The True Story of a New RUP Manager

by **Pan-Wei Ng**
Software Engineering Specialist
Rational Software Singapore

The iterative approach to software development is still new to many project managers, and most are still trying to figure out how to apply it effectively. Through Rational's experience assisting numerous software teams with their projects, we have harvested a body of knowledge about what works and what doesn't. A need to maintain confidentiality prevents us from using an actual project case study to discuss our experiences, so instead we've written a light-hearted story of a successful project that actually reflects the experiences of multiple teams from multiple organizations. It illustrates how to apply the principles of iterative development and steer a project toward success and customer satisfaction. It also illustrates how project metrics play an important part in objective project tracking and control.

We introduce the characters with a sense of humor so that readers can get an inside look not only at what they do, but also at their perspective on software development and life in general. Smith, the hero, understands the dynamics of software development and has qualities everyone desires in a project leader: He manages users effectively, leads the team, and always knows exactly what to do. As you read the story, I encourage you to ask yourself, "What approach would I use if I were in Smith's shoes? What would be the consequences of my actions? And what lessons can I draw from this story and then apply in my environment?" In fact, as you will see, the story ends with a useful discussion of what to do with lessons learned. And now for the story...

Several months ago, the JKL Company completed a preliminary requirements study for REALITY-J, a J2EE project. After much deliberation, senior management finally approved the budget for this project, estimating that it could be completed within six months, given three developers and a project lead. Since past projects had experienced the usual problems -- schedule delays and quality issues -- and because he had read about the benefits of the Rational Unified Process,® or RUP,® as well as lots of articles on software agility, Jones, the senior manager, decided that REALITY-J should adopt an iterative development lifecycle. Smith, who shared Jones' taste in black suits, seemed the best choice to lead the team. A recent hire, Smith had some prior experience with Rational Suite® and RUP, and he was a sharp guy. So Jones

called Smith into his office in January and said, "Complete the project, you must. And may the RUP be with you." Smith scratched his head and wondered what the last sentence meant, but, nevertheless, REALITY-J had begun.

The Three Musketeers

Smith went back to his cubicle. Across the aisle were the three designated developers, Tom, Dick, and Harriet. Tom was a real geek. His watch had a digital camera, and he liked stateless session beans. Because of his enthusiasm, Tom was frequently called upon by Jones to fix problems in other projects. Dick was trying to learn J2EE but not getting very far, because he had recently met someone "special." Harriet was meticulous and wanted everything specified down to the smallest details; she needed to understand everything before proceeding. But she was reliable.

Because he was new to the group, Smith did not have a good understanding of the technical capabilities of any of these three people, but he did get along fine with all of them. So before reaching his cubicle, Smith told the team about the new project; he asked Tom to make sure that all three developers had a proper development environment (IDE, J2EE server, database server, and version control) and then asked Harriet to do the set-up verification.

A Bug's Life

At his desk, Smith quickly scanned the REALITY-J requirement specifications, a fifteen-page document with a whole annex of paper forms and reports. REALITY-J, he discovered, was about automating work processes and converting associated paper forms to online forms. Some of the work processes were within the same department, and had marked similarities. Having had some training on applying use cases, Smith spent the rest of the morning identifying the use cases for REALITY-J; he found twenty of them.

The question that bugged Smith all morning, however, was whether the allocated schedule was feasible. He remembered reading something about use-case points (UCPs) but had not really applied them before. He quickly ran a Google search with the keywords "use-case point" and found the following links:

- http://www.saspin.org/ProjectEstimateMethod.xls [Reader: ignore password prompt if you follow this link!]
- http://www.stud.ifi.uio.no/~kribu/oppgave.pdf

"Great! There's a spreadsheet! I can use it immediately!" Smith exclaimed when he saw the first link. But when he clicked on it to actually look at the spreadsheet, he started to scratch his head. It had several sections:

- Actor Weight Factors
- Use-Case Weight Factors
- Technical Weight Factors
- Environment Factors for Team and Weights

Within each section of the spreadsheet, Smith was required to enter values: either (1) simple, average, or complex, or (2) some rating from 0 to 5. Now, the RUP specifies use-case weight factors as follows: A simple use case has 1-3 transactions; an average use case has 4-7 transactions; and a complex use case has more than 7 transactions. But poor Smith was not sure what simple, average, and complex use cases were, or

transactions either, for that matter. He was not even sure whether the twenty use cases he had identified for REALITY-J were legitimate, or if he had identified all the Actors properly, because he was still new to the organization. Smith looked at the technical weight factors, then looked at the fifteen-page requirement specifications again, and found that they contained little about technical complexity. He also looked at the environment factors and his team, who seemed to have very diverse capabilities. "Maybe I can't use this spreadsheet immediately," Smith thought, wondering how else he would be able to tell if the schedule was feasible.

Late that afternoon, a thought suddenly struck Smith: Instead of trying to find out the number of simple, average, or complex use cases, or to determine the other weight factors, why not try different sets of values and look at the range of possible estimates? As he put some values into the spreadsheet, he noticed that it returned twenty man-hours per UCP. "I have to find out what this means," he thought.

