For the past 15 years, many proposed OO methodologies have sought to deliver full life-cycle support for systems development using OO tools. Many are highly prescriptive, that is, the methodology elements are highly interconnected. This inherent complexity makes it difficult for a methodology to be adapted to project-specific circumstances, especially (and usually) when project managers and developers are advised by the methodology’s creator that they must use all or none of the components of the methodology. Indeed, even with the advent of agile methodologies, frequently touted by their developers as offering more people-focus and flexibility, proponents often say an agile methodology like extreme programming (XP) must be followed in its entirety and that XP without, say, pair programming is not XP.

This metamodel-based framework helps distill the key ingredients in software engineering processes in ways that facilitate method engineering, along with process understanding.
A less popular alternative is provided by way of the theory of method engineering, which subsumes process engineering and product engineering. While discussed in the academic literature and implicit in ISO standards like 12207, it does not seem to be widely acknowledged or practiced by software engineers. Method engineering offers flexibility but is often viewed (unfairly) as also having a costly overhead (in terms of time, money, and people). Method engineering contrasts with the use of out-of-the-box methodologies presented as ready for immediate use. The cost and effort to the organization are often totally ignored when such a prepackaged methodology is used and found to be an inappropriate description of the system-user company’s business processes.

Here, I explore (both theoretically and practically) the rationale behind a method-engineering approach, as well as its potential adoption by systems development teams.

**Method Engineering**

A methodology (or method) includes both process aspects and work-product descriptions. Considering not only the methodology per se but the components that go together to form it, we can use the ideas of method engineering to construct a full methodology from the elements, also known as method fragments. Typically, method fragments are stored in a repository underpinned by a metamodel. Situational (or situated) method engineering is defined as the creation of a method(ology) specifically attuned to the project at hand [1, 2, 11]. I refer to both kinds of method engineering—situational and nonsituational—under the generic label of method engineering.

Methodological approaches compatible with the theory of method engineering include Process Instance Evolution (PIE) [3], the ISO/IEC 12207 standard [10], and, in the OO arena, the Object-oriented Process, Environment, and Notation (OPEN) framework and the OPEN Process Framework (OPF) [4]. However, these approaches tend to focus on the process-engineering elements of method engineering and assume that the product side is addressed through modeling languages like UML. While ISO 12207 and OPF have many similarities in scope and granularity, the ISO model lacks a metamodel and adequate construction guidelines—both found in OPF.

**OPEN Process Framework**

As noted earlier, I bias my discussion of how to apply method engineering toward the more novel and interesting aspects of process engineering. The definition of “process” should include not only descriptions of phases, activities, tasks, and techniques but issues associated with human resources, technology, and the life-cycle model to be used. In contrast, the capability assessment and standards communities tend to use the term “process” at a smaller scale, more like “activity” in OPF and “discipline” in the Object Management Group’s Software Process Engineering Metamodel (SPEM).

Defining process as the transformation of input to output (as in ISO/IEC12207 [10]), the notions of process and process improvement in many ISO standards and
Software Process Improvement (SPI) contexts underline the granularity of interest often referred to as method fragment, method chunk (such as [11]), or process component (such as [4]). Here, I investigate how process engineering can be accomplished in the context of OPF.

An important part of OPF is its comprehensive library of process components used in a variety of software projects and that cover changes in technology. OPF includes five major classes of these components (see Figure 1), all defined in the metamodel:

**Work Product.** “A Work Product is anything of value produced during the development process” [4]. Work Products are the result of Producers (people) executing Work Units and are used either as input to other Work Units or delivered to clients. They also include externally supplied (such as by users) preexisting artifacts used as inputs to Work Units.

**Producer.** “A Producer is responsible for creating, evaluating, iterating, and maintaining Work Products” [4].

**Work Unit.** A Work Unit is a functionally cohesive operation performed by a Producer. The three major classes of Work Unit are Activity, Task, and Technique.

**Language.** A Language is used to document a Work Product.

**Stage.** A Stage is an identified and managed duration within the process or a point in time at which some achievement is recognized.

The OPF repository (see Figure 2) contains a range of predefined instances for each class and subclass in the metamodel; for example, there are about 30 predefined instances of Activity, 160 instances of Task, 200 instances of Technique, and 76 instances of Role. In addition, users can add extra components from their own best practice.

Typically, someone in the development organization, in the role of process engineer or method engineer, selects appropriate process/methodology components from the repository and combines them to form an actual process within the methodology (see Figure 3). This procedure is often called process construction and may (in one step) create a project-specific process directly or a two-step process to construct first an organizational standard process and then a project-specific process. Construction decisions need to account for myriad variables pertinent to the development organization. Variables include, but are not restricted to, Capability Maturity Model level of
maturity, available skills, available tools, quality desired, time scales, degree of ceremony, and number of people on the development team. While most process construction today is carried out by a specialist (the process/method engineer), emerging embryonic software tools assist this engineer in doing it effectively and efficiently.

Other ideas proposed in the academic literature not yet fully utilized include the Method Engineering Process Model of [11] and the method assembly rule set (specified using a formal language) of [1].

