The Software Architecture Renaissance

Paul Kogut
Air Force/NASA CARDS Program (Unisys) and SEI Resident Affiliate

Paul Clements
Software Engineering Institute

Introduction

There is a revival of interest in the software product after several years of intense focus by the software engineering community on the software process. Similarly, within the software design community, there is a revival of interest in higher levels of design (e.g. software architecture) after several years of intense focus on object-oriented (OO) design methodologies (e.g. identifying objects, classes and inheritance relations). The increasing importance of software in systems is also driving the software architecture renaissance. Problems with software architecture have contributed to the difficulties of large projects like the FAA's Advanced Automation System [1]. Because architectural decisions are usually made early in the life-cycle, they are the hardest to change and hence the most critical and far-reaching. Without a good system and software architecture it is difficult to achieve satisfaction of the original performance and behavioral requirements and it is probably impossible to accommodate major design changes. Software architecture also provides the critical context for high leverage reuse of design and components. The importance of architectural context for reuse is derived from observing the common practices in mature engineering fields such as civil engineering and chemical engineering. Within software engineering, common practices and solutions within particular application areas can be reused to provide domain-wide leverage of architectural-level decisions. This article provides a brief overview of some of the important architecture related efforts.

What is software architecture? In this article a software architecture is loosely defined as the organizational structure of a software system including components, connectors, constraints, and rationale [2][3]. Dewayne Perry and David Garlan, the editors of a special issue of IEEE Transactions on Software Engineering (expected fall 94), are working on a more formal and standardized definition.

Software Architecture Science

Currently there is a proliferation of ongoing activities in the science of software architecture. These activities are considered science because the main focus is on the study of the characteristics of existing architectures. At Carnegie Mellon University researchers are identifying and describing architecture styles found in existing software systems [3]. For example, some systems have a pipe and filter style (e.g. Unix like) while others are object oriented, data centered (e.g. transactional database systems) or event systems (e.g. objects broadcast events through a request broker). These styles determine the macro architecture for a whole system. Many systems have been found to have hybrid styles especially when large components are examined in more detail.

In the object-oriented community there is a great deal of interest in OO design patterns found

in existing systems [4]. These OO patterns are micro architectures that facilitate the synthesis and understanding of macro architectures. The OO patterns consist of a cluster of classes that interact to perform a certain common generic function. Researchers are currently identifying and cataloging these OO patterns to help promote design reuse.

The analysis of the quality attributes of architectures is an important area where much works needs to be done. One example of architectural analysis was carried out in the domain of user interface tools [5]. The researchers chose the quality attribute modifiability for their analysis. They then selected an "ideal" reference architecture and a set of benchmarks (e.g. change a component such as a toolkit or a menu) as a basis for comparison. Sample architectures were translated to a standard representation and then evaluated against the reference architecture and benchmarks.

Software Architecture Engineering

There is also a proliferation of ongoing activities in the engineering of software architectures. These activities are considered engineering because the main focus is on building architectures that support the cost effective development of high quality application systems. Many of these engineering efforts have been focused on developing domain-specific software architectures (DSSAs) which are architectures for a family of application systems in a domain. DSSAs support reuse of both design knowledge and components through the use of various models (e.g. structural and behavioral). The development and application of DSSAs fit under the broad umbrella of model-based software engineering which also includes domain analysis and other reuse related activities. The following brief case study discussions cover just a few example DSSAs including their basic approach.

- The Tektronix oscilloscope architecture was an early example of a domain-specific software architecture for a product line [6]. Tektronix experimented with architecture styles such as layered and object-oriented but they found that the pipe and filter style was best for their domain. The filters are parameterized so that they can be instantiated for different models of oscilloscope. They also found that the connections (pipes) were very important for an architecture that supports reuse. They used formal methods to define 6 different pipe specifications that could be chosen.
- The Software Engineering Institute (SEI) has developed several DSSAs for DOD domains. The SEI created several specialized object-oriented architecture styles to support these DSSAs (e.g. object connection update, structured modeling, and object connection architecture). These specialized styles establish certain categories and dataflow/control flow rules for classes/objects. A DSSA based on these styles also includes Ada templates for components. This structure makes it easier to understand and modify a DSSA as well as generate instances of systems. Early experience with building these DSSAs was in the domain of Air Force flight simulators [7]. Extensive experience and guidelines in structured modeling for flight simulators has been consolidated into a guidebook by the Air Force and SEI [8]. Recently the SEI has mapped a domain model and the object connection architecture into a generic design and a set of components for the Army movement control domain [9].

