Where Now for Development Methodologies?

By David E. Avison and Guy Fitzgerald

This brief history of systems development methodologies identifies and explores eras of development and speculates on their future. Today’s “post-methodology” era involves methodologies that can be viewed by developers as outdated and inappropriate for rapid development, Web applications, and other current requirements. Perhaps we are in danger of returning to the bad old days of the pre-methodology era and its lack of control, standards, and training.

Pre-Methodology Era
The early computer applications of the 1960s and 1970s were developed and implemented without explicit or formalized development methodologies. The emphasis was on programming and solving technical problems, particularly those resulting from the rather limited hardware of the time. Developers were trained in computer technology but rarely understood the business or organizational contexts in which the systems were implemented. User needs were rarely well defined, with the consequence that system designs were often inappropriate for meeting genuine user and business requirements. The approach programmers took to development was typically individualistic, often resulting in poor control and management of projects. One emphasis was maintaining operational systems rather than developing new systems and responding to user needs. These limitations led to a new appreciation of standards and a more disciplined approach to developing information systems in organizations. The result was the first development methodologies based on the systems development life cycle.

Early Methodology Era
This era, during the late 1970s and early 1980s, was characterized by an approach to building computer-based applications focusing on the identification of phases and stages it was thought would improve the management of systems development and introduce discipline—an approach that has come to be known as the Systems Development Life Cycle (SDLC), or, more commonly, the waterfall model. It typically consisted of a number of development stages that had to be followed in sequential order, including: feasibility study, systems investigation, analysis, design, development, implementation, and maintenance. One phase had to be completed before the next one could begin (hence the term waterfall), and each phase had a set of defined outputs or deliverables to be produced before it could be deemed complete. SDLC was also associated with a set of techniques (such as flowcharting) for use in particular phases. There was also the notion of iteration around the phases, as problems were encountered or changes required, though in practice it was often ignored.

This approach also involved serious limitations [1], including the way it was used. Notable traps were: failure to meet the real needs of the business (due to concentration on technological efficiency improvements at the operational level of the organization); overly conservative systems design (due to emphasis on the existing system as a basis for
the new system); instability (due to the modeling of processes that are unstable due to changing businesses and markets); inflexibility (due to the output-driven orientation of design processes, thus making changes in design difficult and costly); user dissatisfaction (due to problems with computer-orientated documentation and users' inability to "see" the system before it is operational); problems with documentation (due to its computer rather than user orientation and the fact it is rarely kept current); and application backlog (due to the maintenance workload, as attempts are made to change the system in order to reflect changing user needs).

**Methodology Era**

A number of newer approaches emerged in response to one or more of these limitations, thus launching the methodology era. The term “methodology” was probably first used to describe these different approaches, and methodologies proliferated. A methodology is a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system; we also note the importance of the philosophy behind the methodology, or the set of beliefs and assumptions underpinning it, explaining why the methodology functions as it does. It may embody a belief that the key to successful development is user involvement and that users have a right to participate in developments that affect their lives. These beliefs and assumptions are often not stated explicitly by a methodology’s authors.

The main motivations for adopting a methodology varied by organization and individual but were generally to achieve: better end products (meeting user demands); a better development process (improving developer control and productivity); and a standardized process (enabling better systems integration and the benefits of a common approach in an organization).

We classify the methodology era’s emerging methodologies into seven broad themes, or approaches:

**Structured.** The concepts of structured programming were applied to analysis and design, and techniques (such as data flow diagramming) enabled the top-down analysis and representation of complex processes.

**Data-oriented.** The focus was understanding data as the key element in a system’s development, and the most important technique was entity-relationship modeling.

**Prototyping.** The emphasis was building an approximation or representation of the system, allowing users to visualize and respond to it prior to its physical implementation.

**OO.** The identification of objects and attributes (in whole or part) and classes helped provide the theoretical benefits of inheritance and reuse.

**Participative.** The crucial feature was involvement of users and other stakeholders.

**Strategic.** The emphasis was the planning of information systems and development of an information systems strategy to support and enable the overall objectives of the business; business process reengineering is sometimes viewed as a development approach focusing on strategy.

