JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2002**

Vol. 1, no. 4, September-October 2002

Requirements Engineering

Donald G. Firesmith, Firesmith Consulting, U.S.A.

Abstract

Because all other activities are influenced or driven by it, requirements engineering is arguably the most important activity performed during the development of software-intensive systems. Yet, there has been remarkably little standardization on what requirements engineering is and how it should be done beyond the growing de facto standard of using some kind of use case modeling for functional requirements. The first two articles of this column lay the foundation for my future columns on requirements engineering by defining the reusable requirements related process components of the OPEN Process Framework (OPF) and by providing you with an industry-standard terminology for organizing and communicating requirements engineering concepts.

1 REQUIREMENTS ENGINEERING

Like such activities as testing, operations, and maintenance, the requirements engineering activity has never received as much emphasis on projects, at technical conferences, and in the classroom as have the more popular design and implementation (e.g., programming) activities. One can easily find 50 books on programming languages and either object-oriented or web page design in technical bookstores for every single book one finds on requirements engineering. And although there have been major strides made in requirements engineering over the last decade, there is still much confusion in industry as to just what requirements engineering is, how it should be performed, and how the resulting requirement should be specified.

The first two articles of this column will lay the foundation for future articles of my column on requirements engineering by defining the reusable requirements related components of the OPEN Process Framework (OPF) and providing you with an industry-standard terminology for organizing and communicating requirements engineering concepts. Part I will summarize the OPF for those of you that are unfamiliar with it and briefly describe the various reusable requirements work products that may be produced during the requirements engineering activity. Part II in the next issue of JOT will cover the remaining reusable process components of the OPF including work units (activities, tasks, and techniques), producers (organizations, teams, roles, persons, and tools), languages (natural, modeling, and specification languages), endeavors (enterprises, programs, and projects), and stages (cycles, phases, builds, and milestones).

The OPEN Process Framework

The articles in this column will be based on the reusable requirements engineering process components of the OPEN Consortium's OPEN Process Framework (OPF), a practical, public-domain, industry-standard, general purpose management and engineering process framework that is primarily intended for the object-oriented, component-based development of software-intensive systems. As illustrated in the following figure, OPF consists of a:

- **Repository** of predefined reusable process components.
- **Metamodel** that organizes and provides a theoretical foundation for these components.
- **Guidelines** for using these process components to construct and tailor endeavor-specific processes.