For simplicity, he documented only the sets shown in Table 1. He set all Actor weights as zero, and all ratings of technical and environment factors as average (i.e., a value of 3 on a scale of 0 to 5). He considered three possibilities: all use cases are simple (best case); all are average (average case); or all are complex (worst case).

Table 1: Estimated Schedule: Actor Weight = 0; Technical & Environmental Factors = Average

Use-Case Complexity	Simple	Average	Complex
Number of Use Cases	20.00	20.00	20.00
Use-Case Points (UCPs)	101.49	202.98	304.47
Estimated Effort (man-hours)	2029.80	4059.60	6089.40
Estimated Effort (man-months) (1 man-month = 150 man-hours)	13.50	27.10	40.60
Schedule based on three-person team	4.50	9.00	13.50
Schedule based on four-person team	3.40	6.80	10.50

Smith found that if all twenty use cases were "simple," there would be 101.49 UCPs requiring 2029.8 man-hours of effort, which would translate to 13.5 man-months (see Table 1). With only three developers (Tom, Dick, and Harriet), REALITY-J would take 4.5 months, which was well within the budgeted 6 months. However, if all twenty use cases were of average complexity, three developers would need 9 months to complete the project. If Smith got his hands dirty by being the fourth developer, the schedule would be 6.8 months. This was slightly more than the allotted 6 months, but still not too bad. However, Smith recognized, if he were too involved in the development, the quality of REALITY-J would suffer, and he would have too much overtime and too many sleepless nights. Worst case would be that all the use cases were complex. Smith didn't dare look at that scenario -- the project would take more than a year to complete with three developers.

Smith considered his options and wondered whether he should ask for a schedule extension. He knew that Jones, his senior manager, would want ample justification for such a request. As we noted, Smith did not actually know whether the use cases were simple, average, or complex, and the spreadsheet parameters were not calibrated to his project team -- so he really had no clue about how long the project would take. But he also did not really have a case to ask for a schedule extension. He simply concluded that he must complete the project on time with the resources he'd been given.

Return of the Jedi

That evening, Smith thought again about what Jones had meant when he said, "May the RUP be with you." Maybe it was time to familiarize himself with use-case points. So he returned to the links he had found through his Google search, clicked on the second one, http://www.stud.ifi.uio.no/~kribu/oppgave.pdf, and began downloading the PDF file. An Internet Explorer (IE) dialog box soon popped up (see Figure 1).

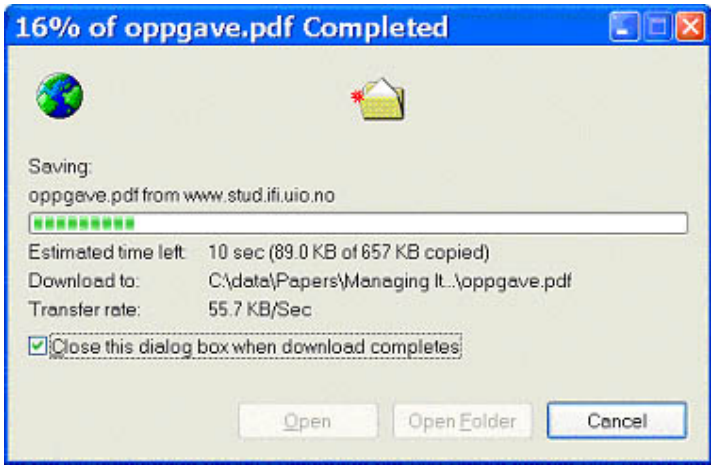

Figure 1: Computing Download Time

"Wouldn't it be nice if I could have a similar indication of project progress?" Smith wondered. At this very moment he suddenly realized that project tracking and control could follow this model, and he could actually do it with the use-case point technique. Smith quickly documented the analogy in Table 2.

Table 2: Project Tracking and Control Analogy

Analogy	Downloading Files	Software Development
Problem Complexity, A_S	File size in kilobytes	Application size in UCPs
Progress, A_P	Kilobytes downloaded	UCPs developed
Productivity, P_T	Download speed or transfer rate	UCPs per month
Remaining Time, T_L	Remaining file size divided by download speed	Remaining UCPs divided by team productivity

Smith recognized that the transfer rate during download was not constant, but varied over time, and he figured that IE must have used some observation window to measure the transfer rate. He could do the same for REALITY-J by monitoring the number of UCPs the team had completed at each month or week, and estimating the remaining time left with the following formula:

$$T_L = (A_S - A_P)/P_T$$

In essence, the formula computes the remaining time by dividing the remaining UCPs for the application (i.e., A_S-A_P) divided by the current team productivity, PT. This

formula would be very useful, because he could tell if he was running late. The project delay at any point in time TD could be computed as:

$$T_D = T_L - T_R$$

TR refers the remaining time in the project deadline. For example, if, at 2 months into the 6-month project (i.e., $T_R = 6 - 2 = 4$) T_L were estimated as 5 months, Smith would immediately know that the project was 1 month late (i.e., $T_D = T_L - T_R = 5 - 4 = 1$). This would mean that he must motivate his team to work harder, because if productivity didn't increase, the backlog of transactions would accumulate and result in further delays.

