Design Records: A Way to Organize Domain Knowledge

Will Tracz – Steve Shafer – Lou Coglianese

IBM Federal Systems Company
MD 0210, Owego, NY 13827
Tel: (607) 751-2169, fax: (607) 751-6025
Email: tracz@vnet.ibm.com

Abstract

This document describes the **Design Records** being developed as part of the Domain-Specific Software Architecture Avionics Domain Application Generation Environment (DSSA-ADAGE) Project[1]. A design record aids in the generation of new avionics applications as well as the maintenance of existing systems built using ADAGE. The purpose of a design record is to serve as a vehicle for software understanding by functioning as a collection point for *domain-specific* knowledge about the components that make up a Domain-Specific Software Architecture (DSSA),

There are **three** kinds of design records used by ADAGE to describe the avionics software architecture and components [1, 2].

1. **Domain Model Design Record** defines a collection of realms[2],
2. **Realm Design Record** defines the interface for a collection of components, and
3. **Component Design Record** represents an (alternative) implementation or design choice.

Keywords: Domain Analysis, Knowledge Representation, Software Reuse.

Workshop Goals: Facilitate the technical interchange of ideas and experience related to the development and use of software components.

Working Groups: Knowledge Representation, Design for Reuse.

[1]This effort is sponsored by the US Department of Defense Advanced Research Projects Agency in cooperation with the US Air Force Wright Laboratory Avionics Directorate under contract F33615-91-C-1788.

[2]See Batory's [3, 4] for a detailed treatment of creating layered designs using realms and type expressions.

1 Background

IBM FSC as part of ARPA Domain-Specific Software Architecture Program has been actively pursueing the developement of tools and processes to support the generation and application of software architectures. One of the major focusses of this effort has been the creation of a "design record". The purpose of a design record is to serve as a vehicle for software understanding by functioning as a collection point for *domain-specific* knowledge about the components that make up a Domain-Specific Software Architecture (DSSA). ADAGE design records play a central role in 1) describing the avionics domain-specific software architecture and 2) integrating the tools that comprise the ADAGE environment.

Two points merit distinction in their bearing on the design record:

1. the contents of the ADAGE **Component Design Record** (i.e., its "data elements") are patterned after those initially proposed by Bill Scherlis in [5].
2. While a design record serves as a collection point for:
   - *domain-specific* knowledge about components or design alternatives and
   - *implementation-specific* knowledge about alternate implementations,

   it does **not** provide information about their application, instantiation, or configuration. This type of information, though similar in nature, is part of an application's **design history**.

2 Position

The goal of a design record is to adequately describe a domain-specific software architecture and its software software components such that design decisions and component selections can be accomplished without looking at implementations.

Asset capture and re-capture is supported in ADAGE by the design record, hypermedia browsing capability, and the domain engineering process [6]. The design record provides a "common data structure for system documentation and libraries [5]".

The **basic** design record data elements, as proposed by Scherlis and arranged according to phases in the software life cycle, include:

1. **name/type**,
2. **description**,
3. **requirement specification fragments**,
4. **design structure**,
5. **design rationale**,
6. **interface and architecture specifications and dependencies**,
7. **PDL texts**,
8. **code**,
9. **configuration and version data**, and
10. **test cases**.

    In addition to the "primary" lifecycle elements listed above, the following "secondary" elements aide in the (re-)use of the components by capturing additional information:

11. **metric data**,
12. **access rights**,
13. **search points**,
14. **catalog information**,
15. **library and DSSA links**, and
16. **hypertext paths**.

    For the avionics domain, the ADAGE design records contain the **basic** data items listed above (with some domain-specific clarifications) in addition to some DSSA-ADAGE specific items including:
17. **models**
18. **constrants**, and
19. **data quality**.

The reader should note that because the design record is a dynamic entity in that its contents grow as a component goes through the various stages in the software development life cycle, all component design records do not contain the same amounts of information. In addition, certain design record elements may "inherit" values for other records they may be associated with (e.g., a realm component design record inherits the constraints of the realm it is a member of).

2.1 Types of Design Records

The goal of design records are to organize information associated with a domain-specific software architecture. There are **three** kinds of design records used by ADAGE to describe the avionics domain software architecture and components [1, 2]:

1. **Domain Model Design Record** defines a collection of realms,
2. **Realm Design Record** defines the interface for a collection of components, and
3. **Component Design Record** represents an (alternative) implementation or design choice.

