Special Report

CMU/SEI-93-SR-21

Process Guide for the DSSA Process Life Cycle

James W. Armitage

December 1993

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1993		**00-00-1993 to 00-00-1993**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Process Guide for the DSSA Process Life Cycle	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**61**	

Special Report

CMU/SEI-93-SR-21
December 1993

Process Guide for the DSSA Process Life Cycle

James W. Armitage
GTE Resident Affiliate

Software Process Definition Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

This work is sponsored by the U. S. Department of Defense. The views and conclusions contained in this document are solely those of the authors and should not be interpreted as representing official policies, either expressed or implied, of Carnegie Mellon University, The U. S. Air Force, the Department of Defense, or the U. S. Government.

Introduction To This Document

Abstract

This document describes the prototype domain-specific software architecture (DSSA) process life cycle developed by GTE as part of the ARPA, formerly DARPA, DSSA program. It is a high-level process description and represents a snapshot of the process as it was in the fall of 1992.

The original version of the document was prepared as part of the Software Engineering Institute's process asset library (PAL) work for the Software Technology for Adaptable and Reliable Systems (STARS) program. That document became the baseline process description for the ARPA DSSA program. The original document is available from the Asset Source for Software Engineering Technology (ASSET) library (asset number ASSET_A_429, file name is PD-081 DSSA-PG-001 Rev 0.2, dated October 16, 1992).

Due to the demand for the document, the original document was reformatted in accordance with related SEI documents so it could be released as an SEI report. The technical content is identical.

Intended audience

This document is intended for those wanting to understand the GTE team DSSA approach to DSSA-based software development.

In this document

This table lists the chapters in the document.

Chapter	Contents
Chapter 1	DSSA Concepts
Chapter 2	Agents
Chapter 3	The DSSA Process Life Cycle
Appendix 1	Glossary
Appendix 2	Index

Not in this document

This document does not address lower level procedural detail.

Source documents

The source documents used in the production of this process guide were:

- DSSA process structured analysis and design technique (SADT) diagrams by Chris Braun of GTE Federal Systems Division.

• Draft proceedings of a panel studying reengineering approaches for the first Joint Logistics Commanders Reengineering Workshop, Chapter 2.

Continued on next page

Methods used

This process guide was developed using

- Software process definition concepts and approach developed by the SEI Software Process Definition Project.
- The information mapping ™ method developed by Information Mapping® Inc.
- IDEF0 notation[1] applied to process modeling.

Tools used

This document was prepared using

- Microsoft Word 4.0
- Tailored version of templates for Word 4.0 by Information Mapping® Inc.
- Design/IDEF tool by Meta Software for construction of IDEF0 diagrams.

Prepared by

This process guide was prepared by Dr. James W. Armitage, GTE Resident Affiliate, for the SEI Software Process Definition Project.

Version

This document is version 0.2a of the process guide.

Its file name is: PD-081 SEI-93-SR-21 .2a

[1]IDEF0 is the functional model notation in the integrated computer-aided manufacturing (ICAM) & definition, aka IDEF, set of notations.

Table of Contents

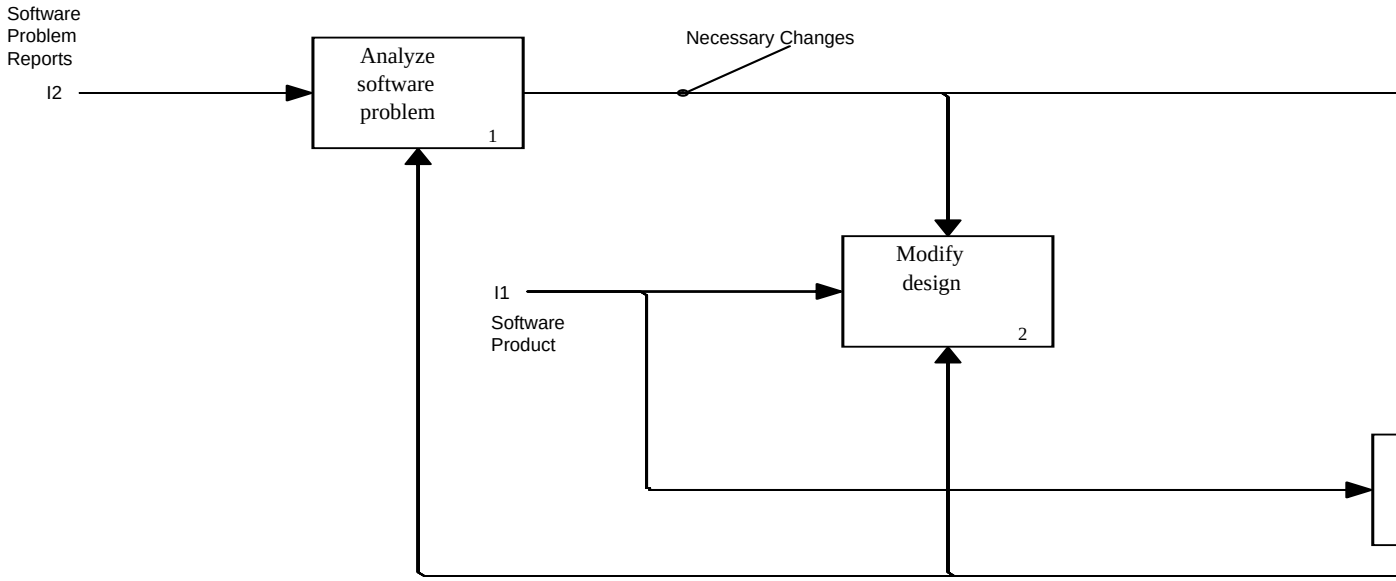
How To Read IDEF0 Process Diagrams

Description

The process was modeled using the IDEF0 notation. Figures from the model are included in this process guide for context and to show the parts of the process.

