Some Experiences in Domain Analysis

Ruben Prieto-Diaz

Reuse, Inc.
12365 Washington Brice Rd.
Fairfax, VA 22033
Tel & Fax: (703) 620-5385
Email: 70410.3014@compuserve.com

Abstract

In this position paper I describe three of my experiences in doing domain analysis and in transferring the technology. Specifically I describe analyses in the domains of command and control systems, flight simulators, and avionics displays. These domains are rich enough to demonstrate the value of domain analysis and show, at the same time, the relatively immature state of domain analysis.

Keywords: domain analysis, the "sandwich" method, command and control systems, avionics systems.

Workshop Goals: to share experiences in the application of domain analysis methodologies.

Working Groups: domain analysis and engineering.

1 Background

Domain analysis is a concept that keeps troubling organizations implementing reuse programs. Although domain analysis is a simple concept, like reuse, its practice is not simple. Domain analysis is similar to systems analysis but for a class or family of systems rather than for a single application. The objective in domain analysis is to find commonalities among systems in the same domain or application area and synthesize generic models or architectures that characterize families of applications. Domain analysis is a key activity in pre-planned reuse.

One of the objectives in pre-planned reuse is to have a repeatable process for doing domain analysis. Some guidelines and methods have been proposed but it is still a research area. These methods are very recent and data on their performance is not available yet. The three leading methods for domain analysis are: FODA (Feature Oriented Domain Analysis) from the Software Engineering Institute in Pittsburgh [1]; the STARS Reuse Library Process Model, also known as the Sandwich method [2]; and the Domain Analysis and Design Process from DISA/CIM (Defense Information Systems Agency/Center for Information Management), in Arlington, Virginia [3].

2 Position

I am an advocate of the Sandwich method since, of course, I am its inventor (btw, a possible name for this method could be Ruben's Sandwich.) I have used this method in several occasions and taught it to several organizations. The following three experiences provide a flavor of what you can expect from doing domain analysis using this method.

My first experience in applying this method was for the domain of command and control systems. My employer at the time was interested in reducing development costs and improving quality by reusing parts from their legacy systems. I was assigned to work with a veteran expert with 17 plus years of experience in developing C2 systems. We decided to focus on three operational systems that were well documented. Requirements documents were made available to us, however, we did not have access to source code. C2 systems are highly classified.

We began bottom-up by inspecting selected samples from the massive requirements specifications. We did a preliminary vocabulary analysis and were able to separate mechanized from manual activities. This separation helped us to properly scope the domain and draw a domain boundary. Human driven activities such as decision making, information review, comparing courses of action, etc. were considered outside the domain. We also found a high incidence of activities related to message processing. This finding gave us a clue to pay close attention to the role of message processing in C2 systems.

Next, we attempted a top-down analysis. We began by comparing the architectures of the three systems and could not find any similarity; each architecture looked completely different. One of the reasons, as explained by the expert, was that the designs of these systems were hardware-driven, that is, the equipment was selected first and the requirements drafted to accommodate the hardware. Software was, in most cases, considered the glue that makes systems work. We stepped back from this approach and reviewed basic principles and theory of C2 systems. We then went back to the architectures and were able to recognize common patterns.

After several iterations and consultations with other experts, we were able to propose a high-level generic architecture, common not only to all three systems, but, to future systems as well. We went back to our bottom-up analysis and did a vocabulary analysis on the data manipulated by C2 systems. We discovered that all data that flows through the system could be formatted in a small set of standard templates. We went back to the architecture postulated during initial top-down and refined it to accommodate for standard (i.e., template) processing of messages. Here we were fully engaged in a sandwich process.

The new re-structuring of the architecture allowed us to discover yet another interesting characteristic of our architecture- almost every subsystem of a C2 system can be represented as a specialized message processor. We then proposed a generic architecture for message processors and demonstrated that a C2 system could be build by assembling different instantiations of message processors. Needless to say, our work was initially received with skepticism, however, as the idea of being able to build C2 systems out of leggo-like parts sank in, the engineers bought into it completely.

