Standards, Agility, and Engineering

François Coallier, École de technologie supérieure

Engineering involves choosing the right tool, which implies an understanding of both the tools and the problem. Such behavior is agile.

Parts of the software and IT engineering community harbor many misconceptions associated with standards, especially process standards. The most common are that standards are rigid, obsolete, and plain boring. Standards are also perceived as being the antithesis of agility.

Yet standards create markets by enabling interoperability and economies of scale through component and interface standardization. In the IT service industry, standards also enable interoperability between organizations through process and competency benchmarking and standardization.

These benefits derive from interoperability between people and organizations that share a common vocabulary, compatible work processes, and the ability to benchmark individuals and organizations. Thus, we see a growing need for standards in areas such as processes, methods and good practices, bodies of knowledge, and formalisms.

STANDARDS DEFINED

Using the International Organization for Standardization (ISO) definition as a baseline, standards are "guideline documentation that reflects agreements on products, practices, or operations by nationally or internationally recognized industrial, professional, trade associations or governmental bodies."

Standards are referred to as guideline documents because they are not compulsory unless mandated so by an individual, an organization or the market. They are agreements because they often reflect a specific level of consensus.

Many types of standards exist and can be elaborated inside an organization; a professional society such as the IEEE International Council on Systems Engineering (INCOSE) or the IT Service Management Forum (itSMF); an industrial consortium such as the Object Management Group (OMG), the Organization for the Advancement of Structured Information Standards (OASIS), and others; formal national bodies such as the Americal National Standards Institute (ANSI), the Standards Council of Canada (SCC), and others; or international organizations such as ISO, the International Electrotechnical Commission (IEC), and the International Telecommunication Union (ITU).

A given standard might be developed in one environment—market, professional, industry, national—and then migrate into a formal international standard. Market, professional, and industry standards might also represent an international consensus or de facto state that differs from formal international standards in the degree of breadth and formality that this consensus embodies.

STANDARDS LIMITATIONS

Standards are usually elaborated through a consensus process. This means that a group of experts creates the document through a series of iterations. Consensus is built among these experts from successful compromises reached through multiple negotiation sessions. Afterward, the document undergoes review by a larger body of experts and organizational representatives, with the goal of reaching a greater level of consensus to meet the final publishing criteria.

All of this underscores the fact that standards, like any other human creations, are far from perfect. Also, because they represent the consensus of a collection of individuals and organizations, they do not necessarily represent the state of the art or the most optimal solution in a given area. Rather, standards are conventions or baselines that a collection of people or organizations reach at a given point. This applies to all types of IT standards.

AGILITY

Either as a philosophy or, more explicitly, as a collection of concepts embedded into methodologies, *agility* has been a popular topic in the software and IT engineering world for many years. Succinctly, an agile approach to development is essentially a result-focused method that iteratively manages changes and risks. An agile method also actively involves customers, making them essentially part of the development team.

Agility is often contrasted to process-oriented approaches in which a rigid, documentation-heavy methodology is followed with very detailed specifications, and the relationships between developers and the customer are formal and sometimes adversarial.

As with many such concepts, the agility characteristic is not binary, as Figure 1 shows (modified from P. Kroll and P. Kruchten's *The Rational Unified Process Made Easy—A Practitioner's Guide to the RUP*, Addison-Wesley, 2003).

In Figure 1, the *x* axis represents the degree of documentation and formality associated with a development process. "High ceremony" means many formal, documented meetings and heavy, process-driven documentation, while "low ceremony" means less formal meetings whose sole documentation might be in the form of updated process artifacts and more, lighter, outcome-driven documentation.

In Quadrant II, the process-based approaches have much in common with industrial engineering practices. In Quadrant III, the focus shifts to people and their skills. At the upper-hand corner of Quadrant II, a fixed-cost contract is in place, and lawyers can be involved when the developers and customer communicate about the project. At the lower corner of Quadrant III, "pay as you go projects" reside. In these, the customer forms part of the team. While agile methods are positioned in Quadrant III, extreme programming (XP) is usually positioned in the lower corner of this quadrant.

Agility is thus both a philosophy and a nonbinary attribute of a development process.

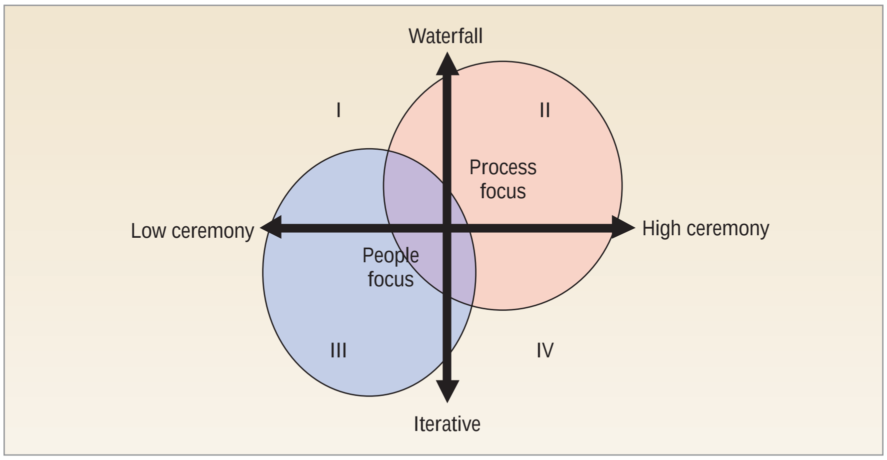


Figure 1. The spectrum of development approaches. The x axis represents the degree of documentation and formality associated with a development process: High ceremony involves many formal, documented meetings; low ceremony involves less formal meetings with less documentation.

