DARE-Web: Domain Analysis in a Web Environment

Omar Alonso
Oracle Corp., oalonso@us.oracle.com

William Frakes
Software Engineering Guild, wbfrakes@yahoo.com

DARE-Web: Domain Analysis in a Web Environment

Omar Alonso, Oracle Corp., Redwood Shores, CA 94065
oalonso@us.oracle.com
William Frakes, Software Engineering Guild, Sterling, VA 20165
wbfrakes@yahoo.com

Abstract

DARE-Web is a Web-based environment that supports domain analysis - the activity of identifying and documenting the commonalities and variabilities in related software systems. DARE-Web supports the capture of domain information from experts, documents, and code in a domain.

Keywords: domain analysis, domain engineering, digital libraries, Web-based applications.

Introduction

The process of creating an environment to support systematic reuse is called domain engineering. Domain engineering has two phases: domain analysis and domain implementation. We define domain as an application area or business area (also called vertical market).

Domain analysis is the activity of identifying, storing, and documenting the commonalities and variabilities in related software systems in a domain. These systems share common design decisions. Domain implementation is the use of knowledge acquired in domain analysis to develop reusable assets for the domain and the creation of a production process for systematically reusing assets to generate new products (Frakes et al., 1998).

DARE

DARE (Domain Analysis and Reuse Environment) is a CASE tool that supports domain analysis—the activity of identifying and documenting the commonalities and variabilities in related software systems (Frakes et al., 1997). DARE supports the capture of domain information from experts, documents, and code in a domain. Captured domain information is stored in a domain book (Arango et al., 1993) that will typically contain a generic architecture for the domain and domain-specific reusable components.

The outputs of DARE include:

1. a domain specific vocabulary.
2. a faceted classification of systems in the domain.
3. domain templates which link facets with explanatory text.
4. a domain feature table.
5. a generic architecture.
6. reusable components.

DARE has been developed in a series of four research prototypes, the latest of which is DARE-Web. In this section we review these prototypes and use them to illustrate the major features and functions of DARE.

The First Prototype

The first DARE prototype was developed in C on a UNIX workstation running X-Windows and Motif during 1994. This version of DARE was used to explore the domain book metaphor, and to investigate automatic and semi-automatic word and phrase extraction and clustering. The concept of a domain book for structuring and representing the outputs of the domain analysis process is important because it solves two hard problems in domain analysis:

- What should the output of domain analysis be?
- What are the criteria for completion, that is, how does the domain analyst know when a domain analysis is done?

The book metaphor helps answer both of these questions: the output of domain analysis is a domain book, and the process is complete when all sections of the domain book are complete.

The book metaphor also brings structure to the whole domain analysis process. Users write a domain book from front to back and the sections of the book are ordered according to the DARE domain analysis method. Thus each part of the domain book becomes a milestone and review point in the domain analysis process. Part 1 of a domain book, Domain Sources, is the input part of DARE where the material used in domain analysis is entered. Parts 2 and 3, Vocabulary Analysis and Architecture Analysis, structure the outputs of the domain analysis process. These parts also correspond to the two main steps of the DARE domain analysis method:

- bottom-up analysis—validate the generic architecture and generic features through analysis of domain documents (text) and source code;
- top-down analysis—postulate a generic architecture and generic features based on expert knowledge and experience.

The domain book structure neatly organizes the outputs of these two activities. The remainder of the domain book, the Glossary, Bibliography, Index, and Appendix, is end-matter that provides reference material and a search mechanism for locating items in the domain book.

The Second Prototype

We found that the C, UNIX, and X-Windows environment did not allow rapid enough development of DARE. We chose Visual Basic as a faster development environment, and the second DARE prototype was developed in Visual Basic 3 on a PC running Windows 3.1 during 1995. We found that this environment cut development time about tenfold. Especially useful was a commercial graphical editor VBX that we were able to tailor to our needs. The book mechanism and the cluster editor from the first prototype were re-created, and several major subsystems were implemented, including:

- the domain expert information entry forms and system feature table;
- the system architecture editor;
- the facet table;
- the generic architecture editor and the generic feature table.

