Incremental Adoption of Software Architecture Technology for Reuse in the DoD[*]

Kurt C. Wallnau

Paramax Systems Corp.
1401 Country Club Road
Fairmont, WV 26554
Tel: (304) 363-1857
Email: wallnau@cards.com

Abstract

Technologically sound and scaleable results of research into software architectures and their representations are beginning to emerge. However, the adoption of these results to support a global DoD reuse strategy is complicated by the business and organizational complexity of the DoD. Architecture-based reuse technology adoption must occur independently within numerous semi-autonomous product development centers in the DoD and within DoD contractor organizations. This adoption must support a gradual convergence to a common architecture representation technology from widely divergent starting points. This paper describes the use of a library of shared software architecture ontologies to support: local architecture representation technology autonomy; evolutionary adoption of common architecture representation technologies; and evolutionary development of representation-independent *domain* architectures from *product-line* architectures.

Keywords: Software reuse; software architectures; architecture description languages; shared ontologies.

Workshop Goals: Learn and share ideas about: design-level reuse; the impact of reusable design representation on DoD reuse and software procurement policies; and technology implications of organizational complexity on design reuse.

Working Groups: Reuse management, organization and economics; Domain analysis/engineering; Design-level, model-oriented reuse.

[*]The views expressed in this position paper are those of the author and do not necessarily reflect the views of Paramax, the CARDS program, or its government spon ors.

1 Background

The need for a disciplined, principled approach to software architectures is as crucial to the DoD Software Reuse Vision and Strategy[1] as the study of software architectures is broad and multi-disciplined. This need, when coupled with the continual advances in a newly-emerging field of software architectonics[1], requires programs such as CARDS to play an active role in accelerating technology transfer of DoD-sponsored architectonics research efforts (such as the ARPA/DSSA program) into practice.

The Central Archive for Reusable Defense Software (CARDS) is a DoD-sponsored program chartered to transition into practice the technology, processes and business practices necessary to institutionalize domain-specific, architecture-based, library-assisted software reuse [2]. The concepts and implementation of the CARDS Command Center Library reflects this accelerated technology transition. The explicit representation of software architectures as the principle organizing framework within the Command Center Library, and the tools which manipulate these architecture models[3], represents the link between the theory and practice of architectonics.

2 Position

CARDS has had some success in representing software architectures in a reuse library framework. However, to achieve our desired technology transfer objectives and to meet our customer's needs we must first understand the nature of the organizations we are dealing with. We recognize two distinct organizational dimensions which must be addressed if we are to achieve (and, possibly, sustain) domain-specific reuse in the DoD. Both dimensions (illustrated in Figure 1) must evolve in cooperation to achieve an evolution from distributed, autonomous product centers to a planned, managed and systematic domain-specific reuse capability of DoD magnitude.

To support this evolution two different kinds of software architecture representations are needed—one which captures detailed, technology- and representation-dependent designs (i.e., product-line architectures), and one which provides information-rich but technology-neutral correlations among product-line architectures (i.e., domain architectures). Current research is already addressing the former; CARDS is developing an ontology-based representation scheme to address the latter.

Section 2.1 elaborates Figure 1 and discusses the architecture representation requirements of product-line and domain organizations. Section 2.2 describes an ontological approach supporting the domain perspective. This represents new and unique work as product-line architecture representation technology is already being addressed by current research.

2.1 Architectures for two Organizational Perspectives: Into the Thickets

Architecture Representation for the Product-Line Organizations

The product-line organizations exist today, and support specific DoD missions within specific application areas (for example, multiple product centers for Command and Control applications exist in the DoD). Each such organization may operate, maintain, evolve and procure new systems as

[1]Architectonics: the study of architectures.

Figure 1: Product Line and Domain Organization Perspectives

part of the normal incremental evolution and accretion of DoD software systems. It is important to note that there are literally hundreds of product-line organiation within the DoD.

Organizational and technological diversity characterize the various product centers. Detailed expertise relative to specific products and customers has been developed within each product center. Lacking a shared domain model, the terminology, concepts and approaches used by these organizations varies tremendously. Thus, independent of development environment technology (which includes the use of software architecture representation technology) there is a high-degree of conceptual impedance mismatch among various product centers' perceptions of the application domain.