Example

An example of an IDEF0 diagram appears below.



Parts and function

This table describes the parts of an IDEF0 diagram

Part	Location	Represents
Input	Left side of box	“Things” used and transformed by activities
Output	Right side of box	“Things” into which inputs are transformed
Control	Top of box	“Things” that constrain activities; often information that directs what activities do
Mechanism	Bottom of box	How activities are realized

Boxes	Diagonally across diagram	Activities of system modeled
Arrows	Between boxes	“Things” (and mechanisms) that have relationships with activities.

An IDEF0 box is read as follows: “Under control ... , inputs ... are transformed into outputs ... by the mechanism ... .”

Note: An arrow that is in parentheses may not be shown on the next higher or lower level diagram (this is called tunneling).

Continued on next page

How To Read IDEF0 Process Diagrams, Continued

How to interpret the process model

The SEI Software Process Definition Project has identified three principle elements of software process to be activities, artifacts, and agents. These are represented in an IDEF0 process model as shown in this table.

Process Element	IDEF0 Notation
Activity - what is done and how	Box
Artifact - things used and produced	Control, input, or output and its associated arrow
Agent - who does it	Mechanism

The IDEF0 process model is read as follows: "Under the constraints imposed by ... artifacts, input artifacts ... are transformed into output artifacts ... by agents ... enacting activity ... ."

Chapter 1

DSSA Concepts

Overview

Introduction This chapter presents an overview of DSSA concepts.

Contents This chapter describes the following concepts.

What is DSSA?

Introduction

The software engineering community has realized that software reuse is a key to improving software quality and productivity. Introducing the concepts of domain-specific software architectures is believed to be a primary vehicle for making that happen.

Definition: domain

A *domain* is a set of current and future applications that share a set of common capabilities and data (also called *application domain*).[2]

It is a class of knowledge, functions, features, etc., common to a family of systems.

Definition: DSSA

A *domain-specific software architecture (DSSA)* is:

- A standard software architecture constructed for a domain, or family, of applications.
- A specification for assemblage of software components that are
  - specialized for a particular class of tasks (domain)
  - generalized for effective use across that domain
  - composed in a standardized structure (topology)
  - effective for building successful applications.[3]
- The high-level packaging structure of functions and data, their interfaces and control, to support the implementation of applications in a domain.[4]

[2]W. G. Vitaletti and R. Chhut, *Domain Analysis*, SofTech, Inc., May 1992, pg. B-1

[3]definition provided by Christine L. Braun

[4]W. G. Vitaletti and R. Chhut, *Domain Analysis*, SofTech, Inc., May 1992, pg. B-3, as defined by the term "software architecture"

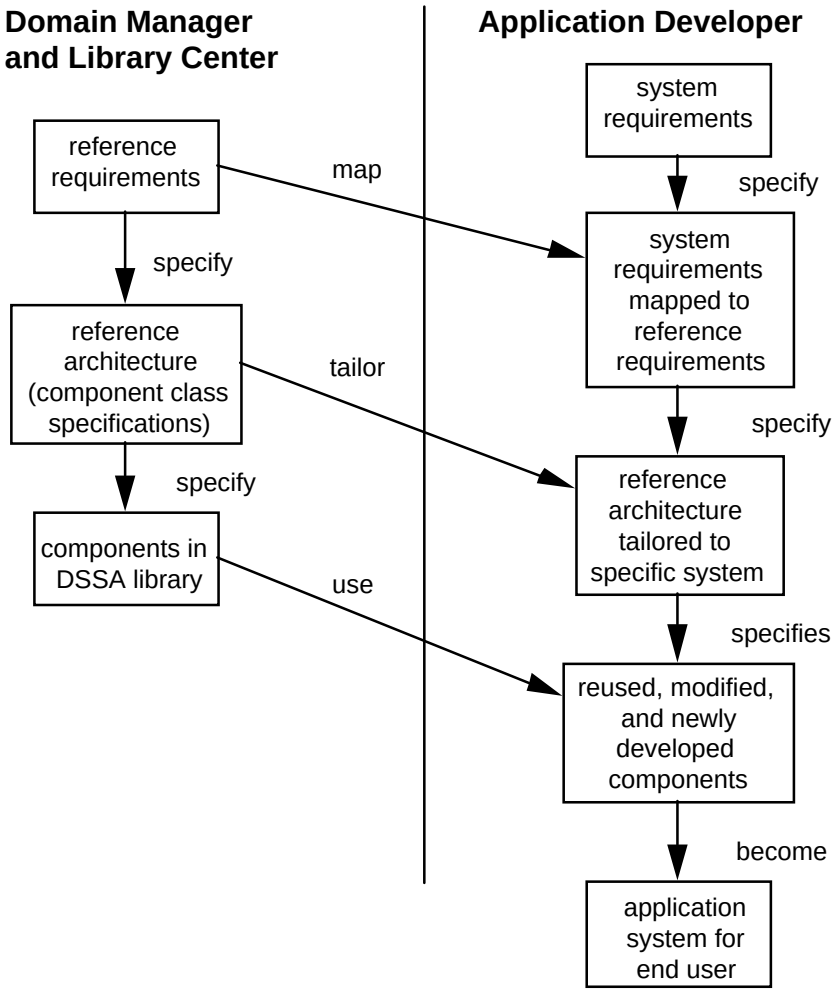
The GTE DSSA Approach

Approach

A domain-specific software architecture, which we call a *reference architecture*, is specified by reference requirements, the product of a domain analysis. Application systems are constructed by tailoring the reference architecture to meet the specific system requirements and populating the architecture with components from the DSSA library.

