I’ve worked with several software organizations that always meet their delivery schedules. As the target date approaches and the project clearly won’t finish on time, the team enters a “rapid descoping phase.” They defer much planned functionality and release on the appointed date a crippled system with serious quality problems that provides no value to anyone. Then they declare the project to be a success because they delivered on time. Yes, they delivered something on time, but that something bore scant resemblance to the expected product.

Defining your product’s release criteria is an essential part of laying the foundation for a successful project. “It’s June 30, so we must be done” isn’t the best plan. Your criteria must be realistic, objectively measurable, documented and aligned with what “quality” and “success” mean to your customers. Decide early on how you’ll determine when you’re done, and track progress toward these goals. Also, make sure everyone understands the implications if they ship before the product is ready for prime time.

How Do You Know When You’re Done?

No one ever builds perfect software products that contain every imaginable function and always behave flawlessly. Every project team needs to decide when its product is good enough to go out the door. “Good enough” means that the product has some acceptable blend of functionality, quality, timeliness, customer value, competitive positioning and supporting infrastructure in place.

There’s no perfect, simple measure of software quality. The customer view of quality depends on factors such as reliability (how long they can expect to use it without encountering a failure) and performance (response time for various operations). You need customer input to the release criteria. How would your customers judge whether the product was ready for use, whether they could cut over to a new application that replaces a legacy app, or whether they’d be willing to pay for the new product?

The internal or engineering view of quality is also important. Can the software be modified and maintained efficiently during its operational life? Is documentation in place so the support staff can help users? The project might end when you deliver the system, but the product will live on for years to come.

Possible Release Criteria

There’s no universally correct set of release criteria for all projects. The indicators you choose must provide some confidence that the project will be an overall success. Weak release criteria that don’t align with the project’s business objectives can give a false sense of...
Table 1 lists actual release criteria for two products described in the public domain. One is a revised internetwork operating system that runs on a family of routers, and the second is the next release of a compiler. These criteria reflect important ideas, but some of them, such as “extensive customer exposure” and “high level of customer satisfaction,” are vague. Such fuzzy statements are subject to individual interpretation—you can’t judge precisely whether they’re satisfied. Some ambiguous words to avoid are: acceptable, adequate, appropriate, comprehensive, enough, extensive, high, improved, low, reasonable, robust, seamless, sufficient and supported.

Management consultant Johanna Rothman recommends that we write SMART product release criteria:

Specific (not vague or imprecise)
Measurable (not qualitative or subjective)
Attainable (not unrealistic stretch goals that you’ll never achieve)
Relevant (related to customer needs and your business objectives)
Trackable (something we can monitor throughout the project)

Try applying these five criteria to the two examples shown in Table 1. The release criteria for the internetwork operating system don’t make the grade. I anticipate energetic debates over what constitutes a “serious” defect and whether the internal testing and deployment were
sufficiently comprehensive. The criteria shown for the new compiler are better, provided the meaning of the word supported is made clear in every case.

To avoid disconnects and bad assumptions, consider the perspectives of various stakeholder groups. Developers don’t always know what’s really important to the users, so you should work with key user representatives to incorporate their perspectives on completeness. Watch out for conflicting release criteria. All team members need to work toward a common set of goals and make the appropriate tradeoffs.

The following sections suggest possible release criteria in several categories. Select conditions from these lists that will help you tell when it’s time to take your product out of the oven and serve it. You might have other criteria that are pertinent to your situation. Tailor these generic examples to suit your own project and keep the SMART characteristics in mind as you write them.

**Defects**

Releasing a buggy product prematurely can lead to high operational costs, user disappointment, poor product reviews, excessive maintenance costs, product returns and even lawsuits. As one quality indicator, monitor the defects discovered during development and testing. Also, you can use historical data about defect densities (defects per thousand lines of code, or KLOC) from previous projects to estimate the number of defects likely to be hiding in your next product.