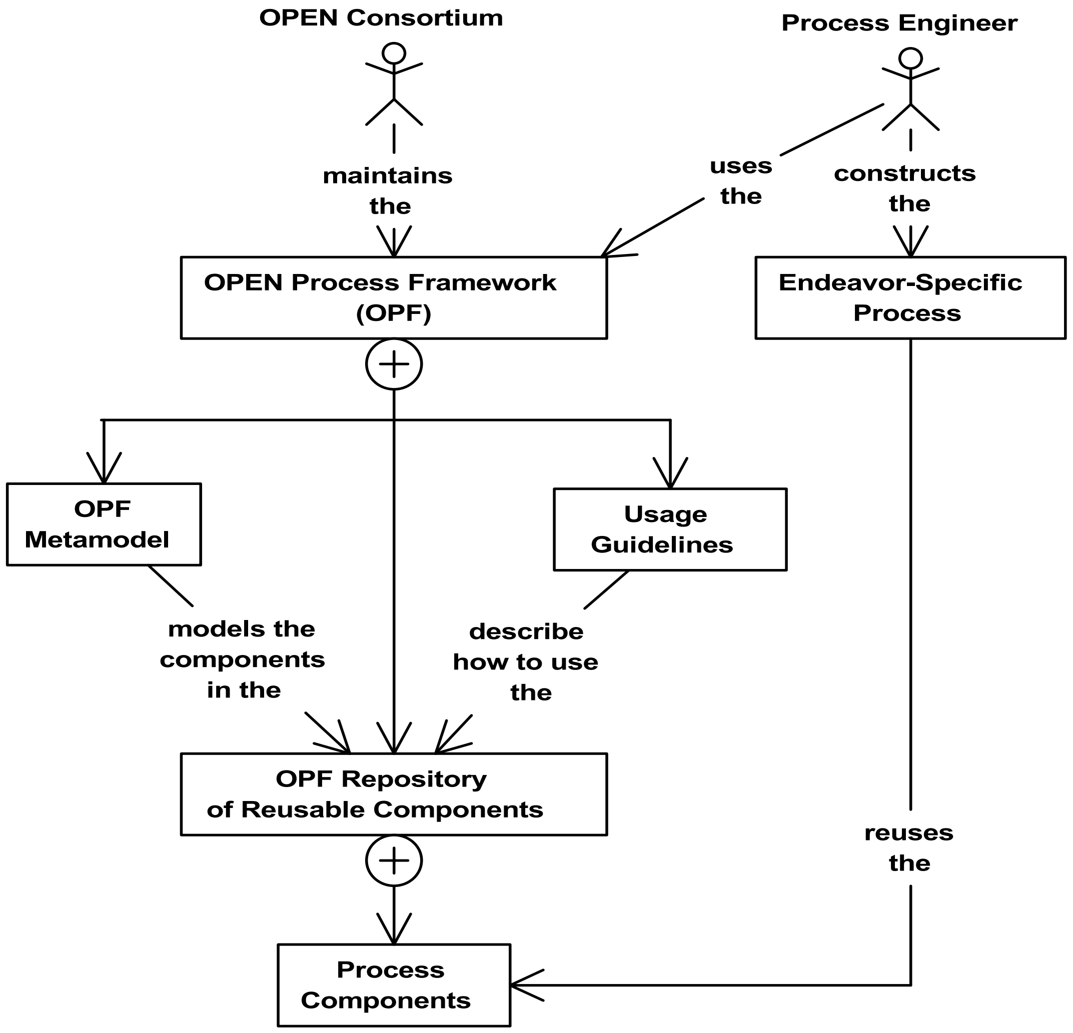

Fig. 1: Overview of the OPEN Process Framework

OPF provides managers, technologists, strategists, user experience personnel, process engineers, methodologists, consultants, trainers, and academics with the best current industry practices for processes to perform:

- **Business (Re)Engineering** including business requirements engineering, business architecting, digital branding, management, etc.
- **Applications Development** including management, configuration management, risk management, quality engineering, training, requirements engineering, architecting, design, implementation, integration, testing, etc., whereby application development may include:
  - **Custom development** of one or more new applications from scratch.
  - **Assembling** one or more new applications from existing, possibly commercial components.
  - **Purchasing** one or more new applications.
  - **Enhancing** one or more existing applications.
  - Any combination of the above.
- **Applications Usage** including operations, maintenance, content management, and eventual retirement.
- **Reusable Component Development** including requirements engineering, architecting, etc. of either individual components or frameworks of related components.

The OPF Metamodel for Process Components

A metamodel is a model of a model. In this case, the OPF metamodel is a model showing the reusable process components that can be used to construct endeavor-specific development processes. As illustrated in the following figure, the OPF metamodel models the different kinds of process components and the most basic relationships between them. In order to simplify the figure, subclasses of process components are indicated by comments rather than individual icons and inheritance arcs. The OPF metamodel includes numerous instances of the following classes of process components:

- **Work products** (e.g., diagrams, models, documents, components, applications) that can be produced during the course of the endeavor-specific processes.
- **Work units** (e.g., activities, tasks, and techniques) that can be performed to produce (e.g., create, evaluate, iterate, and maintain) the work products.
- **Producers** (e.g., organizations, teams, roles, persons, tools) who perform these work units in order to produce the work products.
- **Languages** (e.g., modelling languages such as UML and OML, implementation languages such as Java and HTML, specification languages such as ObjectZ) that the producers can use to implement these work products.
- **Endeavors** (e.g., projects, programs of related projects, entire enterprises) to which the producers are staffed and allocated.

