Criteria for Comparing Domain Analysis Approaches

Steven Wartik Rubén Prieto-Díaz

Abstract

This paper describes a set of criteria for comparing domain analysis approaches. The criteria were derived from a study of several well-known approaches. They help people understand the rôle that domain analysis can play in software development. They are therefore of importance to both practitioners who need to choose an approach, and to researchers seeking to better understand the nature of domain analysis.

Keywords: domain analysis, software process

1 Introduction

Research on domain analysis in recent years has produced many approaches. Practitioners are often confused about the right domain analysis approach for their needs—if indeed a single approach can be said to be "right". At the Software Productivity Consortium, several domain analysis approaches are, or have been, espoused. These include:

- The approach originated by the Consortium's original Synthesis project, and subsequently refined by the Domain Analysis project headed by Dr. Alan Jaworski [Jawo 90].
- The current Synthesis methodology, for which Grady Campbell is the chief architect [Camp 90]. (*N.B.* This approach is called "Synthesis" in the rest of this paper.)
- The approach by Dr. Rubén Prieto-Díaz, first developed at the University of California at Irvine, and refined at GTE, Contel, and the Consortium [Diaz 87].

Three approaches within a single organization has, understandably, led to some confusion. Accordingly, the Consortium decided to study these approaches, as well as those of other researchers. The goal was to help organizations understand which approach, or approaches, would best meet their needs. We also wished to know if we could develop a unified domain analysis approach, incorporating concepts from these and other methods. This paper briefly summarizes our work.

2 Approach

The approach we used was to study the three approaches mentioned above, plus the FODA approach of SEI [Kang 90]. We choose the first three because we were especially interested in making recommendations on in-house methods. We also wanted to include research done outside the Consortium. We selected FODA because it is an amalgamation of various approaches, including those of Prieto-Díaz, Lubars [Luba 88], and KAPTUR [Moor 89].

We studied the approaches with the aim of determining their similarities and differences. This amounts to a domain analysis of the domain analysis approaches. (We did not follow a formal domain analysis process, although what we used was closest to the domain analysis approach of Synthesis.) The result was the following products:

- A set of similarities among all approaches.
- A set of differences among the approaches studied. Each difference corresponded to something we believed would have an organizational impact. We therefore choose fairly high-level criteria for comparison; we deliberately avoided such factors as "approach A derives product X whereas approach B does not", since we could not relate these factors to decisions an organization would want to make in choosing between approaches.
- A terminology glossary that helped us find the similarities and differences.

The four approaches turned out to share certain characteristics. These included high-level objectives (the creation of artifacts that allow for effective reuse, and the capture and formalization of domain knowledge), the sources of domain knowledge (domain experts, reference materials, existing systems), and—to the extent that their objectives are similar—agreement on difficulties in performing domain analysis (the need for precise definitions of domain artifacts, how to validate the results of domain analysis, and economic considerations).

Table 1 shows the criteria we used to study the differences, and how each approach handles each criterion. Space precludes detailed discussion of them. Generally speaking, these criteria are from three categories:

- The relation to the software development process. Domain analysis may or may not impose constraints on software development processes and paradigms. This can have significant organizational impact: the less flexible an organization is to change, the more difficulty it will have in incorporating a domain analysis approach that imposes constraints on process. As Table 1 shows, an approach can be part of another life cycle—typically a pre-requirements activity (FODA, Jaworski, and Prieto-Díaz), or a complete software process in and of itself (Synthesis).
- The paradigm of problem space models. Domain analysis requires analyzing problems within a domain and determining solutions for those problems; ideally, both customers and developers communicate in terms of the domain-level concepts. The paradigm of the problem space model, then, shapes how people think about and communicate in the domain. In the approaches we studied, the problem space model can emphasize generic, reusable requirements

(Jaworski and Prieto-Díaz), it can be a decision model (Synthesis), or it can treat both equally (FODA).

- The primary product (or products) of domain development. This tells whether the paradigm of problem space models—the important analysis focus—is also the focus during development (or reengineering) of the reusable components that are used in applications in the domain. Table 1 shows that the primary product can be a reuse library (FODA, Jaworski and Prieto-Díaz) versus an application engineering process and supporting environment (Synthesis).

3 Results

The differences illustrate some fundamental differences in philosophies towards domain analysis. Most researchers agree that it is helpful for reuse, but differ in just what that approach to reuse should be. The two important characteristics that distinguish the approaches in this regard are the points in the process where reuse based on domain analysis can happen, and the degree to which domain analysis can determine how to do reuse.