ENGINEERING

While the objective of science is to understand nature, engineering's objective is to build useful things using science. Put another way, scientists do reverse engineering while engineers do forward engineering. Engineers must not only build things, they must also do so within budget, schedule, resources, regulatory, and operational constraints.

When developing IT systems, engineers must not only deliver functionalities, but also all those other "ities" commonly referred to as either "nonfunctional requirements" or "quality attributes." The quality attributes include performance, reliability, availability, security, usability, scalability, and maintainability. These attributes are not only important for embedded software systems, but also for modern information systems. Delivering those quality attributes can, in many cases, be much more challenging than delivering the functionality. All of this points out that engineering IT systems is much more involved than doing classical programming.

Many IT systems that engineers build can be safety-critical or, in the case of financials and electronic commerce systems, financially critical. In many cases, the best way to minimize risks when developing such systems, and any other type of software systems is through reuse or incrementally improving proven solutions. This means that engineering is by nature a profession where reuse—be it of proven standardized components, interface specifications and protocols, development methods, or architectural patterns—is a way of life.

Practicing reuse does not mean that engineering is boring technical work without any challenges. On the contrary, it requires a deep understanding of the problem to be solved and its applicable solution paths. Both analytic and holistic systems thinking are necessary. Reuse does not mean the absence of innovation—it means that innovation and risks must be balanced within the project's context.

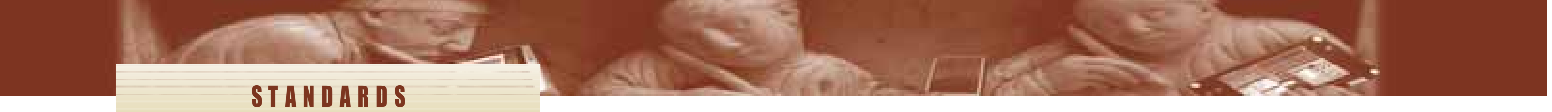
ENGINEERING AND STANDARDS

The relationship between engineering and standards should now be obvious. Because standards facilitate reuse and, usually, document proven and generally acceptable practices, they are an intimate part of every engineer's professional practice. Standards also constitute a significant part of an engineer's teaching and training curriculum.

In software engineering, the relationship with standards goes deeper. While the foundations of software engineering include fields such as computer science, systems engineering, and project management, standards have played a significant role in the development and codification of this rather young engineering discipline.

The US National Bureau of Standards published the first software engineering standard barely eight years after the term "software engineering" was coined in 1968. The same year this standard was published, the IEEE created its software engineering standards committee. In 1977, the joint ISO and IEC standard subcommittee (ISO/IEC JTC 1/SC7, or SC7) was then created.

The IEEE Computer Society and SC7 have been working together to harmonize their respective collection of software and systems engineering standards and to continue joint development. Noteworthy has been the



IEEE's publication of the *Guide to the Software Engineering Body of Knowledge* (SWEBOK Guide) in 2004, followed closely by its adoption as a technical report by ISO and the IEC as TR 19759:2005. As of August 2007, there are 101 published SC7 standards, with approximately 50 from the IEEE CS, focusing on the area of software and systems engineering.

These standards are still evolving, with new ones being elaborated that cover new areas. Six new proposals for software and systems engineering as well as IT service management are currently under consideration at SC7.

AGILITY AND STANDARDS

Some developers perceive standards as the antithesis of agility. This view is difficult to understand because all agile methodologies, including XP, are documented processes. The difference between an agile method and a non-agile one is a matter of content, but it is also a function of the way the methodology's components are applied in a given project.

A project where safety-critical software is developed can be agile even if many of the project artifacts must be produced because of regulatory constraints. The agility in such a case can be in the collaborative nature of the customer relationship, a people-focused work environment, or other factors.

Many process standards also have built-in provisions that, implicitly, enable their use in an agile context. These take the form of a tailoring clause. While machine-to-machine interface standards provide rather straightforward engineering specifications and constraints, the process standards used as a baseline to define a methodology are very different.

ENGINEERING AND AGILITY

Engineers are, in principle, pragmatic professionals strongly focused on their work's outcome. This implies agility. The multitude of patterns, regardless of their nature, that engineers learn through formal training and their professional career are all part of a toolbox. This is true also for process patterns and methods. Competent engineers who understand the contents of their toolbox and who also understand the nature of the problem that must be solved should use the proper tools. Such behavior is agile.

The people dimension of agility involves being able to choose the right tool, which implies an understanding of both the tools and the problem. This means having the right knowledge, skills, and experience. The horror stories that circulate about wasteful work generated by standards are generally more a result of poor engineering and management skills than a problem with the standard.

Not only are standards and agility mutually compatible, both concepts are also key components of any engineering discipline. ■

François Coallier is a professor and chair of the Department of Software and IT Engineering, École de technologie supérieure, Montréal. He is also the chairman of the ISO/IEC JTC 1/SC7, Software and Systems Engineering subcommittee. Contact him at francois.coallier@etsmtl.ca.

Editor: John Harauz, Jonic Systems Engineering Inc., Willowdale, Ont., Canada; j.harauz@computer.org