- The Air Force Command Center Processing and Display System - Replacement (CCPDS-R) was built using the Universal Network Architecture Services (UNAS) tool [10]. UNAS provides middleware to connect communicating Ada processes, tools to monitor the connection traffic, and a CASE tool to architect and generate a system. The UNAS tool provides high leverage architecture based reuse for a communicating Ada processes architecture style.
- The Air Force sponsored Portable, Reusable, Integrated Software Modules (PRISM) Program developed an architecture with the goal of maximizing the reuse of existing commercial and government off the shelf components in command centers [11]. This goal led to an object-oriented event system architecture style. The emphasis of the architecture was on interface standards (e.g. SQL). Wrappers were written for components in order to integrate them.
- The Sematech Computer Integrated Manufacturing Framework is a good new example of a DSSA for a complex domain [12]. Sematech is a consortium of semiconductor manufacturers partially sponsored by ARPA. Sematech adopted the Common Object Request Broker Architecture (CORBA) as its infrastructure and defined standard domain specific objects. CORBA is an evolving standard for object-oriented event system architectures. CORBA defines an interface definition language, an object request broker and object adaptors. Vendor supplied objects must comply to the framework. Sematech defined a compliance testing methodology to ensure this compliance.

There are many issues raised by the thrust to develop DSSAs and analyze architectures:

- What Software Architecture Representation Language (SARL) should you choose? SARLs are critical for capturing the knowledge associated with a DSSA. Commonly used structured and object-oriented design notations (e.g. those found in CASE tools) do not fully support DSSAs. SARLs must help various stakeholders understand and apply DSSA knowledge. SARLs should have associated tools which facilitate the composition, generation and analysis of application systems derived from a DSSA. There are a variety of SARLs emerging from the ARPA sponsored DSSA program and elsewhere. They vary widely in terms of what architecture styles they support and what forms of analyses they permit.
- How do DSSAs fit into your software development process?
- How are DSSAs developed from domain models and/or reengineered from existing system designs?
- How do you build the business case for a DSSA? How do you calculate your revenues from assets (knowledge, components, tools)?
- How do you estimate application development costs in a DSSA based process?

The SEI, CARDS, University of Southern CA, and others are beginning to address these issues.

Pulling It All Together

The SEI and the Air Force/NASA sponsored CARDS (Comprehensive Approach to Reusable Defense Software) program are working together in a Software Architecture Technology Ini-

tiative (SATI) to help transition the most promising architecture-based approaches to practice. The focus is on DSSAs that support model-based software engineering. The goal is to establish a repository of knowledge and expertise about the field of software architecture, make it available to interested parties, and disseminate best practices for architecting a DSSA. Guidance for selecting and using SARLs and their associated tools is also a high priority.

The ongoing SATI activities include a variety of activities to help consolidate architecture concepts and technology:

- develop an on-line annotated bibliography for software architecture.
- perform a domain analysis of SARLs. This will help characterize the features of SARLs to support selection from existing SARLs and development of new SARLs.
- document DSSA case studies to illustrate best practices of architecture selection, evaluation, and usage.
- establish partnerships and collaborative agreements with organizations that are developing a DSSA.

References

1. Gary Stix, “Aging Airways,” Scientific American, May 1994, pp. 96-104.
2. Dewayne E. Perry, Wolf, A., “Foundations for the Study of Software Architecture” ACM SIGSOFT Software Engineering Notes, Vol. 17, No. 4, October 1992, pp. 40-52.
3. Garlan, Shaw “An Introduction to Software Architecture” in Advances in Software Eng. and Knowledge Eng. editors: Ambriola and Vincenzo, World Scientific Publishing Co. 1993
4. Dutton, Sims “Patterns in OO Design and Code Could Improve Reuse” IEEE Software May 1994 pg. 101.
5. Kazman, Bass, Abowd, Webb “SAAM: A Method for Analyzing the properties of Software Architecture 16th ICSE May 1994
6. Garlan, Delisle “Formal Specifications as Reusable Frameworks” In VDM 91 Springer Verlag LNCS 551
7. Lee, Rissman, D'Ippolito, Plinta, Van Scoy “An OOD Paradigm for Flight Simulators” CMU/SEI-88-TR-30 1988
8. “Structural Modeling Guidebook”, ASC/YT Training Systems Program Office, Wright-Patterson Air force Base, April 1994
9. Peterson, Stanley “Mapping a Domain Model and Architecture to a Generic Design” CMU/SEI-94-TR-8 May 1994
10. Royce and Royce “Software Architecture: Integrating Process and Technology” TRW-TS-91-04 Dec. 1991
11. “Generic Command Center Architecture Report for the Portable, Reusable, Integrated Software Modules Program” CDRL Sequence No. A001, Raytheon Company and Hughes Technical Services Company, May 21, 1993.

12. Sematech Collaborative Manufacturing System CIM Application Framework Spec. March 1994