Comparison of Domain Analysis Methods and their supporting Quality Attributes

Thesis

submitted in partial fulfillment of the requirements

for the award of degree of

Master of Engineering

in

Software Engineering

By:

Aman Jatain
(80731001)

Under the supervision of

Mrs. Shivani Goel

Lecturer, CSED,

Thapar University, Patiala.

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

JUNE 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Comparison of Domain Analysis Methods and their Supporting Quality Attributes"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of **Mrs Shivani Goel** and refers other researchers' works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Aman Jatain

(Aman Jatain)

(Roll No. 80731001)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Shivani Goel

(Mrs. Shivani Goel)

Computer Science and Engineering Department

Thapar University,

Patiala.

Countersigned by:

23/06/05.

(Dr. Rajesh Kumar Bhatia)

Assistant Professor & Head

Computer Science & Engineering Department,

Thapar University,

Patiala.

(Dr. R.K. Sharma)

Dean (Academic Affairs)

Thapar University,

Patiala.

Acknowledgement

No volume of words is enough to express my gratitude towards my guide **Mrs. Shivani Goel,** Lecturer (CSED), Thapar University, who has been very concerned and has aided for all the material essential for the preparation of this thesis. She has helped me explore this vast topic in an organized manner and provided me with all the ideas on how to work towards a research-oriented venture.

I am thankful to **Dr. Rajesh Kumar Bhatia**, Head, Computer Science & Engineering Department, TU, Patiala, for the motivation and inspiration that triggered me for the thesis work. I would also like to thank the entire staff member and my colleagues who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of the thesis.

Most importantly, I would like to thank my parents and the almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there is no hope.

Aman Jatain

Aman Jatain
(80731001)
M.E. (Software Engineering)-2nd year
Computer Science & Engineering Department
Thapar University
Patiala –147004.

Abstract

Domain analysis has been suggested by many in the reuse research community as an important process for achieving successful reuse. Reuse of products, processes and all kinds of knowledge has been identified as a goal in software engineering, in order to develop reliable and high quality software systems on schedule and within budget. In software reuse there are producers and consumers of reusable artifacts, which might not be the same people. This duality of issues calls for methods as Domain Analysis to systematically build reusable elements. In this report a detailed description of domain analysis including domain analysis approaches, domain analysis models and domain analysis methods is given.

After studying many domain analysis methods some criteria has been found based on which we compared domain analysis methods. Report discusses the domain analysis methods that share the same objectives: analyzing the domain and developing domain models. However each technique defines a particular way of understanding the domain and capturing domain information as domain models. In general the process, the product and supporting tools, can characterize a domain analysis method. In this thesis report comparison of methodologies is given with the help of a table, we summarized methods according to their use in various domains and then various quality attribute in different domain analysis methods are identified.

.

Abbreviations

ARPA	Advanced Research Project Agency's
CIM	Center of Information Management
COTS	Commercial-Off-The-Shelf
DA	Domain Analysis
DADP	Domain Analysis and Design Process
DISA	Defense Information Systems Agency
DSSA	Domain Specific Software Architecture
FODA	Feature Oriented Domain Analysis
JIAWG	Joint Integrated Avionics Working Group
JODA	Joint Object Oriented Domain Analysis
LSP	Languages for Special Purposes
ODM	Organizational Domain Modeling
RLF	Reuse Library Framework
STARS	Software Technology for Adaptable and Reliable Systems Program

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Software development may not become an engineering discipline if it has not worked on a technology for developing products from reusable assets in regular manner, on an industrial scale. As discipline software reuse must define and promote the managerial, organizational, and technical standards that are required to achieve this goal [2]. One of the most important organizational decisions that having the greatest impact on reuse operations is whether the organizational structure is domain centered or application centered. Under domain-centered organization, domain-engineering team decides the development of reusable assets, leaving it to application engineering team to adjust their design discipline or activity to take the best advantage of available reusable assets. Under application-centered organization, the application engineering team delegates development tasks to the domain engineering team, to serve the goals of their application development activity.

In domain engineering, domain analysis or product line analysis, is the process of analyzing related software systems in a domain to find their common and variable parts. The term was given in the early 1980s by James Neighbors. Domain analysis is the first phase of domain engineering. It is a key method for realizing systematic software reuse [1].

Domain analysis is accomplished by reengineering techniques and domain analysis methods. Domain analysis is the process of identifying, collecting, organizing and representing the relevant information in a domain based upon the study of existing systems and their development histories, knowledge captured from domain experts and emerging technology within a domain. During software development, information of several kinds is generated, from requirements analysis to specific designs to source code. Source code is at the lowest level of abstraction and is considered the most detailed representation of a software system. Complementary key information is also generated during software development. Code documentation, history of design decisions, testing plans, and user manuals are all essential to convey a better understanding of the total system.

One of the objectives of domain analysis is to make all that information readily available for reuse. In making a reusability decision, that is, in trying to decide whether or not to reuse a component, a software engineer has to understand the context, which prompted the original designer to build the component the way it is. The chain of design decisions used in the development process is absent in the source code. By making this development information available, a reuser can learn to make reuse more effective.

An important improvement of the reuse process happens when we succeed in deriving common architectures, generic models or specialized languages by using domain analysis that helps the software development process in a specific problem area [1]. How do we find these architectures or languages? It is by identifying features common to a domain of applications, selecting and abstracting the objects and operations that characterize those features, and creating procedures that automate those operations. This intelligence-intensive activity results, typically, after several of the "same kind" systems have been constructed. It is then decided to isolate, encapsulate, and standardize certain recurring operations. This is the very process of domain analysis: identifying and structuring information for reusability.

Unfortunately, domain analysis is conducted in an ad-hoc manner and success stories are more the exception than the rule [1]. The process of concept abstraction from identifying common features is usually considered as an exclusive human (i.e., intelligent) activity and commonly associated with "experience". Expert programmers, for example, are more proficient in coming up with the appropriate program construct that solves a given programming problem than novice programmers. Through experience, experts have created a larger collection of abstracted templates they can draw from when trying to solve a problem. This is reuse of encapsulated knowledge. Little is known about the process involved in deriving and organizing such collections of abstract concepts. Gaining experience is a slow unstructured learning process. Similarly, domain analysis is a slow unstructured learning process that leads to the identification, abstraction, and encapsulation of objects in a particular domain. Domain analysis research should try to reuse existing research from other disciplines. To make reuse possible, software engineering is divided into two parts: domain engineering to find and implement the

common features and application engineering to produce the individual applications. Domain engineering is then divided into three phases: domain analysis, domain design, and domain implementation. This report concentrates on domain analysis phase of domain engineering.

1.1 Domain Analysis

Domain analysis is first introduced by Neighbors to denote to study the problem domain of a family of applications. Domain analysis is associated with reuse; its purpose is to capture information involved with the domain to be reused in developing further applications of the same domain. However, domain analysis is not necessarily restricted to product-line engineering; instead, it can be used in the context of single-system engineering, too. Domain Analysis is the process that identifies the relevant objects of an application domain.

Domain analysis can be defined as: "a process by which information used in developing software systems is identified, captured and organized with the purpose of making it reusable when creating new systems"[4]. During software development, different information is produced, and the software product delivered is only a part of this heap of data. One of the main goals of domain analysis is to analyze all the information aiming to exploit and reuse most of them in future software development projects. Domain analysis is an activity occurring prior to system analysis. It aims to identify features common to a domain of applications, selecting and abstracting the objects and operations that characterize those features.

The list below offers definitions of several terms which are basic to domain analysis, and which are essential to the discussion of a domain analysis method.

- **Application:** A system, which provides a set of general services for solving some type of user problem.
- **Context:** The circumstances, situation, or environment in which a particular system exists.
- **Domain:** (also called application domain) A set of current and future applications which share a set of common capabilities and data.

- **Domain analysis:** The process of identifying, collecting, organizing, and representing the relevant information in a domain based on the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within the domain.
- **Domain engineering:** An encompassing process, which includes domain analysis and the subsequent construction of components, methods, and tools that address the problems of system/subsystem development through the application of the domain analysis products.
- **Domain model:** A definition of the functions, objects, data, and relationships in a domain.
- **Feature:** A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems.
- **Software architecture:** The high-level packaging structure of functions and data, their interfaces and control, to support the implementation of applications in a domain.
- **Software reuse:** The process of implementing new software systems using existing software information.
- **Reusable component:** A software component (including requirements, designs, Code, test data, etc.) designed and implemented for the specific purpose of being reused.
- **User:** Either a person or an application that operates a system in order to perform a task

A three-phase approach is suggested for the domain analysis [7] as shown in figure 1.1.

1.1.1Context Analysis

The domain analyst discusses with domain expert and users to set the constraints of the domain and set a proper scope for the domain. The analyst also collects information for performing the analysis.

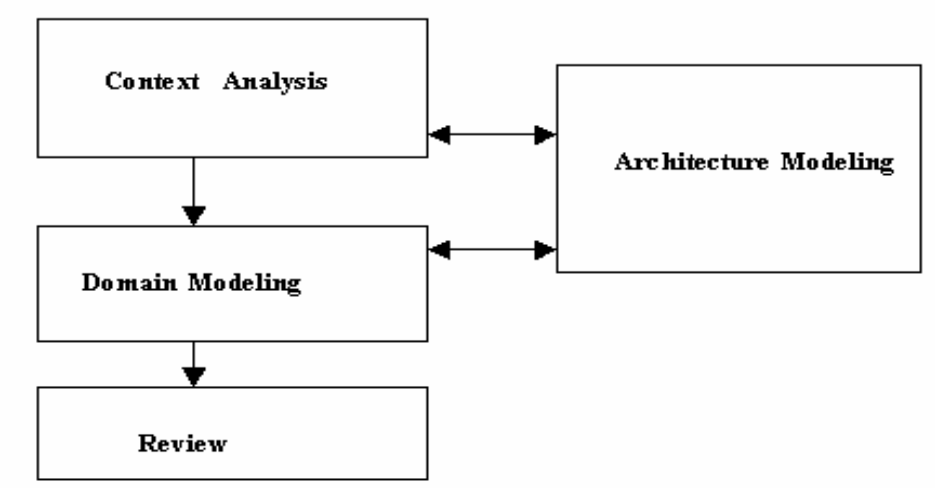

Figure 1.1 Domain Analysis Phases

1.1.2 Domain Modeling

The domain analyst makes use of information collected and other products of context analysis phase to create a domain model. The domain expert, system user and requirements analyst then review the model.