DARE COTS

Even Visual Basic proved too slow a development environment as time and project resources became short. We therefore created another prototype using commercial-off-the-shelf tools (COTS) and freeware. We found that this prototype allowed us to create a version of DARE with much of the planned functionality in a fraction of the time required for either C or Visual Basic.

The DARE COTS implementation using commercial-off-the-shelf tools (COTS) and freeware is made possible through the identification of a backplane that contains the tool slots needed by DARE. These are listed in the table below. Two implementations of DARE-COTs have been done using this backplane model. The first version was done on the Macintosh, and a later one was done in Windows at the request of a client. Table 1 summaries the tools.

Table 1. DARE-COTS Backplane

Backplane: Needed Tools	**Macintosh**	**Windows**
Domain Book	In Control	MS-Word in outline mode
Source Documents	MS Word	MS Word
Text Analysis	Concordance	lexer
Expert forms	MS Word	MS Word
Graphical Architecture Editor	Inspiration	Inspiration
Feature Table	MS Word	MS Word
Glossary and Bibliography	MS Word	MS Word
Stemming	Porter stemmer	Porter stemmer
Code Analysis	Unix Tools, cflow, prof, cia.	Unix Tools, cflow, prof, cia.

.

The main DARE-COTS tools include:

- forms for acquiring general domain information from experts,
- a graphical architecture editor for recording system architectures,
- a feature table for summarizing system commonalities and variabilities,
- a suite of text processing tools for extracting a domain vocabulary from text sources,
- a clustering tool for deriving faceted classification schemes from the domain vocabulary and for identifying commonalities and variabilities,
- an architecture editor for creating and recording generic architectures,
- a system feature table for recording decisions about commonalities and variabilities in systems based on the generic architecture, and
- a glossary, bibliography, index, and appendices for collecting and organizing reference information.

In the next section, we discuss a version of DARE for the WWW.

DARE-Web

In the last couple of years, Web technologies have emerged as an alternative to client-server computing. Given this trend we decided to explore a Web-based version of DARE using a Web-based computing paradigm. This will allow development of and access to DARE domain books via WWW or Intranets. This is important because when developed, domain books will serve as the central resource for developing and maintaining systems in a domain, and all personnel that work in that domain will need easy access. In this particular case we can say that DARE-Web is a domain specific digital library.

Web-based computing allows the integration of disparate systems into a single and coherent environment (Vetter 1999). Web-based infrastructure is a reality in many areas like eCommerce and it is starting to gain momentum for software engineering projects (Gao et al., 1999).

Architecture

Our proposed architecture uses a commercial Internet platform to handle pretty much all the computation (Oracle 1999). The idea behind using this type of technology is to manage all the information in one place while providing its access from everywhere.

Figure 1 shows DARE-Web high level infrastructure. DARE-Web is an application that sits on top of the HTTP-enabled database (Oracle in our case). The application issues SQL statements to query and retrieve all the assets that are stored in the database. When the database sends back the result of a query, DARE-Web constructs the asset content in HTML and display it in the Web browser.

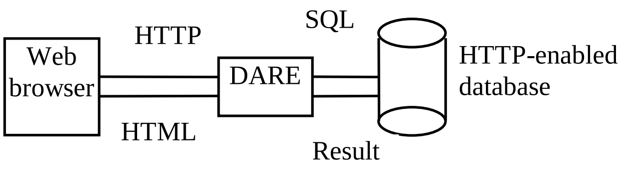


Figure 1. DARE-Web high level infrastructure.

There are three main components of DARE-Web: the asset manager, the book view, and the service manager. The asset manager provide all the features and services for storing assets in the database. The book view component manages the domain book internal structure as well as its visualization. The service manager provides mechanisms for adding new services to the system (in the current implementation there is only one service: search). The storage representation contains all the assets and its metadata with XML tags. Figure 2 describes at high level the software architecture.

