Introduction to the Enterprise Unified Process (EUP)

Scott W. Ambler
Methodologist
www.ambysoft.com/scottAmbler.html

This Version: October 15, 2005

Source Material

This whitepaper summarizes material from the book *The Enterprise Unified Process: Extending the Rational Unified Process* by Scott W. Ambler, John Nalbone, and Michael Vizdos, published by Prentice Hall PTR, February 2005.

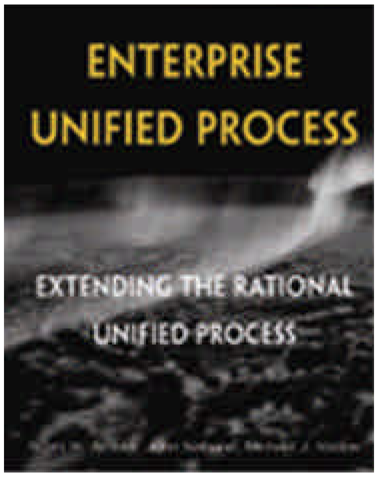

Table Of Contents

A software process is a set of project phases, stages, methods, techniques, and practices that people employ to develop and maintain software and its associated artifacts (plans, documents, models, code, test cases, manuals, etc.). In this white paper I overview the Rational Unified Process (RUP) (Kruchten 2004) and show how it can be extended to form the Enterprise Unified Process (EUP).

1. The Rational Unified Process (RUP)

The RUP is an endeavor of Rational Corporation, now IBM Rational (a division of IBM), the same people who introduced what has become the industry-standard modeling notation, the Unified Modeling Language (UML). The heart of the RUP is the Objectory Process, one of several products and services that Rational acquired when they merged with Ivar Jacobson's Objectory organization several years ago. Rational enhanced Objectory with their own processes, and those of other tool companies that they have either purchased or partnered with, to form the initial version (5.0) of the RUP officially released in December of 1998. Visit www.enterpriseunifiedprocess.com/essays/history.html for a detailed history.

Figure 1 presents the current lifecycle of the RUP, made up of four serial phases and nine core disciplines (formerly called workflows). Along the bottom of the diagram you see that any given development cycle through the RUP should be organized into what Rational Corporation calls iterations. Although I would argue that the term increments is likely a better term than iterations, the basic concept is that at the end of each iteration you produce an internal executable that can be worked with by your user community. This reduces the risk of your project by improving communication between you and your customers. Another risk reduction technique built into the RUP is the concept that you should make a go/no-go decision at the end of each phase – if a project is going to fail then you want to stop it as early as possible in its lifecycle. This is an important concept in an industry with a 65%+ failure rate (www.standishgroup.com).

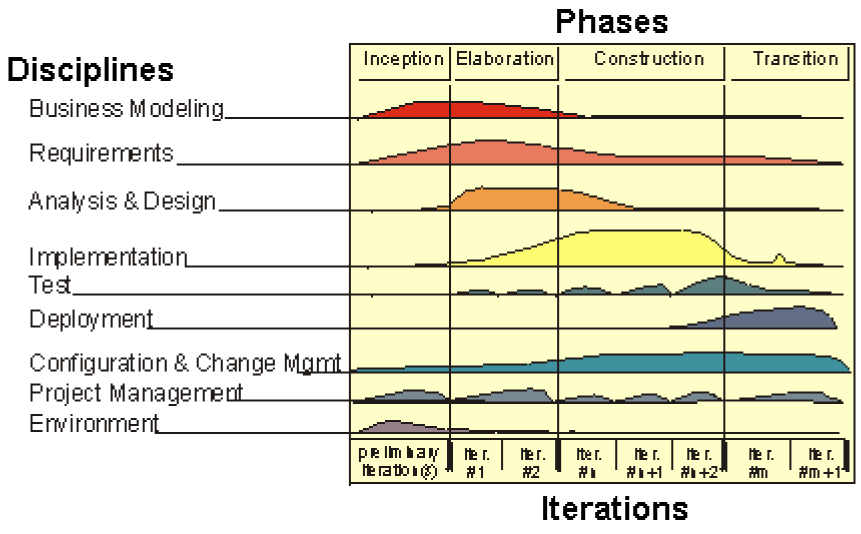

Figure 1. The initial lifecycle for the Unified Process.

