



Agile Product & Project Management Advisory Service
Executive Update Vol. 9, No. 21

Update

To Release No More or To “Release” Always: Part I — The Myth

by Israel Gat

For most of my adult life, I have been perplexed by a Pavlovian phenomenon: whenever I, as an engineering manager, released code to manufacturing,¹ the marketing folks reacted by conducting a three-week world-wide analyst tour. As much as I appreciate good public relations for my products, I viewed this phenomenon as a mystery that I might one day solve, perhaps when I retire. As luck would have it, the need to solve the mystery has become more pressing in recent years because agile has transformed the way we develop software. My resolve to postpone exploring this phenomenon until better days arrived came to an end when a frazzled marketing manager said, “Israel, you guys can code faster than my team can market.” Flattering as this comment was, it led me to some far-reaching conclusions with respect to the way we develop and use software in general, as well as enterprise software in particular. This *Executive Update* series captures the essence of my thoughts on the subject.

This first installment in a three-part series explains why I consider the whole release concept a myth. Subsequent *Updates* will address a more relaxed approach to the engineering, support, customer, and business design aspects of a release. Taken as a whole, this series is a call to reform the software industry, and quite possibly some related industries.

THE MYTH

Conventional wisdom holds the release concept as a pillar of both the software engineering and the go-to-market processes. From an engineering standpoint, closure, whether real or imagined, is reached on

releasing the code. From a business perspective, the release is a well-defined entity that can be discussed with the customer and promoted in the market, irrespective of whether the release is monetized. You can actually think of the release as an “engineering meets the business” construct. Metaphorically, the release is a coin with two faces.

As is often the case with two-faced gods, such as Janus, people tend to forget the dual nature, and that is true of the release. A body of code that delivers certain features and functionalities is one thing. The use of this body of code by marketing and sales to accomplish business results is quite another. Not only do the two activities differ, but they do not necessarily need to be tied together through a 1-to-1 relationship. For example, as an engineering manager, I oversaw several releases that were successful even though we lacked the marketing resources to promote them.

Like a set of nested Russian dolls, the duality of the term “release” often masks a deeper duality. You can view the end user as the primary target of a release or you can view the release as a vehicle for market development. Philosophical as this differentiation of the release might seem, it actually brings to the forefront essential questions about the nature of software and the nature of the market. We will table these important questions for now and deal with them in depth in a subsequent *Update*.

The Drinking Water Pool

Consider the following metaphor: A drinking water pool has two pipes — an in-pipe and an out-pipe. The in-pipe fills the pool with water, while the out-pipe draws water out of the pool. Both pipes have faucets on them, as adding water is independent of drawing water and vice versa. For all we know, water added on 5 September might not be withdrawn until 29 October. Assuming an acceptable level of water treatment between the two dates, this kind of asynchronous operation is perfectly acceptable. Obviously, you do not want the pool to overflow or to be empty. However, as long as these two boundary conditions are satisfied, regular water treatment is all one needs to keep things going.

Think of the in-pipe in this example as engineering and the out-pipe as the business. Engineering can post releases at its own pace. The business can selectively choose from the posted releases. In this paradigm, marketing is not obligated to promote a release upon its completion. Marketing might do so in three months; it might choose to promote the current release with another release due at a later time; it might choose to make a release available on a limited basis; or it might choose never to promote a release.

An intriguing question poses itself if you accept this premise of asynchronous operation. If the business is free to determine how it will promote a release in the market, why should engineering be bound to producing releases in the traditional manner? Can everyone benefit from relaxing the constraints that usually *surround* a release and move toward a more flexible, fluid release concept? For example, how about made-to-order releases for preferred customers? Do not infer that I recommend professional services customization and deployment of a standard release for the needs of a specific customer. Rather, I advocate a “custom-tailored suits” approach; we start by taking “measurements” of the customer and produce a custom release by engineering.

Up to a Point

Right or wrong, orthodox business theory teaches us to aim at the market, not cater to the whims of each and every customer in the market. We, of course, aspire to serve as many customers as possible with every new release through segmentation. However, even with excellent segmentation, in software, we typically serve the customer *up to a point*, not fully. The release represents a compromise — one hopes a good one — between the aggregation of all customer requirements and what we can produce under realistic resource and time constraints.