Smith also knew that the analogy was not perfect. Whereas the file size for downloading was fixed, the application size was not. Requirements could change, and use cases that were developed could need rework and refinement. Smith understood, therefore, that all the terms in the formula were time variable, and he would need to track them regularly.

If he assumed that all use cases were of average complexity, he would have approximately 200 UCPs to complete in 6 months. That would mean that he must strive to complete an average of 33 UCPs each month, or 8 UCPs each week. Smith understood immediately the importance of using an iterative development approach. The earlier development started, the more time the team would have. If he waited for requirements to stabilize and started one month later, he would only have 5 months for development, which would mean completing 40 UCPs each month! He had to proceed, even with his limited knowledge. Time was really not on his side, but he was not going to let time get the best of him!

Ordinary Decent Criminal

But as he launched the project, Smith needed to understand what a use-case point was. That night he studied the thesis he had downloaded and discovered that a transaction is a step in a use case. Since steps can have different granularities, he would require his transactions to be round-trip steps. In other words, a transaction would be an Actor-request/system-response pair. A use-case point is approximately a transaction, Smith concluded.

The next morning, Smith started outlining some use cases, identifying the key abstractions, and drafting the site map (i.e., user-experience model) for REALITY-J. He decided to make some samples for his team so that they would understand the level of detail he wanted, and so that he could quickly delegate some work to the team members. His sketches would also be useful tools for gaining consensus with the end-user department representatives. In fact, one of them had already agreed to meet with him on the next day -- the third day of the project.

On that afternoon, he met with his team. Harriet reported that Tom had set up the development environments. Smith gave his team an overview of REALITY-J and how UCPs could be used to determine if the project was on schedule. He showed them the file downloading analogy (see Table 2), and the remaining-time formula.

Smith also explained that each transaction (unadjusted UCP) was related to an actor-request /system-response pair. He took time to explain this because he would need his team to help him confirm the number of UCPs for each use case. Doing all this counting himself would be too overwhelming.

Tom, the team member most familiar with J2EE, noted that the actor-request and system-response pair mapped very well to the J2EE core patterns. He sketched the

framework (similar to Figure 2) on the whiteboard.

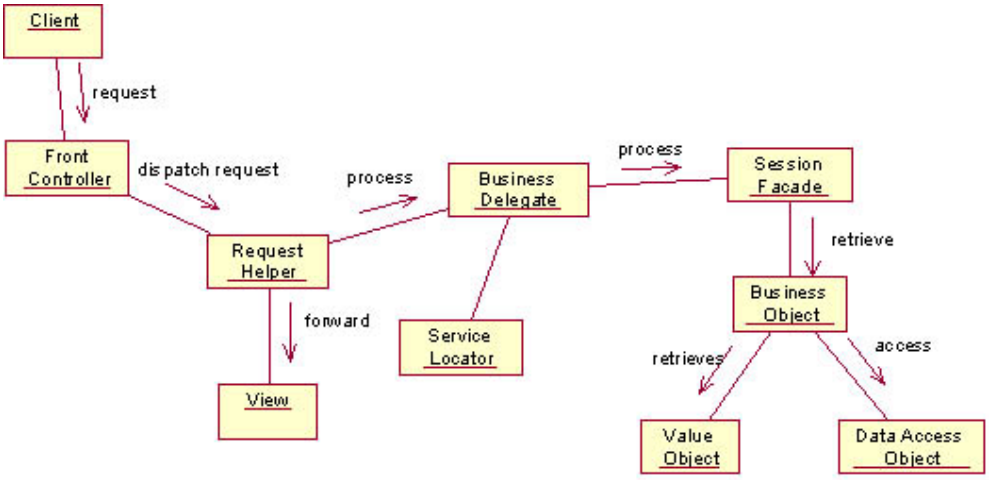

Figure 2: J2EE Core Pattern Framework

Smith asked Tom, "How long will it take to complete one transaction using this framework?" Tom replied, "About half a week." Smith remembered that the spreadsheet he used suggested an effort of 20 man-hours per UCP; this was in agreement with Tom's answer. Then Smith asked the rest of the team how long they would take. Dick and Harriet looked at each other. Harriet said that each transaction would naturally have a different level of complexity, and the time to develop each one would differ. Tom said that Dick and Harriet's work should not take more than twice as long as his, unless the use-case specifications or the use-case storyboards were really vague.

Dick said he could complete a transaction faster if he had some sample code to follow. Smith remembered that he had downloaded the Pearl Circle Online Auction (PCOA) J2EE sample from Rational Developer Network (www.rational.net/rdn; registration required), and that it might be useful. The team discussed how they would package their Java classes and Java Server Pages, and they agreed on a directory structure.

The meeting ended with several action items for the team:

- Tom would develop the infamous "Login" use case and a transaction that allows an actor to submit a record. This would cover two transaction types -- a retrieval (to check password) and an insertion.
- Dick and Harriet would refine the key abstractions and design an initial database, and then add some tables into the test database server. They would also start identifying the different transaction types.
- Smith allocated the use cases and use-case storyboards to Dick, Harriet, and himself, leaving Tom to concentrate on getting the first executable running.
- Everyone would report his or her progress at the end of the week.