1.1.3 Architecture Modeling

By using domain model the domain analyst makes the architecture model and then this model is reviewed by domain expert, requirements and software engineer, user need not participate in this review.

1.2 Definitions of Domain Analysis [6]

- Identify objects and operations of a class of systems.
- Determine characteristics that satisfy the optimum domain.
- Activity that precedes system analysis.
- Specify the domain model of a library.
- Characterize a software domain to support reuse.
- Process of identifying, organizing and representing the relevant information of a domain.
- Identify, organize and model information to produce software requirements.
- Process in which a reusable software architecture and reusable code are defined.
- Identify domains in which reuse of certain experiences is effective.

There is no agreement on the definition of the term DA. As written above, each author's definition is strongly influenced by the expected outcome of the DA process or by the process itself. There are some methods, which prefer not to use the DA concept, as they think it is an ambiguous term due to the host of meanings attributed to it

1.3 Why Domain Analysis

Software reuse has long been argued as a means of improving software quality and increasing development productivity. In recent years, domain analysis methods have emerged as a systematic means of identifying and packaging reusable artifacts in an application area. Domain analysis methods (or approaches) such as Feature Oriented Domain Analysis (FODA) have been criticized for being too code-oriented. However, more recent domain analysis methods such as Organization Domain Modeling (ODM) claim to have a wider appeal as their scope encompasses requirements as well as code [4].

There is a sound argument for using domain analysis methods in domains, which are mature (e.g. where a set of legacy systems exist), reasonably stable (i.e. the domain is not always changing) and economically viable (i.e. new systems are anticipated in the domain). Most appealing, is the notion of producing a 'domain model' or a 'domain architecture' – which can be applied to all systems in the same domain.

Unfortunately, much of the existing documentation on domain analysis concentrates on the results of domain analysis studies, with little in-depth critique of the actual domain analysis process carried out. There is also a scarcity of detailed case studies describing the practical, day-to-day problems and decision-making, which are parts of a domain analysis exercise [4]. To the potential domain-engineer, with no prior knowledge of domain analysis, it would be hard to prepare for the practical issues involved in domain analysis from the literature alone. In the next chapter different domain models are discussed.

Chapter 2: Domain Analysis

2.1 Domain

A domain can be defined by a set of problems or functions that applications in that domain can solve. A domain can also be characterized by a common jargon for describing the concepts and problems in that domain.

The word “domain” can be used in several areas:

• Business area

• Problem domain

• Solution domain

• Area of knowledge with common vocabulary.

2.2 Vertical and Horizontal Domains

Domains can be divided into vertical and horizontal domains

2.2.1Vertical domain

In vertical domains, software systems are classified according to the business area [11]. Such systems are, for example, airline reservation systems, medical record systems, portfolio management systems, order processing systems, and inventory management systems.

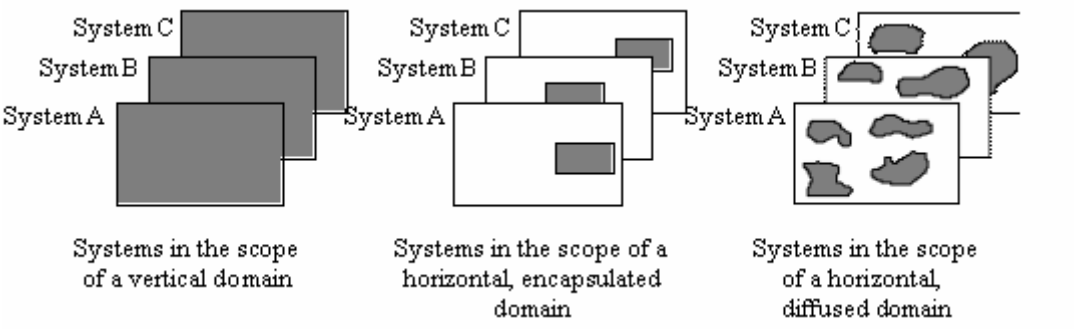

Figure 2.1 Vertical, horizontal, encapsulated, and diffused domains

2.2.2 Horizontal Domain

In horizontal domains, parts of a software system are classified according to their functionality [11]. Examples are database systems, container libraries, workflow systems, Graphical User Interface libraries, and numerical code libraries.

2.2.2.1 Encapsulated Domain

If the domain functionality is covered by a single subsystem in one system, the domain is called encapsulated. E.g. Library of java packages.

2.2.2.2 Diffused Domain

If the domain functionality is scattered through several subsystems of one system, the domain is said to be distributed or diffused domains. E.g. Collection of encryption algorithms.

2.3 Domain Analysis as the Foundation for Reusability [19]

Domain analysis is the systems engineering of a family of systems in an application domain through development and application of reusable assets. The domain analysis must be focused to address a family of systems in a particular application domain. The more definitive these families of systems can be defined, the more focused and precise the domain analysis activity can be. Domain analysis is, in fact, a system engineering activity [7].

Central to the concept of domain analysis is, of course, reusability. Thus, the end goal of a domain analysis activity is to develop a reuse library asset that will be used in the implementation of system instances in the domain family [24]. These assets in the library will include software components and generators, documentation, interface specifications, test plans, procedures and data. In addition, the domain model and the generic architecture are themselves valuable reusable assets.

2.4 Domain Analysis Approaches

Domain Analysis has following approaches [8]:

1. Producing and evaluating literature guides and subject gateways,
2. Producing and evaluating special classifications and thesauri,
3. Research on and competencies in indexing and retrieving information in specialties,
4. Knowledge about empirical user studies in subject areas,
5. Producing and interpreting bibliometric studies,
6. Historical studies of information structures and services in domains,

7. Studies of documents and genres in knowledge domains,
8. Epistemological and critical studies of different paradigms, assumptions and interests in domains.
9. Knowledge about terminological studies, LSP (languages for special purposes) and discourse analysis in knowledge fields,
10. Knowledge about and studies of structures and institutions in scientific and professional communication in a domain.
11. Knowledge about methods and results from domain analytic studies about professional cognition, knowledge representation in computer science and artificial intelligence.

Domain analysis entails developing a complete and rigorous domain model and associated generic architecture as a precursor to developing a set of reusable components for repeated application in developing systems in the domain.

2.5 Domain Analysis Phases

A three-phase approach is recommended for the domain analysis as shown in figure 2.2

2.5.1 Model the domain

Based upon knowledge of the domain and programmatic constraints, a requirements oriented domain model is developed and validated. This model is specifically analyzed to denote the necessary adaptations needed by envisioned future systems in the domain.

The domain modeling activity produces the following products [35]:

- **Domain planning document** that bounds the domain, scopes and plans the domain analysis activities, establishes guidelines and standards, and assesses the costs, risks and benefits of the effort. The document includes an initial assessment as to the level of reusability potential, potentiality achieved through a features analysis of the application domain.
- **Domain dictionary** precisely defining a controlled vocabulary for the domain analysis activities.
- **Validated domain model** representing the specification of application objects and their interaction. This model encapsulates the knowledge of commonalities and differences amongst the family of systems considered in the domain. The

domain model also includes a specification of additional complementary application requirements and constraints and guidelines for using the model.

- **Domain validation** products consisting of scenarios, simulations and executable specifications that are used to validate the correctness and completeness of the domain model.

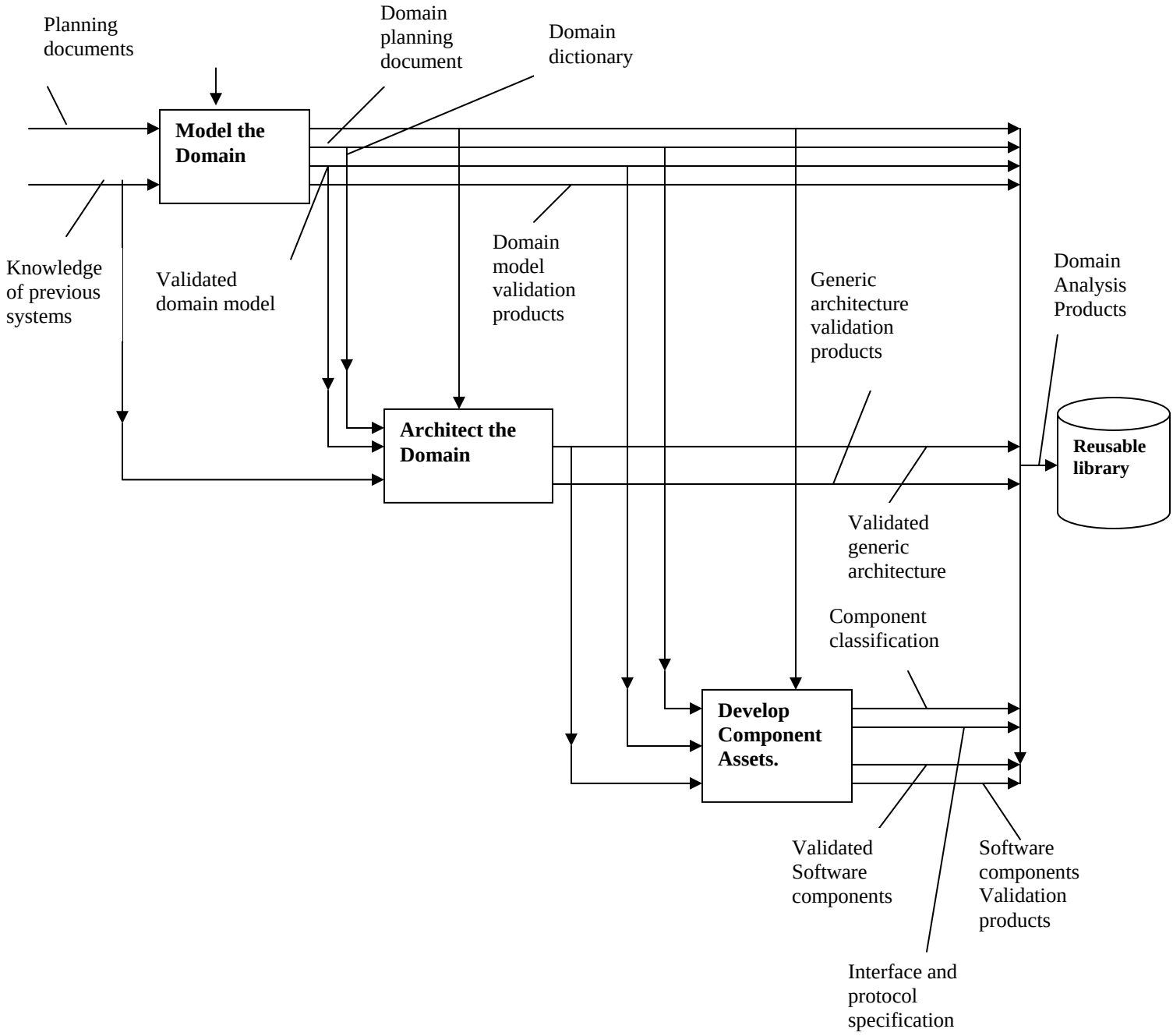

Figure 2.2 Detailed Domain Analysis Phases [7]