The "component interface" found in a **Realm Design Record** includes not only the entry points, type definitions and data formats (e.g., Ada package specification), but a description of its functionality, side effects, performance expectations, degree and kind of assurance of consistency between specification and implementation (reliability), and appropriate test cases.

2.1.1 Domain Model Design Record

The **Domain Model Design Record** defines a collection of realms. These show up as a list in the **library and DSSA links** element of the design record. The **Domain Model Design Record** has, as a minimum: a name, a description, and an interface/architecture specification, as well as a collection of administrative information (e.g., version number, catalog information, access rights, etc.).

2.1.2 Realm Design Record

The **Realm Design Record** defines the interface for a collection of components. These show up as a list in the **library and DSSA links** element of the design record. A **Realm Design Record** defines either *design decision* or *options* (e.g., a list of sensors to choose from), or **implementation alternatives** (e.g., a list of stack implementations).

2.1.3 Component Design Record

The **Component Design Record** may be used to describe three kinds of components:

1. *Domain Specific* – map into problem space,
2. *Implementation Specific* – map into solution space, and
3. *Domain Independent* – general components that may be used in other domains.

Component Design Records normally have some sort of execution capability. This can take the form of an Ada package specification and body, an executable SEDL specification, or a parameterized LILEANNA make statement indicating how such a component could be constructed.

2.2 Design Record Creation

As part of the domain engineering process described in [6], a **Component Design Record** is created for each component in DSSA. As the DSSA reference architecture is configured and extended to meet the requirements of a new application using the Architecture-Based Development Process [7], additional information is recorded as part of a **Design History Record** (i.e., configuration data and design decision rationale). In addition, new design records are generated to account for extensions to the architecture, addition of components with unprecedented functionality, or new implementations of existing components.

2.3 Design Record Evolution

Each generic component in the avionics DSSA has a design record with pointers to instances of the design record (via the DSSA link data element). The design history records for instances of components "inherit" the elements of the generic while specializing those that are application specific. In particular:

- **configuration and version data** indicate parametric values used to instantiate the component and
- **design rationale** describe the reasons for their selection.

The resulting chain or information web is what has been called the "**Design History**" in ADAGE [8].

3 Design Record Comparisons

This section compares the ADAGE design record data elements to the reusable software component information proposed by STARS (Software Technology for Adaptable, Reliable Systems), ASSET (Asset Source for Software Engineering Technology), RIG (Reuse Interoperability Group), and IBM's internal reuse repository.

3.1 STARS Comparison

According to the **STARS Reuse Concept of Operations, Volume I** [9], the following asset information "may be required for asset acceptance:"

- abstract,
- author/ownership information,
- author certificate of originality,
- copyrights/patents,
- distribution rights,
- distribution restrictions,
- liability statements for use/misuse,
- maintenance agreements,
- environmental dependencies, and
- dependencies on other assets.

The ADAGE design record completely supports the inclusion of this information.

3.2 ASSET Comparison

According to the **ASSET Submittal Guidelines** [10], the following documentation is recommended for asset acceptance into the ASSET Repository:

- "an abstract,
- a user's guide or instructions on how to use,
- a list of files making up the asset (preferably in compilation order),
- installation/implementation instructions,
- sample input/output,
- design and/or requirements documents,
- test programs, procedures an/or results,
- description of the environment under which the asset was developed/tested,
- known limitations of the software,
- a list of tools needed to interpret or used the asset,
- warranties or disclaimers,
- statement of distribution rights/licenses, and
- list of special formats of files (e.g., Postscript, InterLeaf, SGML)."

The ADAGE design record completely supports the inclusion of this information.

3.3 RIG Comparison

The Reuse Interoperability Group (RIG), a defacto standards organization, whose goal is to facilitate cross reuse repository information exchange, in **A Basic Interoperability Data Model for Reuse Libraries** [11] identified the following subset of the "Uniform Data Model", which defines the minimal set of information that reuse libraries should be able to exchange about assets in order to interoperate:

- abstract,
- address of contributor/owner,
- cost/fee to use*,
- date of information,
- distribution statement,
- domain,
- element type (e.g., code, test suite, make file),
- email address where asset resides*,
- fax # where asset resides*,
- identification number*,
- keywords,
- language,
- Media asset is obtainable in*,
- name,
- restrictions (e.g., legal),
- security/classification,
- target environment,
- telephone # where asset resides*,
- unique identifier*,
- version, and
- version date.