Each process component should be optimal, say, the best set of Work Units, Activities, Tasks, and Techniques. OPEN recommends construction of a number of matrices linking, for example, Activities with Tasks and Tasks with Techniques. The possibility values in these matrices indicate the likelihood of the effectiveness of each individual pair [8]. These values should be tailored to a specific organization or a specific project. While five levels—mandatory, recommended, optional, discouraged, forbidden—are recommended in published texts, I personally encourage individual organizations to use the matrices for process construction at the project level, perhaps using only two (use/do not use) or three (use/do not use/optional) values.

Table 1 outlines typical Task/Activity pairs for a hypothetical process to build a small piece of software when the requirements are fixed. I have identified eight appropriate Tasks for the six selected Activities. To fulfill these Tasks, a number of Techniques must be identified by the process engineer. In all, 25 are selected (in this example), as shown in Table 2. Finally, some time sequencing is added by using the Stage class of the metamodel, as in Figure 1.

The synergy between the stage view and the process elements results in an overall OPEN process that may be oriented toward a particular organization or project or, more broadly, a specific domain (such as Web applications, business reengineering, or real-time software) (see the sidebar “Examples of Method/Process Engineering” for application domains). Once developers determine which process they’ll use, this process is enacted on a real project (including a method/process instance, as shown in Figure 3) through the allocation of resources, including roles played by personnel and time scheduling, as well as money and hardware.

The method/process engineering approach has many advantages over an out-of-the-box approach. The process is created by members of the organization itself, thus giving them ownership of the resulting OPEN process; everyone has bought into it because they have had the opportunity to contribute to its formation. Along with local ownership, there is also global support because the framework and the repository from which the process was constructed are identical to the ones used by many other organizations worldwide. One can thus readily participate in user group meetings and forums. In addition, there’s a good chance that

<table>
<thead>
<tr>
<th>Task</th>
<th>Activity</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construct the object model</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop and implement resource allocation plan</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>develop timeplan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set up metrics collection program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>specify quality goals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Evaluate quality</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify CIRTs (Class, Instance, Role, or Type)</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Map roles onto classes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x (OOP)</td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write manuals and other documentation</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2. Technique/Task matrix for the same example as in Table 1 [7].**
new hires will have the necessary skills gained from a worldwide community standard rather than from some idiosyncratic in-house approach.

Finally, since many such process instances can be created from a common framework, process engineers can create a family of processes for various industrial demands, including, for example, safety-critical software, background financial processing, and real-time stock market modeling.

REFERENCES
Examples of Method/Process Engineering

The following illustrative examples of method/process engineering are outlined in terms of process instances of OPEN, including an application in Web development, what a process for inserting a process into an organization might look like, an agile application, and how that application can be used with agents.

Web development. How is Web development different from traditional software development? Differences include shorter time scales, a much tighter link between business models and the software architecture, and greater importance on the content management aspects of Web sites for evaluating an application’s business success. Each of these factors has a potential effect on the development process and hence on the form taken by an OPEN instance through the method-processing engineering approach described here. How Web OPEN was created is described in a number of articles, including [5, 8]. The new tasks and techniques introduced in this Web-focused project are summarized in the table here.

Organizational transition. This process is an interesting use of OPF. Rather than choosing instances of Work Units from the repository for creating a software-development process, a process is created by the process engineer to support an organization’s transition from one culture to another—in this case, from non-OO to 00/CBD (component-based development).

The process thus constructed [9] can be used to migrate an organization from a non-process-focused environment to one in which the 00/CBD culture is inculcated and the OPEN process (an instance tailored for software development this time) is used. The experience of one organization—a legal publisher—in adopting OPEN was documented in [12].

Agile processes. To construct an agile instantiation of OPF, one might identify, say, four major activities: coding, designing, reviewing, and testing. The associated tasks and techniques discovered are usually determined more informally, in contrast to the formal matrix template approach outlined in this article. These tasks include, for example, those recommended in XP (such as pair programming, the planning game, system metaphors, and refactoring)—all elements I “borrowed” from XP for inclusion in the more recent versions of the OPF. The current OPF repository thus readily supports agile or lightweight process construction [6].

In this newer, agile environment, process engineers might also wish to augment these OPF process components with new ones of their own design or choosing.

Agent-oriented methodologies. While I focus here on the OO software development environment, a natural extension—in the form of agent-oriented methodologies—is beginning to emerge into the mainstream of computing. As part of this emergence, we have created an initial version of OPEN to specifically support agent-oriented software systems development. In doing so, 13 new tasks and 10 new techniques were identified as needing to be added to the OPF repository. They focus on supporting the internal architecture of agents, including deliberative and reactive reasoning and architectural models, including Beliefs, Desire, Intentions, without having to change the metamodel. Work is ongoing in the Agent OPEN project based at the Centre for Object Technology Applications and Research.

New activity
• Web site management

New tasks
• Design content management strategy
• Create content (on Web site)
• Undertake content management
• Build white site/wireframe prototype
• Design Web site standards
• Design Web site architecture
• Create navigation map for Web site
• Define acceptance criteria for Web site
• Define Web site testing strategy
• Undertake testing of Web site

New techniques
• Web metrics
• Web template
• Branding

New roles
• Web designer (HTML/page construction)
• Graphic designer (visuals)
• Prototype developer

New Work Units useful in Web development projects [7].