To complicate matters further there is also great variety in the DoD contractors which are used to support product-line maintenance and development, the roles they play in the product-line, and the technology and processes they use internally (even within the scope of DoD-STD-2167a) to accomplish their objectives. This both reflects and induces a large measure of technology diversity among the product centers. For example, development environment tooling may be strongly influenced by the preferences and use of commercially-available and proprietary software development tools and processes by contractors.

The need to support product-center diversity in technology and procurement is reflected in recent descriptions of the role of software architectures in the DoD software procurement process[4][2]. To support this diversity, any one of a number of commercial (and research)-off-the-shelf architecture description languages (ADLs) and toolsets might be used—the author certainly claims no special expertise in making recommendations about which ADL to use.

[2]Actually, Saunders does not explicitly tie the concepts in his paper to product centers rather than domain centers. However, applying the concepts discussed in his paper in the current DoD organizational context—one which does not have a strong domain management function—will result in product-line diversity as indicated in this paper.

Architecture Representation for the Domain Organization

Given the diversity of product-line specific terminology, application concepts, development and target-environment capabilities and approaches to software architectures, how will it be possible to evolve a coherent domain architecture?

The domain organization, which does not currently exist (although plans for developing domain centers do exist) will not have the luxury of defining, from "scratch," all-encompassing domain architectures—i.e., generalizations and unifications of the multiple product-line architectures. Instead, domain-specific architectures will have to evolve to accommodate the various product-line architectures because: all of the DoD's software can not be re-written at once; new systems (based on domain architectures) will need to interoperate with existing systems (based on product-line architectures), and; applications will continue to be procured on an application-by-application basis for the forseeable future (irrespective of the acclaimed role of domain analysis in achieving domain-specific reuse).

One approach is to develop detailed, domain-specific technical reference models. Reference models, such as described in [5], can provide a wealth of information about interface standards, conventions, assumptions, and usage models for applications; they can be used to support the procurement of systems (or parts of systems) which adhere to some explicit implementation characteristics. In fact, the CARDS-defined component qualification process makes use of a technical reference model (of command center functional components) to evaluate the "form, fit and function" of components relative to domain-specific conventions and requirements.

However, technical reference models, though useful, are no substitute for software architectures. Architectures describe specific characteristics of systems[3], including: component interconnections, data and control flow, throughput, timing and scheduling, fault-tolerance and security (to name a very few). Which characteristics are described, the form in which they are described, and the level of detail associated with their descriptions will vary across application areas (embedded avionics will differ from information management), product centers and systems within product centers.

There are several possible approaches to managing product-line developed architectures within a domain organization. First, the domain organization could acquire the union of ADL processors and maintain the product line architectures independently—this is a useless option. Second, the organization could invent or adopt a general, all-encompassing "doomsday" ADL, one which could adequately express all of the characteristics described by all of the ADLs in use across all of the DoD domains—don't hold your breath. Third, should no viable approach to managing architecture-representation diversity emerge, the domain management organization could focus primarily on technical reference models; however, this would represent a setback for domain specific reuse within the DoD.

None of the above options are palatable. The next section describes a more practical and flexible approach.

2.2 An Ontological Approach to ADL Adoption: Out of the Woods

Rather than wait for the doomsday ADL, a more flexible approach is to describe, in a formal model, specific characteristics of software architectures and then map product-line software architectures

[3]It should be noted that some reference models masquerade as software architectures, while some software architectures masquerade as reference models: both disguises can introduce serious problems.

Figure 2: Ontology Fragment in Term Classifier Representation

into this formal model. This formal model would describe the elements and relationships of the concepts which underly the representation of software architectures—not the details of specific system architectures.

Describing such a formal model as a *meta*-model for software architectures would be accurate but not sufficiently descriptive. This *meta*-model would model a theory of software architectures, facilitate sharing and exchange (i.e., reuse) of design knowledge (i.e., software architectures and designs) across different technology bases (i.e., product center design tools) and different product lines. Such a *meta*-model would in fact be an ontology[4] of (characteristics of) software architectures[6].

