Wallnau
N93-17528

CARDS: A Blueprint and Environment for Domain-Specific Software Reuse

Kurt C. Wallnau, Anne Costa Solderitsch and Catherine Smotherman
Paramax Systems Corporation
(A Unisys Company)
Farimont, West Virginia and Paoli, Pennsylvania

S29-61
136903

CARDS (Central Archive for Reusable Defense Software) exploits advances in domain analysis and domain modeling to identify, specify, develop, archive, retrieve, understand and reuse domain-specific software components. An important element of CARDS is to provide visibility into the domain model artifacts produced by, and services provided by, commercial computer-aided software engineering (CASE) technology. The use of commercial CASE technology is important to provide rich, robust support for the varied roles involved in a reuse process. We refer to this kind of use of knowledge representation systems as supporting "knowledge-based integration."

1. Introduction

The problem of achieving satisfactory levels of reuse in the development of defense software has been challenged in recent years, but with limited success. A development which will surprise no one in the AI community is a recent focus by the US DoD on attacking the reuse problem on a per-domain basis. A notable example is the CAMP project [1]. CARDS (Central Archive for Reusable Defense Software) attempts to exploit advances in *domain analysis* and domain *knowledge representation* to identify, specify, develop, archive, retrieve, understand and reuse domain-specific software components* — and to do so in a way that is independent of the underlying application domain.

We view the domain analysis and domain knowledge representation as the key to achieving the CARDS objectives — with special emphasis on *understanding* the relationships between software components and the domain model. However, the stipulation that CARDS should be applicable across a variety of application domains has interesting consequences on the construction of a blueprint and environment for domain-specific reuse.

2. Divergent Roles and Environments

The defense department develops systems spanning many domains — exactly how many is a matter of contention and will only be resolved when a concise definition of domain is available and is applied to defense department procurements. Software continues to be a critical component of systems developed in most of these domains. Attaining high-leverage reuse within narrowly focused application domains is well-justified by research, experience and economics. However, to institutionalize domain-specific reuse, a blueprint detailing how to undertake the development of a domain-specific reuse library, and a computer-aided support environment for putting the blueprint in action, is necessary.

The problem confronted by CARDS is the multiplicity and divergence of dimensions, or elements, of any CARDS architecture†. For example, CARDS must support a variety of roles, where roles are task-related personifications of activities necessary to achieve reuse. Examples include: performing domain analysis; using the results of a domain analysis (i.e., the domain model) to identify abstract interfaces; specifying the concrete interfaces; implementing the components; designing a user-friendly library classification scheme; archiving components within the classification scheme; and, ultimately, the end-user role of locating and retrieving components. Figure 2-1 illustrates a strawman architecture for a CARDS environment.

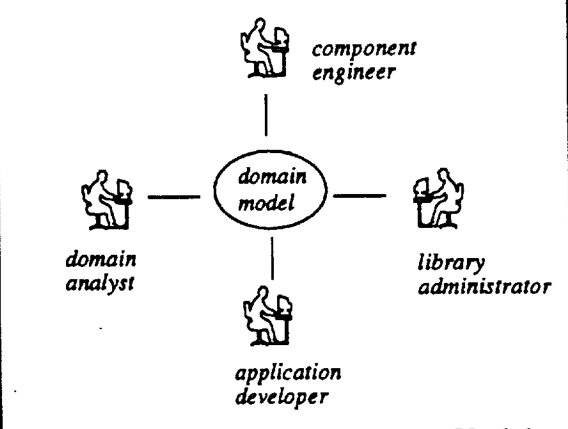

Figure 2-1 Central, Shared Domain Model

Of course, Figure 2-1 is overly simplistic. Since each of these roles represents a different perspective (and several roles are missing), different processes, methods and support technology will need to be brought to bear to support

*. Software components include assets such as requirements and design models, parts generators, programs, etc. See [2] for more details.

†. We use the term "architecture" to refer to the *blueprint* and support technology.

different kinds of tasks. For example, the kinds of information produced and consumed by a domain analyst will be different from that produced and consumed by a component engineer. One way to address divergent roles is to provide alternative views into a shared knowledge base, as illustrated in Figure 2-2. This is the approach that is taken

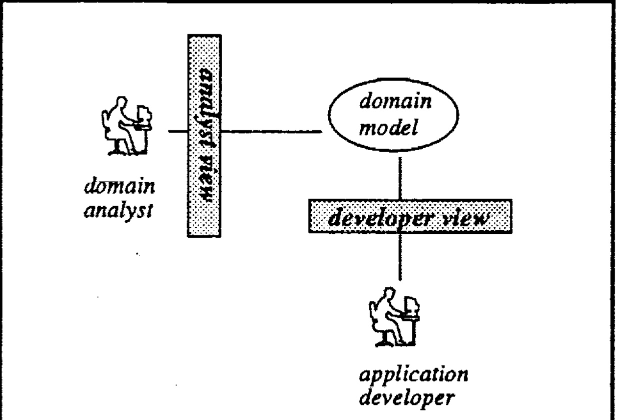

Figure 2-2 Uniform Domain Model & Views

in classical software development environment architectures [3] as well as hypertext-oriented knowledge representation frameworks [16].