This effort took us six months working half time, a total of 3 MM. The following two cases are short experiments on trying to teach domain analysis hands-on. Since domain analysis is a difficult and complex technique, and since it has not evolved into a mature process or method yet, I decided to teach it by example on domains familiar to the participants. I designed a three-day course. The first day for presenting concepts and the method and the last two for starting a domain analysis on a domain selected by the class (i.e., the customer.)

The organization in the first case wanted to use the course to analyze the domain of flight simulators. After an initial scoping, the class realized the domain of flight simulators was broad and complex. I proposed two alternatives: to do a high level, general analysis of the domain, or to look for a subdomain suitable for a more detailed analysis. The class selected the latter. After looking at several components and activities of flight simulators, we found that the design and validation of empirical equations that characterize aircraft behavior was an extremely time consuming endeavor.

Typically, designers work with a virtual aircraft, one that exists on a drawing board and in mathematical models. In order to achieve realistic simulation, the coefficients of several complex equations have to be found. The process is by trial and error. The designer, literally, plays with the coefficients until the expected behavior is observed. To be able to do this, an engineer has to create an interactive environment for plotting the equation and for manipulating the coefficients. Unfortunately, each environment is different because equations are different. As a result engineers spend most of their time creating and tuning the environments and only a fraction of their time playing with the coefficients.

We did a preliminary vocabulary analysis of the concepts in this activity and produced a taxonomy of classes of equations, behaviors, and environments. We then designed generic scripts (i.e., templates) in their high level graphical language for instantiating different kinds of environments. Since this was a two day effort only, we were limited to planning the next steps: to implement several scripts and to make them available in a catalog to design engineers. A few months later I learned that design engineers were reusing predefined templates that they could quickly customize for playing with their equations.

The domain of avionics displays was the topic of the next experience. Again, the class recognized that the domain was broad and complex, but in this case they decided to take a stab at the whole domain. We started bottom-up with a vocabulary analysis of all concepts, keywords, nouns, and verbs used in avionics displays. The concept of "tapes" appear to recur in most kinds of displays. Tapes are display elements with graduation marks used to measure variable parameters. Temperature, airspeed, attitude, pressure, etc., all are displayed in some kind of a tape form. Since each parameter was considered unique, engineers were designing their own tapes. The course helped them to realize that designing a generic tape package or a family of tape classes would make their job easier, promote reuse, and help them to agree on some standards. Although a generic architecture of avionics displays was not derived during the seminar, the concept of standard tapes more than paid off for the unmet expectations.

I hope that these experiences demonstrate the value of domain analysis and show, at the same time, the relative immature state of domain analysis. There is significant research efforts aimed at improving domain analysis and at integrating domain analysis as part of software development. I also hope that these brief descriptions have proven useful and provide some reusable experiences.

References

[1] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November 1990.

[2] "Sandwich Method - A Domain Analysis Method Developed by Reuse, Inc. and Available in Reuse Library Process Model," Tech. Rep. IBM STARS CDRL 03041-001, U.S. Air Force Electronic Systems Center, Hanscom Air Force Base, MA, July 1991. Also available from ASSET, (304)594-3954.

[3] DISA/CIM Software Reuse Program, 701 South Courthouse Rd., Arlington, VA, 22204-2199, *Domain Analysis and Design Process*, March 1993. POC: Sherrie Chubin (703)285-6900.

3 Biography

Ruben Prieto-Diaz is president and founder of Reuse, Inc., a software-reuse consulting company. He is contract consultant to Scientific Applications International Corporation for STARS, developed the STARS Reuse Library Process Model, and is participating in the STARS Demonstration Project. He is co-editor of Domain Analysis and Software Systems Modeling, IEEE Computer Society, 1991, and Software Reusability, Ellis Horwood Workshop Series, Chichester, England, 1993, and the author of several technical papers in software reuse, classification, and domain analysis.

Prieto-Diaz received a BS in aerospace engineering from St. Louis University, a MS in engineering design and economic evaluation, and a MS in electrical engineering both from University of Colorado, Boulder, and a PhD in computer science from University of California, Irvine. He is a member of IEEE, IEEE-CS, and ACM.