It was only the second day of the project, but one of the developers was already starting on the coding. And Tom was very happy that he was beginning to code. Dick had a "date" with his friend and needed to leave slightly earlier. Harriet, who had also read parts of RUP, wondered, "I didn't know that RUP projects start coding on the second day. What ever happened to all the artifacts that must be produced? What about the Vision document, the supplementary specifications, the software architecture document, the configuration management plan, the development case? Well, Smith is the boss; we'll see how this goes."

Smith, meanwhile, was very happy with himself. The team had a plan, and he would see some results by the end of the week. That night, he continued refining the use cases, storyboards, and key abstractions in preparation for a meeting with Angela, an end-user representative, the next day.

American Beauty

Smith understood that maintaining good communication with the end user was crucial, and that meant more than just communicating about the project requirements. It also meant establishing good rapport with the customer. On the third day of the project, Smith went to Angela's office. Angela remarked, "You dress like someone else in the company; I just can't remember who."

First, Smith went through the requirement specifications, including the annexes. Angela pointed out that the sample forms and reports were not really up-to-date; she handed Smith the new set. Overall, however, the business workflow was largely the same as that described in the requirement specifications. Smith walked through the use-case specifications as well as the use-case storyboards, and Angela could visualize the screens and flows. Together, Smith and Angela brainstormed the exceptions that had to be handled. Next, Smith went through the entity classes with Angela to check if REALITY-J had captured enough data fields. Angela had never seen use cases, storyboards, or entity classes before, but Smith's walkthrough was sensible, and she could understand the presentation.

It was a fruitful morning, and Smith obtained a good understanding of what Angela needed. Although there were still some gaps in the business workflow, they reached consensus on which parts were stable, and Smith knew his team could begin developing these parts. Angela was impressed because Smith was very well prepared and asked intelligent questions, so she was ready to meet with him two weeks later to finalize the process gaps, and then a month later to see some prototypes. After the session, Smith suggested that they go have lunch together.

The next day, as Angela entered her office, her secretary handed her a box."Here are some muffins the new IT chap bought." Somehow, through their lunch conversation, Smith had discovered that she liked muffins. After she started Outlook, she saw an email from Smith summarizing their discussion.

By the end of the second week, Smith had met with all three end-user representatives. He had a good idea of the real requirements. He understood that there were still gaps to be plugged, but knew which parts were stable. The end-user representatives had also agreed to meet with Smith monthly, each for a half-day session.

A Time To Kill

At the end of the first week, the team met again to discuss the different types of transactions. Dick and Harriet had identified eight:

- Record insertion
- Single record retrieval
- Updating records in a single table
- Updating records in multiple tables
- Multiple record retrieval, resulting in an HTML table display
- Multiple record retrieval, resulting in multiple HTML pages; the user can go to the next page or the previous page

- Multiple record retrieval to populate a drop-down list
- Multiple record retrieval to fill a calendar on an HTML page

"I have the first two transaction types in the executable. I can show you," Tom said.

"Let's see it," Smith said.

Tom, who had been working late for the past few days, admitted, "I need to get used to the J2EE server. The way they deploy Enterprise Java Beans now is completely different from the previous version! But anyway, I got the two transactions running, although it's really crude. The user interface will need more work, but it's functional." Tom began clicking through the executable prototype, demonstrating the login and the record submission. Tom verified the application further by checking that the record had indeed been inserted into the database.

"How many hours did you take to do this?" Smith asked.

Tom started counting. "I spent 14 hours a day for the past two days, and I spent some time this morning finishing the record submission. That would be (2 x 14) + 4 = 32 hours."

"That would be 16 hours per transaction," Smith concluded.

"That would be *two days* per transaction. Don't count on me to do a 14-hour day. I have a life to live," Dick quickly added.

"But we should get faster as we progress through the project," Tom argued.

"What kind of testing did you do? I doubt these transactions are foolproof," Harriet remarked.

"Well, I spent 2.5 days, and this is the first week of the project. You can't expect too much," Tom answered.

"As our project progresses, we'll need to do regression testing. Do we have any test automation tools in this company?" Smith asked.

"We have a copy of Rational® Robot. I recently renewed the license -- the telesales rep was quite persistent. I've used it before, but our user interface has to be fairly stable; otherwise, we'll waste time rerecording the scripts. At this point, we don't have any use-case sign-off or user interface sign-off," Harriet said.

"We can apply test automation to those parts of the user interface that the end-user representatives have seen and agreed to," Smith commented, knowing that he could not afford too much rework. "Have Rational Robot installed on your machines."

The team proceeded to evaluate their project status. They found that, instead of twenty use cases, they had thirty; but these use cases averaged about six transactions each to total 180 transactions, which was slightly less than the previous estimate of 200 transactions. They also estimated that it would take eighteen hours to complete each transaction (including some testing), compared to the previously estimated twenty hours. They then summarized this project progress assessment, as shown in Table 3.