Trends in defect discovery rates can provide an indication of how many defects might remain undetected in the product. Various software reliability engineering models can estimate the number of remaining defects based on the rate at which new ones are being discovered through testing. Some of those remaining defects will be inconsequential, but others could be showstoppers—you just don’t know. Consider adapting defect-related release criteria from the following general statements:

- There are no open defects of severity 1 or 2 on a 4-level severity scale.
- The number of open defects has decreased for X weeks and the estimated number of residual defects is acceptable.
- The rate of arrival of new defects is fewer than X defects per 1,000 hours of testing.
- The density of known remaining defects is fewer than X defects per KLOC.
- All errors and warnings reported by source-code static analyzers and run-time analyzers have been corrected.
- Specific known problems from previous releases are corrected, and no additional defects were introduced while making the corrections.

**Testing**

Most software teams rely heavily on testing to discover defects, although other quality practices, such as inspections and code analyzer tools, are also valuable. But testers may not really know when they’re done testing or what it means when they declare testing complete—they often stop testing just because they’re out of time. Some better testing-related release criteria are:

- All code compiles, builds and passes smoke tests on all target platforms.
- 100 percent of integration and system test cases are passed.
• Specific functionality has passed all system and user acceptance tests (for example, the normal flows and associated exceptions for the most commonly executed use cases).
• Test case execution specified in the test plan is complete for all documented functional requirements.
• The mean time between failures is at least 100 hours (that is, your interactive application runs continuously for at least 100 hours without something going wrong).
• Predetermined testing coverage targets for code or requirements (for example, functional requirements, use-case flows or product features) have been achieved.

You need multiple, complementary criteria. For example, if no one does any testing in a certain period, obviously you won’t find any defects that way. Therefore, the rate of defect discovery alone isn’t an adequate quality indicator.

Quality Attributes

Another aspect of software quality concerns various attributes that characterize the product’s behavior. These quality attributes include reliability, security, integrity, usability, portability, maintainability, efficiency, robustness and interoperability, among others. You can assess the product’s readiness for release partly by whether it satisfies its critical quality attribute requirements. For example, quantitative performance goals should be satisfied on all platforms. Reliability goals also should be satisfied. As an example of a reliability goal, the mean time between failures is at an acceptable level and is increasing with each build that goes through system test. These days it’s imperative that your product satisfy specified security goals and requirements. See http://rssr.jpl.nasa.gov/ssc2/questions.htm for a sample checklist of specific security issues and threats to examine. In some cases, your product must meet specified conditions that will enable it to pass a necessary certification audit or evaluation.

Functionality

Every product needs a minimal set of functionality that provides appropriate value to the customer. Thoughtful requirements prioritization lets your team deliver a useful product as quickly as possible, deferring less urgent and less important requirements to later releases. Consider the following requirements-related release criteria:

• All high-priority requirements committed for this release are implemented and functioning correctly.
• Specified customer acceptance criteria are satisfied.
• All requirements for accessibility by disabled users are satisfied.
• All localization and globalization tests to ensure that the software runs properly in different languages are passed.
• Specified legal, contractual, standards-compliance and regulatory goals are met.
• All functional requirements trace to test cases.

Configuration Management

Configuration management involves uniquely identifying the components that are assembled into the product, managing and controlling versions of these components, and building them properly into the deliverable files and documents. Before pushing your product out the door, ask some configuration questions. Can the product be built reproducibly on all target
platforms? Has a physical configuration audit confirmed that the correct versions of all components are present? Does the product install correctly on all target platforms, including reinstall, upgrade from a previous release, uninstall and recovery from aborted or failed install attempts? And are the release image and media free from viruses?

Support

You might think the software is done, but that doesn’t mean the world is ready for it. Make sure you’ve lined up the other elements necessary for a smooth rollout and implementation. Be sure that release notes have been prepared that identify defects corrected in this release, enhancements added, any capabilities removed, and all known uncorrected defects. Log any known uncorrected defects in the project’s defect-tracking system. Confirm that software release and support policies and procedures are published and understood by affected stakeholders. The support function must be ready to receive and respond to customer problem reports. Also, the operating environment needs to have all necessary infrastructure in place to execute the software. In addition, the software manufacturing and distribution channels should be ready to receive the product. Manufacturing is especially important when the product includes software embedded into hardware components.