A reuse library which defined, and was organized around, one or more ontologies[5] could provide numerous encyclopedic reuse services:

- model and relate idiomatic architecture patterns such as those identified by Shaw and Garlan[7], and model (in the same formalism) actual systems as refinements/specializations of these idioms to support analysis of single systems and comparison of multiple systems.
- model the abstractions and semantics of specific architecture/design representation systems so that tool-specific semantics can, at least in a limited form, be normalized with respect to explicitly-defined, shared principles.
- relate, through formal term classification, various idiosyncratic vocabularies used to describe common domain "concepts."

[4]Ontology: a particular theory about the nature of being or the kinds of existents. (Webster's Ninth New Collegiate Dictionary. More thorough definitions may be found in AI literature.)

[5]For pragmatic issues related to the design of *good* ontologies it may be more convenient to define several small intersecting ontologies than to define one large ontology.

Such a reuse library might appear to be overly-visionary, but in fact such libraries are within reach. The reuse library framework employed by the CARDS program— RLF[8]—makes use of a term classification knowledge-representation formalism derived from KL-ONE[9]. As such, the system provides a limited but useful (certainly for the near-term) capability for describing architecture ontologies. A trivial fragment of an ontology for software architectures[6] which might appear in a CARDS library is provided in Figure 2. The reader should be able to see how some of the architectural idioms described by Shaw, such as pipes and filters, can be represented as specializations of components, connections and interactions.

2.3 Summary: What's Real and What's Next

The CARDS program has developed an architecture-based reuse library for Command Centers. The library makes use of a formally-encoded generic command center architecture (GCCA), encoded in a knowledge-representation scheme derived from KL-ONE. The encoded software architecture supports two automated reuse assistants. One, the system composer, uses the encoded GCCA to guide library users through an interactive architecture refinement process to compose portions of the GCCA (related to message processing); over twenty compositions are possible, with several of these resulting in executable load images. A second tool, the component qualifier, uses the encoded GCCA to guide library administrators in an architecture-tailored component qualification[7] process.

The CARDS program does not *currently* employ an architecture ontology. That is, we encoded the GCCA through the use of informal modeling conventions. As a result, although we have developed impressive demonstration capabilities for model-based reuse libraries, these capabilities are somewhat sensitive to the particular form of the architecture as represented in the library model. CARDS is undertaking a kind of domain analysis (using Simos' Organizational Domain Modeling approach[10]) on the field of software architectures. Our intention is to develop a "starter ontology" as a result of this work that will allow us to generalize the composition and qualification tools, make more systematic the development of CARDS reuse libraries by CARDS franchisees, and provide a basis for evolving the domain perspective illustrated in Figure 1. As the space of possible of design notations is immense[11] the scope of the CARDS efforts will be defined relative to a limited number of reuse services provided by CARDS libraries; the more general discipline of "ontological architectonics" will need to be elaborated elsewhere.

3 Comparison

3.1 ARPA Domain Specific Software Architectures (DSSA)

A number of concepts being explored by the DSSA program are consistent with (and in some cases provided the basis for) CARDS concepts. The notion of a domain-specific reference architecture and tool support for the refinement of this architecture to support specific application needs is a direct analogue to the CARDS GCCA encoding and system composer. The notion of distinguishing between domain engineering and application engineering functions is also shared between DSSA and CARDS. The system composition capabilities[12] and type expression formalisms[13] of the

[6]The example illustrates only a fragment of an ontology; it is not clear that this is even an "interesting" ontology.

[7]We use the term "qualification" to avoid confusion with more widely known (but not better understood!) term, "certification."

DSSA/ADAGE system have many concepts related to the CARDS composition and modeling formalism; however, more detailed description defies brief summary.

There are some interesting differences as well. First, DSSA is chartered to undertake domain analysis and architecture definition efforts in several domains. CARDS, on the other hand, is not chartered to undertake domain analysis efforts, but must instead seek out partnerships with product centers in the DoD (a.k.a. franchises) and encourage and support them in domain analysis, architecture recovery, and/or architecture definition. Second, DSSA is in the process of defining a consensus ADL. CARDS, as already mentioned, is more interested in capturing and modeling the principles which underlie whatever ADL(s) DSSA converges upon. Last, DSSA is addressing a broad array of system and software development issues, while the CARDS technology effort is more narrowly focused on supporting architecture-based, library-assisted software reuse.