Table 3: Project Progress Assessment

Week	Application Size	Application Progress	Application Remaining	Productivity Hours Per Transaction	Required Hours	Required Months
0	200	0	200	20	4000	8.9
1	180	2	178	18	3204	7.1

The Negotiator

The team recognized that, in addition to an assessment of project progress, they needed an indicator of project quality. Having coded a transaction was not enough; the application needed to be thoroughly tested. They also recognized that they needed end-user consensus on what they had delivered. For each transaction, they had to capture more information about how complete it was. Each transaction was given the attributes shown in Table 4.

Table 4: Transaction Attributes

Transaction Attribute	Attribute Values	Attribute Description
Use-Case Sign-Off	Yes/No	Use-case specification signed off and accepted by the user. The value can be reverted to a "No" if the use-case specification is not clear and requires further refinement.
Use-Case Storyboard	Yes/No/NA	A class diagram showing the participating screen (i.e., JSP pages) is documented. If this is unnecessary, indicate Not Applicable (NA).
User Interface Prototype Sign-Off	Yes/No/NA	The screen prototype (HTML/JSP) has been accepted by the end-user representative.
Test Cases Identified	Yes/No	The test cases are systematically identified to explore various input combinations. In our story, Smith personally identified the test cases and used them as a contract for the developers to evaluate whether they had completed their transactions.

Analysis and Design	Yes/No/NA	Identify the classes and packages required for coding. If this is unnecessary, indicate Not Applicable (NA). In our story, Tom played the role of architect and assisted the other developers with this.
Implemented	Yes/No	The deployment unit has been deployed to the server, and the developer can click on it and do some informal testing.
Tested	Yes/No	Test cases have been designed and executed with the results logged.
Passed	Yes/No	The deployment unit has passed all identified tests.
Accepted	Yes/No	The user has seen the use case and accepted it, albeit informally.

Smith and his team also realized that they could leverage the sample code that Tom had developed -- specifically, by identifying parts that could be cut and pasted. Since Tom was the most proficient in J2EE, he was tasked to create samples of the identified transaction types.

To clarify roles and responsibilities in the project team, Smith sent an email. This would be the process the team would adhere to.

> Hi team,
>
> We will be following a weekly iterative development cycle, as indicated below.
>
> **Requirements.** I will meet with respective end-user representatives and attempt to get the use-case specifications, use-case storyboards, and the user-interface prototypes signed off, or at least get some consensus from them on these artifacts.
>
> **Planning.** At the beginning of each week, we will decide which transactions can be implemented. To avoid unnecessary rework, the transactions will be selected from those that have consensus from the end-user representatives.
>
> **Analysis, Design, and Implementation.** Initially, Dick and Harriet will need Tom's assistance and guidance on identifying classes and the Java packages. Harriet is responsible for the database design; any database changes go through Harriet, who

will publish these changes. Common J2EE services will be implemented by Tom. Other than that, each developer will instantiate her or his own set of deployment units, and deploy them separately on their machines.

Testing. When the transactions for a use case are implemented, we'll present them to the end-user representatives to gain their agreement at the end of each month. If there were any changes, we will note them and forward them to the respective use-case owners. When the end-user representatives are satisfied with the system for the use case, the use-case owner will record a set of test scripts for the use case. Scripts will be executed at the end of every week on the test databases.

Assessment. Progress tracking will be performed at the end of every week. All progress metrics will be in terms of weeks. We will discuss which transactions or rework should be allocated to the next week. I will personally track the progress of requirements gathering.

We should have a first prototype ready for Angela by week four, since her department had the most use cases. By week eight we should do some performance testing, since REALITY-J needs to support fifty concurrent users.

Cheers

--Smith

Figure 3: Roles and Responsibilities e-Mail

It was the end of the first week; the team had a plan, and they understood their roles and responsibilities. The developers began to respect to Smith because he knew exactly what he wanted to achieve. Harriet was very happy because instructions were spelled out. She printed the Transaction Attributes (Table 4) and the e-mail (Figure 3) and pasted them on her cubicle wall.

Attack of the Clones

Smith decided that it was time to create a spreadsheet to help track the progress of REALITY-J. Over the course of the project, he disciplined himself and his team to keep the spreadsheet updated. The results are shown in Table 5. Column 1 shows the week; there were 24 weeks in all. Column 2 shows the estimated project size in terms of transactions. The third set of columns shows the new and cumulative defects found over the weeks. The fourth set of columns shows the team's productivity in terms of total number of transactions completed, transactions not implemented (i.e., new ones) and transactions that fixed defects (i.e., rework). The fifth set of columns shows the progress in terms of new transactions incorporated and total transactions incorporated (i.e., new and rework).