It is important that the domain analysis address the future system requirements of the designated family of systems. Analysis of previous systems is valuable for acquiring a basic understanding of the application domain [10]. However, the primary focus must be to understand the needs, requirements, commonalities and differences of future systems.

The best way to scope the domain precisely is to specifically enumerate the system instances that comprise the application domain family to be addressed. These systems instances will define the scope of variability that the domain analysis products will be expected to address. This will allow a more accurate cost-benefit assessment to drive the domain analysis decision process.

The primary focus of the domain modeling activity is to develop a validated domain model consisting of

- Rigorous requirements model of the application, serving as a model for commonality.
- Determination and specification of the adaptation requirements across the family of systems.
- Other complementary domain requirements and constraints (e.g., performance, fault tolerance, security, survivability)
- Tradeoffs and rationale for the domain model
- Guidelines for using the domain model

While most domains modeling activities focus on the commonality between system instances in the domain, it is more as important to understand the differences between system instances. Adaptation analysis is critical to deriving a generic architecture and component library that can adapt to future system needs. This analysis of required adaptation may be based upon anticipated mission, threat, environment, site, platform or technology changes. Trends observed from an analysis of changes or evolutions of previous systems may also prove helpful.

2.5.2 Architect the domain

A generic architecture is developed and validated for the family of systems, identifying those software components that are potentially reusable. Standard systems engineering disciplines are applied to ensure that the architecture is a feasible and satisfactory basis

for developing future systems. The domain analysis process is directed toward bounding and analyzing an application domain in order to produce a set of reusable domain analysis products. The goal of this phase is to transit the abstract domain model into a concrete generic architecture or family of architectures that form the blueprint for the domain family of systems [25]. The outputs of this phase are:

- **Validated generic architecture** and associated tradeoffs and systems analyses for the application domain family of systems. The generic architecture product includes guidance for applying it.
- **Generic architecture validation products**, specifically simulations, prototypes and partial implementations that demonstrate the architecture.

A generic architecture, or family of generic architectures, is derived for the application domain. The resulting generic architecture is composed of:

- Architectural specification
- Alternative analysis, tradeoffs and rationale for the generic architecture
- Guidelines for using the generic architecture

2.5.3 Develop software component assets

A set of reusable software components is built and validated to comply with the interfaces and protocols required by the generic architecture. These components are cataloged into the reusable component library for the domain.

The generic architecture defines the software components to be developed in this phase. This final phase produces the following products:

- **Component classifications** identify and categorize those software components that will be developed.
- **Interface and protocol specifications** document key interfaces to components and component sets and guidelines for using them.
- **Validated software components** are developed for adaptation and reuse in developing systems in the application domain. A software component includes the code, its adaptation mechanism, documentation, classification characterization, trade-offs and rationale, and guidance for reusing it.

- **Software component validation** products are developed, representing the test suites used for component validation. This may include test plans and procedures, test cases, test data, test results, proofs, or demonstrations.

2.6 Domain Analysis Models

The domain analysis is the activity that formally describes the commonalities and variabilities within a domain. The domain engineer capture and organize this information in set of domain models with the intent of making it reusable when creating new systems. The output of domain analysis is a domain model [9]: an explicit representation of knowledge about the domain.

The domain model consists of:

2.6.1 Domain Dictionary

Domain Dictionary provides and describes the terms concerning the domain [34]. Its purpose is to make communication among developers and stakeholders easier and more precise.

2.6.2 Context Models

A context model specifies the boundaries of the domain. The model considers both the commonalities and variabilities of the application in domain. The analysis include

- Description of the domain and its relation to other domains and a record of important decisions and alternatives
- Commonalities: A structured list of assumptions those are true for every member of family.
- Variabilities: A structured list of assumptions about family members which differ.
- Parameters of variation: A list of parameters that refine the variabilities.

2.6.3 Feature Models

It is a hierarchical decomposition of features. Feature models that also tell about which combination of features are meaningful can depict feature. Feature models provide notation for different kinds of features such as FODA- like features [34]. Details about various DA methods are discussed in next chapter.

Chapter 3: Domain Analysis Methodologies

By using domain analysis methods a more general perspective of domain analysis is taken by looking at the existing methodologies to analyze a domain. In general a domain analysis method cane be characterized by the process (steps to conduct the analysis), the product (usually domain models), and supporting tools. There are number of domain analysis methods but there is no category of these. Although some authors have compared different methods according to different criteria.

A domain analysis method should have two things:

- A domain theory, along with an anatomy of this theory both of which will appear in domain model.
- A process, which, if strictly held, will allow construction of the generic domain model.

The sequence of steps to be defined in the process will be clearly influenced by the form of the domain model. The question is, however, what should the domain model specify; domain model should set out the information required (now and in the future) to allow reuse in the domain. Traditionally, it has been accepted that the information required for modeling consisted of identifying entities, operations, events and relationships in the domain [31].

However, if DA is applied to other software elements, apart from software products (architectures, code, etc.) the modeling and the way models are built will have to be changed. This means that both the domain model and the model construction process will be very much influenced by the type of element to be reused.

3.1 Classification of DA Methods

In so far as DA methods depend on the type of element to be reused, all the methods will have to be classified so as not to compare pears with apples. So, DA methods will be classified according to the type of element they intend to reuse. Accordingly, there are:

- DA methods for software product reuse.
- DA methods for software process reuse.
- DA methods for software technology reuse.

- DA methods for software experience reuse.

DA methods for software product reuse have been more widely addressed. Despite the fact that the idea of reusing code evolved into the reuse of other kinds of software products, most of the effort that has gone into reuse research has addressed software product reuse. These DA methods can be divided into four subclasses, as listed below:

1. Methods for software component reuse.
2. Methods for asset reuse.
3. Methods for software architecture/design reuse.
4. Methods for software requirements reuse.

- **Software component reuse** can be seen as a search for components that supply the functionality needed by the reuser. The underlying idea is to build a software component library around the domain.
- **Asset reuse** claims to be more than just component reuse, but it is not clear what, as nobody has given a precise definition of what an asset is. The basic difference between a software component and an asset seems to be how they are obtained; while software components are produced from a mentality of code reuse, assets are architecture-driven. In both cases, code is the element to be reused (the architecture is never reused in the case of assets).

With regard to **software architecture/design reuse**, software architecture is one of the software elements to be reused during the software process (of course, there will be software components to support this architecture, but nothing is said in any of the approaches about how they should be built).

The ultimate step in software product reuse would be to try to bring reuse into the analysis phase of the software process, that is, **software requirements reuse**. If requirements are reused, there will be some software architecture to support these requirements, and also software components to support the architecture.

Not much research has been carried out on **software process reuse**. This kind of reuse deals with the construction of reusable software processes as a means of improving an organization’s software process.

One of the fundamental misconceptions regarding the nature of software and its creation is that principles, techniques, and tools are generally applicable, and there is no need to investigate their limits in different project.

Finally, there are DA methods for **software experiences reuse**. These methods try to reuse every useful experience in software systems development. It is not easy to determine experiences that are relevant to a given project context; that is, the kind of experiences that are applicable to the project.

The purpose of any DA Methods are:

- Improve reusable elements,
- Populate libraries of reusable elements,
- Integrate DA into the software process,
- Decrease adaptation costs, and
- Construct reusable elements

3.2 Common Domain Analysis Processes in All DA Methods

All the DA methods may appear at first glance to follow a different process for obtaining the domain model, but some early research performed showed that all DA methods follow some common process, as shown in figure 3.1, Definitions of theses common processes are as follows:

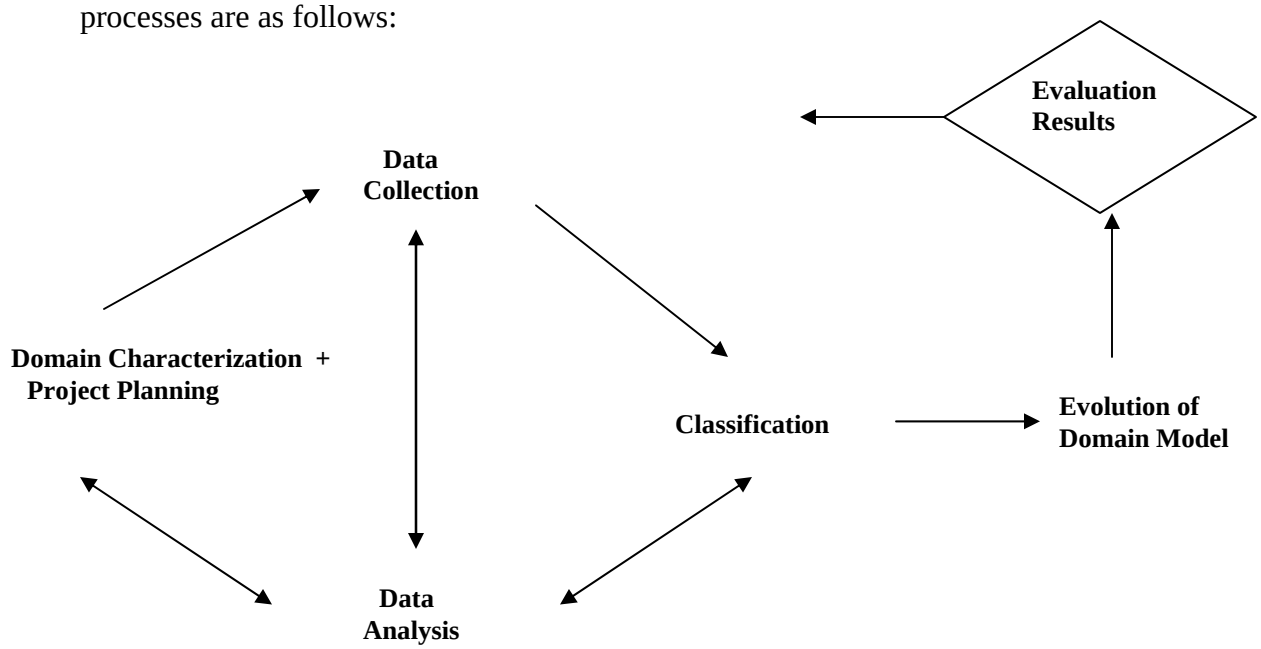

Figure 3.1 Representation of common processes followed by all DA methods [10].