The Inception phase is where you define the project scope and define the business case for the system. The initial use cases for your software are identified and the key ones are described briefly. Use cases are the industry standard technique for defining the functional requirements for systems, providing significant productivity improvements over traditional requirement documents because they focus on what adds value to users as opposed to product features. Basic project management documents are started during the Inception phase, including the initial risk assessment, the estimate, and the project schedule. As you would expect, key tasks during this phase include business modeling and requirements engineering, as well as the initial definition of your environment including tool selection and process tailoring.

The Elaboration phase focuses on detailed analysis of the problem domain and the definition of an architectural foundation for your project. Because use cases aren't sufficient for defining all requirements (Ambler 2004) a deliverable called a Supplementary Specification is defined which describes all non-functional requirements for your system. A detailed project plan for the Construction phase is also developed during this phase based on the initial management documents started in the Inception phase.

The Construction phase often the heart of software projects (typically to their detriment) is where the detailed design for your application is developed as well as the corresponding source code. I say that projects focus on this phase to their detriment because organizations often do not invest sufficient resources in the previous two phases and therefore lack the foundation from which to successfully develop software that meets the needs of their users. The goal of this phase is to produce the software and supporting documentation to be transitioned to your user base.

The purpose of the Transition phase is to deliver the system to your user community. There is often a beta release of the software to your users, typically called a pilot release within most businesses, in which a small group of users work with the system before it is released to the general community. Major defects are identified and potentially acted on during this phase. Finally, an assessment is made regarding the success of your efforts to determine whether another development cycle/increment is needed to further enhance the system.

The RUP has several strengths:

1. It is based on sound software engineering principles such as taking an iterative, requirements-driven, and architecture-based approach to development.
2. It provides several mechanisms, such as a working prototype at the end of each iteration and the go/no-go decision point at the end of each phase, which provides management visibility into the development process.
3. Rational has made, and continues to do so, a significant investment in their Rational Unified Process product, an HTML-based description of the RUP that your organization can tailor to meet its exact needs.

The RUP also suffers from several weaknesses:

1. It is only a development process. The current version of the RUP does not cover the entire software process, as you can see in Figure 1 it is very obviously missing the concept of operating and supporting systems once in production. Nor does it include the concept of eventually retiring systems.
2. The RUP does not explicitly support multi-system infrastructure development efforts such as organization-wide architectural modeling, missing opportunities for large-scale reuse within your organization.
3. The iterative nature of the lifecycle is foreign to many experienced developers, making acceptance of it more difficult. Although I think that the "hump chart" of Figure 1 is incredibly well drawn and information packed, it isn't the bubbles connect by lines diagrams which many people expect. The iterative nature of the Unified Process is both a strength and a weakness.

2. The Enterprise Unified Process (EUP)

So how do you enhance the RUP so that it meets the real-world needs of typical organizations? The first place to start is to expand the scope of the RUP to include the entire software process, not just the development process. Your organization likely has several software projects that it's currently managing. You likely have some systems that you are currently operating and supporting in production. The actual focus of most organizations isn't on the development of a single project, it's on the development, operation, support, and maintenance of a collection of systems. This implies that processes for operations, support, and maintenance efforts need to be added to the RUP. Second, to be sufficient for today's organizations the RUP also needs to add support for the management of a portfolio of projects, something other processes have called programme, multi-project, infrastructure management, or enterprise management. These first two steps result in an enhanced version of the Unified Process lifecycle of Figure 2 that Larry Constantine and myself presented in CMP book's Unified Process series[1] (Ambler & Constantine 2000a; Ambler & Constantine 2000b; Ambler & Constantine 2000c; Ambler & Constantine, 2002). This lifecycle describes an advanced instantiation of the Unified Process called the Enterprise Unified Process (EUP).