Table 5: Project Metrics

Week	App Size	Defect		Productivity			App Progress	
		New	Cumulative	Total	New	Rework	New	Total
1	180	0	0	2	2	0	2	2
2	180	0	0	4	4	0	6	6
3	180	0	0	5	5	0	11	11
4	180	5	5	6	6	0	17	17
5	180	5	10	8	6	2	23	23
6	185	2	12	8	6	2	29	33
7	185	0	12	9	9	0	38	42
8	185	0	12	9	9	0	47	51
9	185	0	12	9	9	0	56	60
10	185	8	20	9	9	0	65	69
11	185	5	25	11	8	3	73	80
12	185	5	30	11	8	3	81	91
13	185	3	33	12	9	3	90	103
14	185	2	35	12	10	2	100	115
15	185	0	35	12	11	1	111	127
16	190	0	35	13	12	1	123	140
17	190	3	38	13	12	1	135	153
18	190	3	41	13	10	3	145	166
19	190	3	44	13	10	3	155	179
20	190	2	46	13	9	4	164	192
21	190	2	48	12	7	5	171	204
22	190	1	49	12	6	6	177	216
23	190	1	50	12	6	6	183	228
24	190	0	50	12	7	5	190	240

Smith and his team delivered the project on time, within the allotted 6 months. By that time, his team had another member, Freddy, who had joined in the middle of the project. As Smith compiled his post-implementation report, he included the charts shown in Figures 4, 5, 6, and 7. Figure 4 shows the defects found each week, and has three spikes occurring at weeks 4, 10, and 17.

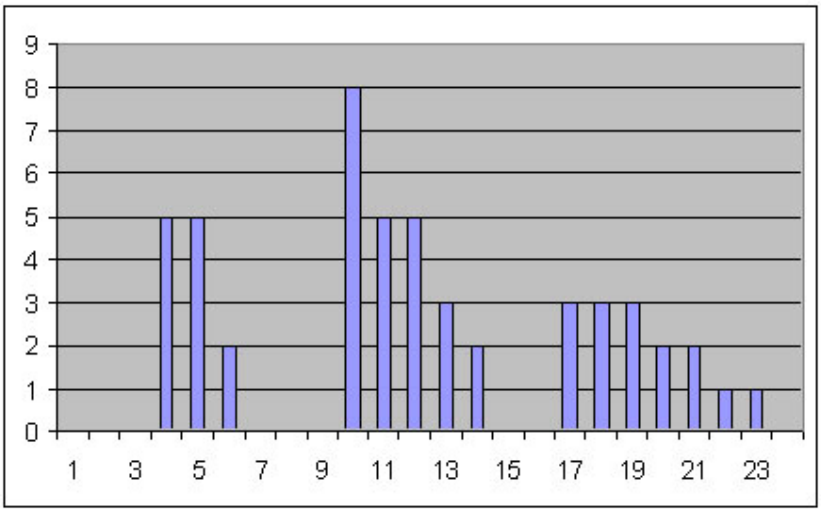

Figure 4: Defects Found by Week

At **week 4** of Project REALITY-J, the team started giving prototype demonstrations to various end-user representatives. Even though the system was rudimentary, the team was able to confirm many of the requirements and identify flaws in others. This resulted in some rework. There were more flaws than defects, but they were not implemented yet. Since the prototype had code that hit the back-end database, the team was able to evaluate the impact of identified rework. By actually doing the rework, they began to experience how to support the system, and they outlined some concrete steps for handling rework. And because they applied the J2EE core patterns, they were able to localize the part of the application that needed modifications.

At **week 10**, the team did some stress testing. This quickly identified defects, especially transactions involving large queries and pages that had drop-down lists with many options. Tom, who had subscribed to J2EEPATTERNS-INTEREST@JAVA.SUN.COM got some answers on how to quickly improve performance.

At **week 16**, Jones, the senior manager, asked Smith, "My managers have asked for a progress report on the system, and whether parts of it can be used by Angela's department. Manual processing in her department is costing too much in terms of delays for our customers." Smith answered, "We've been showing the developing system to her and her staff almost monthly. Let us finish up some of the functionality, and we'll let her do the formal user acceptance testing on week 18."

On **week 18**, Angela's staff found some defects, but not many. They were largely satisfied. Over the past months, they had become familiar with the system. Moreover, Smith had the system tested early and had focused on stabilizing the essential functionalities. Over the next few weeks, other end-user departments started their formal user acceptance testing. Defects were uncovered, but they were not major.

Figure 5 shows the progress of the application in terms of new transactions incorporated and total work done (new and rework). The new transactions incorporated followed a typical s-curve: Productivity is slow at the beginning and gains momentum as a project progresses. During the later part of the project, the team spent more time on fixing defects.

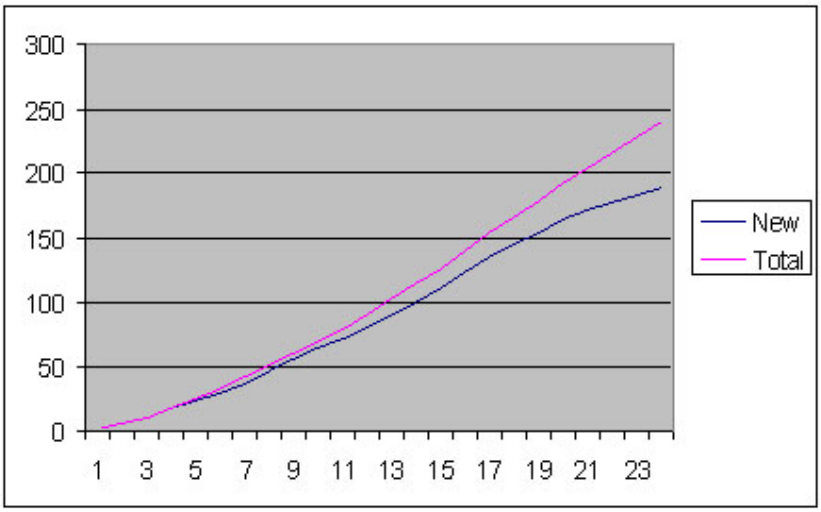

Figure 5: Application Progress

The proportion of new transactions and rework at each week is depicted in Figure 6, which shows team productivity over time, with momentum gaining in the first 8 weeks. Team productivity averaged nine transactions/week from weeks 7 to 10, and then twelve transactions/week after week 12, which was after Freddy joined the team. Overall, each team member had a productivity of about three transactions/week. The peak was at week 16, which corresponded with a push to complete the functionality, so that Angela's department could do acceptance testing.