**Systems.** The complexities of human activity systems were addressed by adopting a holistic view far beyond a system’s single-application boundaries.

These approaches were not necessarily mutually exclusive. The era saw the coming together, or blending, of a number of themes within a single methodology; the method-engineering movement embraces this view. Associated tools supporting many of the approaches were also developed, including project management software, systems repositories, drawing tools, and computer-assisted software (or systems) engineering (CASE) tools.

Although they characterize the methodology era, not every organization at the time used a methodology, particularly one with a name and that was developed by and bought from a vendor, whose claims, in any case, would certainly have to be viewed with caution. A 1999 survey of U.K. organizations [5] found that 57% of the sample claimed to be using a methodology for systems development, but only 11% reported using an unmodified commercial development methodology; whereas 30% reported using a commercial methodology adapted for in-house use, 59% used a methodology they claimed as unique to their organization, often internally developed. Thus, methodologies were in relatively widespread use, though they were often developed in-house rather than as commercial products. Despite the claims of being unique, many were clearly influenced by existing commercial methodologies. Even those organizations not using explicit methodologies were influenced by the methodology movement through, say, associated tools and techniques.

**Post-Methodology Era**

The success or failure of development efforts cannot be attributed exclusively to the use, misuse, or nonuse of methodologies, but the post-methodology era starting in the late 1990s has been characterized by a serious reappraisal by researchers and practitioners alike of the concepts and usefulness of the earlier methodologies. As a result, some organizations continue to turn to yet different (perhaps newer) methodologies and
approaches, while others have abandoned methodologies altogether. For many organizations, adoption of a methodology has not always worked out or been the success its advocates touted. Indeed, it was highly unlikely that methodologies would ever achieve their more overblown claims, like productivity increases, made by some vendors and consultants. Real-world performance has led some developers to reject methodologies in general terms and attack the concepts (such as step-by-step development and meticulous documentation) on which they are based.

Many reasons for this developer backlash have been suggested. The most common is probably disappointing productivity; another is that the methodologies are overly complex, usually designed for the largest and most complex development projects. They may lead to developing requirements to the ultimate degree, often over and above what is legitimately required, sometimes encouraging users to create unrealistic wish lists. They also require highly technical skills that can be difficult and expensive for developers and end users to learn or acquire. Moreover, the tools advocated by methodology proponents can be costly, difficult to use, yet still not deliver enough benefit.

Methodologies are often not contingent on the type or size of a project, nor upon the technology environment and organizational context. A methodology is often said to be one-dimensional, that is, it adopts only one approach to the development of projects that may well not address a particular organization’s underlying issues or problems. Few recognize or address the critically important social, political, and organizational dimensions of development.

A methodology may be inflexible, not allowing changes to requirements during development. Most methodologies make a number of simplifying yet invalid assumptions (such as a stable environment, a well-documented business strategy, users knowledgeable about their own requirements, or that a consensus of requirements can be achieved). Rarely do such conditions exist in practice. The use of a methodology in an organization may lead to its rote implementation and to a focus on following the procedures of the methodology rather than on addressing the real implementation and business issues. Strict adherence to the methodology rule book has been described as slavish adherence to the methodology and the fetish of technique, that is, the methodology is allowed to inhibit creative thinking [8]. Some organizations have found it difficult to adopt methodologies in practice, confronting resistance from both developers and users. Finally, and perhaps the acid test of whether a methodology is really worth using, is the conclusion of some organizations that a methodology led to poor systems, for whatever reason. A typical reaction might be, “We’ve tried it, and it didn’t help and may have actively hindered systems being implemented.”

Some organizations have rejected the use of methodologies altogether, returning to less-formal, more off-the-cuff, perhaps more flexible approaches. One area where methodologies are not widely used is development of Web-based applications [7]. They are typically developed in an ad hoc, trial-and-error manner, relying on the skills and experience of a few key people in an organization [3], not unlike the development approach usually associated with the pre-methodology era of the 1960s and 1970s.