Figure

This figure depicts the overall DSSA approach being developed by the GTE DSSA team.



Note: feedback paths to the reference requirements, reference architecture, and DSSA library are not shown in the above diagram.

Support tools

This DSSA approach is supported with tools. Requirements for these tools are stated as policies in this document.

The DSSA Process Life Cycle

Introduction

The DSSA process is a software life cycle based on the development and use of domain-specific software architectures, components, and tools. It is a process life cycle supported by a DSSA library and a development environment.

Phases

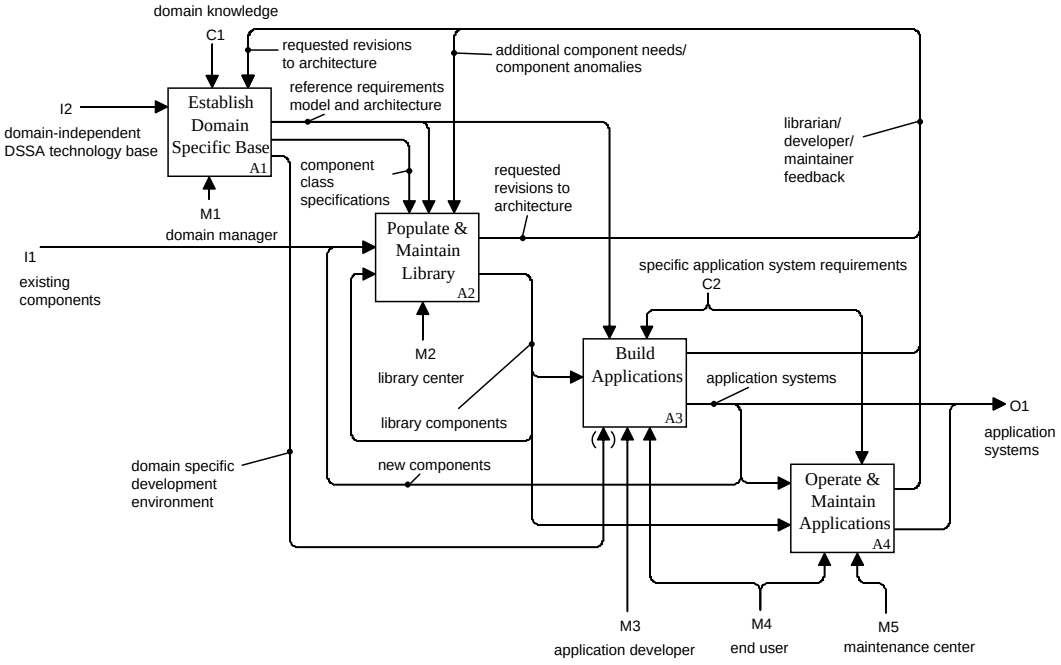
The DSSA process has four distinct activities:

1. Develop domain-specific base
2. Populate and maintain library
3. Build applications
4. Operate and maintain applications

These activities are described in Chapter 3. The agents that perform the process are described in Chapter 2.

IDEF0 diagram

This figure shows the four activities in the DSSA life cycle.



Key concepts

As can be seen in the above diagram, the building of DSSA application systems (A3) is driven by reference requirements, reference architecture, and library components. The library components are developed and maintained by a DSSA library (A2) and are driven by the reference requirements, reference architecture, and component class

specifications. These concepts are described in the remainder of this chapter.

What is a DSSA Library?

Definition

A *DSSA library* is a library containing domain-specific software assets for reuse in the DSSA process.

The DSSA library may be a collection that is part of a larger collection or library.

The DSSA library may be administered by a library organization.

Library functions

The library's main purpose is for component version control rather than query. The DSSA tool set should find the applicable components.

Contents

The DSSA library contains

Content	Description/Examples
Requirement specification templates	Standard forms for requirements specifications
Reference requirements statements	Standard statements of requirements for systems in the application domain
Reference requirements model	Model of the requirements statements
Reference architecture	Architecture that satisfies the reference requirements
Component class specifications	Specifications for components of the reference architecture
Software design information	Design documents and other design information (for example, from CASE tools), etc.
Components	Software code components that meet component class specifications
Design records	Revision history, modification provisions (how to tailor, etc.)
Manuals	Operation and maintenance manuals, etc.
Test materials	Test plans, procedures, drivers, data, results
Component subassemblies	Subsystems of modules, modules with associated documentation and tests, etc.

Example

The Reusable Ada Products for Information Systems Development (RAPID) library could be used to manage a DSSA library.

What Are Reference Requirements?

Definition	A *reference requirement* is a generic requirement for the domain.
Policy: machine readable	Reference requirements shall be available in a machine-readable form. Example Reference requirements are maintained in a tool such as RDD-100.
Policy: traceability	Reference requirements shall be electronically referenced to (alternative) elements of the reference architecture. Example A requirements traceability database is maintained in the DSSA library.

What is a Reference Architecture?

Definition

A *reference architecture* is a generic set of architectural component specifications for a domain (and at least one instance).

A reference architecture is composed of component class specifications.

Comment

The reference architecture defines the solution space, whereas the reference requirements define the problem space.

Definition

A *component class specification* is an element of the reference architecture that specifies what elements of the architecture do and what their interfaces are. A particular system substitutes a specific element. There may be multiple elements in the DSSA library that meet the specification.

Note: "Class" does *not* imply inheritance in the object-oriented programming sense.