- **Stages** (e.g., development cycles, phases, builds, and milestones) of the endeavors during which the producers perform their work units to produce the work products.

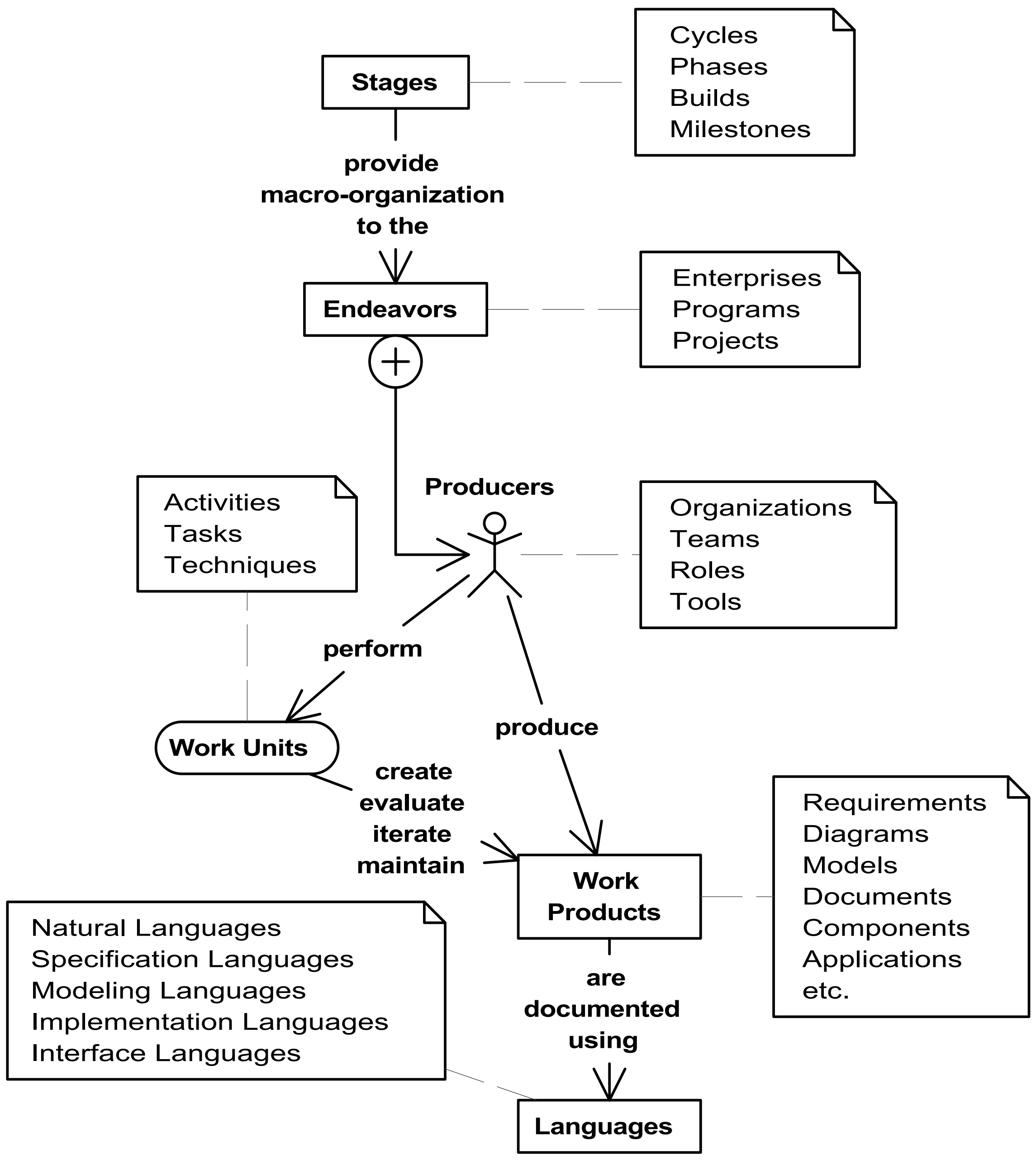

Fig. 2: OPF Metamodel for Describing Process Components[1]