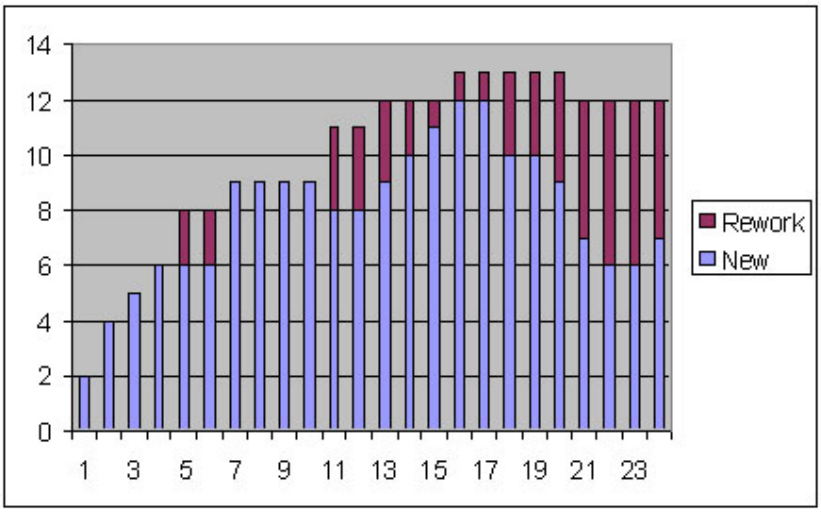

Figure 6: Productivity (Number of New Transactions and Rework Each Week)

Figure 7 shows the estimated delay T_D of the project in weeks. There are two curves: One estimates the delay in incorporating new functionality; the other estimates the delay in incorporating both new functionality and rework. Since productivity at the beginning of the project was low, the estimated delay was very high. The team started to catch up at about week 7. However, as the performance test identified some flaws in week 10, and as resources were assigned to fix them, the project schedule began to slip. The delay peaked at week 12, with a delay of 3.5 weeks -- almost a month. Smith knew that it would be very difficult to catch up if productivity didn't increase, and sought help from Jones. Jones, who understood the importance of REALITY-J, assigned Freddy to the team. Since many code samples already existed, Freddy was easily able to assimilate into the team and get to work.

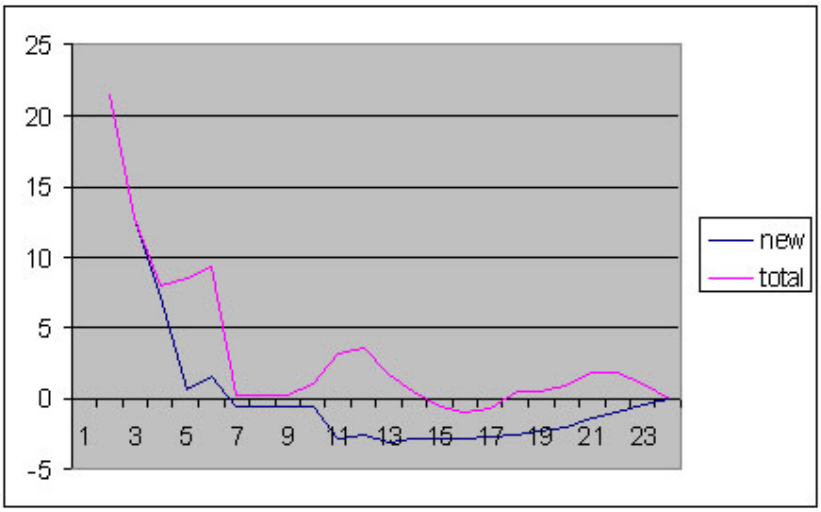

Figure 7: Estimated Delay in Weeks

Fellowship of the Rings

Jones was glad he had chosen Smith to lead the REALITY-J team. He wanted other project teams to apply the same approach to software development, so he asked Smith to make a presentation on his project execution to the rest of the IT staff. As Smith looked back at the past 6 months, he realized that it was an interesting experience. He gathered his team for a round of beer to celebrate and recall their experiences applying iterative development with the Rational Unified Process. Here's what they talked about.

Development Case

"What is the development case? I heard that all RUP projects must be tailored to the project's specific needs," Harriet asked.

"Remember the meeting we had at the end of the first week of the project? That e-mail I sent to all of you is the development case," Smith said.

"But you didn't follow the RUP template!" Harriet exclaimed.

"So what? It worked fine for me," Dick said.