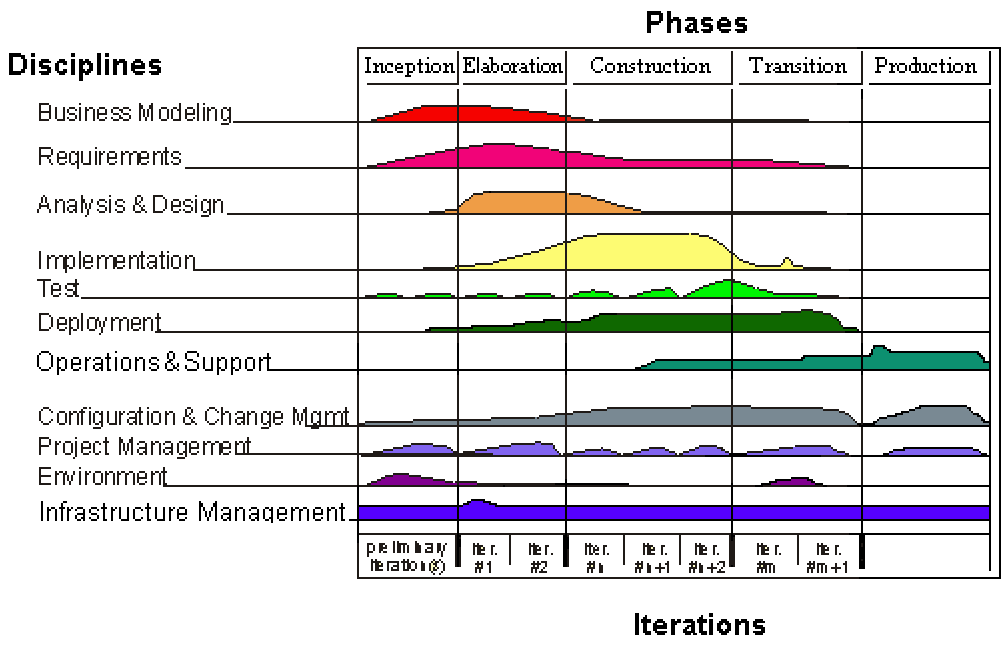

Figure 2. The enhanced lifecycle for the Enterprise Unified Process (EUP) (v1.0).

In November of 2002 I extended the lifecycle further to include a Retirement phase, reflecting the fact that systems are eventually removed from production, as you see in Figure 3. I also renamed the *Infrastructure Management* discipline to *Enterprise Management* as many people seemed to think that this discipline only dealt with technology-oriented issues.

[1] This book series is a collection of leading-edge articles from Software Development (www.sdmagazine.com), plus some new material, that describe techniques that can be used to enhance the RUP. I'm happy to say that some of the techniques have now been included in new releases of the RUP, it's actually an interesting exercise to compare the contents of each book with the subsequent release of the RUP product to actually see how far ahead we were writing the series. However, there is still significant material in the books that the RUP does not yet include, particularly material pertaining to the *Operations and Support* discipline and the enterprise disciplines.