[1] This informal diagram uses the plus in a circle icon from the OPEN Modeling Language (OML) to represent the whole-part aggregation relationship, whereby endeavors consist (among other things) of producers. Think of the icon as representing the end of a Philips head screw that binds the producers into the aggregate endeavor.

Requirements Engineering Overview

By using the OPEN Process Framework metamodel, we can now see that any requirements engineering process can be modeled and understood as a collections of reusable process components including:

- **Requirements work products** (e.g., requirements, requirements diagrams, requirements models, and requirements specifications and other documents).
- **Requirements work units** (e.g., the requiremens engineering activity, requirements engineering tasks, and techniques for performing them).
- **Producers of requirements** (e.g., the requirements team, requirements engineers, and requirements management tools).
- **Languages** (e.g., English for textual requirements, UML or OML for requirements models, and specification languages for formally specifying requirements).
- **Endeavors** (e.g., projects developing application-specific requirements, programs and enterprises developing common reusable requirements).
- **Stages** (e.g., phases during which requirements are elicitated, analyzed, and specified, milestones by which requirements are baselined and delivered).

2 REQUIREMENTS WORK PRODUCTS

As an activity, requirements engineering exists in order to produce the requirements work products that are the foundation upon which the majority of the other major activities (e.g., architecting, design, implementation, testing) are built. Thus in many ways, they are some of the most critical work products produced during development endeavors.

As the following paragraphs will show, there are a great many reusable requirements work products that can be produced on an endeavor. However, not every endeavor will require every one of these work products. The process engineers working with others on the endeavor will select only those work products that are appropriate and cost effective.

The requirements work products can be categorized (in order of increasing size and complexity) as either requirements, diagrams, models, documents, or baselines[2]. As the following simplified figure illustrates, requirements work products are produced by producers (e.g., requirements engineers, requirements teams, requirements management tools) during the performance of requirements tasks using associated techniques and that requirements baselines must typically be delivered by certain milestones.

[2] Technically speaking, diagrams, models, and baselines are not really requirements work products because they can be produced during the performance of multiple activities. For example, class diagrams can be produced during requirements engineering as well as the design of packages of software or the design of databases. Thus, diagrams, models, and baselines form their own sets of work products, and they only the requirements-related ones are covered in this section.

