral, to address these challenges. The pmiCentral project includes wholesale replacements of some systems and an initiative to extend the useful life of the legacy system. The pmiCentral system replaces PMI's claims and delinquency management systems, while the project to extend the legacy system's useful life is focused on the policy servicing, billings and earnings systems. The final part of pmiCentral requires an entirely new system to achieve compliance with the Sarbanes-Oxley Act for a line of business that had been relying on Excel spreadsheets — an approach no longer feasible given new government regulations.

Adopting RUP

Hovey's responsibilities included managing the entire pmiCentral project and ensuring all requirements were fulfilled to ultimately deliver substantial business value. In the past, PMI had used a waterfall-based methodology, in which each phase of development — including analysis, design, coding, and testing — starts only after the previous phase has been completed. Hovey notes, "In previous projects, PMI enforced a quasi-waterfall methodology.
We have used various outsourcing vendors, who usually wanted to follow the waterfall methodology as well. In my experience, the waterfall methodology is inefficient because it tends to delay the availability of inspectable results until late — often too late — in the project lifecycle. I prefer an iterative methodology that embraces key principles like actively managing risk, inspecting results early and regularly, focusing on quality, and modeling visually, among others."

When PMI started the pmiCentral project, it became clear that the complexity and scale of the project would require a new approach for PMI and its vendors. PMI selected the IBM Rational Unified Process, or RUP, to establish a uniform, consistent process foundation for the entire project. Hovey explains, "pmiCentral required hundreds of individuals on two continents to work around the clock on 20 subprojects that were all part of a single master project. I wanted all of this effort to conform to a single set of principles and practices so that they could all produce results in a uniform way, and report their status and progress in a uniform way. Since RUP has already formalized these key softw-are development principles and practices, it was obvious that RUP was the best solution for us. It was easy to convince management that we needed to really embrace RUP to drive this project because of the difficulties we had in the past with waterfall-based approaches."

While establishing a consistent process was a primary factor in the decision to adopt RUP, the best practices promoted by RUP were also seen as providing significant advantages to PMI. "Many of the principles of RUP are absolutely essential success factors. In addition to continuously verifying quality and visual modeling, I believe that working with component-based architectures — and developing iteratively to produce inspectable results and mitigate risk — are critical for understanding how the project will succeed. RUP was a natural match for us," says Hovey.

PMI also considered other iterative development methodologies, but decided that RUP was a much better fit for its enterprise-scale project. "The other nontraditional methodologies were, I felt, tailored towards much smaller projects. While they were likely going to get the right results from an iterative development or testing standpoint, they did not have the robustness and scale that were matched to the size of this project. Whereas the traditional waterfall model would have that kind of robustness to scale, it really didn't have the kind of continuous production of inspectable results that we were looking for. It also had no emphasis on architecture, which is essential. Again, RUP was right in line with what we were looking for."

Getting the Systems Integrator on Board

While deciding to adopt RUP was fairly easy for PMI, convincing the outsourcing vendors to use the RUP methodology was much more challenging. Hovey notes, "In general, the development model for outsourcing vendors aims to define and validate all the requirements at one point in time. And then, in order to get the cost benefits of outsourcing, they send the requirements and specifications to a lower cost resource pool, produce the code over a long period of time, and then bring it back and try to get it to work. Their entire business model is essentially based on a waterfall approach, and getting them to change it was very difficult."

Once PMI's outsourcing vendors agreed to follow RUP, they ran into another challenge. "Despite the fact that they essentially only used the waterfall model, they believed they already knew everything about RUP. But what they thought RUP was, and what we wanted, were not the same. We needed to ensure that they weren't just paying lip service to RUP, but they really understood what we were trying to do and would embrace it. We had to conduct a number of meetings to work with their team in a partnership and in a collaborative manner to establish a framework — essentially RUP guidelines for this project. We established a baseline for what we all meant by developing iteratively, for example, and for visual modeling with UML and so on. Once that was done, it became a lot easier and smoother to move forward."

