The Commercial Application of Domain Analysis

James M. Neighbors
Bayfront Technologies, Inc.
1280 Bison B9-231
Newport Beach, CA. 92660
tel/fax:(714)436-0322

Keywords: domain analysis, domain engineering, application software, domain-specific architectures

Workshop Goals:
Compare our scheme of domain-specific application development with other approaches.

Working Groups:
domain analysis/engineering, reuse process models, reuse and formal methods

1 Background

My interest and experience with reusable software components and automatic programming led me to the concept of domain analysis as a basis for reuse. The key domain analysis idea is the reuse of concepts not just program code. This approach was embodied in an experimental system called Draco which has been well documented and will not be further discussed here (concept [Neighbors80], theory [Neighbors84], overview [Freeman87], problems [Arango86], rationalization [Neighbors92]).

My current interest in reuse follows along the same lines. Use domain analysis [PrietoDiaz91] but from a different psychological set. All of my previous work really emphasized the tool, deemphasized the domains, and completely ignored the users. My current interest is how to deliver the power of a problem-specific domain to a user without making the user into problem domain expert, tool expert, or modeling domain expert. This goes deeper than human interface issues. The detailed information should still be available but I would doubt that it is used very much. This kind of information is only of use to a domain expert and it is my experience that for most application domains there aren't very many experts.

2 Position

There are many problem domains. Some domains are core domains of many systems (e.g., graphics, database, network). These are the usual computer science kind of modeling domains. They are supported by hard-won theory and experience through 40 years of computer science. The more application-specific domains which use these core domains (e.g., CAD, CAM, CAE) have less theory and fewer common notations attached to them. These general areas seemed to have formed about 25 years ago. The very application-specific domains on top of these (e.g., parametric geometry, stereoscopic lithography, transport phenomena) have little (or evolving) theory and notation. These seem to bloom and many die within a 10 year lifecycle. The essence here is that the formation of domains seems to be an evolutionary process. New strains show up in different major branches. Some last, some die out. Some, like spreadsheets, are major paradigm shifts.

Given this evolutionary domain cycle I could make a case for codifying the known information about the core set of domains in some kind of knowledge net. For our purposes here the form of the knowledge net is unimportant. The core set of domains have lasted the longest, are least likely to have major changes, and are the most used. As a computer scientist it is easier to communicate my work to peers. The following avenues of research are open to me:

1. Codify what is already known about the core domains. Sometimes this is not viewed as acceptable research because it does not create any "new" knowledge. I believe it provides structure to what we already know and argue [Neighbors92] that work like [Batory88] is much more important than this weeks new but unproven theory. All of the core domains in the computer science literature are ripe for this kind of exploitation.

2. Work on the form of the general knowledge net describing the structure of the domains. AI has been working on this for 30 years so you could probably spend your entire career doing this. Each different use of the knowledge requires a different view which requires representation in the knowledge net. This is a rich area of research.

3. Work on the general process of having an organization recognize, codify, and reuse domain models as part of ongoing development. Mankind has studied the knowledge acquisition process for thousands of years. Philosophy even comments here as what is knowable and unknowable. The dynamics of organizational infrastructure continually change as we continue the information revolution. This is another interesting area of research.

All of the above areas of research are intellectually rich and challenging. We need research in all these areas. Does research in these areas help users build domain-specific applications? No, not directly. There is no application domain-specific information in any of these areas.

For me this brings up an interesting point. Just who exactly do we expect to produce application domain-specific information? As a researcher I've enumerated rich research areas with important problems. General solutions I find to these problems will be well received by my research peers. I could spend a lifetime researching these problems and doing examples. I'm not exactly motivated to investigate, say, general 5-axis machining fillet solutions under surface-surface intersections for a derivative CAM NC domain. I could (and have been known to) make this a process issue by hypothesizing a person (or entity, e.g., corporate committee) with enough application knowledge and modeling abstraction knowledge to form coherent domains. Who knows if this works under any methodology? All the results of this research will remain simple intellectual curiosities until we try these ideas out in a commercial environment.

Armed with the above point of view, four years ago I attempted to interest some large commercial and government organizations in doing a pilot study. The problem I found was that in large organizations even a simple project is very expensive. Only development divisions have the resources to perform such a study. Development divisions are development oriented. They are the ones with the application domain knowledge. They have a development schedule which is directly tied to cash flow. Performing a classical systems analysis of a collection of the developed systems can result in a first approximation of a domain analysis. The CAMP project is an example of this approach. These models can accelerate development and subsequent cash flow. The danger is that the research analyst ends up in the stream of development because it is worth a lot to the developing division. So research divisions in large organizations have to hold their corresponding development divisions at arms length. This makes pilot studies in large organizations difficult.