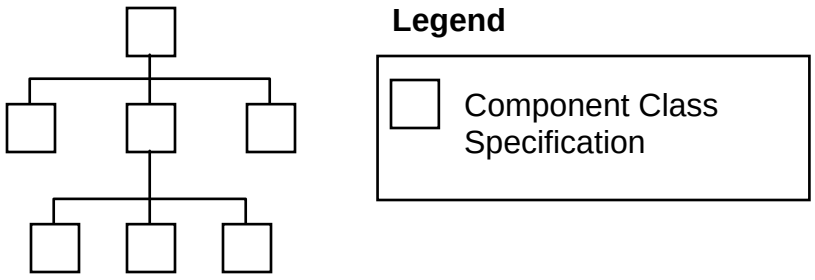
Counter-example

Typically, an Ada generic package is *not* a component of the reference architecture; it is a component of a particular system architecture. It is a tangible, specific artifact, as opposed to the abstract, intangible specifications in the reference architecture. It should reside in the DSSA library and may be a component of one or more system architectures.

(An Ada generic package could be a component of the reference architecture if Ada was used as the specification language.)

Figure

This figure represents a reference architecture, composed of component class specifications.



Policy

The reference architecture is available in a machine-readable form.

Example
The behavioral portion of the reference architecture is maintained in a tool such as RDD-100.

What is a System Architecture?

Definition

A *system architecture* is an instance of an architecture that meets the specifications in a reference architecture tailored to meet the requirements of a specific system.

A system architecture is composed of components that meet the component class specifications, plus additional components.

Synonyms

Other terms used for the system architecture include

- application architecture
- target application architecture

Analogy

A reference architecture is to a system architecture as Posix is to a Posix-compliant Unix operating system for a specific target machine (e.g., VAX Ultrix).

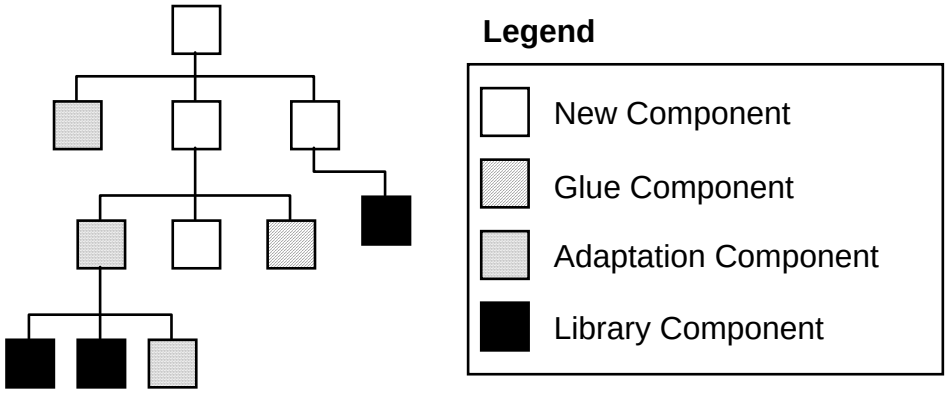
Component types

There are different types of components used to implement the system architecture. These are listed in increasing order of original design preservation.

Type	Changes
New component	Entirely new
Glue component	Only differences are developed (see page 11)
Adaptation component	Only parameters change (see page 10)
Library component	No changes, used as is

Components

This figure represents a system architecture derived from a reference architecture, composed of components.

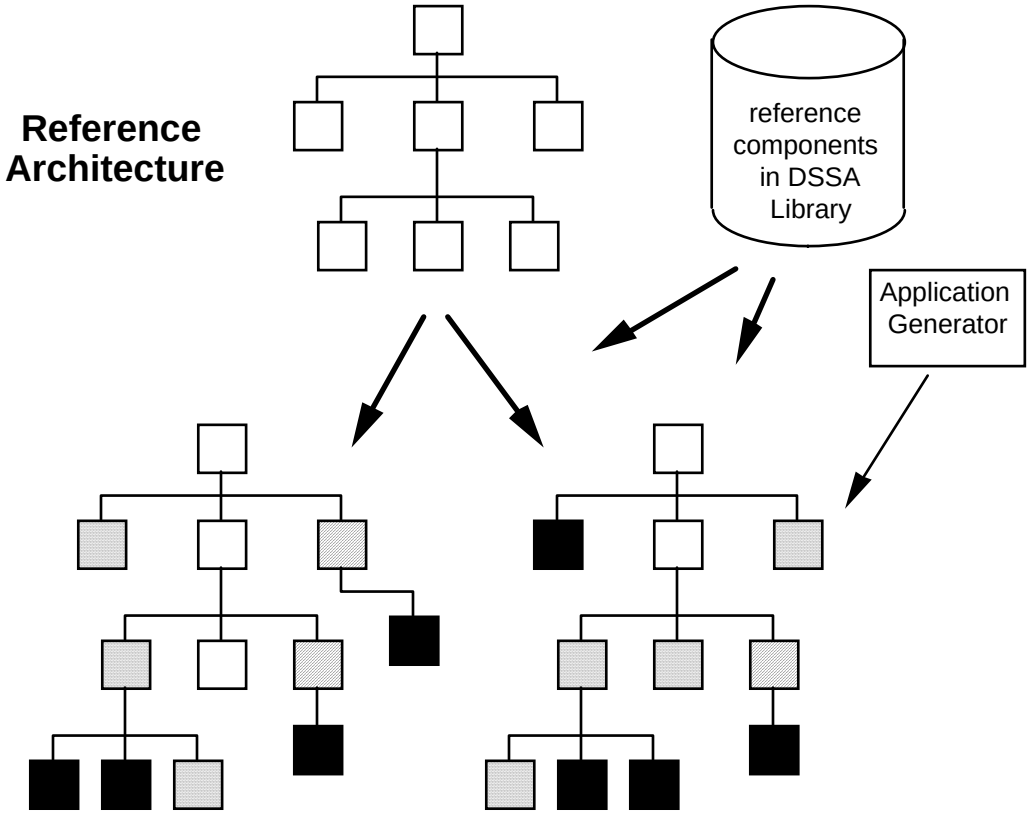


Continued on next page

What is a System Architecture?, Continued

DSSA concept drawing

This drawing depicts the DSSA concept. The component class specifications in the reference architecture are realized in multiple system architectures with existing and reengineered components from the DSSA library, generated components, and new components.