3.2.1 Domain characterization and project planning: This is a feasibility analysis and planning phase [22]. Here the main characteristics of application in said domain is identified.

3.2.2 Data collection: This phase is driven by the modeling needs. The data collection sources may vary from experts to documents, etc.

3.2.3 Data analysis: The purpose of this phase is to build descriptions of reusable elements, identifying the similarities and differences between them.

3.2.4 Classification: The information modeled in the previous step is refined, clustering together similar descriptions, abstracting relevant common features from the descriptions in each cluster, adding new descriptions to existing or to new clusters, or reorganizing the clusters. Abstractions are organized into generalization hierarchies.

3.2.5 Evaluation of the domain model: This activity evaluates the domain model obtained, and any defects found are put right. Some methods include references to evaluation steps in their process, though they fail to explicitly specify tasks or performance objectives for the models.

Therefore, although it initially looked interesting, these above discussed parameter will not be taken into account for comparing different methods, as they all follow a similar process and, accordingly, this parameter does not discriminate. These researches performed also points out that not all the techniques used in Domain Analysis are Domain Analysis techniques. As a result, only the following used techniques seem helpful.

Analysis technique*:* Different techniques can be used for data analysis during subsidiary modeling, are:

- Object Oriented
- Functional (and data)
- Qualitative Analysis
- Case-based technology

Classification technique: There are a wide variety of techniques used in primary modeling. These are:

- Facets
- Features
- Capabilities
- Requirements
- Coad-Yourdon
- Partial patterns

Representation of variability*:* Not all the methods represent variability in the Domain in the same way. Some methods do not consider variations. All the different techniques identified for representing variability in the domain are:

- Combinations
- Compromises
- Specialization
- Refinements
- Relations
- Parameterization

Table 3.1 explains a more detailed list of Domain Analysis activities:

Domain Analysis major process components	**Domain Analysis activities**
Domain characterization (Domain planning and scoping)	• Select domain • Perform business analysis and risk analysis in order to determine which domain meets the business objectives of the organization.
	• Domain description • Define the boundary and the contents of the domain.
	• Data source identification • Identify the sources of domain knowledge.
	• Inventory preparation • Create inventory of data sources.
Data collection	• Abstract recovery • Recover abstractions

	• Knowledge elicitation • Elicit knowledge from experts
	• Literature review
	• Analysis of context and scenarios
Data analysis (Domain modeling)	• Identification of entities, operations, and relationships
	• Modularization • Use some appropriate modeling technique, e.g. object-oriented analysis or function and data decomposition. Identify design decisions.
	• Analysis of similarity • Analyze similarities between entities, activities, events, relationships, structures,
	• Analysis of variations • Analyze variations between entities, activities, events, relationships,
	• Analysis of combinations • Analyze combinations suggesting typical structural or behavioral patterns.
	• Trade-off analysis • Analyze trade-offs that suggest possible decompositions of modules and architectures to satisfy incompatible sets of requirements found in the domain.
Taxonomic classification (Domain modeling)	• Clustering • Cluster descriptions.
	• Abstraction • Abstract descriptions.
	• Classification • Classify descriptions.
	• Generalization • Generalize descriptions.
	• Vocabulary construction
Evaluation	• Evaluate the domain model.

Table 3.1 Common Domain Analysis process [6]

3.3 Feature Oriented Domain Analysis Method

The Feature-Oriented Domain Analysis (FODA) methodology resulted from an in-depth study of other domain analysis approaches [17]. Successful applications of various methodologies pointed towards those approaches, which focused on the process and

products of domain analysis. The feature-oriented concept of FODA is based on the emphasis placed by the method on identifying prominent or distinctive features within a class of related software systems. These features lead to the creation of a set of products that define the domain.

3.3.1 Foundations of the FODA Methodology

The FODA methodology was founded on a set of modeling concepts and primitives. These concepts and principles are used to develop domain products that are generic and widely applicable within a domain. The basic modeling concepts used in the creation of the domain products are abstraction and refinement. Abstraction is used to create domain products from generic applications in the domain. These generic domain products abstract the functionalities and designs of the applications in a domain. Abstracting away "factors" that make one application different from other related applications creates the generic nature of the domain products.

The FODA method advocates that applications in the domain should be abstracted to the level where no differences exist between the applications [13]. Refinements are used to both refine the generic domain products and to refine the domain products into applications. Once the abstraction of the applications in the application domain is completed, the factors that make each application unique are incorporated into the generic domain products as refinements of the abstractions. Specific applications in a domain may be developed as further refinements of the domain products by using the general abstraction as a baseline and selecting among alternatives and options to develop the application abstracting the applications in the application domain is accomplished by using the modeling primitives of aggregation or decomposition, generalization or specialization, and parameterization.

The FODA method applies the aggregation and generalization primitives to capture the commonalities of the applications in the domain in terms of abstractions. Differences between applications are captured in refinements. An abstraction can usually be refined in many different ways depending on the context in which the refinements are made [32]. Parameters are defined to uniquely specify the context for each specific refinement. The

result of this approach is a domain product consisting of a collection of abstractions and a series of refinements of each abstraction with parameterization. Understanding what differentiates applications in a domain is most critical since it is the basis for abstractions, refinements, and parameterization. Domain products are produced through a number of activities [31].

The following subsection discusses the activities of the FODA process and the models that are produced from the process.

3.3.2 FODA Process and Products

Three basic phases characterize the FODA process [18]:

- **Context Analysis:** defining the extent (or bounds) of a domain for analysis
- **Domain Modeling:** providing a description of the problem space in the domain that is addressed by software
- **Architecture Modeling:** creating the software architecture for implementing solutions to the problems in the domain

Figure 3.2 provides the structure of the FODA methodology and Table 3.2 summarizes the inputs, activities, and products of each phase in the FODA process and the relationships between their products.

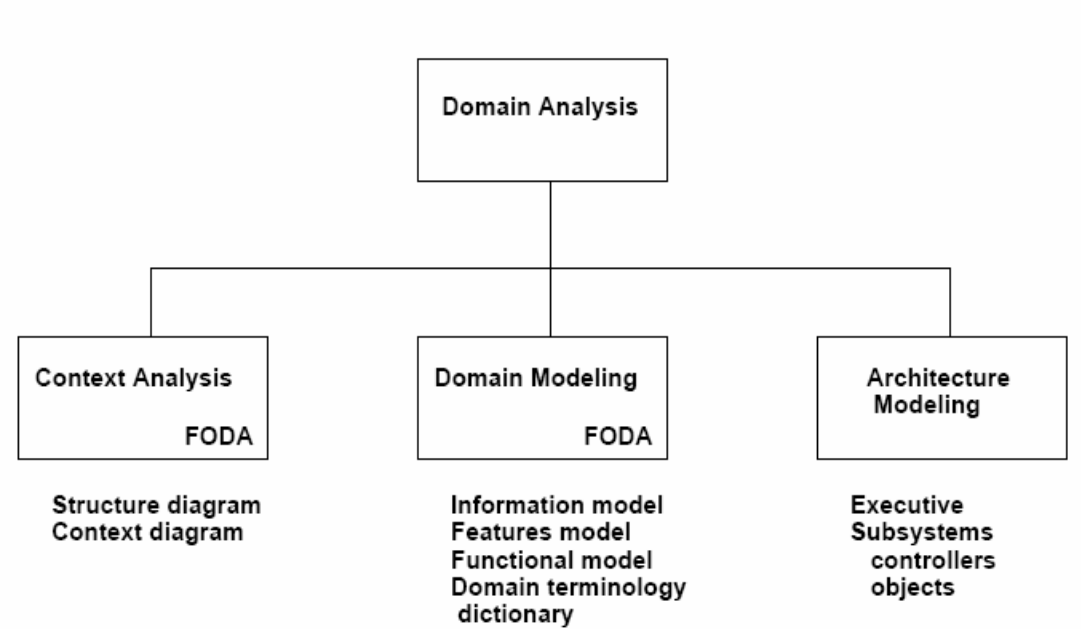

Figure 3.2 FODA Phases [12]

3.3.2.1 Context Analysis

Context analysis defines the scope of a domain that is likely to yield useful domain products. During the context analysis of a domain, the relationships between the "domain of interest" and the elements external to it are established and analyzed for variability. The kinds of variability to be accounted for are, for example, when applications in the domain have different data requirements and/or operating environments. The results of the context analysis, along with other factors such as availability of domain expertise, domain data, and project constraints, are used to limit the scope of the domain.

The product resulting from the context analysis is the context model [18]. This model includes a structure diagram and a context diagram. The structure diagram for this domain is an informal block diagram in which the domain is placed relative to higher-, lower-, and peer-level domains. Higher-level domains are those of which the domain under analysis is a part or to which it applies. Lower-level domains are those within the scope of the domain under analysis, but which are well understood. Any other relevant domains must also be included in the diagram.