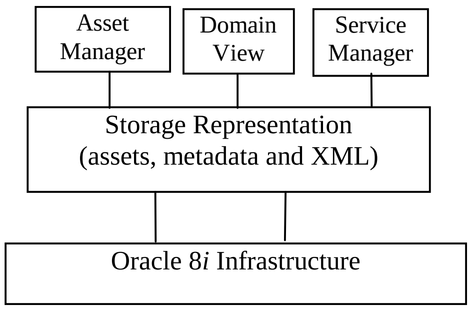

Asset Manager
Domain View
Service Manager
Storage Representation (assets, metadata and XML)
Oracle 8*i* Infrastructure

Figure 2. DARE-Web architecture.

We are experimenting with two alternatives ways of implementing DARE-Web with Oracle tools. The first one is to write the application in PL/SQL using an HTTP package and a Web server that supports PL/SQL cartridges. The second alternative is to use Java servlets for the application and JDBC for connecting to the database. A comparison of the two approaches is beyond the scope of this paper.

Relevant Features

There are several advantages of using a database for the Web implementation. A database backend implementation provides data security, data integrity, scalability, and flexibility. All those things are very important when building Web-based infrastructure.

Apart from all the DARE features we added a search mechanism and the ability to define users (and therefore) roles.

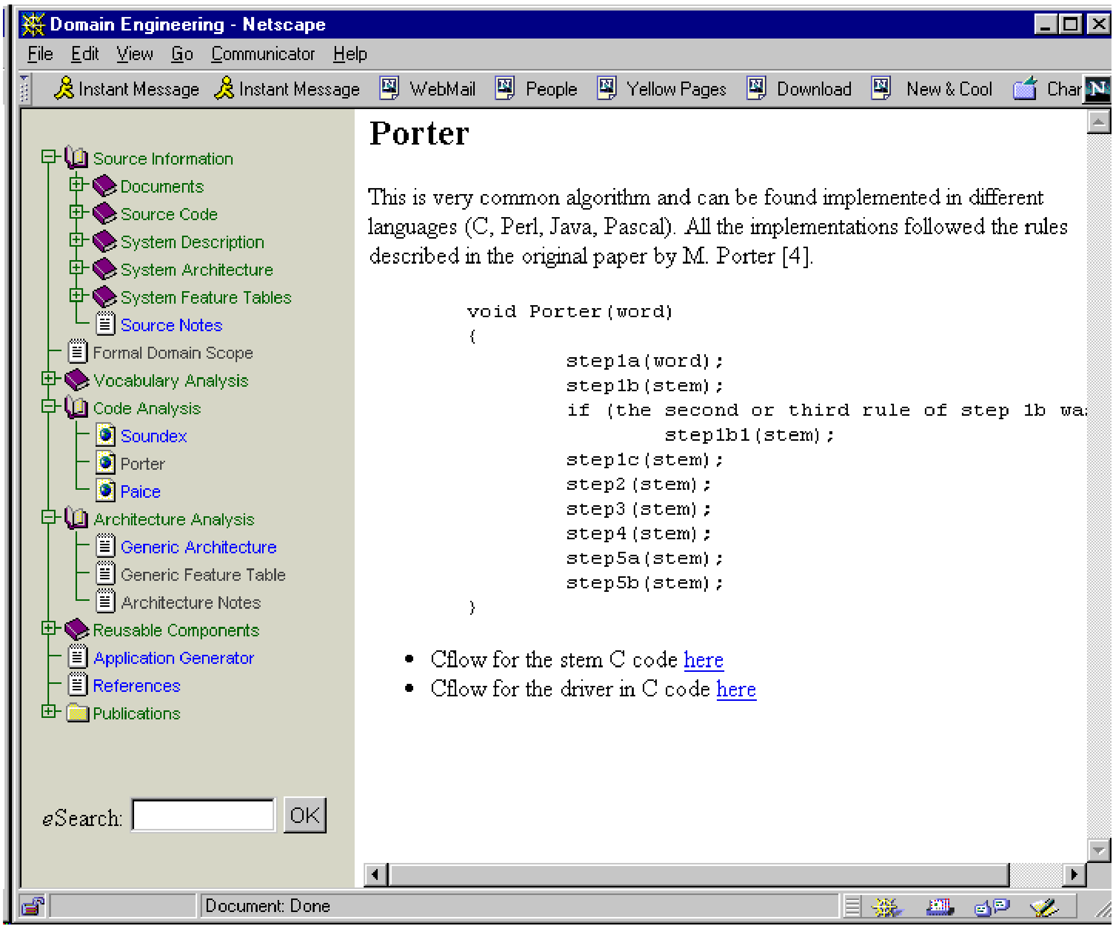
While searching and retrieval is not the main purpose of DARE, the search mechanism allows a user to search for any asset within the domain book. For some assets like documents the user can perform full-text searching. For other assets like diagrams and source code, metadata searching is available.