Adaptation Component

Definition	An *adaptation component* is a component that is built using (conforms to) the adaptation provisions of a component class specification. Adaptation provisions are highly dependent on the internals of a reference component.
Comment	An adaptation component preserves the integrity of the original design. The designer of the component supports adaptation by providing some form of parameterization. Its reuse does not require modification of the component. The integrity of the design is preserved across a range of adaptation parameters.
Limitations	An adaptation component cannot be tailored beyond its provided range of parameterization without knowledge and modification of its internals (then it would be a "new component" produced by reengineering).
Examples	Examples of adaptation mechanisms are • Macro expansions • Generic units (for example, Ada generic unit) • Callback routines • Parameters for a component generator

Glue Component

Definition	A *glue component* is a component that • Uses the interface specifications of reference components to define new application-specific objects. • Converts outputs of one reference component for suitable input to another in a way not available in the reference architecture.
Comment	The integrity of the original component in the DSSA library is preserved, as it is never modified. The additional functionality and differences are implemented in the glue component.
How created	To create a glue component, one does not have to know the internals of a component in the library; the new component uses the existing component through its interface. The glue component extends the functionality of the existing component by adding to the functionality already provided.
Example	A reference architecture contains a component class specification for a component that determines an aircraft's position and speed. For a specific system, that component class specification is tailored to include the aircraft's identification. A component that meets that tailored component class specification in the system architecture can be implemented with a glue component and an associated existing component in the DSSA library that meets the original specification. In an object-oriented programming language, the glue component would define a subclass of the aircraft class defined in the existing component, add the identification attribute and associated methods, and override any other methods as appropriate.
What they indicate	Glue components are precursors of new reference architectures. They arise when two or more subarchitectures without a common parent are used in the same application.

What is Unique About This Approach?

Uniqueness

This approach is unique because it

- Supports reuse across each of several dimensions (no single prescription - one or the other or both)
  - compositional vs. generative
  - small-scale vs. large-scale reuse
  - as is vs. reuse with modification
  - generality vs. performance
- Supports multiple forms of reuse - The most effective reuse is code reuse; however reuse of specs, designs, tests, and documentation are also important by themselves in support of code reviews.
- Recognizes the need to develop correlated user (application domain) and application developer environments.
- Provides a blueprint for standardization of parts.
- Shifts the focus away from product to process, potentially boosting productivity further.

Benefits

Among its benefits, this approach

- Provides a mechanism to allocate resources to domain-unique components while taking advantage of cross-domain components.

  Example

  - separately identify non-domain-specific components that can be obtained commercially (for example, DBMS)
  - put mission dollars into domain-unique components
- Incorporates prototyping (not explicit in model yet) — builds on existing architecture.
- Breaks down barriers associated with presumed untrustworthiness of existing components by providing demonstrably validated components for a domain.

Chapter 2

Agents

Overview

Introduction

This chapter describes those who participate in the process, whom we call "agents."

Definitions

In this process guide, the following process terms are used:

The term *agent* refers to those who participate in, or enact, a process. An agent can be an organization or a role within an organization.

An *organization* is responsible for performance of a process element, typically an agency, command, or company.

A *role* is a uniquely identified class of individuals based on qualification, skills, or responsibilities that performs specific activities in a process element.

Note: Some methodologies include tools (that automate process enactment) in the definition of agent. Here we only addresses the humans that enact the process, shown as mechanisms in the IDEF0 diagram.

Organizations

The following organizational agents enact the DSSA process:

Organization	See Page
Domain manager	14
Library center	15
Application developer	16
End user	17
Maintenance center	18

Roles

Two roles are explicitly described in this process guide.

Role	See Page
Domain architect	19
Domain expert	20

Domain Manager

Definition	A *domain manager* is an organization that manages a family of related systems within a domain. The domain manager would be the organization motivated to develop a domain-specific software architecture, benefiting from a common technology base.
Responsibilities	The domain manager's responsibilities are to • Manage a family of related systems within a domain. • Manage program managers. • Resolve differences among program managers. • Forecast future system needs. • Control budgets and schedules. • Set strategic direction.
Example: military	A Program Executive Office (PEO) manages a family of application systems. Examples include • Army Tactical Center • AF MIS • Navy logistics
Example: commercial	A product line manager's organization is a "domain manager" responsible for a company's line of business.
Roles	The domain manager organization includes staff that fulfill the technical roles defined in this process: • Domain expert • Domain architect

Library Center

Definition

A *library center* is an organization responsible for acquiring and maintaining the domain-specific components and managing the library.

The library may be part of a domain manager's organization or an independent, external organization.

Policy

Changes to component class specifications must be approved by the domain architect.

Changes to the requirements and architecture are made by the domain architect.

Responsibilities

The library center's responsibilities are to

- Classify and install components in DSSA library.
- Maintain library components.
- Perform configuration management.
- Collect component usage metrics.
- Provide library concept of operations and mechanism.
- Develop component acquisition strategy.
  - evaluate existing components
- Provide other user services (such as help desk).

Examples

The library center could be an externally run asset library such as

- Asset Source for Software Engineering Technology (ASSET) — DoD funded through ARPA/STARS.
- Defense Information Systems Agency/Center for Information Management (DISA/CIM) Defense Software Repository System (DSRS) — a national network connecting instances of [Army] Reusable Ada Products for Information Systems Development (RAPID) libraries.
- Software Technology for Adaptable, Reliable Systems (STARS) reuse libraries.
- Central Archive for Reusable Defense Software (CARDS) — funded by the Air Force.