Of course Figure 2-2 is also overly simplistic. First, there is no consensus regarding domain analysis process, method or representation. It appears that the choice of domain analysis technique depends to some extent upon the desired end-result of the analysis — e.g., supporting reuse, understanding a system, comparing different systems, etc. For example, Diaz's analysis technique [4] for reuse differs substantially from Brown's informal [5] technique for comparing software environment architectures, while LaSSIE makes use of a uniform, formal knowledge representation scheme for managing the complexity of a layered system [6].

Second, domain analysis techniques will vary across application domains. For example, information management application domains may be suitably modeled using classical structured analysis and structured design techniques; real-time systems may require the addition of behavioral models and temporal logics; complex, interactive systems may be best modeled using object-oriented techniques. While, in theory, each of these techniques has an analogue in a more generic knowledge representation formalism, such a mapping would not be practical.

Third, even within isolated application domains it may be useful to employ a variety of domain analysis and representation techniques. For example, the SEI feature-oriented domain analysis method (FODA) [7] employs an eclectic assortment of representations. Besides FODA, the notion of refinement, crucial in various formal design methods, implies a mapping among various representations, for example Z [17] specifications to program source. Thus, focusing on support for the domain analyst role, a more realistic CARDS architecture is illustrated in Figure 2-3.

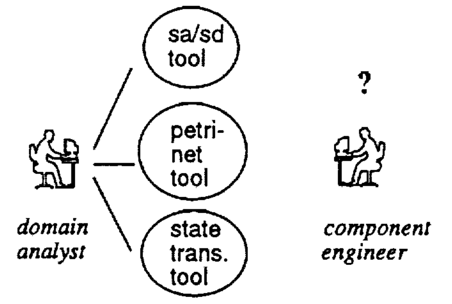

Figure 2-3 Eclecticism and Tool Coalitions

There is an underlying pragmatic basis for Figure 2-3 as well — while domain analysis and knowledge representation are better understood today than just a few years ago, the technology is still unstable. Further, there is an existing body of commercial CASE tools available which can support practical application of domain analysis techniques.

There are severe problems underlying Figure 2-3. The collection of analysis tools employed by the domain analyst — in essence the domain analysis environment — are not likely to be well integrated with respect to the domain analysis process, the logical services provided by the tools, nor the underlying tool mechanisms [8]. The tools themselves are at worst completely egocentric and at best wired together in some loose form of tool coalition [9]. This makes it difficult to verify the completeness and consistency of domain models.

Just as serious is the lack of integration of the domain analysis environment with the environment required by the component engineer. Not only will it be difficult for the component engineer to locate and understand the portions of the domain model relevant to the construction of software components, but the component engineer will also have specialized tools to support development tasks, e.g., coding, performance, annotation, testing and configuration management tools. The conceptual distance between the analysis tools and development tools makes even tool coalitions an unlikely prospect. A similar impedance mismatch exists between other roles in the CARDS architecture.

3. Knowledge-Based Integration

Figures 2-2 and 2-3 illustrated the dichotomy between an idealized view of a domain-specific reuse environment, and the view most likely to emerge from the combination of state-of-the-practice tool support and the requirement for domain-independence of the CARDS architecture. These views need to be merged. That is, we must provide a semantically meaningful view, for each role, into a domain model, while not sacrificing the tool support necessary to support the processes associated with a particular role.

Our approach is to merge these views, initially using the STARS* Reusability Library Framework (RLF) [10] as a meta-model for relating, and integrating, services provided by and artifacts produced by different tools. The hybrid knowledge-representation system in RLF combines a semantic network system based upon KL-ONE, with an extensible, *typed* rule-base system. A high-level view of this architecture is depicted in Figure 3-1.