We were primarily interested in understanding the Consortium approaches. We made the following observations based on our analysis:

- Prieto-Díaz's method is useful when there is an existing software base for a domain, when current and future projects can benefit from reuse but must design and possibly implement a significant portion of an application from scratch, and when application developers foresee a need for formalizing solution-level characteristics of a domain.

- Synthesis is useful when systematic, managed development and evolution of work products are concerns, when requirements are likely to change, and when long-term commitment to and investment in a business area is justified by corporate objectives.

The most important conclusion we have drawn is that there is no single "best" domain analysis approach. An organization should choose the one that best suits their software process needs, existing software base, and business objectives. However, we have noted that the approaches of Synthesis and Prieto-Díaz are not incompatible, and we are working on creating a software process that incorporates both at different points in the process—Synthesis as a framework, Prieto-Diaz where there is a need to organize an existing software repository in hopes of locating specific components needed in the larger context. We are also continuing our efforts by studying other approaches to domain analysis. We hope to determine other differences that might influence an organization's decision on which approach to use, and to uncover any variations within a criterion. In this way we shall gain a fuller understanding of the potential of domain analysis.

References

[Moor 89] Moore, John, and Sidney Bailin. *The KAPTUR Environment: An Operations Concept.* CTA Incorporated, Rockville, Maryland. June 1989.

[Camp 90] Campbell, Grady Jr. *Synthesis Reference Model.* Software Productivity Consortium, Herndon, Virginia. 1990.

[Jawo 90] Jaworski, Allan, Fred Hills, Tom Durek, Stuart Faulk, and John Gaffney. *A Domain Analysis Process.* Software Productivity Consortium, Herndon, Virginia. 1990.

[Kang 90] Kang, Kyo, Sholom Cohen, James Hess, William Novak, and Spencer Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania. 1990.

[Luba 88] Lubars, Mitchell. *Domain Analysis and Domain Engineering in IDeA.* Microelectronics and Computer Technology Corporation, Austin, Texas. September 1988.

[Diaz 87] Prieto-Díaz, Rubén. *Domain Analysis for Reusability.* Proc. COMPSAC'87. Tokyo, Japan, pp. 23-29. October 1987.

Table 1: Criteria for Comparing Approaches

	Method			
	FODA	**Jaworski**	**Prieto-Díaz**	**Synthesis**
Definition of "Domain"	Application area	Business area	Application area	Business area
Determination of Problems in the Domain	Top-down	Top-down	Bottom-up	Top-down
Specific Objectives and Products of Domain Analysis	Development of canonical architecture	Development of domain knowledge base	Learning more about immature domains, discovering facts about a domain	Products for application engineering
Permanence of Domain Analysis Results	Permanent	Mutable	Permanent	Mutable
Relation to the Software Development Process	Pre-requirements	Pre-requirements, waterfall model	Pre-requirements	Meta-process yielding application engineering process
Focus of Analysis	Decisions	Objects and operations	Objects and operations	Decisions
Paradigm of Problem Space Models	Decision model and generic requirements	Generic requirements	Generic requirements	Decision model
Purpose and Nature of Domain Models	Specification for software products	Repository of domain knowledge	Specification for software products	Specification for software process, products, environment
Organizational Model of Domains and Projects	Not specified			Projects are components of a domain organization
Approach to Reuse	Opportunistic	Systematic	Opportunistic	Systematic
Focus of Formalization Effort	Formalizing canonical models	Formalizing canonical models	Formalizing objects and operations	Formalizing canonical models
Primary Product of Domain Development	Reuse library	Reuse library	Reuse library	Application engineering process

Steven Wartik received the Ph.D. degree in Electrical and Computer Engineering in 1984 from the University of California at Santa Barbara. From 1981 to 1984 he worked on software development environments as part of TRW's Software Productivity Project. From 1984 to 1988 he was an assistant professor of Computer Science at the University of Virginia, where his research interests included software requirements and software configuration management. From 1989 to the present, he has been employed by the Software Productivity Consortium. He is currently part of the Synthesis project, studying systematic reuse strategies.

Rubén Prieto-Díaz is Principal Member of Technical Staff at the Software Productivity Consortium. He recommends reuse strategies within SPC and consults with member companies on establishing reuse programs. He was Princial Scientist at the Contel Technology Center responsible for the technical direction of the Software Reuse Project. Previously he was the lead architect for GTE's Asset Library System at GTE Laboratories.

Dr. Prieto-Díaz's research interests are in software reusability with emphasis in library systems, classification, retrieval, and domain analysis. He holds a B.S. in Aerospace Engineering from St. Louis University, an M.S. in Engineering Design and Economic Evaluation and an M.S. in Electrical Engineering, both from the University of Colorado at Boulder, and a Ph.D. in Computer Science from the University of California at Irvine. He is a member of IEEE, IEEE-CS, and ACM.