Application Developer

Definition	An *application developer* is a contractor or government organization that develops new application systems.
Examples	An application developer can be • a defense contractor • a government organization • a commercial company
Responsibilities	The application developer's responsibilities are to • Understand the reference model, reference architecture, and library components. • Build systems to meet requirements. • Tailor and modify components in the DSSA library. • Submit new components to the DSSA library.
Domain expert	Included in the application developer organization (as well as in the domain manager and maintenance center organizations) is a domain expert.

End User

Definition	An *end user* is the organization that uses the system, including hands-on users and managers. The end user is not necessarily aware of the domain-specific technology applied to the development of their system.
Responsibilities	The end user's responsibilities are to • Use the information and capabilities the system provides. • Provide a source of domain knowledge (a domain expert). • Provide inputs for the requirements for new systems. • Request fixes/changes to existing system and documentation. • Assess system effectiveness.
Example	The "soldier in the field" is an end user.

Maintenance Center

Definition	A *maintenance center* is an organization that changes and improves fielded systems.
Responsibilities	The maintenance center's responsibilities are to • Update the application system. • Provide feedback to the library center.
Example	A post deployment software support (PDSS) center is a maintenance center, as is a life cycle support center.
Domain expert	Included in the maintenance center organization is a domain expert.

Domain Architect

Definition

A *domain architect* is a system/software engineer responsible for analyzing domain requirements, developing the domain-specific architecture, and specifying domain-specific components.

The domain architect resides in the domain manager organization.

Skills needed

A domain architect must have the following qualifications:

- Understand the overall DSSA process.
- Have some experience in the domain.
- Know various requirements elicitation techniques.
- Have proven interviewing and interpersonal communication skills.
- Be familiar with requirements allocation.

- Understand requirements modeling techniques.
- Be able to use at least one requirements modeling technique.
- Use the method selected for architecture description.
- Be able to design a software system architecture.
- Be able to specify components.

Duties

A domain architect's responsibilities are to

- Elicit requirements.
- Define reference requirements.
- Model reference requirements.
- Establish consensus model.
- Allocate requirements to architecture.
- Develop component class specifications.

- Establish relationship with reuse library.
- Specify reusable components.
- Tailor environment to domain.
- Modify reference requirements model in accordance with feedback from application developer and maintenance center.
- Modify architecture in accordance with feedback from application developer and maintenance center.
- Modify component specification in accordance with feedback from application developer and maintenance center.
- Approve changes to components.

Domain Expert

Definition

A *domain expert* is an expert in the application domain.

Domain experts can be found in the following organizations:

- Domain manager
- Application developer
- Maintenance center

Skills needed

A domain expert must have the following qualifications:

- Have experience in the domain (more than one application).
- Understand requirements modeling techniques.
- Be able to use at least one requirements modeling technique.
- Be able to express user needs as requirements.
- Understand end user needs and requirements.
- Be able to evaluate design decisions from the user's perspective.

Duties

A domain expert's responsibilities are to

- Define reference requirements.
- Model reference requirements.
- Review the domain reference model.
- Interface with end users and understand their needs.
- Capture user needs as requirements.

Examples

A domain expert in the application developer organization may be someone formerly from the end user organization.

A requirements developer in US Army Training and Doctrine Command (TRADOC) is a domain expert in the domain manager organization.

In the commercial world, someone from a vendor's marketing organization who specifies what the market wants would be the domain expert.

Counter-example

An engineer experienced in the development of an application is *not* a domain expert. In addition to understanding a specific application, a domain expert understands the domain thoroughly, from the perspective of the user's current and future needs.

Chapter 3

The DSSA Process Life Cycle

Overview

Introduction

This chapter presents the DSSA process life cycle model.

In this chapter

The top levels in the process life cycle model hierarchy are described in this chapter as indicated in the following table:

0 - The DSSA Process Life Cycle

Introduction

The DSSA process is a software life cycle based on the development and use of domain-specific software architectures, components, and tools. It is a process life cycle supported by a DSSA library and a development environment.

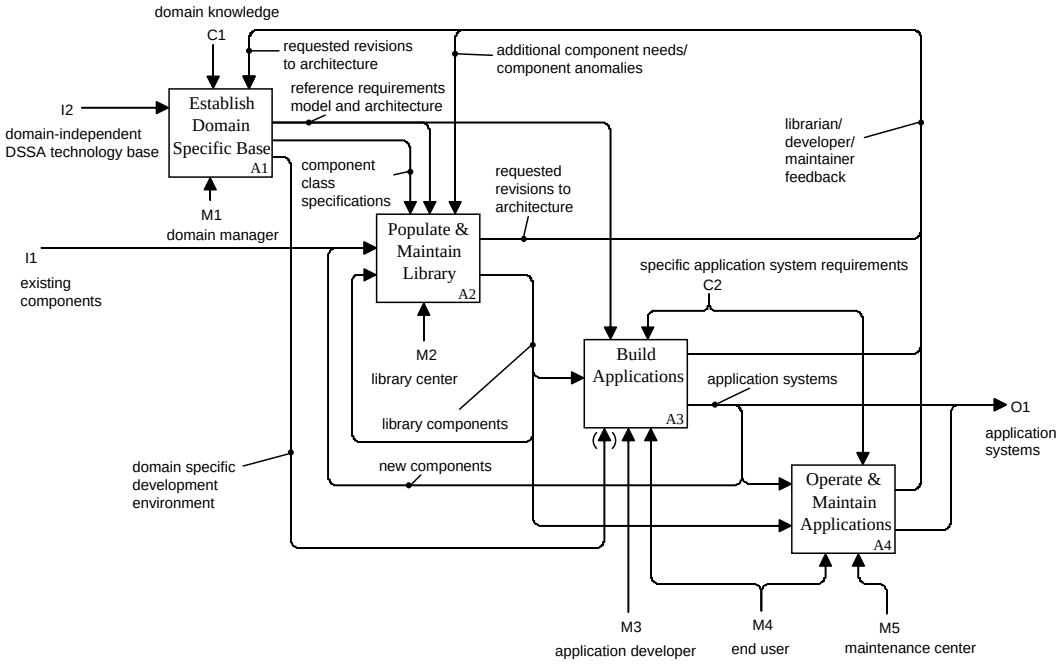
Phases

The DSSA process has four distinct activities:

Activity	Description
Develop domain-specific base	The domain is analyzed and a domain-specific software architecture (reference architecture) and development environment are produced.
Populate and maintain library	Components meeting the component class specifications in the reference architecture are collected, modified, and /or developed.
Build applications	An application system is constructed using the DSSA library and DSSA tool set.
Operate and maintain applications	The application system is operated in the field, maintained, and feedback is provided to the domain manager and DSSA library.

IDEF0 diagram

This figure shows the four activities in the DSSA life cycle.



1 - Establish Domain-Specific Base

Introduction

This first phase of the DSSA process will

- Construct multiple views of the domain independently by the domain experts, ensuring greater coverage of concepts.
- Construct a consensus requirements list and requirements model.
- Define reference requirements and reference architecture.
- Identify components.
- Construct component class specifications.
- Instantiate the DSSA tool set for the specific domain.
- Put together the domain-specific software development environment.

Who performs

This phase is performed by the domain manager. The activities are led by the domain architect with participation of domain experts. The domain experts may come from different organizations managed by the domain manager, providing different user perspectives.

Inputs

The inputs to this activity are listed in the following table:

Artifact	Description
Domain knowledge	• Strategies/directions for the domain. • Technology constraints. • Domain expert knowledge of user needs.
Requested revisions to architecture	Request for change. Can come from library center (activity A2), application developer (activity A3), or maintenance center (activity A4).
Domain-independent DSSA technology base	This is a broad term that includes the set of things that come out of the ARPA DSSA program including • Methods for domain modeling. • Process model, process definition. • Tool set. - requirements analysis tools to identify and collect reference requirements - architecture tools • Knowledge base of DSSA. • DSSA software development environment.

Note

In tables of inputs we are not distinguishing inputs and controls as

shown on the IDEF0 diagram.

Continued on next page

Outputs

The products of this phase are the following artifacts:

Artifact	Description
Reference requirements model and architecture	The requirements model is a model of the reference requirements (problem space model). It may be a multi-paradigm model with different technical views (for example, object, data flow, behavior views). The reference architecture is the generic set of architectural component specifications for the domain, that is, the solution space model.
Component class specifications	Specify what elements of the reference architecture do and what their interfaces are.
Domain-specific development environment	A tailoring of the tool set provided by the DSSA program.

Continued on next page

1 - Establish Domain-Specific Base, Continued

Activities The following activities are performed in this phase:

Activity	Inputs	Tasks	Outputs
Model multiple views	• Domain knowledge	• Multiple views of the domain are constructed independently by the domain experts, ensuring greater coverage of concepts.	• Individual requirements models (different user perspectives)
Establish consensus model	• Domain knowledge • Individual requirements models • Requested revisions to architecture	• Select a consistent terminology (glossary). • Develop a consistent family of model views. • Conduct a workshop to get domain expert feedback.	• Reference requirements model
Allocate requirements to reference architecture	• Domain knowledge • Reference requirements model • Requested revisions to architecture	• Construct the reference architecture by allocating the reference requirements in the reference requirements model.	• Reference architecture
Specify reusable component classes	• Domain knowledge • Reference architecture	• Make specifications for a selected subset of components.	• Component class specifications

Tailor environment to domain	• Reference requirements model • Reference architecture • DSSA software development environment • Tool/platform availability	• Adapt the domain-independent DSSA technology base for the domain. - e.g., create parameters for generic application generator	• Domain-specific development environment

2 - Populate and Maintain Library

Introduction

This phase of the DSSA process will

- Identify sources for components that will meet the component class specification in the reference architecture.
- Collect, modify existing components, and develop new components for the library.

Who performs

This phase is performed by the library manager.

Inputs

The inputs to this phase are listed in the following table:

Artifact	Description
Reference requirements model and architecture	The requirements model is a model of the reference requirements (problem space model). It may be a multi-paradigm model with different technical views (for example, object, data flow, behavior views). The reference architecture is the generic set of architectural component specifications for the domain, that is, the solution space model.
Component class specifications	Specify what elements of the reference architecture do and what their interfaces are.
Additional component needs/component anomalies	New things wanted and changes requested — can come from application developer, or maintenance center.
Existing component	A component existing outside of the library — includes new components constructed by the application developer.
Library components	Components in the library, including fixed and reengineered components.

Continued on next page

2 - Populate and Maintain Library, Continued

Outputs

The products of this phase are the following artifacts:

Artifact	Description
Requested revisions to architecture	Request for change from library center (activity A2) to the domain manager (activity A1).
Library components	Components in the library, including fixed and reengineered components. One or more components may be developed for each component class specification. (Any developed application generators become part of tool set supporting the domain.)

