Iterative and Incremental Development

Engineering Notebook

C++ Report, Feb, 1999

In 1970 Dr. Winston Royce published a paper entitled "Managing the Development of Large Software Systems". This paper appeared on pages 1-9 of *Proceedings of IEEE WESCON*. It is commonly agreed that this paper was the progenitor of the waterfall model of software development. And in some distorted fashion, perhaps it was. It may be that some pointy haired managers ignored Royce's words and simply used one or two of his illustrations to create and promulgate the idea that software can be completely analyzed before it is designed, completely designed before it is coded, and completely coded before it is tested. I.e. – Waterfall.

Were I Dr. Royce, I would wince at the thought of how thoroughly and inextricably Waterfall has permeated the software development community. It is not surprising, even today, to hear of managers chiding their developers about doing design while in the implementation phase. The manager want's to be able to manage the schedule by drawing an X through the design phase.

Royce's paper describes what we would today call an Iterative and Incremental development style. In his paper he described how analysis must be constrained by design issues, how design issues fed back into analysis and coding issues fed back into design. He also described the process of releasing a project incrementally where each increment had its own analysis, design, coding, and test activities.

In my next few columns we are going to walk through the fundamentals of Iterative and Incremental development. We will begin, this month, by describing the Waterfall process, and discussing some of its weaknesses. In future months we will describe Iterative and Incremental Development Processes (IIDP) and show how it addresses the weaknesses of Waterfall.

Waterfall

The date is January 2nd, and you head still aches from the recent revelry. You are sitting in a conference room with several managers and a group of your peers. You are a project team leader. Your boss is there, and he has brought along all of his team leaders. His boss called the meeting.

"We have a new project to develop." Says your bosses boss. Call him BB. The points in his hair are so long that they scrape the ceiling. Your boss' points are just starting to grow, but he eagerly awaits the day when he can leave Brylcream stains on the acoustic tiles. BB describes the essence of the new market they have identified and the product that want to develop to exploit this market.

"We must have this product to market by fourth quarter – October first." BB demands. "How long will it take you to do the analysis?"

You raise your hand. Your boss tries to stop you, but his spitwad misses you and you are unaware of his efforts.

"Sir, we can't tell you how long the analysis will take until we have some requirements."

"The requirements document won't be ready for three or four weeks." BB says, his points quivering with frustration. "So, *pretend* that you have the requirements in front of you now. How long will you require for analysis?"

No one breathes. Everyone looks around at everybody else to see if they have some idea.

"If analysis takes any longer than April first, then we have a problem. Can you finish the analysis by then?"

Your boss gathers his courage and says: "We'll find a way, sir!" His points grow 3mm; and your headache increases by two Tylenol.

"Good." BB Smiles. "Now, how long will it take to do the design?"

"Sir," you say. Your boss visibly pales. He is clearly worried that his 3mms are at risk. "Without an analysis, it will not be possible to tell you how long design will take."

BB's expression becomes stern. "*PRETEND*, you have the analysis already!" He says, while staring you down with his vacant beady little eyes. "How long will it take you to do the design?"

Two Tylenol are not going to cut it. Your boss, in a desperate attempt to save his new growth babbles: "Well, sir, with only six months left to complete the project, design had better take no longer than three months."

"I'm glad you agree, Smithers!" BB says, beaming. Your boss relaxes. He knows his points are secure. He starts lightly humming the Brylcream jingle.

BB continues, "So, analysis will be complete by April 1st, Design will be complete by July 1st, and that gives you three months to implement the project. This meeting is an example of how well our new consensus and empowerment policies are working. Now, get out there and start working. I'll expect to see TQM plans and QIT assignments on my desk by next week. Oh, and don't forget your cross-functional team meetings and reports will be needed for next months quality audit."

"Forget the Tylenol." You think to yourself as you return to your cubicle. "I need bourbon."

Visibly excited, your boss comes over to you and says, "Gosh, what a great meeting. I think we're really going to do some Earth shaking with this project." You nod in agreement, too disgusted to do anything else.

"Oh," your boss continues, "I almost forgot." He hands you a thirty page document. "Remember that the SEI are coming to do an evaluation next week. This is the evaluation guide. You need to read through it and memorize it and then shred it. It tells you how to answer any questions that the SEI evaluators ask you. It also tells you what parts of the building you are allowed to take them to, and what parts to avoid. We are determined to be a level 3 organization by June!"