This Basis Interoperability Data Model (BIDM) is derived from the Common Data Model defined in the *Asset Library Open Architecture Framework* (ALOAF) [12] and the *STARS Repository Guidelines and Standards* [13].

With the exception of those items indicated with an "*", the ADAGE design record supports the inclusion of the RIG local "attributes" for component classes.

3.4 IBM Corporate Reuse Environment (CRE) Comparison

IBM's internal use only software reuse tool, CRE (Corporate Reuse Environment) is a sophisticated repository of tools for managing libraries of reusable software components. The "Software Element Types" required for entry into the repository include:

- abstract,
- change history,
- dependencies,
- design,
- interfaces,
- legal,
- load module,
- metrics,
- miscellaneous project information,
- object module,
- performance analysis,

- product documentation,
- references to other documents or components,
- requirements for use and adaptation,
- restrictions and side-effects,
- reuse,
- sample use,
- source code,
- specification (formal),
- test materials,
- usage information, and
- variations:

References

[1] S. Shafer and L. Coglianese, "Avionics Type Expressions: Realm Definitions and an Example System," Tech. Rep. ADAGE-IBM-93-07, IBM Federal Systems Company, April 1993. Preliminary Version.

[2] D. Batory, "A Domain Model for Avionics Software," Tech. Rep. ADAGE-UT-93-03, University of Texas at Austin, May 1993.

[3] D. Batory and et al., "GENESIS: An Extensible Database Management System," *IEEE Transactions on Software Engineering*, March 1986.

[4] D. Batory and S. O'Malley, "The Design and Implementation of Hierarchical Software Systems," Tech. Rep. TR-91-22, University of Texas, 1991.

[5] W. Scherlis, "DARPA Software Technology Plan," in *Proceedings of ISTO Software Technology Community Meeting*, June 27-29 1990.

[6] W. Tracz and L. Coglianese, "DSSA Engineering Process Guidelines," Tech. Rep. ADAGE-IBM-92-02A, IBM Federal Systems Company, December 1992.

[7] L. Coglianese and W. Tracz, "Architecture-Based Development Process Guidelines for Avionics Software," Tech. Rep. ADAGE-IBM-92-02, IBM Federal Systems Company, December 1992.

[8] W. Tracz and L. Coglianese, "DSSA-ADAGE Operational Scenarios and System Vision," Tech. Rep. ADAGE-IBM-92-01B, IBM Federal Systems Company, April 1992.

[9] "STARS Reuse Concepts Volume I – Conceptual Framework for Reuse Processes," Tech. Rep. STARS-TC-04040/001/00, The Boeing Company, IBM FSC, and Paramax Systems Corp., February 1992.

[10] "ASSET Submittal Guidelines Version 1.0," Tech. Rep. SAIC-92/7625&00, Asset Source for Software Engineering Technology and IBM FSC, December 1992.

[11] R. T. C. . (TC2), "A Basic Interoperability Data Model for Reuse Libraries," Tech. Rep. SDS-0001 v.2, Reuse Library Interoperability Group (RIG), February 1993.

[12] "Asset Library Open Architecture Framework," Tech. Rep. Contract No. F19628-88-D0031, Publication No. GR-07670-1317, Boeing Company, IBM FSC, and Unisys Defense Systems, Inc., April 1992.

[13] "Repository Guidelines & Standards for the STARS Contract," Tech. Rep. Contract No. F19628-88-D-0032, CDRL No. 0460, IBM System Integration Division, March 1989.

4 Biography

Will Tracz is a senior programmer at the Owego Laboratory of the IBM Federal Systems Company where he is currently a DARPA PI (Principal Investigator) on the DSSA-ADAGE Project. He is a member of IBM Corporate Reuse Council and IBM FSC Reuse Steering Committee as well as editor of the IBM Corporate Programming Reuse Newsletter and column editor for IEEE Computer. His book, *Software Reuse: Emerging Technology*, published (1988) by IEEE Computer Society Press, paints a broad picture of the technical, economic, pedagogical and social issues facing the transfer of software reuse technology into the workplace.