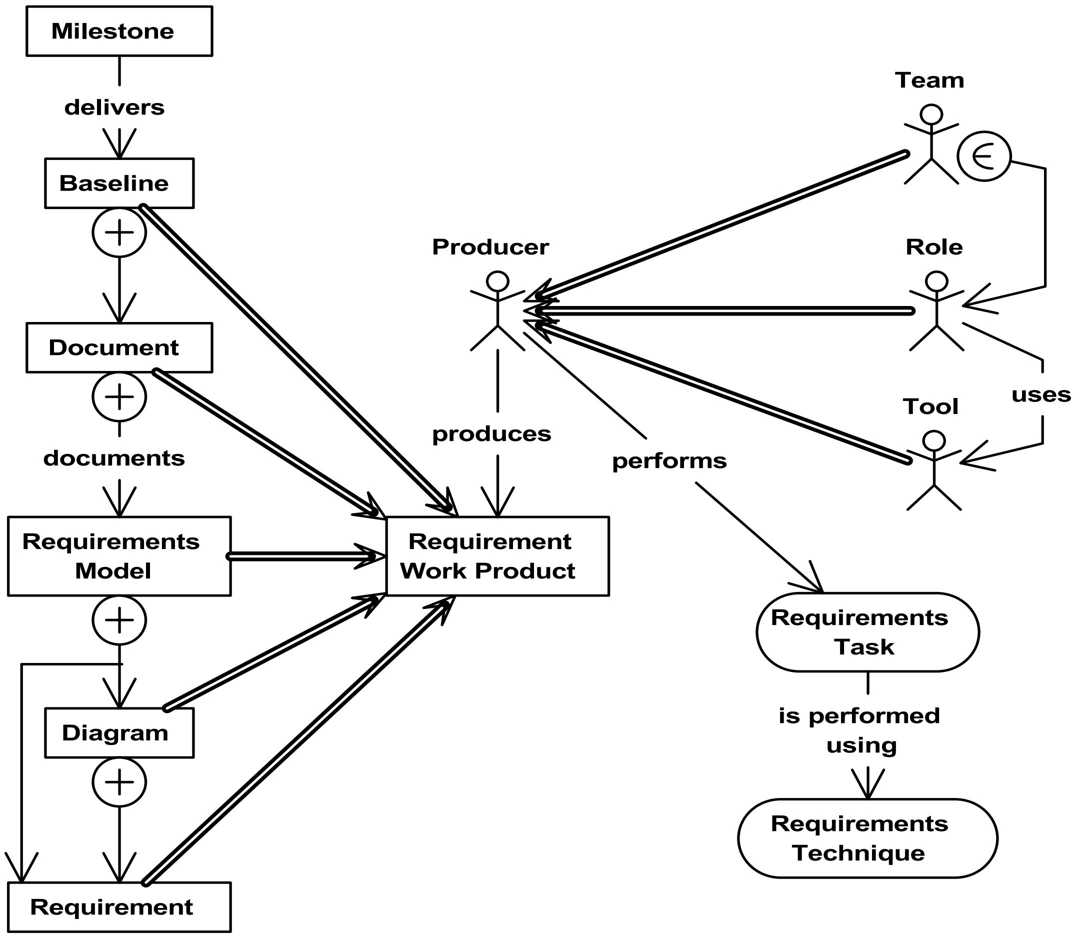

Fig. 3: Requirements Work Products[3]

Requirements

A **requirement** is a work product in the requirements set of work products that formally specifies a mandatory, externally-observable, and validatable (e.g., testable) aspect of something being produced or modified. Whereas most requirements specify mandatory functions or characteristics of an application being developed, one can also specify requirements for components, frameworks, contact or data centers, or even a business unit during business reengineering. Some organizations even produce reusable requirements for individual application domains. Thus, requirements are capabilities that are needed by the customer and user organizations to achieve their objectives (e.g., by enabling users to better perform their current tasks, enabling users to perform new tasks, or to perform tasks that users cannot or should not perform themselves). Requirements

[3] This informal diagram uses several relationship icons from the OPEN Modeling Language (OML). There is the plus sign in a circle representing the end of a Philips head screw to symbolize configurational aggregate relationships from wholes to their parts. It also uses the epsilon in circle to represent nonconfigurational membership relationships. Finally, it clearly uses double-line arrows to distinguish the tight coupling of inheritance relationships from the single-line arrows of referential relationships.

are also capabilities that are formally imposed (e.g., in a contract, statement of work, requirements specification, industry standard, law or governmental regulation, etc.) by the customer organization on the development organization.