After agreeing upon specific guidelines for artifacts such as use case specifications, Hovey and his team codified those guidelines by customizing RUP to their specific needs. He recalls, "Early on, we found it important to agree upon what a use case is. The outsourcing vendors asserted that use cases were essentially system flows, such as a log-in screen. However, PMI defines a use case primarily as a distinct business process flow. To resolve this, we convened all of the key stakeholders from both groups to establish a guideline on what a use case is. After that, we said, 'This is what a use case is. We all agree. Let's go forward.' If we hadn't done that, we would have had too many different opinions of what constitutes a use case, which would have created confusion and disruption, especially for our counterparts in the business. We repeated that same exercise on a number of other key issues, ultimately creating a set of guidelines that was signed by executives from both PMI and the vendor."

The Value of UML

As with RUP, PMI also had to ensure that the outsourcing vendor was using IBM Rational Rose Enterprise and UML to model in a way that facilitated efficient and unambiguous communication between the two organizations and across all development locations. Hovey reports, "Another good example of reaching a common understanding was the use of UML. Very early on, we decided to use UML as a standard, common language between teams. At one point, one of the vendor's architects produced an architectural diagram in which the modeling was based on the particular stencil and shape preferences in the architect's diagramming software. We explained that it is important and necessary to use UML because the modeling elements are standardized and have well-defined meanings. That meaning is critically important, because it enables us to avoid creating a dictionary to go along with the models to explain what these elements mean — that has already been invented for us. "

While the outsourcing vendor did have experience using UML for sequence diagrams, they had little experience using it to express higher-level architectural diagrams, use case realizations or activity diagrams. Defining requirements in terms of use case models within Rational Rose was a key to clearly communicating all of the projects requirements to everyone on the distributed team. Bob Jensen, Enterprise Architect for PMI explains, "Getting everyone to understand and utilize use cases was a real advantage to us because we were able to transform primarily text-based requirements into a visual model. We were able to establish a common understanding — for the business, the analysts and the designers — of what the UML elements meant. That gave us a level of precision and completeness that enabled us to overcome some of the challenges we had with offshore development prior to using UML. It helped offshore developers better understand the architecture and it helped decrease rework and increase productivity as a result."

Hovey adds, "To enforce our UML guidelines, we made UML 2.0 compliance part of the acceptance criteria defined in our contracts. Any diagram that was not in compliance with UML 2.0 would not be accepted. Such contract language was key to deriving proper behavior from the vendor."

Metrics Help in Managing Outsourced Development

Once the process and modeling guidelines had been defined and agreed upon, the outsourcing vendor took over the entire development effort. The outsourcing vendor conducted meetings with PMI business teams to gather and define requirements. The vendor was also responsible for the design, implementation, integration, and verification of all of the pmiCentral systems. PMI project managers monitor and track progress and direct user acceptance testing. Hovey notes, "We laid out the framework. That starts with identifying the goals of the system and goes forward through requirements specification, analysis, implementation, and test across an iterative schedule. Our outsourcing vendor conducts that entire development effort, following our framework throughout the project."

PMI is closely involved at a number of stages, and in a range of capacities throughout development. Hovey explains, "On PMI's side, we had many business people who were involved as subject matter experts. Also, every key area had PMI oversight. For example, although the vendor does all the quality assurance (QA), there is a PMI QA lead who oversees all that work to make sure that it conforms to standards and guidelines. And of course, PMI staff conducts all the user acceptance testing."

Hovey adds, "That same kind of oversight covered the system architecture as well. Our project managers use a scorecard, which essentially reflects the maturity of the architecture over time through a series of well-established states. The maturity of the architecture is very metrics- and measurement-focused. The architecture is expressed as the set of components that manifest it. For instance, in order to satisfy a particular use case, the vendor needs to build through the layered architecture — the user interface, the business logic layer, the data access layer, the database layer, and so on. The vendor is building a series of components, which ultimately realize these use cases. Before any critical work is done, we make them think about how that architecture will mature over time and then articulate their assumptions in a scorecard which we use to track them through a series of inspectable results. Every week, they must account for any deviations that have surfaced in the scorecard — including missing scheduled checkpoints — and they must show us how they are going to correct those deviations. And then, at the end of each iteration, they must account for the net of all of those deviations across their iterations and figure out what the impact is on the remaining plan."