With the definition of users and roles in the database there is a level of security that was not presented in earlier versions of DARE. It is also possible to add extra security in the Web server.

Example

Figure 2 shows a snapshot of the DARE-Web implementation of stemming algorithms. The left frame shows the domain book (in this case implemented as a Java applet) and the search service.

When the user clicks on any item of the outline structure, DARE-Web displays the content in the right frame. In the example, the user clicked on the Code Analysis section of the outline and then on the Porter asset. In this particular case the asset is document that describes the Porter algorithm with additional links to view the `cflow` output for a C implementation.



Porter

This is very common algorithm and can be found implemented in different languages (C, Perl, Java, Pascal). All the implementations followed the rules described in the original paper by M. Porter [4].

```
        void Porter(word)
        {
                step1a(word);
                step1b(stem);
                if (the second or third rule of step 1b wa
                        step1b1(stem);
                step1c(stem);
                step2(stem);
                step3(stem);
                step4(stem);
                step5a(stem);
                step5b(stem);
        }
```

- Cflow for the stem C code here
- Cflow for the driver in C code here

Figure 2. A domain book for stemming algorithms using DARE-Web

Comparison

Table 2 summarizes all the main features of DARE, DARE-COTS, and DARE-Web.

Table 2. DARE, DARE-COTS and DARE-Web main features.

Feature	**DARE**	**DARE-COTS**	**DARE-Web**
Model	Client-Server	COTS	Web
Storage	Flat files	Flat files	Database
Cluster editor	Yes	Yes	No (*)
Architecture editor	Yes	Yes	No (*)
Search	No	No	Yes
Security features	No	No	Yes
Platform independent	No	No	Yes

(*) DARE-Web does not currently provides this component. However DARE-Web allows a user to upload the file to the system for check in/out purposes.

Conclusions

This paper reviewed the development of domain books using DARE, discussed several implementations of DARE, and presented a Web version of DARE that provides support for domain analysis. The advantage of using the Web version is that users can access the assets from different places with a Web browser. Difficulties include lack of a reusable configurable graphical editor component for Java.

We plan to continue to work on the Web version and to explore different visual representations of the domain book's assets.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments and suggestions.

References

Guillermo Arango, Eric Schoen, and Robert Pettengill. “Design as Evolution and Reuse”, *Proceedings of the Second International Workshop on Software Reusability*, IEEE Computer Society Press, Los Alamitos, CA (1993), pp. 9-18.

William Frakes, Rubén Prieto-Díaz, and Chris Fox. “DARE: Domain Analysis and Reuse Environment”, *Annals of Software Engineering*, Vol. 5 (1998), pp. 125-141.

William Frakes, Rubén Prieto-Díaz, and Chris Fox. “DARE-COTS: A Domain Analsys Support Tool”, *Proceedings of SCCC ’97*, IEEE Computer Society Press, Los Alamitos, CA (1997), pp. 73-77.

Jerry Gao, Chris Chen, Yasufumi Toyoshima, and David Leung. “Engineering on the Internet for Global Software Production”, *IEEE Computer*, May (1999), pp. 38-47.

Ron Vetter. “Web-Based Enterprise Computing”, *IEEE Computer*, May (1999), pp. 113-116.

Oracle 8*i* Reference Manual, Redwood Shores, CA (1999).