OPF classifies requirements into the following multiple types as follows:

- **Functional requirements** are requirements that specify mandatory behaviors. They are typically specified using normal narrative text, specification languages, use case models, functional models, state models, and/or object models.
- **Data (a.k.a., informational) requirements** are requirement that specify some mandatory property of a data type or value. They are typically specified using logical data models, object models, or data dictionaries.
- **Quality requirements** are requirements that specify mandatory levels of quality factors (i.e., the "ilities"). Quality requirements can be either developer-oriented (e.g., extensibility, installability, maintainability, portability, reusability, scalability, testability) or user-oriented (e.g., accessability, auditability, configurability including personalization and internationalization, correctness, efficiency, interoperability, look and feel including banding, operational availability, operational environments, performance, reliability, robustness, safety, security, and usability). They are typically specified using normal narrative text, either separately as a group of related requirements or else individually with the relevant functional and data requirements.
- **External API requirements** are requirement that specify mandatory application programmer interfaces to external systems, typically either the customer organization's legacy systems or systems not owned by the customer organization. They are typically specified using interface and protocol specifications.
- **Constraints** are architectural, design, or implementation decisions that are imposed on the development organization by the customer organization and are thus treated as if it were requirements. Constraints include physical constraints, business rules, data and content constraints, hardware constraints, software constraints, industry standards, legal and regulatory constraints, and production environment constraints.

Requirements Diagrams

Requirements diagrams are diagrams that are typically used to create and document requirements models for a blackbox application, application domain, framework, component, contact or data center, or business enterprise. Typical examples include:

- **Context diagrams,** which are used to document the required context (environment surrounding) of a blackbox application (etc.) in terms of externals and the relationships between them.

- **Use case diagrams,** which are used to document the top-level required behaviour of a blackbox application (etc.) in terms of externals (a.k.a., actors) and the blackbox ways they interact (i.e., a use case).
- **Sequence diagrams,** which are used to document the required interactions between the top-level required behaviour of a blackbox application (etc.) and its externals.
- **Data flow diagrams,** which are used to document the required data flows between the major required logical functions of the application, application domain, framework, component, contact or data center, or business enterprise.
- **Control flow diagrams,** which are used to document the required control flows between the major required logical functions of the application (etc.).
- **Entity relationship attribute diagrams,** which are used to document the required major data types of an application (etc.), their attributes, and the relationships between them.
- **Class diagrams,** which are used to document the required major types of objects, their attributes and operations, and the relationships between them.
- **State transition diagrams,** which are used to document the required life cycles of required objects in the application (etc.) in terms of their states and transitions.

Requirements Models

Many different kinds of models have been used over the last 30 years to analyze and specify requirements, including the following:

- **Use case models** are models based on use cases, which when used during requirements engineering are general ways of specifying a functionally cohesive set of interactions between one or more externals/actors (e.g., roles played by users, external applications) and a blackbox application, application domain, framework, component, contact or data center, or business unit in order to provide an observable benefit to one of the actors. Use case models are the current industry-defacto modeling approach, and they are especially popular among developers trained in object technology, although use case models are more closely related to functional models than object models. They typically consist of actor specifications, use case specifications, use case path specifications, use case diagrams, sequence diagrams, and context diagrams. In addition to use cases and the paths through them, use case models sometimes include change cases (for specifying likely changes to the functional requirements) and misuse/abuse cases (for specifying security requirements by specifying how to handle malicious usage) [Sindre&Opdhal2000].
- **Functional models** are models based on the use of functional decomposition in order to identify and specify mandatory, functionally-cohesive functions and subfunctions of a blackbox application, application domain, framework, component, contact or data center, or business unit. Functional models were the de facto industry-standard approach used during the 1970s and 1980s, and they were typically developed using Structured Analysis (SA) or some related

requirements engineering method. Functional models typically consist of data-flow diagrams (DFDs), function specifications, and control-flow diagrams (CFDs), and they are typically backed up by data models and state models.