3.2 NASA/KAPTUR

The notion of case-based reuse supported by KAPTUR represents a similar view of incrementally evolving a domain knowledge base from a series of specific application instances, as illustrated in Figure 1. The KAPTUR system also provides a number of distinct architectural views of systems—this is a capability which is currently under development for CARDS libraries (in fact, the integration of KAPTUR or some of its subsystems with RLF is being investigated by the STARS program to provide the Army demonstration project technology support for ODM[10]).

The fundamental differences between KAPTUR and the CARDS effort—and these differences may narrow as RLF/KAPTUR integration is explored and implemented, concerns the use of the library toolset to support interactive architecture refinement and system composition. In KAPTUR the purpose of maintaining architecture cases is to support the cognitive aspects of selecting among design alternatives; in CARDS one purpose of encoding the GCCA was to support library-assisted (i.e., semi-automated) composition of systems.

3.3 ARPA Knowledge Sharing Initiative (KSI)

The Knowledge Sharing Initiative is exploring the use of ontologies to specify content-specific specifications shared among autonomous reasoning agents. The goal is to "enable libraries of reusable components and knowledge-based services that can be invoked over networks[6]." This effort is clearly relevant to the development of software architecture ontologies in support of the correlation and exchange of design information developed by semi-autonomous product centers (again, refer to Figure 1).

One significant difference in approach is that KSI ontology specifications represent multi-lateral knowledge sharing agreements among reasoning agents; in CARDS, the ontology is uni-lateral and is intended (at least in the near-term) to support systematic description efforts to support library assisted reuse. That is, the ontology is not intended to support independently-developed library assistance tooling.

References

[1] DoD, "DoD Software Reuse Vision and Strategy," Tech. Rep. DISA 1222-04-210/40, Depart-

ment of Defense, 1992.

[2] K. Wallnau, "Cards overview," *CROSSTalk, The Journal of Defense Software Engineering*, no. 32, 1992.

[3] K. Wallnau, "Towards and Extended View of Reuse Libraries," in *Proceedings of the 5th International Workshop on Software Reuse*, 1992.

[4] T. Saunders, B. Horowitz, and M. Mleziva, "A New Process for Acquiring Software Architecture," Tech. Rep. M 92B0000126, MITRE, 1992.

[5] NIST, "Reference Model for Frameworks of Software Engineering Environments," Tech. Rep. NIST Special Publication 500-201, ECMA TR/55, 2nd Edition, ECMA, 1991.

[6] T. Gruber, "Toward principles for the design of ontologies used in knowledge sharing," Tech. Rep. Unpublished Technical Report, Stanford Knowledge Systems Laboratory, 1993.

[7] D. G. M. Shaw, "An Introduction to Software Architecture," *Advances in Software Engineering and Knowledge Engineering*, vol. 1, 1993.

[8] K. W. J.J. Solderitsch, J.Thalhamer, "Construction of Domain-Specific Reuse Libraries," in *Proceedings of Seventh Annual National Conference on Ada Technology*, 1988.

[9] J. S. R.J. Brachman, "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science*, vol. 9, no. 2, pp. 171–216, 1985.

[10] M. Simos, "An Introduction to Organizational Domain Modelling: A Domain Analysis process Model," in *Proceedings of the International Workshop on Software Reuse, Pisa, Italy*, 1993.

[11] D. Webster, "Mapping the Design Information Representation Terrain," *IEEE Computer*, December 1986.

[12] D. Batory, "A Process and Retrospection on Creating a Domain Model for Avionic Software," Tech. Rep. ADAGE-UT-93-04, Department of Computer Science, University of Texas, Austin, 1993.

[13] D. Batory, "le: A Type Expression Language," Tech. Rep. ADAGE-UT-93-02, Department of Computer Science, University of Texas, Austin, 1993.

4 Biography

Kurt C. Wallnau is a research scientist for Paramax Systems Corporation. He currently is system architect for the CARDS program, an Air Force program in domain-specific reuse. Prior to the CARDS program, Mr. Wallnau was a member of the technical staff at the Software Engineering Institute, where he performed research in the area of software development environments, specifically in the area of environment integration. Before that, Mr. Wallnau was chief programmer of various STARS tasks, including interface standards and user interface systems. Mr. Wallnau was one of the principal designers and developers of the RLF, a knowledge-based reuse library framework, also developed by the STARS program.