Serving our customers with up-to-a-point software might sound innocuous enough, but I believe it represents the most fundamental characteristic of software. None of us would normally buy a car that lacks a major feature we deem intrinsic to the car’s concept and function; say, a steering wheel. We might not be getting a great steering system that affects us viscerally in a certain car, but we certainly get a functional steering system in any car.

Unlike software, *up to a point* applies only up to a point with respect to cars: the car is complete without any compromise as to fulfilling the common understanding of what constitutes a car. You might not be able to take an ordinary car off-road or drive it at 200 miles per hour on a racetrack, but a car restricted to reasonable speeds on paved roads definitely satisfies the common definition of a car for most buyers.

If we carefully examine a software release, it is not at all clear whether it is *complete* in the sense discussed in the previous paragraph. Software is categorized as supply chain management (SCM), customer relation management (CRM), network management, or some other kind of software, just as a sedan, an SUV, or a cabriolet defines categories in the car industry. However, a universal definition of completeness does not really exist for any of these software categories. As a matter of fact, the all-too-frequent request for enhancement (RFE) “race” between customer and vendor clearly indicates lack of completeness. Dozens of RFEs per product is not an uncommon phenomenon. Some products actually reach the distinction of hundreds of RFEs cut against them.

The RFE race is shocking enough, but compounding it is a low average percentage of functions and features that customers in enterprise software actually exercise. At *XP 2002*, Jim Johnson of the Standish Group reported that 45% of application features and functions provided in enterprise software releases are never used, while 19% are rarely used.² In other words, close to two-thirds of features and functions in a software product are practically worthless to both customer and vendor.

Bottom line, the value of the release construct from the customer’s perspective is dubious; a small percentage of functions and features are actually exercised, while a great many are flagged as missing (in the form of the RFE). If you accept this premise, the natural question to ask is, “Well, then, is the release concept really useful to engineering?”

Worshipping the Gods We Created Ourselves

In reality, the release is a fiction that we in engineering maintain through the software configuration and change management (SCCM) tool. Look under the wraps of an SCCM, and you will find any number of

The *Executive Update* is a publication of the Agile Product & Project Management Advisory Service. ©2008 by Cutter Consortium. All rights reserved. Unauthorized reproduction in any form, including photocopying, faxing, image scanning, and downloading electronic copies, is against the law. Reprints make an excellent training tool. For information about reprints and/or back issues of Cutter Consortium publications, call +1 781 648 8700 or e-mail service@cutter.com.

branches in the code. The code tree, with all of its coding history, is the real representation of the software. The release is merely a branch in the code tree. One branch or another can usually be designated as “The Release.” The decision as to which branch constitutes the official release can, and often does, happen at the last minute. For most practical purposes in the engineering context, the release is primarily a packaging and distribution decision.

Viewing the release as a packaging and distribution decision demystifies the concept and puts the rituals that developed around it in proper perspective. Developing or building software (or any other entity) is one thing; packaging and distributing it is quite another. One cannot help wondering whether a one-size-fits-all release philosophy is appropriate for current circumstances and technologies. What if I chose to package and distribute my software in one way to benefit one customer but in quite a different way for another customer, thus providing a different set of features for each? Customer support might not be excited about maintaining two releases simultaneously, but the two customers would love it!

Obviously, the question of scalability is of paramount importance here. Maintaining two or three releases simultaneously is a fairly common practice and is often part of a software company’s currency policy with respect to support for previous releases. The question is: what about 50 simultaneous releases?!

Release No More

During a recent presentation at *Agile 2008*, Jeff Sutherland indicated that PatientKeeper produced 45 releases in one year.³ This data is consistent with the way significant parts of the retail division at Amazon.com function.⁴ Obviously, the notoriously long-duration beta exercised by Google is of the same ilk.

A release-per-week pace practically renders the traditional release concept obsolete. The software becomes much more of a fluid, alive, and evolving entity that is not subordinate to an old-fashioned packaging and distribution pattern. For most practical purposes, the software becomes continuous.