- **Data models** (during requirements engineering) are logical models used to analyze and specify the mandatory externally-visible data used by a blackbox application, application domain, framework, component, contact center, or business unit. Data models were the de facto industry-standard approach used during the late 1980s and early 1990s, and they were typically developed using Information Engineering (IE) and information-intensive applications involving complex databases. Data models typically consist primarily of data dictionaries and some form of Entity Relationship Attribute (ERA) diagrams.
- **State models** (during requirements engineering) are logical models used to analyze and specify the mandatory life cycles of actors, externally visible data, and the blackbox application, framework, or component in terms of states and the transitions between them. State models typically consist primarily of state transition diagrams (a.k.a., state charts, state diagrams) or possibly state transition tables.
- **Object models** (during requirements engineering) are logical models used to analyze and specify the domain object model and the data requirements, and to begin to identify required behavious (functional requirements).

Requirements Documents

Requirements documents are document work products that either specify requirements or primarily capture requirements-related information. They may be manually produced paper documents, or electronic documents automatically generated from digital content by a requirements tool. Examples include:

- **Customer, competitor, and user profiles**, which document the various kinds of stakeholders.
- **Customer, market, and user analyses**, which document the result of analyzing the customer organization, the market in which the customer or user competes, and the users of the application or component.
- **Technology needs assessment and technology analysis**, which document information about relevant technology, technology trends, and how they will impact the requirements.
- **Business vision statements**, which document the customer organization's vision of its business enterprise.
- **User-task matrices**, which document the tasks that the users perform.
- **Features matrices**, which document the features of an application and their associated attributes.
- **Requirements prototypes**, which identify new requirements and iterate existing requirements.

- **Quality grids**, which are matrices that document the importance of the different quality requirements.
- **Application vision statements**, which document the customer organization's vision of a single application.
- **Requirements executive summaries**, which formally summarize the requirements for executives and managers (e.g., of the customer organization).
- **System requirements specifications**, which formally specify the functional requirements, data requirements, quality requirements, and constraints.
- **External API specifications**, which formally specify any external application programmer interfaces.
- **Glossary**, which formally define the abbreviations, accronyms, and terms used on an endeavor.
- **Domain model document**, which uses an object model to formally document the relationships between application domain concepts defined in the project glossary.

Requirements Baselines

Requirements baselines are baselines that primarily contain a cohesive collection of requirements work products (and possibly other kinds of work products) that are to be at a certain level of completion by a certain milestone. Examples include:

- **Initial requirements baseline**, which contains all requirements work products whereby the system requirements specification should be approximately 80% complete and specify all architecturally-significant requirements.
- **Requirements complete baseline**, which contains all requirements work products, whereby each work product is essentially complete and under configuration control.
- **Requirements frozen baseline**, which contains all requirements work products, whereby all requirements are frozen and may not be changed until the next version of the application, etc

.

3 CONCLUSION

As the preceding article shows, requirements engineering is not trivial and can involve the production of a great many reusable work products that must be selected, integrated, and tailored when producing a project-specific requirements process. The next article will complete the discussion of the reusable requirements process components and lay a foundation for future articles. These articles will cover individual requirements-related process components of the OPEN Process Framework such as quality requirements, the requirements engineering tasks, requirements tools, and requirements engineering teams. For more detailed information about the OPEN Process Framework, go to the official OPEN website at www.open.edu or my website at www.donald-firesmith.com.

REFERENCES

[Firesmith2001]	Donald Firesmith and Brian Henderson-Sellers: The OPEN Process Framework, Addison-Wesley-Longman, 2001.
[Sindre&Opdhal2000]	G. Sindre and A. L. Opdahl, "Eliciting Secutiry Requirements by *Mis*use Cases", Proc. TOOLS Pacific 2000, pp 120-131, 20-23 Nov 2000.

About the author

Donald Firesmith runs Firesmith consulting, which provides consulting and training in the development of software-intensive systems. He has worked exclusively with object technology since 1984 and has written 5 books on the subject. Most recently, he has developed a 1000+ page informational website on the OPEN Process Framework. His most recent book is *The OPEN Process Framework* (Addison-Wesley-Longman, 2001), and he is currently writing a book on requirements engineering. He can be reached at donald_firesmith@hotmail.com.