Continued on next page

2 - Populate and Maintain Library, Continued

Activities The following activities are performed in this phase:

Activity	Inputs	Tasks	Outputs
Develop acquisition strategy	• Library components • Existing components • Component class specifications • Problems with strategy • Component needs/anomalies	• Identify sources of components. • Select sources and state in acquisition strategy.	• Acquisition strategy • Identified similar or matching components
Provide components	• Similar components • Matching components • Acquisition strategy • Library components • Reference architecture • Component class specifications	Provide components as stated in the acquisition strategy: • Use as-is existing component. • Reengineer existing components. • Develop application generator. • Develop components manually.	• Problems with strategy • Requested revisions to architecture • Components that meet specification • Application generator

Install in DSSA library	• Components that meet specification • Library components • Reference architecture • Component class specifications • Component needs/anomalies	• Place components in DSSA library.	• Library components

3 - Build Applications

Introduction

This phase of the DSSA process will

- Tailor reference requirements for the specific system.
- Produce an instance of the reference architecture the meets a specific system requirements specification.
- Develop the software.

Who performs

This phase is performed by the application developer. The end user participates in reviews during the development.

Inputs

The inputs to this phase are listed in the following table:

Artifact	Description
Reference requirements model and architecture	The requirements model is a model of the reference requirements (problem space model). The reference architecture is the generic set of architectural component specifications for the domain, that is., the solution space model.
Specific application system requirements	Requirements for a particular system procurement.
Library components	Components in the library, including fixed and reengineered components.

Outputs

The products of this phase are the following artifacts.

Artifact	Description
Application systems	Systems developed for use by the end user.
Developer feedback	Request for change.

Relationship to domain analysis

One may consider the *establish domain-specific base* activity (A1) as a domain knowledge life cycle and the *build application*s activity (A3) as a software development life cycle which are linked by the *DSSA library* (activity A2).

Continued on next page

Activities The following activities are performed in this phase:

Activity	Inputs	Tasks	Outputs
Develop requirements	• Reference requirements model • Reference architecture • System requirements • Requirements modifications (feedback)	• Compare system requirements to reference requirements with help of tools. • Restate requirements in terms of reference requirements with help of tools.	• Requirements specified in terms of reference model • Feedback on reference requirements model
Design application system	• Reference architecture • Requirements specified in terms of reference model • Library components • Design modifications (feedback)	Tailor reference architecture to meet system requirements: • Select reference components. • Design adaptation components. • Design glue components. • Develop new component class specifications if needed.	• System architecture • Requirements modifications • Feedback on reference architecture
Implement system	• Requirements specified in terms of reference model • System architecture • Library components	• Plan and monitor. • Code the adaptation components. • Code the glue components. • Code the new components. • Integrate components.	• Application system • Unmet component needs • Requirements modifications • Design modifications

Note These activities may be concurrent. This depiction does not imply the waterfall life cycle model.

4 - Operate and Maintain Applications

Introduction

This phase of the DSSA process will

- Operate the system in the field and maintain.
- Provide feedback to the DSSA library and the domain architecture.

Who performs

This phase is performed by the end user and maintenance center.

Inputs

The inputs to this phase are listed in the following table:

Artifact	Description
Specific application system requirements	Requirements for a particular system procurement.
Application systems	Systems for the end user produced by the application developer.
Fixed components	Updated components from the library center.

Outputs

The products of this phase are the following artifacts:

Artifact	Description
Maintainer feedback	Request for change — incompleteness, what is wrong with architecture or components, faults or enhancements.
Application systems	Systems revised by the maintenance center.

Continued on next page

4 - Operate and Maintain Applications, Continued

Activities The following activities are performed in this phase:

Activity	Inputs	Tasks	Outputs
Carry out application	• System requirements • Mission inputs	• End user performs missions using the application system.	• Mission outputs
Assess effectiveness	• System requirements • Mission outputs • Mission changes	• End user assesses the effectiveness of the system in accomplishing the missions. • End user assesses the effectiveness of the system with respect to future missions.	• Needed change or correction
Maintain system	• System requirements • Needed change or correction • Application system • Fixed components	• Maintenance center corrects and enhances the application systems it is responsible for. • Maintenance center procures fixes to reference requirements model, reference architecture, and/or library components.	• Revise application system for end user • Needed changes to reference model • Needed changes to reference architecture • Needed changes to library components

Appendix 1

Glossary

Introduction	This appendix contains a glossary of terms and acronyms used in this process guide.
adaptation component	A component that is built using (conforms to) the adaptation provisions of a component class specification.
agent	One who participates in, or enacts, a process. An agent can be an organization or a role within an organization.
application developer	A contractor or government organization that develops new application systems.
component class specification	An element of the reference architecture that specifies what elements of the architecture do and what their interfaces are.
domain	A class of knowledge, functions, features, etc., common to a family of systems
domain architect	A system/software engineer responsible for analyzing domain requirements, developing the domain-specific architecture, and specifying domain-specific components.
domain expert	An expert in the application domain.
domain manager	An organization that manages a family of related systems within a domain.
DSSA	Domain-specific software architecture — a standard software architecture constructed for a domain (family of applications); a specification for assemblage of software components.

Continued on next page

Glossary, Continued

DSSA library	A library containing domain-specific software assets for reuse in the DSSA process.
end user	The organization that uses the system, including hands-on users and managers.
glue component	A component that uses the interface specifications of reference components to define new application-specific objects, converts outputs of one reference component for suitable input to another in a way not available in the reference architecture, etc.
library center	An organization responsible for acquiring and maintaining the domain-specific components and managing the library.
maintenance center	An organization that changes and improves fielded systems.
organization	The agent responsible for performance of a process element, typically an agency, command, or company.
reference architecture	A generic set of architectural component specifications for a domain.
reference requirement	A generic requirement for the domain.
reference model	A model of the reference requirements (problem space model).
role	A uniquely identified class of individuals based on qualification, skills, or responsibilities that performs specific activities in a process element.
system architecture	An instance of a system that meets the specifications in the reference architecture.

Appendix 2

Index

Introduction This appendix contains an index of terms used in this process guide.