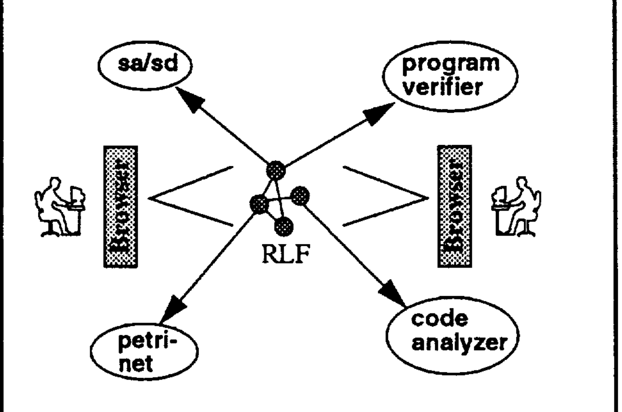

Figure 3-1 Meta-Model Integration in RLF

The architecture highlighted in Figure 3-1 has several interesting properties. First, the RLF knowledge base provides a single meta-model which a) uses the semantic network to relate the artifacts produced by various tools, and b) uses *action* rule types to tie tool services to tool artifacts†. The first property increases the visibility to relationships among elements of the domain model that are created by one role but semantically meaningful to other roles in the CARDS environment. The second property leverages the substantial investment in existing CASE technology and preserves a convenient, comfortable and functional environment already tailored to role-specific processes.

Second, the browser allows various users in the CARDS environment to view only those portions of the knowledge base that are appropriate for their role. Two forms of view filters are possible: through the graphical browser, and through the use of *advisor* librarians (also available through the browser). The former is a relatively straightforward user-interface problem. The latter is supported through the use of various rule types which are used by a special advisor inference engine — TAU.

Third, the use of RLF provides the basis for the development of other specialized types of inferencers to support the reuse process. One inferencer — Gadfly [11], has already been prototyped to support component specification and qualification. Other inferencers have been developed using a similar hybrid knowledge representation system for systems diagnostic maintenance [12] and (more closely related to software component reuse) hardware configuration [13].

4. CARDS and the Reuse Process

The architecture in Figure 3-1 is sketchy and only briefly discussed because the real problem is not the mechanisms of the CARDS environment, but the use of it within the context of an overall reuse process. A number of questions will need to be answered, perhaps some of them on a per-domain basis:

- How much of the domain model should be captured in the knowledge base, versus its use as an index into tool artifacts and tool services?
- What are the appropriate views into the knowledge base? For example, should an application developer's view be based upon models of architectures [14] or requirements [15]?
- When is a domain *ripe* for reuse [2]?

While these discussions have focused on the integration of different user roles with a reuse repository‡, another dimension of integration can be found when viewing a domain-specific reuse library as a bridge between supply-side and demand-side reuse processes. As illustrated in Figure 4-1, the scope of a repository can vary according to the nature of the domain analysis processes, e.g., how close is the "fit" between the domain analysis process and the domain modeling services provided by the repository, and the nature of the demand-side processes, e.g., who on the demand side will be using the repository?

In Figure 4-1, two parallel life-cycle processes are depicted: domain engineering and software engineering represent the supply-side and demand-side reuse processes, respectively. The repository can be scoped to capture the by-products of different domain engineering subprocesses; such decisions about scoping can result from, or can result

*. STARS — Software Technology for Adaptable, Reliable Systems.

†. A similar integration approach is provided in Frame Technology's *Live Links* and in several other systems.

‡. The use of a domain model as a kind of repository has been implicit throughout the discussion. The terms "archite," "library" and "repository" are also used synonymously.

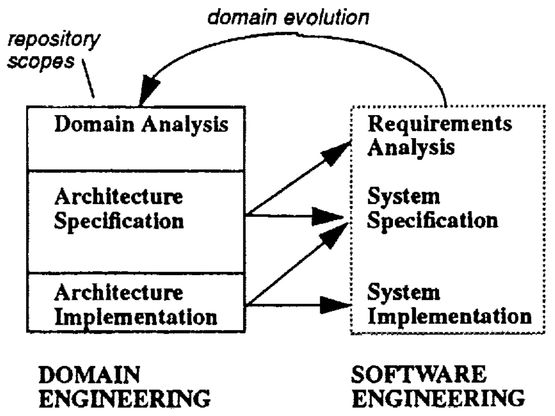

Figure 4-1 Repository Scopes and Process

in, different demand-side processes. For example, scoping the repository to include only the implementation components produced by domain engineering processes will result in a "conventional" parts library. Such a design decision can be motivated by various factors, including the possibility that the demand-side processes are still too chaotic to support more systematic reuse. Thus, in Figure 4-1 the "parts" library could support, at worst, ad hoc opportunistic reuse during system implementation, and, at best, could support a system specification that takes some advantage of existing reuseable components.

There are clearly potential advantages to extending the scope of the repository to address the entire spectrum of domain-engineering by-products, including domain analysis. In Figure 4-1 the primary benefit illustrated is the potential for closing the loop between domain engineering and software engineering through a feedback and domain-evolution path. Such a feedback loop can probably only be supported if the domain model is captured and represented in a reasonably formal manner.

