Defining Families: The Commonality Analysis

Mark A. Ardis
Bell Laboratories
Lucent Technologies
1000 E. Warrenville Rd.
Naperville, IL 60566 USA
+1 630 979 0042
maa@research.bell-labs.com

David M. Weiss
Bell Laboratories
Lucent Technologies
1000 E. Warrenville Rd.
Naperville, IL 60566 USA
+1 630 979 8392
weiss@research.bell-labs.com

ABSTRACT
A recent trend in both the software engineering research and industrial communities has been to seek ways systematically to engineer software domains. One approach is to develop families of software and to invest in facilities for rapidly producing family members. Success in such an endeavor requires that the software engineers be able to identify the desired family members. This tutorial describes the commonality analysis process, a systematic approach to analyzing families. Commonality analysis was developed at Bell Labs and is being tried in Lucent Technologies as part of a process for engineering domains that is known as family-oriented abstraction, specification, and translation (FAST). The result of the analysis forms the basis for designing reusable assets that can be used to produce rapidly family members. The tutorial teaches the participants the principles underlying the approach and gives them a chance to perform a practice commonality analysis guided by experienced users of the process.

Keywords:
software engineering, domain analysis, domain engineering, families, software process, application-oriented languages, reuse, requirements engineering

INTRODUCTION
A recent trend in both the software engineering research and industrial communities has been to seek ways systematically to engineer software domains. This tutorial describes the commonality analysis process, a systematic approach to analyzing domains that was developed at Bell Labs and that is in experimental use at Lucent Technologies. The tutorial teaches the participants the principles underlying the approach and gives them a chance to perform a practice commonality analysis guided by experienced users of the process.

The commonality analysis process views domains as families, as described in [5], and is an analytical technique for deciding what the members of a family should be. This technique is in use at Lucent Technologies as part of a domain engineering process known as family-oriented abstraction, specification, and translation (FAST). The goal of the FAST process is to develop facilities for rapidly generating members of a family; it is a variation on the Synthesis process described in [1] and [2]. Performing a commonality analysis is an early step in the FAST process, and is described in detail in [9].

ICSE 97 Boston MA USA

ENGINEERING FAMILIES
FAST is a software production strategy in which one plans for a system to exist in a number of variations, attempts to predict those variations, identifies what they have in common, and reuses the common aspects in producing the variations. Such a set of variations on a system may be considered to be a family, a relatively old idea in software engineering, suggested by Dijkstra and others in the software engineering literature as early as 1972 [3]. Parnas and others described approaches for building software families in the mid-1970s [5], [6], [7], [8]. This work emphasized the design and development of program families, but said little about how to decide what the members of a family should be. More recently, an area of study known as domain engineering has developed whose intent is to define families and assemble the assets needed to produce family members rapidly [2], [4].

The success of family-oriented software development processes depends on how well software engineers can predict the family members that will be needed. This problem is hard because the idea of a family is not well formalized, there are no rules that enable engineers to identify families easily, prediction of expected variations is difficult, and there is usually no time allocated in the development process for conducting an analysis of the family. Nonetheless, the payoff for conducting such an analysis can be quite high; it potentially reduces drastically the time and effort needed for design and for production of family members.

Commonality analysis is an early step in the FAST process, but it repays its practitioners in a variety of ways that are independent of FAST. It is performed as a moderated group discussion among domain experts that is organized into phases with specific objectives for each phase. As the

discussion proceeds, the domain experts produce a document, also known as a commonality analysis, that captures the results of each phase.

The analysis engenders a deep understanding of their domain among the experts, and helps them to develop standard terminology for the domain, a set of assumptions about what is common to all members of the domain, and set of assumptions about how domain members can vary from each other. This information is critical for use in creating reusable assets for the domain, such as a specification language from which domain members can be generated, a design common to all members of the domain, and reusable, adaptable components that can be used to create members of the domain very rapidly.

TUTORIAL CONTENTS

The tutorial is designed to teach participants how to perform a commonality analysis. It consists of a set of lectures and small exercises that introduce participants to the motivations underlying a commonality analysis, the structure of the commonality analysis process, and the structure of the commonality analysis document. The participants gain an understanding of the following:

- when the process is useful,
- the benefits of performing a commonality analysis,
- how and where the results may be used,
- who should participate in a commonality analysis,
- the phases of the analysis and the motivation for each phase, and
- the form of the commonality analysis document.

In addition, participants are organized into groups and guided through an example commonality analysis. By the end of the day they are prepared to participate fully in an analysis in a domain of their own choosing.

The lectures are conducted in an informal style, encouraging audience participation. We often ask participants for personal examples of artifacts or processes relavant to the discussion.

This tutorial is based on a course developed by Bell Labs researchers for internal use at Lucent Technologies. A proprietary version of it has been taught several times.

REFERENCES

1 Campbell, G.H. Jr., Faulk, S.R., Weiss, D.M.; *Introduction To Synthesis*, INTRO_SYNTHESIS_PROCESS-90019-N, 1990, Software Productivity Consortium, Herndon, VA.

2 Campbell, G.,O'Connor, J., Mansour, C., Turner-Harris, J.; "Reuse in Command and Control Systems," *IEEE Software*, September, 1994.

3 Dijkstra, E. W., "Notes on Structured Programming," *Structured Programming*, O.J. Dahl, E.W. Dijkstra, C.A.R. Hoare, eds., Academic Press, London, 1972.

4 Neighbors, J., "The Draco Approach to Constructing Software from Reusable Components," *IEEE Transactions on Software Engineering*, SE-10, 1984.

5 Parnas, D.L., "On the Design and Development of Program Families," *IEEE Transactions on Software Engineering*, SE-2:1-9, March 1976.

6 Parnas, D.L., "Designing Software For Ease Of Extension and Contraction," *3rd International Conference on Software Engineering*, May 1978.

7 Parnas, D.L., Clements, P.C.; "A Rational Design Process: How and Why to Fake It," *IEEE Transactions on Software Engineering*, SE-12, No. 2, February 1986.

8 Parnas, D.L., Clements, P.C., Weiss, D.M.; "The Modular Structure Of Complex Systems," *IEEE Transactions on Software Engineering*, SE-11., pp. 259-266, March 1985.

9 Weiss, David M., "Defining Families: The Commonality Analysis," submitted to *IEEE Transactions on Software Engineering*.