

Panel

Domain Analysis: From Tar Pit Extraction to Object Mania?

Steven Fraser, Moderator

Panelists: Spencer Peterson, Douglas Schmidt, Mark Simos, Will Tracz, Nathan Zalman

Abstract

Steven D. Fraser
Northern Telecom, Santa Clara, CA.
sdfraser@bnr.ca

Features — The Heart of Object Integration

A. Spencer Peterson
Software Engineering Institute, Pittsburgh, PA.
asp@sei.cmu.edu

Domain Analysis has been researched, debated, and applied for fifteen years. There are many documented methods available. The collective heritage of these methods covers the spectrum from the procedurally-oriented to the object-oriented paradigms. Application focus includes re-engineering (tar pit extraction?), requirements engineering, organizational learning, reuse, class/object identification, and design pattern identification (object mania?).

This panel brings together perspectives on domain analysis by five practitioner methodologists. Diverse experiences including technology insertion, stakeholder analysis, and team techniques are represented. The intent is to focus on the application of domain analysis methods. The objective of the panel is to share the experiences of the participants (both panelists and audience) and to offer practical advice.

Keywords: Design patterns, domain analysis, DSSA, FODA, ODM, software reuse

Steven Fraser is an Advisor with the Software Design Process Engineering team at Northern Telecom's (Nortel) Lab in Santa Clara, California. Previous to this he spent four years at Bell-Northern Research's Computing Research Laboratory in Ottawa, Canada. In 1994 he was a Visiting Scientist at the Software Engineering Institute (SEI) collaborating with the Application of Software Models project on the development of team-based domain analysis techniques. Since joining BNR in 1987, Fraser has contributed to the ObjecTime project, an OO-based CASE-Design Tool and to the BNR BCS software development process. Fraser completed his doctoral studies at McGill University in Electrical Engineering. He holds a Master's degree from Queen's University at Kingston in applied Physics and a Bachelor's degree from McGill University in Physics and Computer Science. He is an avid operatunist and photographer.

The Feature-Oriented Domain Analysis (FODA) method was developed by the Software Engineering Institute (SEI) in response to a perceived need for a domain analysis method that emphasized *the identification of prominent or distinct user-visible aspects ... in a domain*, i.e. features. This method, first published in 1990 as [1], has had a major impact on the state of domain analysis (DA) methods. FODA was one of the DA methods compared in [2] and has been cited as a basis for many DA approaches, as seen in [3] and by eight papers/authors in [4].

The notion of identifying features in a domain is the heart of the FODA method. Features are the chief mechanism for defining the commonalities of and differences between various capabilities in a domain. FODA, and many of its spin-offs, are DA methods that place great emphasis on features. In [1], three distinct classes of features were defined (mandatory, alternative, and optional) depicted as an And/Or tree. Also, an initial categorization of features was presented. In a later report [5], the categories were refined into three distinct groups: Context, Operational, and Representation features. These categories and the features tree, along with the Information (using Entity-Relationship or semantic data models, or both) and the Operational Models of FODA (and FODA-like methods), are the basis for developing object-oriented software components that will integrate together easily into systems that satisfy all of the user requirements.

The context features branch of the Feature Model tree provides valuable knowledge of the reasons for variability of the software products within a product family/domain. In many cases, context features are the driving force behind the selection of various alternatives. They can specify global constraints on software components (such as time/space performance issues) that cannot be allocated singly to components that embody operations, i.e. lower-level operational features. Similarly, the selection of various

representation features can play a major role in the operational capabilities and information needs of potential software products derivable from domain analysis results ([5] and [6] provide evidence for these claims).

Object-oriented (OO) software can be readily developed from FODA models. The kinds of data needed and their interrelationships are portrayed in the Information Model. The approach taken in [7] for developing software components from DA models describes a Domain Design process for creating object specifications and their implementations. Steps included in this process include object identification (what data entities are and are not encapsulated as objects), derivation of object operations/methods from operational features with their classes, and appropriate examination of the Operational model, particularly for data input and outputs. Furthermore, this approach describes a process for grouping objects at a higher level of abstraction than the objects themselves, thus avoiding the flat and overly broad designs seen in many OO design methods. The main criterion for grouping objects is the usage of the operational features to collect the objects (and their operations) together that provide the properties of a higher-level feature. These collections of objects, called subsystems in [7], provide a means for looking at software requirements and designs at the levels of abstraction needed to make reuse of designs a reality.

[7] also contains an example of the usage of the design approach described briefly above. The example focuses on the use of a particular architecture (the Object Connection Architecture (OCA), described in [7]) and implementation language (Ada83), but the overall approach is amenable to the use of other architectural styles and/or languages, as suggested in the report. The use of design patterns in implementing objects and higher-level abstractions within such an approach is not yet fully resolved.

A Features Model can describe the interrelationships between objects and various qualities of software products beyond objects and their methods, along with denoting the objects and methods themselves. These abilities makes DA methods that place high importance on feature collection and categorization highly valuable methods for performing domain analysis with object-oriented intentions. FODA and its descendants are good examples of such methods.

References

- [1] Kang, Kyo C.; Cohen, Sholom G.; Hess, James A.; Novack, William E.; and Peterson, A. Spencer. *Feature-Oriented Domain Analysis (FODA) Feasibility Study* (CMU/SEI-90-TR-21). Pittsburgh, PA.: Software Engineering Institute, Carnegie Mellon University, 1990.

- [2] Wartik, Steven; and Prieto-Diaz, Ruben. *Criteria for comparing reuse-oriented domain analysis approaches*, p. 403-431, International Journal of Software Engineering and Knowledge Engineering, vol. 2:3; Sept. 1993.
- [3] Tracz, Will; Coglianesi, Lou; and Young, Patrick. *A Domain-Specific Software Architecture Engineering Process Outline*, p. 40-49, ACM SIGSOFT Software Engineering Notes, vol. 18:2; April 1993.
- [4] Collected Papers from OOPSLA'95: Workshop 14, Application of Domain Analysis Techniques to Object-Oriented Systems, Editors: Honna Segel, Steven Fraser, and Jim Coplien, Oct. 1995.
- [5] Cohen, Sholom G.; Krut, Robert W.; Peterson, A. Spencer; & Stanley, Jay L. *Application of Feature-Oriented Domain Analysis to the Army Movement Control Domain* (CMU/SEI-91-TR-28). Pittsburgh, PA.: Software Engineering Institute, Carnegie Mellon University, 1992.
- [6] *In-Transit Visibility Modernization (ITVMOD) Domain Modeling Report* (STARS-VC-H002a/001/00), Reston, VA.: Comprehensive Approach to Reusable Defense Software (CARDS), Unisys Corporation, Aug. 1995.
- [7] Peterson, A. Spencer; & Stanley, Jay L. *Mapping a Domain Model and Architecture to a Generic Design* (CMU/SEI-94-TR-8). Pittsburgh, PA.: Software Engineering Institute, Carnegie Mellon University, 1994.

Peterson is a Member of the Technical Staff at the Software Engineering Institute in Pittsburgh, PA. He has worked on all of the teams dealing with the development and maturation of FODA during his tenure there, starting in 1987. He has (co)authored numerous reports and papers on FODA and related topics, including the application of FODA in several domains. He holds a BS in Computer Science from California State University, Hayward and a Master of Software Engineering from Carnegie Mellon.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.