Annualized Software Delivery

Prospecting for programmer's gold.

PROJECTS IN THE COMMERCIAL SOFTware industry are often characterized by frenzied attempts to leapfrog the competition. Companies try to deliver innovative products quickly, and in their ill-fated attempts to serve two masters, both development time and product innovation suffer. A history of the GigaCorp company, a composite of companies I have worked with, describes the current situation in more detail.

GigaCorp produced a successful widget formatter called GigaMat 1.0. It was well received in the marketplace and offered significantly more functionality than its main competitor, MegaMat. A few months after GigaMat was released, MegaCorp responded by releasing a new version of MegaMat, which provided incrementally more functionality than GigaMat 1.0.

Because of the competitive situation, GigaCorp then set a target of releasing a minor upgrade, GigaMat 1.2, within three months. The project planners recognized this was an ambitious schedule but felt that to remain competitive they had no other choice. As a result of the short schedule, the GigaMat 1.2 team rushed through the early stages of analysis and design—doing little requirements analysis and only a sketchy design.

In the later stages of the project, the team found that some of its assumptions about the product's requirements were invalid, and it then spent a great deal of time correcting defects that it had inserted during the requirements and design stages.

Although it developed an initial implementation of the intended functionality within the desired three-month time frame, the quality of the product was poor enough that it could not be released to the public. The GigaMat 1.2 project then spent months in an extended integration, test, and defect-correction cycle, and finally released the product after 12 months.

In later versions of GigaMat, the code base became extremely brittle. Even minor changes became time consuming and error prone. Time that might have been spent developing innovative new capabilities was instead spent correcting the problems arising from incomplete design, rushed implementation, and short-term planning decisions that had been made in earlier releases.

The pattern at GigaCorp isn't unusual. Companies create products under intense schedule pressure, which produces low-quality design and code bases. A company typically knows that its product is built on a shaky foundation, but because of pressure to release timely upgrades, it continues to extend its low-quality code base in hopes of delivering the next version as quickly as possible. As problems mount, the company realizes that the notion of using a poor-quality code base to support quick development is a cruel joke; it severely hobbles the company's ability to extend its product and ultimately delays delivery.

A TWO-PART PLAN. Annualized releases provide a way to escape from pressure-cooker upgrade cycles and low-quality code bases. Suppose you work for the GigaCorp company and that you released the first version of your GigaMat widget formatter in 1996. With annualized delivery, you can strategically divide your future development efforts into two parts, which provide for incremental releases in 1997, 1998, and 1999 and for a completely redesigned product in 2000.

Part 1: Incremental enhancement. One team maintains and enhances the product thread that started with GigaMat 1996 and that will continue through 1999. Its task is to deliver minor, incremental improvements that can be readily supported by the existing code base.

This team is committed to release incremental updates approximately once a year. Functionality is developed in increments, perhaps using daily builds to drive to a potentially shippable state once every two months or so. Keeping the product close to a shippable state at all times increases the likelihood of actually shipping the product close to its target delivery date.

Part 2: New development. While the first team

Continued on page 103

Editor:
Steve McConnell
Construx Software Builders
PO Box 6922
Bellevue, WA 98008
stevemcc@construx.com

Continued from page 104

delivers incremental improvements, a second team develops a completely new GigaMat 2000 that sheds much of the baggage associated with GigaMat 1996. This team's task is to make wholesale conceptual improvements in both the product and its underlying design and implementation.

Separating the new development effort from the enhancement effort lets this team focus on innovating without being preoccupied with short-term delivery pressures. Because of the longer time frame, it can treat the development of a new GigaMat 2000 as a truly innovative exercise in redefining the state of the art in its product area.

More effective development. For all but the richest companies, the major objection to this strategy will be that it is prohibitively expensive to conduct two separate development threads. But this strategy is much less expensive than it might at first appear, because separating development into two branches allows each branch to operate more efficiently.

For new products, the current crash-development style leads to poor designs, poor implementation, poor quality, excessive rework, developer burnout, high turnover—in short, to significant inefficiencies. Development takes place in a pressure cooker, hardly the ideal environment for creating innovative products. By stretching out the development cycle, product redesign can be conducted at a more leisurely, contemplative pace with a smaller staff. The smaller staff can operate more efficiently and innovatively, and product integrity is likely to benefit.

Having an independent product redesign underway reduces the enhancement team's need to leapfrog the competition with each release. Cramming major new functionality into an incremental release increases the risk of late delivery. When incremental enhancements are separated from new development, you can instead plan an enhancement release as a strategic, low-risk effort to provide timely, incremental improvements to your customers.

More proactive improvements. Currently, most companies will create a new code base only after their old code base has become so brittle that it is virtually impossible to maintain. This puts a company in a reactive rather than proactive posture and virtually guarantees that the last development cycle using the old code base will be characterized by enormous struggles to make even the tiniest improvements in the outdated design and code.

The annualized delivery strategy accounts for the reality that most code bases deteriorate under maintenance—especially if major portions of the code base were originally developed under intense schedule pressure. The strategy puts a company on track to renew the code base for a strategic product every few years, proactively, before the old code base becomes unmaintainable and sabotages new development efforts.

Products should be so well designed that they stay competitive for three to five years.

More regular product releases. From a software purchaser's point of view, software products are currently delivered at irregular intervals, perhaps as infrequently as every two or three years. Although software vendors continue to make more ambitious schedule promises, the commercial software industry's track record does not instill much confidence. Because of uncertainty in product-release cycles, companies that use software don't know whether to budget for a product upgrade this year, next year, or the year after that. Software vendors thus fail to provide the support needed for mid- to long-range planning.

With annualized releases, the vendor's customers can decide once a year whether to upgrade. Annualized releases reduce the planning risks that now arise from companies that announce release dates they have no reasonable chance of meeting.

Less baggage for established companies. One nagging problem in the commercial software industry has been how a company with a large installed base can enhance its products with the same agility that a startup company can. The obligation to support an existing customer base can become a ball and chain that restricts an established company's ability to innovate.

Annualized product releases allow an established company to recapture some of the startup's agility. An established company can announce a policy like this: "We guarantee that we will support the 1996 model of GigaMat in enhancement releases through the year 1999. After that, we will introduce a redesigned GigaMat 2000, which will not support an upgrade path from previous versions." Each customer will have to decide whether to upgrade to GigaMat 2000 or continue using GigaMat 1999. This approach shifts part of the responsibility for staying with an outdated product back onto customers' shoulders, and since three to five years is a lifetime in the computer industry, limiting upgrade support to that time frame seems inherently reasonable.

BACK TO THE PROGRAM. Segments of the commercial software industry already use forms of annualized product releases. Tax programs such as Turbo Tax have, of necessity, been on annualized release cycles since their inception. Multimedia products such as Encarta joined the annualized-release bandwagon later. Time will tell whether product names such as "Windows 95" and "Office 97" are driven by an annual-release strategy.

The broader commercial software industry needs to recognize that significant upgrades of major products typically take an amount of time measured in years rather than weeks or months. If a company can step back from the current frenzy of new-product development and invest the time needed to develop a truly innovative product, it should be able to deliver products that are so well designed and implemented that they can remain popular and competitive for three to five years—which will provide enough time to complete the next innovative design cycle. ◆