Alternatives

For other organizations, the problem is not the concept of a methodology but the inadequacy of the current methodologies, prompting them to keep looking for different and better ones. Still others seek, not better methodologies, but alternatives to traditional in-house systems development; some are outlined in the following sections:

Development with tools. Some organizations pin their faith on the tool evolution, including automatic code generation, to automate the development process. Using tools without a methodology to guide their use can, however, be considered a return to ad hoc development. OO approach. Although part of the methodology era, the OO approach is still evolving with new methods and techniques. Component-based development, which views systems development as the combination and recombination of existing components (usually OO), is being adopted by some software engineers.

Incremental development. Incremental, or evolutionary, development reflects the characteristic of building upon and enhancing the previous versions of systems, rather than developing whole new systems each time. It aims to reduce the amount of time needed to develop a system, especially in the form of Rapid Application Development. It addresses the problem of changing requirements during the process of development; for example, the Dynamic Systems Development Method is an evolutionary methodology that has been adopted by a number of companies since the mid-1990s [6].

External development. Some organizations try satisfying their systems needs by buying commercially developed methodology packages. Although their purchase and implementation has been commonplace for some time, the post-methodology era is characterized by some organizations purposely not embarking on in-house systems development activities, satisfying all their requirements in the form of packaged systems. This is regarded by many organizations as a quicker and more cost-effective way of implementing systems.
Only systems that are strategic or for which a suitable package is not available would be considered for in-house development. The market for packaged development software offers increasingly sophisticated products and features and includes more and more tailor-made packages. Enterprise resource planning packages are especially popular with large corporations, as are customer relationship management packages. However, there are notable disadvantages associated with being locked into a particular supplier and of not being in control of the features in a particular package, a risk discounted by many organizations.

Outsourcing. For still other organizations, the continuing problems of systems development and the perceived failure of methodologies to deliver promised performance, has resulted in their outsourcing development to third parties. The client organization is no longer as concerned with how a system is developed and which development approach or methodology is used, but with the effectiveness of the system ultimately delivered. A client organization has to develop sharp skills to be able to select appropriate vendors, specify requirements in detail, and write and negotiate contracts, rather than just apply methodologies [4].

Contingency. Most methodologies today are designed for situations that fit some ideal specification, whether stated or unstated. However, some researchers and practitioners argue that each situation is different and there is no such thing as an ideal in the real world. Such thinking advocates a contingency approach to systems development whereby a structure is presented, but tools and techniques are expected to be used or ignored (or used and adapted), depending on the situation. Situations might differ depending on, say, the type of project and its objectives, the organization and its environment, the users, and the developers and their respective skills. The type of project might also differ as to purpose, complexity, structuredness, degree of importance, projected life, and potential contribution to overall corporate performance. Different environments might exhibit different rates of change, number of users affected by the system, user skills, and analyst skills. All these characteristics can influence the choice of required development approach. A contingent methodology (such as Multiview [2]) allows for different approaches, depending on the situation.

Diversity
Thus there is diversity in terms of types of methodologies and their components, functions, and potential results in the post-methodology era. For some developers it represents the abandonment of methodologies altogether; for others, it represents improved methodologies while moving away from the highly bureaucratic types of methodologies of the methodology era; and for others it represents moving beyond in-house development altogether.

Diversity characterizes the current post-methodology era. However, today’s flood of methodology options should not be interpreted as the death of methodology; many traditional methodologies are still being used, along with alternative methodologies and approaches. Moreover, even with the developer backlash against formal methodologies, they remain influential in practice. They might be adapted or other techniques and tools used but still contribute in practice; they are also important in training and education, teaching good practice and forming a solid basis for understanding systems development.

Conversely, the current era may indeed be more a methodology era than is often supposed. Systems development methodologies may not be adopted by all organizations, but they weren’t during any particular period. We might even claim methodologies are being used more than ever. In light of today’s diversity, organizations are much more likely to find an appropriate approach for their systems development work, even though they rarely find a single clear strategy for doing so. Finally, the risk for organizations not using any methodology at all should be recognized and the lessons of history not completely ignored.  

References

David E. Avison (avison@essec.fr) is a professor in the Information and Decision Systems Department of the ESSEC Business School, Cergy-Pontoise, France.
Guy Fitzgerald (guy.fitzgerald@brunel.ac.uk) is a professor of information systems in the Department of Information Systems and Computing at Brunel University, Uxbridge, Middx, U.K.