The context diagram is a data flow diagram showing data flows between a generalized application within the domain and the other entities and abstractions with which it communicates [13]. One thing that differentiates the use of data flow diagrams in domain analysis from other typical uses is that the variability of the data flows across the domain boundary must be accounted for with either a set of diagrams or text describing the differences [28]. These products provide the domain analysis participants with a common understanding of:

- The scope of the domain
- The relationship to other domains
- The inputs/outputs
- Stored data requirements for the domain

Phase	Inputs	Activities	Products
Context Analysis	Operating environments, Standards	Context analysis	Context model
Domain Modeling	Features, Context model	Features analysis	Features model
	Application domain knowledge	Information modeling	Information model
	Domain technology, Context model, Features model, Information model, Requirements	Functional analysis	Functional model
			Behavioral model
Architectural Modeling	Implementation technology, Context model, Features model, Information model, Design information	Architectural modeling	Structured executive Subsystems model

Table 3.2 Summary of FODA Method

3.3.2.2 Domain Modeling

Domain modeling identifies the commonalities and differences that characterize the applications within the domain. The domain-modeling phase consists of three major activities. A brief description of each activity and its results is given below.

1. Features Analysis captures a customer or end user's understanding of the general capabilities of applications in a domain1. For a domain, the commonalities and differences among related systems of interest were designated as features and are depicted in the features model. These features, which describe the context of domain applications, the needed operations and their attributes, and representation variations, are important results because the features model generalizes and parameterizes the other models produced in this domain analysis [12].

The features model is the chief means of communication between the customers and the developers of new applications. The features are meaningful to the end users and can

assist the requirements analysts in the derivation of a system specification that will provide the desired capabilities. The features model provides them with a complete and consistent view of the domain.

2. Information Modeling captures and defines the domain knowledge and data requirements that are essential for implementing applications in the domain. Domain knowledge typically is information that is deeply embedded in the software and is often difficult to trace. Those who maintain or reuse software need this information in order to understand the problems the domain addresses. The information model may take the form of an entity-relationship (ER) model, a semantic network, or other representations such as object modeling.

Primarily the requirements analyst and the software designer to ensure that the proper data abstractions and decompositions are used in the development of the system use the information model [33]. The information model also defines data that is assumed to come from external sources.

3. Functional Analysis identifies the control and data flow commonalities and differences of the applications in a domain. This activity abstracts and then structures the common functions found in the domain and the sequencing of those actions into a model. Common features and information model entities form the basis for the abstract functional model. The control and data flow of an individual application can be instantiated or derived from the functional model with appropriate adaptation. The functional model is the foundation upon which the software designer begins the process of understanding how to provide the features and make use of the entities selected.

The domain modeling process also produces an extensive Domain Dictionary of terms and/or abbreviations that are used in describing the features and entities in the model and a textual description of the features and entities themselves [13].

The domain dictionary has been found to be one of the most useful products of a domain analysis. The dictionary helps to alleviate a great deal of miscommunication by providing the domain information users with:

- A central location to look for terms and abbreviations that are completely new to them

- Definitions of terms that are used differently or in a very specific way within the domain

3.3.2.3 Architectural Modeling

Architectural modeling provides a software solution for applications in the domain. An architectural model (also known as a design reference model) is developed in this phase and de-tailed design and component construction can be done from this model [12]. This architectural model is a high-level design for applications in a domain. It focuses on identifying concurrent processes and domain-oriented common modules and on allocating the features, functions, and data objects defined in the domain models to the processes and modules.

FODA does not have any specific process for requirements specification, verification and management.

3.4 Joint Object Oriented Domain Analysis Method

The Joint Object Oriented Domain Analysis method advocates the idea that software objects are more understandable and customizable than traditional functions and subroutines. The Joint Integrated Avionics Working Group (JIAWG) reuse subcommittee developed JODA [14].

JODA is the domain analysis part of the reuse based software development approaches defined by JIAWG. This approach also includes business and methodologies planning. Business planning identifies the high level domain to which the analysis approaches will be applied. JODA domain analysis defines the domain structure and requirements and captures them in domain models [33]. These domain models are implemented and stored as a repository of reusable software objects.

3.4.1 Foundation of JODA Methodology

Joint Object Oriented Domain Analysis, JODA, is based on the Coad-Yourdon Object-Oriented Analysis method (OOA). Business and methodology planning are the preceding phases of domain and application engineering processes. JODA is a technology-oriented method that focuses on reusable requirements. Domain services that are visible to

subsystems are listed as well as the dependencies of services that must be available from subsystems for the domain to meet its requirements.

The main contribution of JODA is the separation of domain engineering from application engineering and their interaction. JODA also emphasizes business and methodology planning as a starting-point of domain and application engineering, as well as the reuse-asset library as a part of domain engineering. The method is a comprehensive, “pure” DA method and can be applied straightforward, if the OOA method is used. However, more often a method has to be adapted to the needs of an organization,

The following subsection discusses the activities of the JODA process and the models that are produced from the process.

3.4.2 JODA Process and Products

Three basic phases characterize the JODA process:

- **Preparing the domain:** It concerned with collecting information about the domain under consideration either by interviewing domain experts or by reengineering existing system.
- **Domain Scoping:** It includes definition of the domain glossary, services, dependencies, and the development of high whole part, subject, and inheritance diagrams.
- **Modeling the Domain:** The last phase in JODA in modeling the domain by defining the object life histories and state event response, investigating operation scenarios, and packaging and grouping reusable objects.

JODA uses OOA/D instead of functional methods to cover the domain analysis phase. Object-oriented analysis techniques are used to define the structure and capture requirements. This domain analysis phase identifies what is reusable, how it can be structured, and how it can be reused [2]. The domain analysis activities within JODA are iterative. Details of three basic phases that characterize the JODA process is as follows:

3.4.2.1 Domain preparation

This activity is in charge of identifying and collecting source material, references and software artifacts linked to the domain. The result is a domain source material and

support from domain experts; this is help from experienced people. It concerned with collecting information about the domain under consideration either by interviewing domain experts or by reengineering existing system.

Phase	**Inputs**	**Activities**	**Products**
Domain preparation	Reengineering of Existing system, Standards	Preparing the domain	Domain Information
	Source Material	Identification and collection of source Material	A domain source material Support from domain experts
	References	Identification and collection of references	
	Software artifacts	Identification and collection of software artifacts	
Domain scoping	Domain glossary, Services, Dependencies, And the development of high whole part, subject, and inheritance diagrams	Bounding of Domain and its analysis	A top-level subject diagrams, Whole-part diagrams, Generalization-specialization diagram, Domain services, Domain dependencies, Domain glossary and textual description.
Modeling the Domain	Object life histories, State event response, Operation scenarios	Architectural modeling	Domain Model suitable for reuse

Table 3.3 Summary of JODA Method

3.4.2.2 Domain definition

It bounds the domain and the analysts use this domain to clarify what can be reusable. This phase generates a top-level subject diagrams, top-level whole-part diagrams, top-level generalization-specialization diagram, domain services, domain dependencies, domain glossary and textual description. Finally, the context for potential re-users is identified. It includes definition of the domain glossary, services, dependencies, and the development of high whole part, subject, and inheritance diagrams.

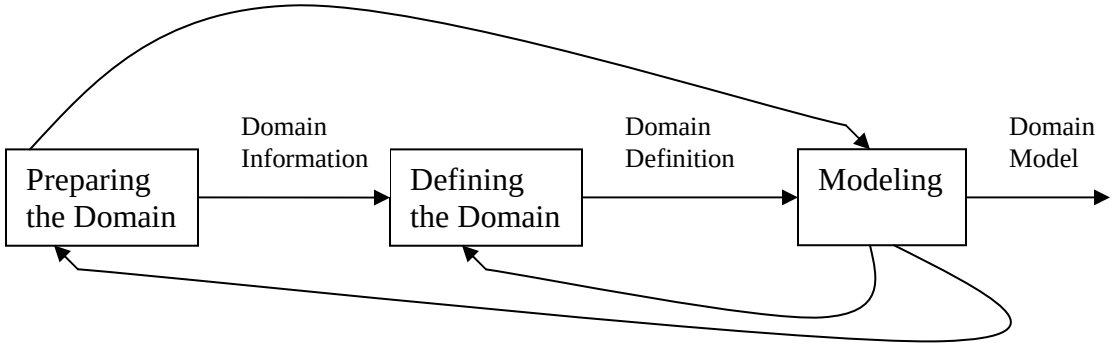

Figure 3.3 Phases in JODA Domain Analysis Method

3.4.2.3 Domain modeling

This last phase is the one, which extends from OOA and it, follows the next steps, definition of attributes and services, identification of objects and their relationships, identification, definition and simulation of the domain scenarios. Objects are abstracted and grouped to enable reuse. This phase in JODA is used to model the domain by defining the object life histories and state event response, investigating operation scenarios, and packaging and grouping reusable objects.

The JODA method is an object-oriented approach that produces scenarios. Scenarios relate special cases to the normal case. They describe the major threads of processing from a stimulus entering the system until it completes processing. Scenarios support the utilization of the domain model and components by relating external events to the services of the domain.

These three steps are iterated until the final domain model is obtained. It must be accurate enough to define the domain. This method has already by the JWG Reuse

Subcommittee's Domain Analysis Group to analyze an avionics domain (stores management).

3.5 Organization Domain Modeling (ODM)

Organization Domain Modeling (ODM) is a systematic approach to domain engineering developed by Mark Simos of Organon Motives over the past eight years. ODM was developed to address a gap in the domain engineering methods and processes available to reuse practitioners. Because work in domain analysis for software reuse emerged out of the software engineering research community, many of the methods were developed by technologists [15].

Organizational domain modeling prescribes a general method for conducting domain analysis in an organization. ODM offers a domain analysis method that is part of a larger domain engineering lifecycle [16]. Although ODM mainly focuses on organizational issues and transition to reuse discipline, it defines domain engineering technical activities.