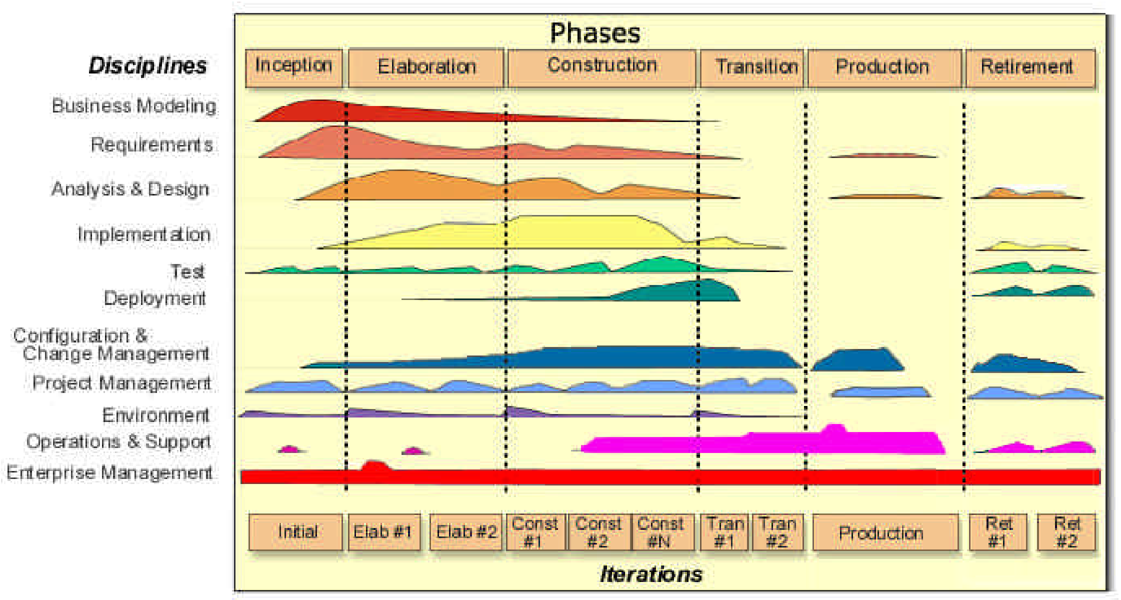

Figure 3. The updated lifecycle for the EUP (v2002).

During the Spring of 2004 Mike Vizdos, John Nalbone, and myself refactored the *Enterprise Management* discipline into seven disciplines, as you see in Figure 4. We also organized the disciplines into three categories: Development, Support, and Enterprise.