To successfully operate in a “the software is alive and always evolving” manner, four questions need to be answered:

1. How do engineering and support not lose control and cohesion over numerous releases?
2. How does the customer keep up with the flow of releases?
3. What business design is appropriate for an “avalanche” of releases?
4. How does “the software is alive and always evolving” thinking relate to adjacent trends in software?

I will address these questions in forthcoming *Updates*.

ACKNOWLEDGMENTS

I am indebted to Annie Shun, Melody Locke, and Walter Bodwell for reviewing an earlier version of this *Update* and astutely commenting on it.

ENDNOTES

¹The term “RTM” (release to manufacturing) will be used in this series. In a conventional software process, RTM precedes general availability by a few weeks.

²Johnson, Jim. *Chaos Report*. The Standish Group, 2002. Presented at *XP 2002*.

³Sutherland, Jeff et al. “Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams.” *Proceedings from Agile 2008*. IEEE Computer Society, 2008, pp. 339-344.

⁴Furthermore, Amazon’s retail division tends to skip the preproduction stage.

ABOUT THE AUTHOR

Israel Gat is broadly recognized as the father of the agile transformation at BMC Software, a systems management company that, under his leadership, has increased to 1,000 Scrum users from none in four years. Dr. Gat’s executive career has spanned some of the world’s top technology companies, including IBM, Digital, Microsoft, and EMC. In addition to his experience with industry giants, he is also well versed in growing smaller companies and has held advisory and venture capital positions for companies in new, high-growth markets. Dr. Gat holds a PhD in computer sciences from the Israeli Institute of Technology and an MBA from Clark University. He can be reached at Israel_Gat@bmc.com.

Workshop Developers/ Presenters

Every workshop is led by one of Cutter Consortium's expert Senior Consultants — experienced IT professionals who have honed their skills and developed their methodologies over years in the field, at companies like yours.

Verna Allee
 Sanjiv Augustine
 Rob Austin
 Christopher Avery
 Sam Bayer
 Kent Beck
 E.M. Bennatan
 Bob Benson
 Tom Bugnitz
 David J. Caruso
 Ken Collier
 Rachel Davies
 Tom DeMarco
 Sid Henkin
 Jim Highsmith
 Wendell Jones
 Jeff Kaplan
 Joshua Kerievsky
 Bartoz Kiepuszewski
 Tim Lister
 Michael Mah
 Terry Merriman
 Larissa Moss
 Ken Orr
 Carl Pritchard
 Ken Rau
 Thomas Redman
 Suzanne Robertson
 Mike Rosen
 Michael Schmitz
 David Spann
 Rob Thomsett
 John Tibbetts
 Jim Watson
 Karl Wiig
 Bob Wysocki

Cutter Consortium
 37 Broadway, Suite 1
 Arlington, MA 02474-5552, USA

Tel: +1 781 648 8700
 Fax: +1 781 648 1950
 Web: www.cutter.com
 E-mail: sales@cutter.com



Workshops

In these times of intense pressure to make every development dollar and every development minute count, the maxim *you are only as strong as your weakest link* has never rung truer.

Moving your development organization up the productivity curve will improve the ROI of every one of your projects. Just trace this back and you'll discover the ROI in training is immense. And with training and workshops designed and delivered by Cutter Consortium's Senior Consultants, you can add to that equation the peace of mind you get from being trained by the best of the best.

Cutter Consortium offers inhouse training solutions from IT project management techniques to software development methodologies, improving data quality, architecting Web services applications, aligning business and IT objectives, and more.

Workshop Topics

- Agile Development Methodologies
- Agile Project Management
- Business-IT Alignment
- CIO Dashboards
- Data Quality
- Data Warehousing
- Enterprise Architecture
- Estimation Techniques
- Extreme Programming
- IT Strategic Planning
- Knowledge Management
- Metrics/Benchmarking
- Outsourcing
- Requirements Management
- Risk Management
- Software Development Practices
- Teamwork and Leadership
- Web 2.0 Technologies

For details about the courses offered in each of these areas, contact Dennis Crowley at +1 781 641 5125 or dcrowley@cutter.com, or visit www.cutter.com.