Staying on Schedule

On the pmiCentral project, staying on schedule is crucial because the software development effort is only one part of the entire program. Hovey explains, "You cannot just unplug a 20-year-old system, plug in a new one overnight and expect that people will be able to do business the next day. The central effort of developing software is surrounded by satellite issues like user training, systems integration, data migration, and legacy system change. So being ahead of schedule in software development on this kind of project really is not an advantage. What is most important is the precise orchestration of all the moving parts towards a successful promotion to production."

More than halfway through the planned schedule, the pmiCentral project team is consistently on schedule — and under budget. Hovey reports, "One of the claims projects successfully completed its last iteration of transition and the other is completing its last iteration of construction. Another pmiCentral project has already been deployed, and is in a maintenance cycle. That was completed on schedule and appreciably under budget — and that's a run rate that we're consistently looking at."

Supporting Requirements Traceability

Based on the success PMI has had with IBM solutions on the pmiCentral project, the company is planning to adopt IBM Rational® RequisitePro® for requirements management and IBM Rational Software Architect for model-driven design and development. Jensen explains, "We are in the process of piloting a Rational RequisitePro implementation, based upon the specifications that we have introduced in the Rational Unified Process. We are looking at that to help us manage traceability, impact assessment, and overall requirements and test coverage for requirements. We are also looking at using Rational Software Architect for UML modeling as well. Its integration with Rational RequisitePro will enable us to better ensure that the requirements that we have modeled are traceable and have coverage through test scenarios. We expect it to be a very powerful solution for us."

Top Benefits

In addition to remaining consistently on schedule and under budget, Hovey cites improved quality, risk management, uniformity, and predictability as key benefits of implementing the Rational Unified Process methodology, Rational Rose and UML.

Hovey explains, "By modeling our use cases to represent the business process, and then by making our tests be scenarios of that business process, we ensure that we are delivering not just well written code, but the right code. And the breadth and depth of coverage of our tests really improves quality. From a risk mitigation perspective, the continuous, very rigorous forcing of these projects to report with absolute transparency, their real status and progress toward our goals, really helps us contain and manage risks on this geographically-distributed development project. Also, looking at the architecture they develop in a really well-defined language helps us figure out where we are going to hit problems, so we can deal with those things first.” He adds, “We have a dedicated group of subject matter experts from the business on all these projects, and the uniformity that we have enforced with RUP and UML is a key benefit for them. If they had to learn and interpret a new language for each project, that would add an enormous amount of time and inefficiency to their effort. By making the process uniform — everything from how the vendor reports status, to how the vendor specifies requirements, design and so on — we have made the subject matter experts a great deal more efficient and productive, while making the jobs of managing geographically-distributed development less stressful. We have also increased predictability by adhering to the RUP phase definitions which called for, among other things, sending each project through an elaboration phase to stabilize the architecture and work the unknowns and risks out of our system. We found that following RUP helped us think thoroughly and comprehensively about the system we were tasked with building and generally provided us a stable basis for predicting how the project would perform as it proceeded forward, which led to credible and accurate estimates for building it.”

In a geographically-distributed development environment, particularly one that involves outsourcing, clear communication is vital to the success of each project. Hovey concludes, “Ultimately, software development is a social activity and clear, precise and meaningful communication is key to that activity being productive and efficient. With RUP and UML we have a uniform set of syntax and semantics that help us bridge the geographic, linguistic and experiential gaps that are common in such projects. RUP and Rational Rose are helping us accomplish a larger volume of work across shorter timeframes and with greater momentum than would be possible without them.”

IBM Corporation
Route 100
Somers, NY 10589
U.S.A

Produced in the United States of America
04-05
All Rights Reserved.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

This case study is an example of how one customer used IBM products. There is no guarantee of comparable results

All statements regarding IBM future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only.