5. Summary

We have described the problem of constructing an environment to support the construction of domain-specific reuse libraries in terms of integration. The integration problem involves integration of:

- roles in the reuse process
- domain analysis tools with each other
- domain analysis tools with a reuse process

We briefly outlined the use of a hybrid knowledge representation system, RLF, to act as an integrating agent to provide role-specific views into the domain model, and to support the use of an eclectic assortment of modeling techniques by tapping into a large, robust CASE market.

The CARDS program will focus, in the next year, on creating a *blueprint* for achieving reuse in the DoD. This blueprint will address technical as well as non-technical issues, and will provide guidelines for the use of a hybrid knowledge-representation/CASE tool architecture for developing domain-specific reuse libraries and using domain-specific software architectures and assets to create application systems.

The CARDS program will also be experimenting with domain-specific reuse environment and system composition techniques tailored to the command center subdomain of C^2 applications. The conceptual model for this composition is similar to that of hardware configuration [13] — a user *configures* a system of software components based upon a inferencer-directed dialogue designed to elicit system requirements.

References

[1] Anderson, C.M., McNicholl, D.G., "Common Ada Missile Packages (CAMP); Preliminary Technical Report, Vol. 1," in *STARS Workshop Proceedings*, April 1985, FO 8635-84-C-0280.

[2] Simos, M.A., "The Growing of an Organon: A Hybrid Knowledge-Based Technology and Methodology for Software Reuse," *Domain Analysis and Software Systems Modeling*, Prieto-Diaz, Arango, Eds., IEEE Computer Society Press, ISBN 0-8186-8996-X.

[3] *Integrated Project Support Environments: The Aspect Project*, A.W. Brown (Editor), Academic Press, 1990.

[4] Prieto-Diaz, R., "Domain Analysis for Reusability," *Domain Analysis and Software Systems Modeling*, Prieto-Diaz, Arango, Eds., IEEE Computer Society Press, ISBN 0-8186-8996-X.

[5] Brown, A.W., Feiler, P.H., *An Analysis Technique for Examining Integration in a Project Support Environment*, Technical Report CMU/SEI-92-TR-3, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, January 1992.

[6] Devanbu, P., Brachman, R.J., Selfridge, P., Ballard, B., "LaSSIE: A Knowledge-Based Software Information System," *Domain Analysis and Software Systems Modeling*, Prieto-Diaz, Arango, Eds., IEEE Computer Society Press, ISBN 0-8186-8996-X.

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

[8] Brown, A.W., Feiler, P.H., Wallnau, K.C., "Understanding Integration in a Software Development Environment," accepted for *Second IEEE International Conference on Systems Integration*, Irvine, CA, for May, 1992.

[9] Brown, A.W., Feiler, P.H., Wallnau, K.C., "Past and Future Models of CASE Integration," submitted to *Fifth International Workshop on Computer-Aided Software Engineering*, for July 1992.

[10] Solderitsch, J., Wallnau, K., Thalhamer, J., "Constructing Domain-Specific Ada Reuse Libraries," in *Proceedings of the 7th Annual National Conference on Ada Technology*, Atlantic City, NJ, 1989.

[11] Wallnau, K., Solderitsch, J., Thalhamer, J., et. al., "Construction of Knowledge-Based Components and Applications in Ada," *Special Issue of the IEEE Intelligent Systems Review*, Spring 1989.

[12] Matuszek, P., Clark, J., Sable, J., Corpron, D., Searls, D., "KSTAMP: A Knowledge-Based System for the Maintenance of Postal Equipment," United States Postal Service Advanced Technology Conference, May 1988.

[13] Searls, D., Norton, L., "Logic-Based Configuration with a Semantic Network," in *The Journal of Logic Programming*, Volume 8, 1990, pages 53-73.

[14] D'Ippolito, "The Context of Model-Based Software Engineering," in *Proceedings of the Workshop on Domain-Specific Software Architectures*, Hidden Valley, PA, DSSA Program Manager, DARPA/ISTO, 1400 Wilson Blvd., Arlington, VA 22209, July 1990.

[15] *A Domain Analysis Process, Interim Report*, Domain_Analysis-90001-N, Version 01.00.03, Software Productivity Consortium, 2214 Rock Hill Road, Herndon, VA, 22070, January 1990.

[16] Mayfield, J., Nicholas, C., *Using Semantic Networks to Enrich Hypertext Links*, Technical Report, Computer Science Dept., University of Maryland Baltimore County, January, 1992.

[17] *Z — An Introduction to Formal Methods*, Diller, A., John Wiley and Sons, ISBN 0-471-92489-X.