Making the Call

You also need to know who will monitor progress toward satisfying the release criteria and who will make the final release decisions. I use the plural here because you’ll probably have several types of product releases. You might deliver a late-stage build to quality assurance as a release candidate for final system testing. After the product passes its QA checks, you might release it to a beta site for user acceptance testing, deliver it to manufacturing, or approve it for general availability release. Information systems being developed for internal corporate use typically go through a sequence of quality verification, beta test in a staging environment, and release to production. Different stakeholders might make these various release decisions, and different release criteria will pertain to each decision point. Think about how, when, by whom and to whom the results of the measurements will be reported. Assign responsibility to those who will evaluate and interpret the measurements. Very importantly, determine who will make the ultimate decision that the criteria are satisfied and the product is ready for release.

Track your release criteria in a binary fashion: Each one either is or is not completely satisfied. You might create a color-coded tracking chart for your release criteria. Green indicates that the criterion is completely fulfilled, red means that the criterion is not yet fulfilled, and yellow indicates a risk of possibly not being able to completely satisfy the criterion. Avoid using yellow as an indicator for partial satisfaction of the release criterion. If the board isn’t all green, you aren’t done.

Make sure the project’s management sponsor agrees that the release criteria are appropriate indicators of likely business success. Also, obtain the sponsor’s commitment to rely on these criteria to make the release decisions. If management elects to override the objective release criteria indicators, the team won’t take release criteria seriously on future projects.

As the scheduled delivery date fast approaches, you might conclude that you won’t be able to satisfy certain release criteria. You might need to reevaluate those criteria or to modify them. Maybe marketing’s concerned about missing the optimum time to release, but is it in the company’s best interests to release an incomplete or flawed product on schedule? The release criteria you select early on should reflect the appropriate balance of product features, quality,
timeliness and other factors that align with business success. You shouldn’t have to change these criteria unless business objectives or other key project realities change. Identify the risks of failing to satisfy each release criterion so the stakeholders understand the implications of ignoring or finessing them.

Next Steps

It’s never too late too develop release criteria for your product. Start by listing several indicators that tell you when the product is ready to go. Make sure they’re specific, measurable, attainable, relevant and trackable. Determine what you can measure to see whether you’re approaching satisfaction of that criterion. For those that can be quantified, state the target value that would indicate satisfaction of the criterion. In addition, identify the project stakeholders who will make each of your various release decisions and describe the decision-making process each group will use. Agreeing on these end-game issues early on helps lay a solid foundation for achieving project success.

SIDEBAR: Precise Release Criteria with Planguage

Release criteria are often written in a qualitative, subjective style that makes it difficult to know exactly what is intended or when you’ve satisfied them. To address this problem, consultant Tom Gilb has developed a notation that he calls “Planguage”; see Gilb’s book Competitive Engineering (Butterworth-Heinemann, 2005). Planguage—derived from “planning language”—permits precise specification of requirements, project business objectives and release criteria.

In Planguage, each release criterion receives a unique tag or label. The ambition describes what you’re trying to achieve. Scale defines the units of measurement, and meter describes precisely how to make the measurements. You can specify several target values. The must criterion is the minimum acceptable achievement level for the item being defined. You haven’t satisfied the release criteria unless all must conditions are completely met. The plan value is the nominal target, and the wish value represents the ideal outcome. Expressing your product release criteria using Planguage provides a precise and explicit way to know when your product is ready to go out the door.

Here’s an example of how to express a release criterion using Planguage:

TAG: Reliability.Conversion
AMBITION: Very low frequency of failures when converting archive data files into current data formats.
SCALE: Percent of conversion failures when performing published supported file conversions.
METER: Convert 400 files representing an operational profile of the supported archive data file formats. Express the number of failed conversions as a percentage of the attempted conversions.
MUST: No more than 5 percent.
PLAN: No more than 1 percent.
WISH: No more than 0.1 percent.