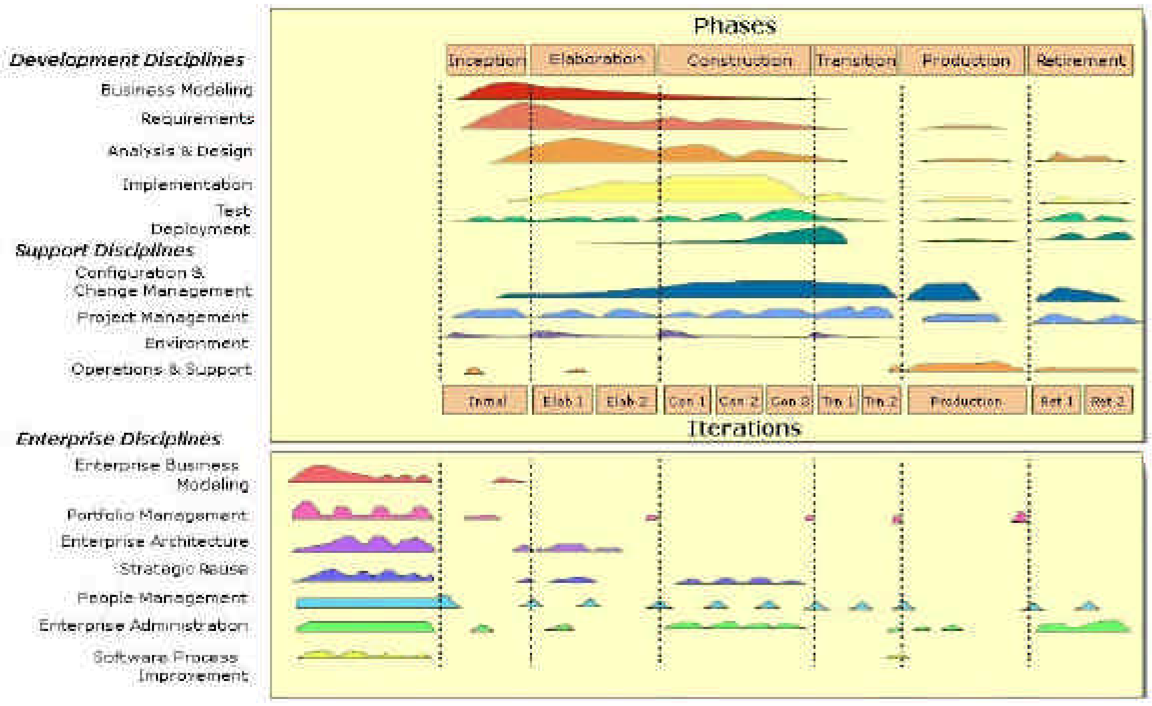

Figure 4. The EUP lifecycle v2004.

2.1 The Operations and Support Discipline

The purpose of the Operations & Support discipline is exactly as the name implies, to operate and support your software. Operations and support are both complex endeavors, endeavors that need processes defined for them. This discipline spans several phases, as do all the others. During the Construction phase, and perhaps as early as the Elaboration phase, you will need to develop operations and support plans, documents, and training manuals. During the Transition phase you will continue to develop these artifacts, reworking them based on the results of testing and you will train your operations and support staff to effectively work with your software. During the Production phase your operations staff will keep your software running, performing necessary backups and batch jobs as needed, and your support staff will work with your user community in working with your software. This will also happed during the Retirement phase during the process of turning down the legacy system.

This discipline basically encompasses portions of the OOSP's Release stage and Support stage (Ambler 1998b); the OPEN Process's Implementation Planning activity and Use of System activities (Graham, Henderson-Sellers, and Younessi 1997); ISO 12207's *Operational Support* and *Dispensation* phases (www.iso.org); and the Information Technology Information Library (ITIL) (www.itil-itsm-world.com).

2.2 The Enterprise Disciplines

The EUP includes seven new enterprise management disciplines that tackle the cross-system issues that organizations should address to be successful at IT. These disciplines are:

1. **Enterprise Business Modeling**. The goal of this discipline is to explore the business structure and processes of the enterprise. It provides a common understanding of the business activities, customers, and suppliers of the business. Enterprise business modeling helps to identify problems and areas that are candidates for automation.
2. **Portfolio Management**. Organizations often have suites of applications, called programs, which can be better managed as a whole than as individual applications. This discipline enables you to track and plan your organization's entire software portfolio as well as individual programs within your overall portfolio. Doing so allows you to schedule and implement new requirements in a more strategic fashion. This also helps to avoid implementing the same functionality within different applications.
3. **Enterprise Architecture**. This discipline addresses the overall architecture issues associated with your organization. It consists of models that define it, prototypes and working models that demonstrate how it works, and frameworks that make it easier to use. The Enterprise Architecture discipline helps ensure consistency across systems and greatly facilitates application architecture efforts.
4. **Strategic Reuse**. This discipline promotes development and reuse of assets across projects, the goal of which is to enable you to develop higher-quality applications more quickly by reusing assets instead of developing them anew each time. It also helps improve quality, as it allows you to use artifacts that have already been tested and proven to work.
5. **People Management**. Managing software development efforts includes far more than the technical tasks of creating and evolving project plans and schedules. People exist within your organization, and you need to manage your staff and mediate the interactions between them and other people. This discipline describes the process of organizing, monitoring, coaching, and motivating people in a manner to ensure they work together well and successfully contribute to projects within the organization.
6. **Enterprise Administration**. This discipline includes setting up and administering tools, processes, and facilities that are key infrastructure components of your IT organization.
7. **Software Process Improvement**. This discipline addresses the need to manage, improve, and support the multiple processes in use across your organization. Remember, one process does not fit all.

The new enterprise disciplines come at a cost. Any attempts at implementing them should be applied judiciously and with minimal interruption to your current business functions. These new disciplines can provide significant savings in the long run by streamlining the way that your IT department operates. if you choose to implement them in a non-bureaucratic manner.