3.5.1 Foundation of ODM Methodology

ODM was first prototyped as part of the design process for the Reuse Library Framework (RLF). Considerable support and collaboration in refining the ODM method has come from Hewlett-Packard Company, and from Unisys Corporation, as a prime contractor for the Advanced Projects Research Agency (DARPA) Software Technology for Adaptable and Reliable Systems (STARS) program [23].

ODM is structured in terms of a core domain-modeling life cycle, distinct from and orthogonal to the system development life cycle. The processes and work products of this core life cycle specifically address domain-engineering concerns; all other activities are allocated to sets of supporting methods. The core is represented by a process and work product model that can be tailored and instantiated in a variety of sequences and project structures.

Since each organization and to some extent each domain will have unique constraints and preferences, the goal is a core domain engineering method that can be integrated with a broad variety of supporting methods. Supporting methods include system modeling techniques and taxonomic modeling techniques (among other methods). In addition to

supporting methods, invoked at particular points in domain modeling, ODM can be extended with optional layers, which have pervasive impact across the entire life cycle. Because the core ODM process model excludes activities classified as supporting methods, it omits some steps conventionally considered part of domain engineering.

3.5.2 ODM Process and Products

ODM divides the domain engineering process into three different processes:

- **Plan Domain** This is the domain scoping and planning phase corresponding to Context Analysis in FODA
- **Model Domain**. In this phase the domain model is produced. It corresponds to Domain Modeling in FODA
- **Engineer Asset Base**. The objective of this phase is to scope, architect and implement an asset base that addresses the needs of multiple systems.

This method is useful for a wide range of organizations and domains and it may be integrated with a variety of software engineering processes, methods and implementation technologies. However, it does not have support for creating domain-specific languages and application generators. ODM is most successful when it is used to support domain-engineering projects for domains, which are mature, reasonably stable and economically viable [29]. Details of three basic phases that characterize the ODM process is as follows:

3.5.2.1 Plan Domain and Analysis

The phase focuses primarily on explicit descriptive and prescriptive analysis phases. “An explicit distinction is made between domain modeling activities that are descriptive, ‘as is,’ and prescriptive,’ to be.’ This distinction aims to prevent a modeler from unconsciously modeling aspects of legacy systems in terms of how they should be designed rather than how they are actually designed”. The stakeholder’s goals are reconsidered as critical points throughout the domain-modeling life cycle. According to these goals and the characteristics of the domains of interest, a domain has to be selected for the target project. This domain is called “domain of focus”. Finally, the domain is defined and it is also specified what is in and out of the domain, this is bound the domain.

Phase	Input	Activities	Products
Plan domain	Legacy Systems, Stakeholders Goal, Domain Sources.	Planning the Domain Bounding the domain	Validated set of Objectives
Architecture Modeling	Artifacts, Features, Past experiences Domain Definition	Creation of domain architecture, Identification of system features, Mapping of feature combinations onto the domain settings	Descriptive Model Prescriptive Model Features Model Domain Model Domain Dossier Domain Lexicon
Asset Implementation	Asset Base	Scoping asset base Architect asset base Implementing asset base	Domain Assets

Table 3.4 Summary of ODM Method

3.5.2.2 Architecture Modeling

This phase is concerned with gathering and documenting relevant domain information. A descriptive domain model is developed from legacy systems, artifacts, and past experiences. This descriptive model is transform into a prescriptive domain model that documents the features that the domain architecture will support. Domain architecture is then created and represented in a concrete and analyzable format. Once the information is integrated and the most relevant system features have been identified, the domain has to be described [33]. A domain model is produced based on the domain definition produced during the plan domain phase. This model represents the commonality and variability within the domain and tries to formalize the space of possible alternatives for individual systems in the domain. This model includes feature profiles that map specific features or

feature combinations onto the domain settings to which they may be applied. As well as the domain model, a domain dossier, domain lexicon, and feature models are generated.

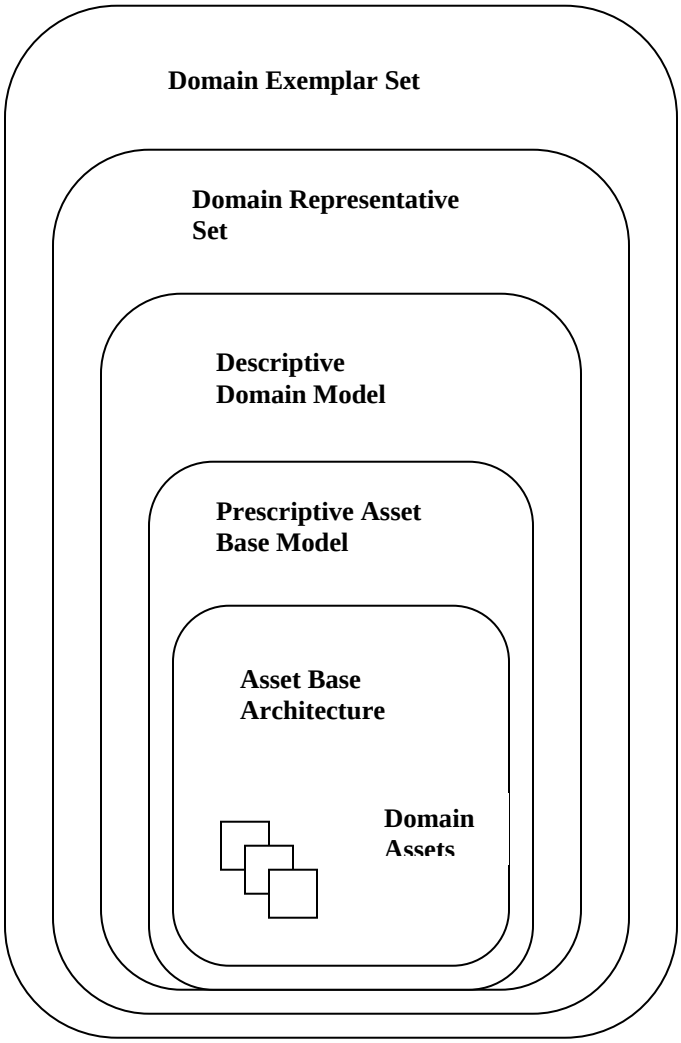

Figure 3.4 Iterative scoping steps in ODM [16]

3.5.2.3 Asset implementation

The objective of this phase is to scope, architect and implement an asset base that addresses the needs of multiple systems.

The scope sub-phase is in charge of correlating features with customers and selecting which will be implemented. Architect sub-phase determines the external and internal architecture constraints and the definition of architecture. Finally, the implementation sub-phase plans the implementation and implements the assets and infrastructure and

then finally domain asset are developed that conform to the architecture.

3.5.3 ODM Objectives

ODM works for following objectives:

1. Make the domain engineering process more systematic, formal, manageable and repeatable. In particular, document where key boundary negotiation and scoping activities take place, to avoid the phenomenon of ad hoc designs being imposed as domain models.
2. Ground domain engineering projects in a specific organization context. Support a view of modeling activities as not only describing the state of the organization, but also directly contributing to its evolution.
3. Maximize use of legacy artifacts and knowledge as:
    - An empirical basis for systematic scoping of domain definitions, models and asset bases.
    - A source of domain knowledge.
    - Potential resources to use in reengineering.
4. Reveal the hidden constraints embedded in legacy systems and artifacts and render them visible in domain models and assets.
5. Encourage exploration of maximum variability within the domain, including opportunities in various dimensions:
    - Reuse of artifacts from across the software life cycle.
    - Reuse of process as well as product assets.
    - Reuse in ways that facilitate use of static components and generative techniques, as well as hybrid strategies for asset implementation.
    - Reuse via definition of flexible, highly developed architectures for the asset base.
6. Provide effective strategies for selecting an intended scope of applicability for asset bases that is strategically appropriate for the performing organizations. In particular, apply techniques that limit the impact of combinatorial sets of choices in selecting range of variability.

7. Support evolution of the asset base, and the scale-up of the technology to support new kinds of organizations, organized around domains rather than around systems or products in a conventional sense.

3.6 Domain Specific Software Architecture (DSSA)

Domain engineering is the process of developing and implementing domain specific software architecture. A Domain-Specific Software Architecture (DSSA) is architecture for a specific domain; however, it should still be general enough to support several applications of the domain. A DSSA consists of common requirements and the process how to refine it. Thus, a DSSA denotes very much the same as product-line architecture.

A Domain-Specific Software Architecture (DSSA) has been defined as:

"An assemblage of software components, specialized for a particular type of task (domain), generalized for effective use across that domain, composed in a standardized structure (topology) effective for building successful applications"[17]

Or, alternately

"A context for patterns of problem elements, solution elements, and situations that define mappings between them.

3.6.1 Foundation of DSSA

The DSSA approach to Domain Engineering was developed under the Advanced Research Project Agency's (ARPA) DSSA Program. The DSSA approach emphasizes the central role of the concept of software architecture in Domain Engineering. The overall structure of the DSSA process is compatible with the generic process structure. Domain- Specific Software Architecture (DSSA) covers the entire software lifecycle required to develop a software system from which concrete architectures pertaining to that domain can be substantiated. The phases involved in the development of DSSA are domain engineering and application engineering. 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. This domain analysis process often involves several domain

"experts" who are intimately familiar with legacy systems of this kind or other aspects of the domain of interest.

Components of DSSA

1.Domain model

2.Reference requirements

3.Reference (parameterized) architecture

- Standardized architecture describing all systems in domain
- Focuses on fundamental domain abstractions

4.Supporting infrastructure/environment

5.Process/methodology to instantiate, refine, and evaluate

DSSA domain analysis method is more concerned with defining the models to be produced rather than the analysis process. These domain models are function, dynamic and object model.

3.6.2 DSSA Process and Products

The main work products of the DSSA process include the following:

- **Domain Model**: The DSSA Domain Model corresponds to the concept model (i.e. concept model in ODM or information model in FODA) rather than a full domain model.
- **Reference Requirements**: The DSSA Reference Requirements are equivalent to the feature model in. Each reference requirement is either mandatory, optional, or alternative. The DSSA Reference Requirements include both functional and non-functional requirements.
- **Reference Architecture**: DSSA Reference Architecture is an architecture for a family of systems consisting mainly of an architecture model, configuration decision tree, design record (i.e. description of the components), and constraints and rationale.