The Analysis Phase.

You and your peers start working on the analysis of the new project. This is difficult because you have no requirements. But, from the 10-minute introduction given by BB on that fateful morning, you have some idea of what the product is supposed to do.

The requirements document arrives on the 15th of February. And then again on the 20th, 25th, and every week thereafter. Each new version contradicts the previous. Clearly the marketing folks who are writing the requirements, empowered though they might be, are not finding consensus.

In spite of it all, you and your teammates continue with your analysis work. And then, a miracle happens!

On April 1st you are *done* with the analysis!

You go to your boss and complain. "How could you have told BB that we were done with the analysis?"

"Have you looked at a calendar lately? It's April 1st!"

The irony of that date does not escape you. "But we have so much more to think about. So much more to analyze!"

"Where is your evidence that you are not done?" inquires your boss impatiently.

"Whaaa…."

But he cuts you off. "Analysis can go on forever, it has to be stopped at some point. And since this is the date it was scheduled to stop, it has been stopped. Now, get back to your cubicle and start designing."

As you shuffle back to your cubicle, you begin to consider the benefits of keeping a bottle of bourbon in your bottom desk drawer.

The Design Phase

They threw a party to celebrate the on-time completion of the analysis phase. BB gave a colon stirring speech on empowerment. And your boss, another 3mm taller, congratulated his team on the incredible show of unity and teamwork. Finally, the CIO takes the stage and tells everyone that the SEI audit went very well, and thanks everyone for studying and shredding the evaluation guides that were passed out.

As the weeks flow by, you and your team work on the design of the system. Of course you find that the analysis that the design is supposedly based upon is flawed. But when you tell your boss that you need to go back and work some more on the analysis to shore up its weaker sections, he simply states: “The analysis phase is over. The only allowable activity is design. Now get back to it.”

So, you and your team hack the design as best you can, unsure of whether the requirements have been properly analyzed or not. Of course it really doesn’t matter much since the requirements document is still thrashing with weekly revisions.

Midway through the design cycle, the marketing folks announce that the have rethought the focus of the system. Their new requirements document is completely restructured. They have eliminated several major feature areas, and replaced them with feature areas that customer surveys have shown to be more appropriate.

You tell your boss that these changes mean that you need to reanalyze and redesign much of the system. But he says: “The analysis phase is over. The only allowable activity is design. Now get back to it.” So you do.

Hack, hack, hack, hack. You try to create some kind of a design document that might actually reflect the new requirements documents. However, the revolution of the requirements has not caused them to stop thrashing. Indeed, if anything, the wild oscillations of the requirements document have only increased in frequency and amplitude. You slog your way through them.

Then, on July 1st a miracle happens!

You are *done* with the design!

Rather than go to your boss and complain, you stock your middle desk drawer with some vodka.

The Implementation Phase

They threw a party to celebrate the on-time completion of the design phase, and their graduation to CMM level 3. This time you find BB’s speech so stirring that you have to use the restroom.

There are new banners and plaques all over your workplace. They show pictures of eagles and mountain climbers, and they talk about teamwork and empowerment. They read better after a few scotches. That reminds you that you need to clear out your file cabinet to make room for the brandy.

You and your team begin to code. But you rapidly discover that the design is lacking in some significant areas. You convene a design session in one of the conference rooms to try to work through some of the nastier problems. But your boss catches you at it and disbands the meeting saying: “The design phase is over. The only allowable activity is coding. Now get back to it.”

Your boss hires a consultant to build tools to count the number of lines of code that are being produced. He puts a big thermometer graph on the wall with the number 1,000,000 on the top. Every day he extends the red line to show how many lines have been added.

Three days after the thermometer appears on the wall, your boss stops you in the hall. “That graph isn’t growing fast enough. We need to have a million lines done by October 1st.”

“We aren’t even sh-sh-shure that the proshect will require a m-million linezh.” You blather.

“We have to have a million lines done by October 1st.” your boss reiterates. His points have grown again, and the Grecian formula he uses on them creates an aura of authority and competence. “Are you sure your comment blocks are big enough?”