It is important to understand that although some overlap exists between the enterprise management discipline activities and existing RUP activities, the scope is different. For example, the Enterprise Architecture discipline includes many of the same activities as the RUP Analysis and Design discipline. Your enterprise architecture is concerned with the architecture that spans across multiple systems, whereas the architecture that is done during the Analysis and Design discipline of a project is for a single system only. Similarly, Enterprise Business Modeling (EBM) and Business Modeling (BM) disciplines are similar as far as the activities are concerned. The difference is that EBM takes a broad and shallow look at the entire business, whereas BM typically looks at the business only as it relates to a single system. BM will most likely go into more detail in a specific area, as it may be required to do so in order for a particular system to be developed.

2.3 The Production Phase

As you see in Figure 4 the EUP includes a fifth phase, Production, representing the portion of the software lifecycle after a system has been deployed. As the name of the phase implies, its purpose is to keep your software in production until it is either replaced with a new version, from a minor release such as a bug fix to a major new release, or it is retired and removed from production. Note that there are no iterations during this phase, or there is only one iteration depending on how you wish to look at it, because this phase applies to the lifetime of a single release of your software. To develop and deploy a new release of your software you need to run through the four development phases again.

2.4 The Retirement Phase

The focus of the *Retirement Phase* is the successful removal of a system from production. Systems are removed from production for several reasons:

1. They are no longer needed. An example of this is a system that was put in production to fulfill the legal requirements imposed by government legislation. Now that the legislation has been repealed there is no longer any need for the system.
2. The system is being replaced. For example, it is common to see home-grown systems for human resource functions be replaced by commercial off-the-shelf (COTS) systems.

Activities of the Retirement phase include:

1. A comprehensive analysis of the existing system to identify its coupling to other systems.
2. A redesign and rework of other, existing systems so that they no longer rely on the system being retired. These efforts will typically be treated as projects in their own right.
3. Transformation of existing legacy data, perhaps via database refactoring (www.agiledata.org/essays/databaseRefactoring.html), because it will no longer be required or manipulated by the system being retired.
4. Archival of data previously maintained by the system that is no longer needed by other systems.
5. Configuration management of the removed software so that it may be reinstalled if required at some point in the future (this is easier said than done).
6. System integration testing of the remaining systems to ensure that they have not been broken via the retirement of the system in question.

2.5 Enhancing Existing Disciplines

Comparing the enhanced lifecycle for the EUP of Figure 2 with the initial lifecycle for the Unified Process of Figure 1 you will notice that several of the existing disciplines have also been updated. First, the Test discipline has been expanded to include activity during the Inception phase. You develop your initial, highlevel requirements during this phase, requirements that you can validate using techniques such as walkthroughs, inspections, and scenario testing. Two of the underlying philosophies of the OOSP are that you should test often and early and that if something is worth developing then it is worth testing. Extreme Programming (XP) (Beck 2005) takes this one step further by insisting that you take a test-first approach! My experience is that testing should be moved forward in the lifecycle. Also, the Test discipline can be enhanced with the techniques Full Lifecycle Object Oriented Testing (FLOOT) method (Ambler 2004)[2].

The second modification is to the Deployment discipline, extending it into the Inception and Elaboration phases. This modification reflects the fact that deployment, at least of business applications, is a daunting task. Data conversion efforts of legacy data sources are often a project in their own right, a task that requires significant planning, analysis, and considerable effort to accomplish. Furthermore, my belief is that deployment modeling should be part of the Deployment discipline, and not the Analysis & Design discipline as it currently is, due to the fact that deployment modeling and deployment planning go hand in hand. Deployment planning can and should start as early as the Inception phase and continue into the Elaboration and Construction phases in parallel with deployment modeling.