DSSA domain analysis method is more concerned with defining the models to be produced rather than the analysis process. These domain models are function, dynamic and object model.

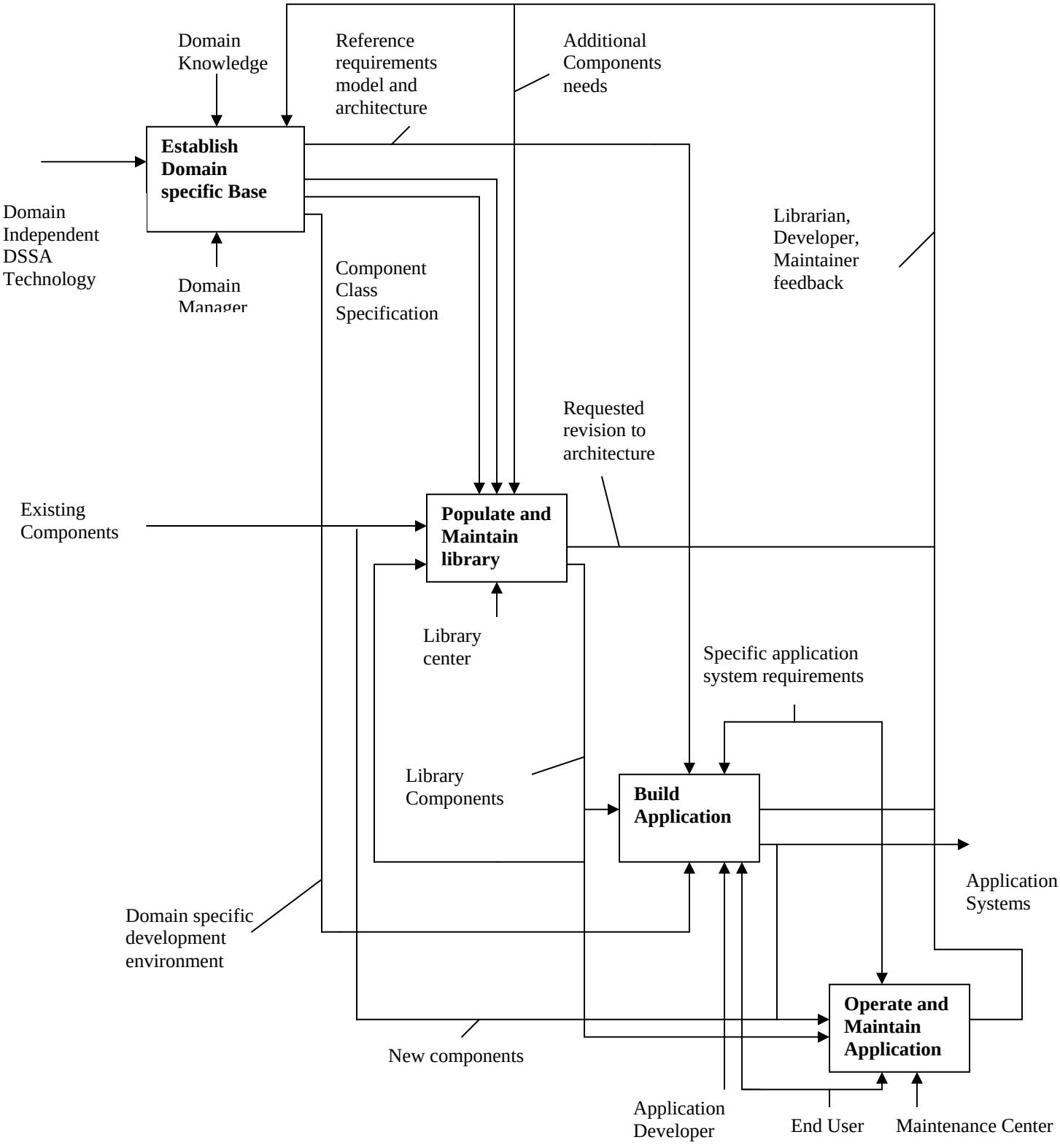

Figure 3.5 Activities in DSSA Life Cycle [27].

- **Function and Domain Models**

  Function and dynamic model describe the functional process in a manner similar to the structured analysis diagram. These diagrams include data flow diagram and control flow diagram. These models offer a hierarchical decomposition of the domain functionality.

- **Object Models**

  Object model are developed using the object modeling technique. These models include class diagrams containing class attributes, class methods and generalization and association relationship. The choice of object- oriented analysis diagrams in the DSSA application seems flexible.

As can be seen in the figure 3.5, the building of DSSA application systems is driven by reference requirements, reference architecture, and library components. The library components are developed and maintained by a DSSA library and are driven by the reference requirements, reference architecture, and component class specifications.

A DSSA consists of common requirements and the process how to refine it. Thus, a DSSA denotes very much the same as product-line architecture. DSSA domain analysis is comprised of:

- Domain analysis is used to capture and identify components and operations in a class of similar systems in a particular domain. Once the elements are identified, the relationships among them (inheritance, aggregations, etc) are defined and the dataflow and control flow among elements are captured. The result is a requirements document.
- Later, it is needed to identify constraints and software, hardware and performance requirements on the implementation.

Finally, the architecture is developed. The components are identified and the decision for mapping variabilities to changes is captured.

Phase	Input	Activities	Products
Establish Domain Specific Base	Domain Knowledge, Requested revisions to architecture, Domain independent DSSA technology base.	Model multiple Views, Establish consensus model, Allocate requirements to Reference Architecture, Specify reusable Component classes, Tailor environment to domain.	Reference requirements model and architecture, Component class Specifications Domain-specific development environment.
Populate and Maintain Library	Reference requirements model and architecture, Component class Specifications, Existing Component, Library components, Additional component Needs.	Develop acquisition strategy, Provide components, Install in DSSA library.	Requested revisions to architecture, Library components.
Build applications	Reference requirements Model and architecture, Specific application System requirements, Library components.	Develop Requirements, Design application System, Implement system,	Application Systems, Developer Feedback,
Operate and Maintain Applications	Specific application System requirements, Application systems, Fixed components.	Carry out application, Assess effectiveness, Maintain system.	Maintainer feedback, Application systems.

Table 3.5 Summary of DSSA Method

3.6.2.1 Establish Domain-Specific Base

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.

The domain manager performs this phase. The domain architect with participation of domain experts leads the activities [26]. The domain experts may come from different organizations managed by the domain manager, providing different user perspectives.

3.6.2.2 Populate and Maintain Library

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.

The library manager performs this phase [26].

3.6.2.3 Build Applications

This phase of the DSSA process will

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

The application developer performs this phase. The end user participates in reviews during the development

3.6.2.4Operate and Maintain Applications

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.

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

3.7 Domain Analysis and Design Process (DADP)

DADP is a process model for domain analysis and design. It is based on an object-oriented approach to analysis and design. The domain analysis process focuses on identifying commonalities and determining common object adaptation requirements (the differences among domain common objects are not described in terms of variability, but rather in terms of tailoring the information to particular needs).

Most of their diagrams are based on the Coad-Yourden Object-Oriented Analysis diagrams and method, but there is no specific requirement in their guidelines to use any particular method. In fact, they reference most of the current object-oriented analysis methods, including Bailin, Booch, Rumbaugh, and Schlaer-Mellor. This method is among the newest of the group and is likely to undergo still more changes.

3.7.1 Foundation of DADP

The domain analysis and design process (DADP) is a process established for review of areas, developed by Defense Agency. The Defense Information Sytems Agency (DISA) Center of Information Management (CIM) develops it. DADP identifies associations and adaptations of objects common. The DADP method takes a problem or solution space approach [4]. The domain is defined in the problem space where system engineer are mostly involved. System engineers define the problem and the system constraints. Software, hardware, and human factor engineer are then involved to define and evolve a solution space. The analysis aspect of DADP is concerned with identifying and defining problems within a group of related system in the domain. The design aspect of DADP is concerned with the development of domain- specific solutions in terms of architecture and reusable assets.

3.7.2 DADP Process and Products

DADP consist of four phases:

3.7.2.1 Identifying the Domain

The outcome of the step is a set of business models, definition of system capabilities, domain models, and definitions of external interfaces, knowledge reuse opportunities, groups of systems sharing sane capabilities, and some descriptions and documentation of current systems and anticipated systems.

Phase	**Input**	**Activities**	**Products**
Identifying the Domain	Definition of domain, Domain Knowledge, Definitions of external interfaces.	Identification of system capabilities, Descriptions of current systems,	A set of business models, Knowledge reuse opportunities, Documentation of current systems.
Scoping the Domain	System constraints, Domain Sources.	Identification of opportunities for reuse among systems.	Documentation of domain knowledge and team experiences
Analyzing the Domain	Problem space information, Common objects adaptation requirements.	Analyzing the problem space information, Identification of Commonalities.	Domain models, Validated domain models.
Designing the Domain	Domain models	Analyzing solution space. Implementations of domain objects.	Common designs, Implemented domain objects.

Table 3.6 Summary of DADP Method

3.7.2.2 Scoping the Domain

The domain is scoped with more than three systems in mind. In this phase, the domain analyst also identifies opportunities for reuse among systems in the same domain and

reuse across other domains. The domain knowledge and team experiences are also documented.

3.7.2.3 Analyzing the Domain

This step involves analyzing the problem space information. Commonalities and common objects adaptation requirements are identified. Domain models are constructed and verified.

3.7.2.4 Designing the Domain

At this step the solution space is addressed by providing solution in terms of common designs and implementations of domain objects.

The DADP advocates an object oriented technology approach where most of the diagrams are based on the old object oriented analysis models of Coad-Yourdon. However, no particular process is emphasized. The main advantage of this method is that it corrects much of the vagueness in the STARS process, particularly in domain model construction and definition.

Chapter 4: Problem Statement

Domain analysis is "the process of identifying, collecting, organizing, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain". Domain analysis is a process by which we are able to exploit commonalities in applications in the domain, capture experiences, and identify variabilities.