Once again armed with this large organization experience we decided to try out the use of domain analysis as a small commercial company, Bayfront Technologies Inc. Two years ago we chose a domain we thought would interest other companies and would show off the power of a domain analysis. We chose the development of telephone and data communications protocols. We were pleased to see that other researchers also saw this as a domain [OMalley90]. In fact this area has since become known as computer-aided protocol engineering (CAPE).

As a commercial product our users are not primarily interested in the theory of the tool or the knowledge it encapsulates. They are primarily concerned with the power ratio of the tool (effort with tool divided by effort without tool) and the quality of the results. As researchers we would proudly unfurl the innards of our modeling domains for other researchers. The customers could care less. When code generation is performed, the final code is usually embedded in the customers system. They do not want to make low level decisions unless it impacts their interface. If it impacts their interface they prefer a simple method to state their request (e.g., compiler switches). They want a quick development cycle, a lot of quality analysis, and a final optimize capability. This should not surprise us because the same is true of language compiler tools.

We believe the concept of domain analysis is holding up well. Our CAPE tool uses a single application-specific language (protocol description language) tied to modeling domains (streams, layer definitions, message handling, timers, memory management) to generate code, perform static analysis, perform dynamic analysis, generate simulations, and draw documenting diagrams. We have successfully specified, tested and generated real protocols such as the IEEE802 data communications protocols and the ISDN Q931 telephony protocol. All the modeling domain languages are hidden from the users. We use the general framework presented in [Neighbors84] for manipulating the domains. We are able to use the same framework and modeling domains to attack new commercial problem areas.

Our chosen problem area is in one of the core domains – networks. We chose this area for many of the reasons it is attractive to research with core domains. There is a literature, theory, experience and experts. The surprising part to us has been that the users have taken over the analysis of many of the domains. I do not mean just the application level protocol description language. I also mean the low-level modeling domains. Because our generated code is embedded in customer systems, the customers provide us with tricks for implementing protocols. Some of these we have later found in the literature. Many we have not found in the literature. This is in the best tradition of domain analysis.

To answer my earlier question "Just who exactly do we expect to produce application domain-specific information?" for us the surprising answer seems to be the end users. It is a pump priming problem. Our end users are development oriented. They do not want to have anything to do with a tool builder until the tool aids their development process. This is a problem for domain-specific tool builders because the end users are really the ones with significant domain-specific knowledge. A knowledge based tool is always weak without their input. The benefit of this situation is that once the pump is primed the organizational leverage of development can be used to power the tool development. As yet we cannot report absolute success or failure but at least the experiment is underway.

3 References

[Arango86] G. Arango, I. Baxter, P. Freeman, and C. Pidgeon,
"TMM: Software Maintenance by Transformation,"
IEEE Software, pp. 27-39, May 1986.

[Batory88] D. Batory,

"GENESIS: An Extensible Database Management System", *IEEE Transactions on Software Engineering*, vol. SE-14, no. 11, pp. 1711-1730, November 1988.

[Freeman87] P. Freeman, "A Conceptual Analysis of the Draco Approach to Constructing Software Systems," *IEEE Transactions on Software Engineering*, vol. SE-13, no.7, pp.830-844, July 1987.

[Neighbors80] J. Neighbors, *Software Construction Using Components*, Ph.D. Dissertation and Report UCI-ICS-TR160, University of California, Irvine, ICS Dept., 1980.

[Neighbors84] J. Neighbors, "The Draco Approach to Constructing Software from Reusable Components," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 5, pp. 564-574, September 1984.

[Neighbors89] J. Neighbors, "Draco: A Method for Engineering Reusable Software Systems", in *Software Reusability*, T. Biggerstaff and A. Perlis eds., vol. 1, pp. 295-319, Addison-Wesley 1989.

[Neighbors92] J. Neighbors, "The Evolution from Software Components to Domain Analysis", *International Journal of Knowledge Engineering and Software Engineering*, to appear in special issue on domain analysis, J. Butler and G. Arango eds.

[OMally90] S. OMalley, *AVOCA: An Environment for Programming with Protocols*, Ph.D. Dissertation and Report TR90-31, University of Arizona, Tucson, August 1990.

[PrietoDiaz91] R. Prieto-Diaz and G. Arango, *Domain Analysis and Software Systems Modeling*, IEEE Press, 1991.

4 Biography

James M. Neighbors is a principal in the consulting firm of SADA. He specializes in the assessment, restoration, and management of very large software systems at risk of failure. He is also a founder of Bayfront Technologies Inc. working on the application of domain analysis to commercial application domains. He was previously on the faculty of the Information and Computer Science Dept. of the University of California, Irvine. He received B.A. Physics '74, B.S. Computer Science '74, and Ph.D. Computer Science '80 all from the University of California, Irvine.