The Environment discipline has been updated to include the work necessary to define the Production environment, work that would typically occur during the Transition phase. The existing Environment discipline processes effectively remain the same, the only difference being that they now need to expand their scope from being focused simply on a development environment to also include operations and support environments. Your operations and support staff need their own processes, standards, guidelines, and tools, the same as your developers. Therefore you may have some tailoring, developing, or purchasing to perform to reflect this need.

The Configuration & Change Management discipline is extended into the new Production phase to include the change control processes needed to assess the impact of a proposed change to your deployed software and to allocate that change to a future release of your system. This change control process is often more formal during this phase than what you do during development due to the increased effort to update and re-release existing software. It is also extended into the Retirement Phase to support the configuration management issues pertaining to the removal of the system from production.

Similarly, the Project Management discipline is also extended into the new Production and Retirement phases to include the processes needed to manage these efforts. New features should be added to the Project Management discipline. As mentioned earlier it is light on metric management activities, although this is an area that Rational Corporation has improved on recently. It also needs processes for subcontractor management a key need of any organization that outsources portions of its development activities or hires consultants and contractors. People management issues, including training and education as well as career management, are barely covered by the RUP. There is far more to project management than the technical tasks of creating and evolving project plans, you also need to manage your staffs and mediate the interactions between them and other people.

3. The Agile Unified Process (AUP)

One way to enhance the EUP is to swap in the Agile Unified Process (AUP), which can be downloaded free of charge from www.ambysoft.com/unifiedprocess/agileUP.html, instead of the RUP. The AUP is an agile instantiation of the RUP which simplifies it as well as shows how to apply agile techniques such as Agile Model Driven Development (AMDD), Test Driven Development (TDD), refactoring, and database refactoring within the scope of an UP project.

[2] Since I first started writing about enhancing the Unified Process (Ambler 1999b) in Software Development (www.sdmagazine.com) Rational Corporation has made great strides in improving the Test discipline. I'm impressed with their progress and look forward to further improvement.

4. The EUP Mailing List

If you have any questions about the Enterprise Unified Process (EUP), please feel free to post them on the EUP mailing list. It's a free service and the details for joining are posted at www.enterpriseunifiedprocess.com/feedback.html. If you're interested in the EUP, or simply open discussion concerning real-world tailorings of the RUP, then you should seriously consider joining this list.

5. In Closing

In this paper you were also introduced to the Enterprise Unified Process (EUP), an instantiation of the Unified Process to make it meet the actual real-world needs of typical organizations, tailored to meet the mission-critical needs of large-scale development. Software development, maintenance, and support are complex endeavors, ones that require good people, good tools, good architectures, and good processes to be successful. The software process is a significant part of the solution to the software crisis, something that your organization has likely ignored to its peril.

6. References and Suggested Reading

6.1 Web-Based Resources

Agile Data Home Page, www.agiledata.org

Agile Modeling Home Page, www.agilemodeling.com

Enterprise Unified Process, www.enterpriseunifiedprocess.com

The OPEN Website. www.open.org.au

The Process Patterns Resource Page. www.ambysoft.com/processPatternsPage.html

Rational Unified Process. www.rational.com/products/rup

Software Engineering Institute Home Page. www.sei.cmu.edu

6.2 Printed Resources

Ambler, S.W. 1998a. *Building Object Applications That Work*. Cambridge University Press. www.ambysoft.com/buildingObjectApplications.html.

Ambler, S.W. 1998b. *Process Patterns: Building Large-Scale Systems Using Object Technology*. Cambridge: Cambridge University Press. www.ambysoft.com/processPatterns.html.

Ambler, S.W. 1999a. *More Process Patterns: Delivering Large-Scale Systems Using Object Technology*. Cambridge: Cambridge University Press. www.ambysoft.com/moreProcessPatterns.html.