"Good enough for me too," Freddy said.

Project Risks

"How about project risks? I don't recall that you did a risk list," Harriet said.

"The greatest risk is not knowing the risk," said Smith. "Not knowing much about any of you, I figured the best way to identify risk was to walk through an iteration as quickly as possible. That would force all to admit what they do and don't know. Also, any problems with the development processes would be highlighted immediately," he added. "Rather than waiting for typical project problems to surface, we steered the project toward highlighting them."

"Like the performance test you did at the beginning of the project," Tom recalled.

"And the frequent prototype demonstrations to the end-user representatives," Harriet remembered.

Minimal Artifacts

"How about the other RUP artifacts? I thought there were lots of them," said Harriet, who seemed to have the most questions.

"We had minimal artifacts, but minimal does not equate to nothing," Smith explained. "We only developed the artifacts that we really needed: a use-case model, use-case specifications, use-case storyboards, database design diagrams, and a set of diagrams for each transaction type. The test plans and results are all in the tools," he went on. "As for use-case realizations, we reverse engineered them after we implemented them. This allowed us to quickly evaluate the impact of rework, and to do some verification against the architecture framework and some refactoring."

"Yes, a few good artifacts are better than many useless ones!" Harriet concluded, and everyone nodded in agreement.

Software Architecture

"Tom did a great job with the transaction types. The diagrams for those and the associated code samples were enough for me to get started right away. And I am not that experienced in J2EE!" Freddy remarked.

"Yes, architecture is about communicating the structure and behavior of the system to the team, and about partitioning it effectively for resilience and performance," said Tom. Then he became philosophical: "I focused on visually modeling the framework (see Figure 2) we used and the variations for the different transaction types. And we supported the visual modeling framework with sample codes. Those codes became the templates for the rest of the team."

"As for me, so long as my team knows how to code, to do it the same way as everyone else on the team, and to produce code that works and meets performance characteristics -- that is good architecture!" Smith said. "It doesn't have to be perfect. We need to learn, but we don't need to learn everything in a single project. We still have a lifetime ahead of us."

"In the past, we had many projects that got stuck at the design stage, and we argued a lot," Harriet recalled. "I guess this time we knew enough to get beyond that."

"On this project I finally learned the J2EE core patterns," said Dick, smiling broadly.

Requirements and User Involvement

"You know, we really did not have requirements sign-off for this project, either," Harriet remembered.

"Well, we didn't have formal sign-off, or freezing, as most people call it, but we had continual agreement," Smith said. "We also had all the meeting minutes recorded and disseminated promptly, which is a form of sign-off. But the sign-off is more like requirements baselining. It indicates that this is our common understanding at this point."

"Even if people sign off on the requirements, they can still change them," Dick said, recalling past experiences.

"Yes, continual involvement with the customer is crucial," Tom said.

"I spent quite some money on them," Smith mused. "I bought drinks and donuts for our meetings, and paid for their lunches after that."

"Yes, good relationships go a long way," Dick said.

"But, we were professional in our dealings, too; we delivered a quality product, and no amount of lunches can save you if you turn over a poor-quality product," added Smith.

Project Phases and Iterations

"What about the Inception, Elaboration, Construction, and Transition phases? Where were they in the project?" Harriet wanted still more answers.

"Does it matter? The early part of a project is driven by risk and the latter part by a need for completion. As long as you remember this, you will be fine," said Smith. "Inception was completed by the third week, when I met with all the end-user representatives. Elaboration ended when we fixed some of the performance problems, and Construction ended when I found that we were quite ready for the user acceptance tests," Smith answered.

"How long were our iterations?" Harriet asked.

"You could say a month, because that was when I told the team which use cases or transactions needed to be completed for the month," Smith said. "Or you could even say a week, because that was when everyone completed certain transactions and submitted their status reports to me -- but really, we had continual status reporting. The key is to have a good grasp of where the project is at all times, and to steer it toward customer satisfaction."

Project Tracking and Control

"My past understanding was that use-case points are about project schedule estimation, but you used them to determine whether we were experiencing delays," said Tom, who was clearly impressed with Smith.

"This is the first time I applied the technique. I think it worked perfectly, and I used it as justification to get Freddy to join the team." Smith was also impressed with himself.

"In the past, we followed plans," Harriet said, "but now, we use objective numbers!"

"I'm glad I joined the team. It was a great experience!" Freddy said.

"We're glad, too. We would have had serious delays if you hadn't joined," Harriet added.

"I've got to go now," Dick said as he saw his friend enter the bar.

As Dick started to leave his seat, everyone said to him, "May the RUP be with you!"

Pay It Forward

I frequently use the above story, especially the part on software metrics and the weeklong iterations, to help teams shift from the waterfall approach to the iterative development paradigm. For smaller teams, iterations last for 1 week; for larger teams, iterations average about 2 weeks.

Feedback on the approach has been quite encouraging. And as one software team pointed out, their project evolved in much the same way as the project in this story; the story actually highlighted what they needed to do next. A story can make the

iterative approach easy to understand in an enjoyable way, and it's also easy to relate to, so I urge you to try it out if you need to convince anyone in your organization that iterative is the way to go.

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!