Then, in a flash of managerial insight he says: “I have it! I want you to institute a new policy amongst the engineers. No line of code is to be longer than 20 characters. Any such line must be split into two or more -- preferably more. All existing code needs to be reworked to this standard. That’ll get our line count up!”

You decide not to tell him that this will require two unscheduled man months. You decide not to tell him anything at all. You decide that intravenous injections of pure ethanol are the only solution. You make the appropriate arrangements.

Hack, hack, hack, and hack. You and your team madly code away. By August 1st your boss, frowning at the thermometer on the wall institutes a mandatory 50-hour workweek.

Hack, hack, hack, and hack. By September 1st, the thermometer is at 1.2 million lines and your boss asks you to write a report describing why you exceeded the coding budget by 20%. He institutes mandatory Saturdays.

Hack, hack, hack, and hack. Tempers are flaring; people are quitting; QA is raining trouble reports down on you. Customers are requesting installation and user manuals, salesmen are requesting advance demonstrations for special customers; the requirements document is still thrashing, and the liquor store won’t accept your credit card anymore. Something has to give. On September 15th BB calls a meeting.

As he enters the room, his points are emitting clouds of steam. When he speaks, the bass overtones of his carefully manicured voice cause the pit of your stomach to roll over. “The QA manager has told me that this project has less than 50% of the required features implemented. He has also informed me that the system crashes all the time, yields wrong results, and is hideously slow. He has also complained that he cannot keep up with the continuous train of daily releases.”

He stops for a few seconds, visibly trying to compose himself. “The QA manager estimates that, at this rate of development, we won’t be able to ship a product until December!” Actually, you think it’s more like March, but you don’t say anything.

“December!” BB roars. People duck their heads as though he were pointing an assault rifle at them. “December is absolutely out of the question. Team leaders, I want new estimates on my desk in the morning. I am hereby mandating 65-hour workweeks until this project is complete. And it better be complete by Nov 1st.”

As he leaves the conference room he is heard to mutter: “Empowerment – Bah, Humbug!”

The Feeding Frenzy that ensues.

Your boss is bald; his points are mounted on BB’s wall. The fluorescent lights reflecting off his pate momentarily dazzle you.

“Do you have anything to drink?” he asks. Having just finished your last bottle of Boone’s Farm, you pull a bottle of Thuderbird from your bookshelf and pour it into his coffee mug. “What’s it going to take to get this project done?” he asks.

“We need to freeze the requirements, analyze them, design them, and then implement them.” You say callously.

“By Nov 1st?” your boss exclaims incredulously. “No way! Just get back to coding the damned thing.” He storms out, scratching his vacant head.

A few days later you find that your boss has been transferred to the Test division. Turnover has skyrocketed. Customers, informed at the last minute that their orders cannot be fulfilled on time, have begun to cancel their orders. Marketing is reevaluating whether or not this product aligns with the overall goals of the company, etc, etc. Memos fly, heads roll, policies changes, and things are, overall, pretty grim.

Finally, by March, after far too many 65-hour weeks, a very shaky version of the software is ready. In the field, bug discovery rates are high, and the technical support staff are at their wit’s end trying to cope with the complaints and demands of the irate customers. Nobody is happy.

Conclusion

Although this story is satirical, it is also not that far off the mark. The events I described here may not all happen, all the time, at the same company; but I have seen each and every one happen in various companies over the last couple of decades.

The problem with the waterfall process can be summed up on one sentence. Waterfall produces no reliable data with which to manage the development process. Analysis and design are not the kind of activities whose completeness can be measured; thus you never know when you are done with them. If you don't know when you are done with an activity, then the schedule will tell you when you are done. All management perceives from that is that the project is on schedule.

Implementation, on the other hand, can be tested for completion. Therefore, it is only after implementation is well advanced that managers get an inkling that their original schedule might not be sufficient.

As we saw, the analysis and design documents produced during waterfall are also inadequate. By the time implementation is well under weigh, it doesn't look very much like the design, and it bears almost no resemblance to the analysis. Oftimes the analysis and design documents fill huge volumes that sit in a bookshelf gathering dust. It is rare that anyone actually refers to them.

In my next column we will explore how these deficits can be addressed using an iterative and incremental development process (IIDP). We will discuss how schedules can be managed using a process that produces schedule data, rather then one that consumes schedule data.