Ambler, S.W. 1999b. *Enhancing the Unified Process*. Software Development Magazine, October 1999. www.sdmagazine.com/documents/s=753/sdm9910b/9910b.htm

Ambler, S.W. (2001). Agile Modeling and the Unified Process. www.agilemodeling.com/essays/agileModelingRUP.htm

Ambler, S.W. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. New York: John Wiley & Sons. www.ambysoft.com/agileModeling.html

Ambler, S.W. (2004). *The Object Primer 3rd Edition: Agile Model Driven Development with UML 2*. New York: Cambridge University Press. www.ambysoft.com/theObjectPrimer.html.

Ambler, S.W. (2005). *The Elements of UML 2.0 Style*. New York: Cambridge University Press. www.ambysoft.com/elementsUMLStyle.html

Ambler, S.W. & Constantine, L.L. (2000a). *The Unified Process Inception Phase*. Gilroy, CA: CMP Books. www.ambysoft.com/inceptionPhase.html.

Ambler, S.W. & Constantine, L.L. (2000b). *The Unified Process Elaboration Phase*. Gilroy, CA: CMP Books. www.ambysoft.com/elaborationPhase.html.

Ambler, S.W. & Constantine, L.L. (2000c). *The Unified Process Construction Phase*. Gilroy, CA: CMP Books. http://www.ambysoft.com/constructionPhase.html.

Ambler, S.W. & Constantine, L.L. (2002). *The Unified Process Transition and Production Phases*. Gilroy, CA: CMP Books. www.ambysoft.com/transitionProductionPhase.html

Beck, K. (2005). *Extreme Programming Explained 2nd Edition – Embrace Change*. Reading, MA: Addison Wesley Longman, Inc.

Boehm, B.W. 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21(5), 61-72.

DeMarco, T. 1997. *The Deadline: A Novel About Project Management*. New York: Dorset House Publishing.

Emam, K. E.; Drouin J.; and Melo, W. 1998. *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. Los Alamitos, California: IEEE Computer Society Press.

Graham, I.; Henderson-Sellers, B.; and Younessi, H. 1997. *The OPEN Process Specification*. New York: ACM Press Books.

Graham, I.; Henderson-Sellers, B.; Simons, A., and Younessi, H. 1997. *The OPEN Toolbox of Techniques*. New York: ACM Press Books.

Humphrey, W. S. 1997. *Managing Technical People: Innovation, Teamwork, And The Software Process*. Reading, Massachusetts: Addison-Wesley Longman, Inc.

Jacobson, I., Booch, G., & Rumbaugh, J., (1999). *The Unified Software Development Process*. Reading, MA: Addison Wesley Longman, Inc.

Jacobson, I., Griss, M., Jonsson, P. (1997). *Software Reuse: Architecture, Process, and Organization for Business Success*. New York: ACM Press.

Kruchten, P. (2004). *The Rational Unified Process: An Introduction 3/e*. Reading, MA: Addison Wesley Longman, Inc.

Software Engineering Institute. 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading Massachusetts: Addison-Wesley Publishing Company, Inc.

Vermeulen, A., Ambler, S.W., Bumgardner, G., Metz, E., Misfeldt, T., Shur, J., & Thompson, P. (2000). *The Elements of Java Style*. New York: Cambridge University Press.

Wiegers, K. (1996). *Creating a Software Engineering Culture*. New York: Dorset House Publishing.

Wiegers, K. (1999). *Software Requirements*. Redmond, WA: Microsoft Press.

7. About the Author

Scott W. Ambler is an IT methodologist living just north of Toronto Canada. Scott is the (co-)author of several books, including *The Enterprise Unified Process* (2005) published by Prentice Hall PTR, *The Object Primer 3rd Edition* (2004) and *The Elements of UML 2.0 Style* (2005) published by Cambridge University Press, and *Agile Modeling* (2002) and *Agile Database Techniques* (2003) published by Wiley Publishing. Scott is a contributing editor with *Software Development* (www.sdmagazine.com) and the thought leader behind the Enterprise Unified Process (EUP), Agile Model Driven Development (AMDD), the Agile Data method, and the Agile Unified Process (AUP). He can be reached at www.ambysoft.com/scottAmbler.html