Domain analysis can be seen as an extension of the conventional requirements analysis. The primary advantage of domain analysis is that it provides flexibility. Domain analysis is suited for suitable, mature, and well-understood domains. However it suffers some barriers due to the complexity and huge amount of time required for development.

Good domain analysis depends upon several kinds of skills, but by and large, a domain analysis method has to issue three sets of issues:

1. How to organize domain analysis activities into systematic and controllable process.
2. How to integrate them into the normal application development process in such a way that both benefit, and in a way that optimizes the use of the resources of the organization.
3. How to find widely useful components, and how to package them in such a way that they are easily usable.

The first two sets of issues are more organizational and process-oriented, while the third is purely technical.

The problems, which are addressed in this thesis, are as follows:

4.1 How to compare domain analysis methods on the basis of common analysis steps, main product and various reuse methods used in various domain analysis methods steps.

4.2 When to choose which domain analysis method depending upon the type of the problem.

4.3 Identification of quality attributes for various domain analysis methods.

Chapter 5: Proposed Solution

Proposed solutions for the above problems are described as follows:

5.1 Comparison of Domain Analysis Methods

	FODA	**DSSA**	**JODA**	**ODM**	**DADP**
Base Model	**ER**	**ER +OO**	**OO**	**OO**	**OO**
Analysis Steps	1. Context Analysis 2. Domain Modeling 3. Architecture Modeling	1.Define scope of the domain. 2. Refine domain specific requirements 3. Refine domain Specific Design Implement 4. Develop domain Models 5. Gather Reusable Work Products	1. Prepare Domain 2. Define Domain 3. Model Domain	1. Define the Domain 2. Acquire Domain Information 3. Develop Descriptive Model 4. Refine Domain Models 5. Scope the Asset Base 6. Architect the Asset Base 7. Implement the Asset Base	1. Identifying the Domain 2. Scoping the Domain 3. Analyzing the Domain 4. Designing the Domain
Main Product	1. Domain Model 2. Feature Model	1. Domain Model 2. Domain Specific Software Architecture 3. Domain Reusable Component	1. Domain definition 2. Domain Model	1. Descriptive Domain Model 2. Prescriptive Domain Model 3. Domain Architecture	1. Business Model 2. Domain Models
Reuse Methods	1. Feature Model 2. Reuse library	1. Reusable Components	1. Domain Models 2. Active Repository	1. Asset reuse	1. Reusable Architecture 2. Design Reuse

Table 5.1 Comparison Table for Domain Analysis Methods

In table 5.1 comparison of domain analysis methods is done based on their base model, Analysis steps and Main product produced and reuse methods. This table shows that generic, common steps for domain analysis exist and the entity relationship and object oriented technology is applied to domain analysis. Reuse methods are not only reusable domain models but also reuse guidance, or reuse opportunities.

5.2 Choosing of Domain Analysis Methods depending upon type of the Domain

Table 5.2 below describes the use of domain analysis methods where they can be applied. These are not the only areas where a particular method can be applied. Research on domain analysis in recent years has produced many approaches.

	Area where Domain Analysis Methods can be applied	**Types of Methods Used**	**Domain**
1.	Family-oriented software engineering	FODA	Functionality
2.	Object Oriented Software engineering	JODA	Functionality
3.	Diverse organizations and implementation technologies	ODM	Functionality
4.	Domains based on Commonalities and Differences	DSSA	Functionality
5.	Opportunistic Domain engineering Approaches	DADP	Functionality

Table 5.2 Table for Domain Analysis Methods based on their Use

5.3 Identification of Quality Attributes for Various Domain Analysis Methods.

Various quality attributes of product analysis must also be included as common requirements in domain analysis [20].

The key quality attributes of the domain analysis of a product are summarized as follows:

- **Reusability** - The domain analysis must provide the foundation for systematic, large-scale reuse of system assets including requirements, architecture, components, test cases and documentation.

- **Scalability** - Systems can be scaled to execute on a single laptop computer supporting a single Input/Output (I/O) interface to a large, distributed, multiprocessor system capable of supporting many interfaces. Scaling the system can require configuration changes but does not require system software modifications.
- **Portability** - The use of open system standards is strategically implemented to the maximum extent feasible. The system provides insulation from the operating system and Commercial-Off-The-Shelf (COTS) solutions for portability. Portability is measured by the amount of effort required to re-host the system on different platforms [21]. Generally, increased portability should not be acceptable at the expense of increased recurring costs (maintenance, training, etc.).
- **Modifiability -** The system is designed to maximize flexibility (accommodate Change). Modifiability includes or is related to extensibility, maintainability and modularity, and is greatly affected by software coupling and cohesion. Changes can occur as the result of new COTS versions or products, new or improved standards, new requirements, or defect fixes. The measure for modifiability will mostly be how easily changes can be made and whether changes can be localized with little ripple effect to the system at large.
- **Performance** - Performance is the measure of latency and throughput.
- **Security** - Access to the system including its constituent components, capabilities, and data products is strictly controlled and the systems are protected under all circumstances.
- **Usability -** The system is user-friendly providing an intuitive environment through conceptually consistent interfaces without unnecessary complexity. Usability includes the ability to readily configure the system.
- **Dependability -** This is the confidence that reliance can justifiably be placed in the services provided by the system. The measures of product line dependability emphasize availability, reliability and the system's continuity of service. The quality attributes are often very interrelated and highly affected by architecture and implementation decisions. It is important to understand

their interrelationships, prioritize and develop a strategic plan for realizing each attribute, and document the necessary requirements to assure

None of the domain analysis method focuses on quality attributes which are actually essential to be included in domain analysis methods, so there is a need to identify these quality attribute in above discussed domain analysis methods.
List of Quality Attribute for domain analysis methods, which are discussed in report, are as follows:

5.3.1 Quality Attributes For Feature Oriented Domain Analysis Method (FODA)

FODA emphasize on identifying features that characterize a domain and hence gives the approach its name. In FODA feature is considered as a user visible functional requirement [30] e.g. in mobile phone F.M radio is considered as one of the feature.
List of Quality Attributes for features:

- **Modifiability:**
  Architecture of domain supports the variability and commonalities in feature, so there is need of change management.
- **Compatibility:**
  The extent to which the feature works well with associated products or systems.
- **Reliability:**
  The capability of a feature to maintain its level of performance under stated conditions for a stated period of time.
- **Portability:**
  The ability of a feature to be transferred from one product to other.
- **Understandability:**
  Modeling features need to understand their dependencies, and their function.
- **Usability:**
  It is ease of use of features.
- **Dependability:**
  Features are interdependent, like in mobile phone music player and playing ring tones.

- **Security:**

  The extent to which features of a product can be protected from internal intrusion.

5.3.2 Quality Attributes For Joint Object Oriented Domain Analysis Method (JODA):

The joint object oriented domain analysis method advocates the idea that software objects are more understandable and customizable than traditional functions and subroutines. So there is need to identify the quality attributes of objects under domain.

Quality attribute for objects:

- **Reusability:**

  Domain model produced in JODA is used to produce reusable elements or objects.

- **Stability:**

  How stable are the objects to be used for reuse.

- **Coupling:**

  It is measure of the strength of association established by a connection from one object to other.

- **Usefulness:**

  How efficient is the object to be reused.

- **Cohesion:**

  How effectively objects work together to provide well-bounded behavior.

5.3.3 Quality Attributes For Organizational Domain Modeling (ODM):

Organizational domain modeling prescribes a general method for conducting domain analysis in an organization. ODM mainly focuses on organizational issues and transition to reuse discipline; it defines domain engineering technical activities.

Quality attributes considered for an organization:

- **Modifiability:**

  Ability of organizational domain change after deploying it.

- **Productivity:**

How useful is the domain identified for organization.

- **Security:**

  There is need to maintain the information integrity of organization

- **Scalability:**

  When the organization change in size to meet the user's needs or to accommodate the increasing number of services**.**

- **Modifiability:**

  There is always a need of addition of information in domain or removing of some content or updating of domain, so there is need of change management.

- **Audit ability:**

  With ever increasing need of organization to comply with business and regulatory legislation, so there is a need to audit the organization for compilation

5.3.4 Quality Attributes For Domain Specific Software Architecture (DSSA)

Domain Specific Software Architecture is multi point solution to a set of application specific requirements. The DSSA process is a software life cycle based on the development and use of domain-specific software architectures, components, and tools.

List of quality attributes for DSSA:

- **Functionality:**

  The ability of the architecture to do the work for which it was intended.

- **Modifiability:**

  The ease with which the architecture can accommodate changes to its software.

- **Conceptual integrity:**

  The integrity of the overall structure, which is composed from a number of small architectural structures and components.

- **Correctness:**

  Accountability of architecture to satisfy all requirements of the system.

- **Completeness:**

Whether the maturity of architecture model reached a level from where implementation can be start.

5.3.5 Quality Attributes For Domain Analysis and Design Process (DADP)

The DADP method takes a problem or solution space approach [4]. The analysis aspect of DADP is concerned with identifying and defining problems within a group of related system in the domain. The design aspect of DADP is concerned with the development of domain- specific solutions in terms of architecture and reusable assets.

Quality attributes for DADP Method:

- **Construct ability:**

  The extent to which the design of the product enables assembly or construction.

- **Expandability:**

  The ease with which the product can be modified or extended to encompass additional functionality

- **Flexibility:**

  How well the design can be used or applied in other domain.

- **Reusability:**

  Does the design of architecture and asset produces support reuse.

Chapter 6: Conclusion and Future Scope

In this thesis report, we have tried to summarize the common basic analysis steps, main product and various reuse methods used in various domain analysis methods. The common steps followed in all domain analysis methods are domain characterization, data collection, data analysis and classification. Some of areas where these methods can be applied are family oriented software engineering, object oriented engineering, diverse organizations and implementation technologies, domains based on commonalities and differences.

After the detailed study of domain analysis and domain analysis methods it is concluded that there is no single “best” domain analysis approach for all types of product line software development. An organization should choose the one that best suits their software process needs, existing software base, and business objectives. A summary of domain analysis and domain analysis methods is given in this thesis report. Also the quality attributes for DA methods are discussed, which a particular DA method should take care when applied for a particular domain. This shows the efficiency of DA methods incorporating the quality in domain analysis.

More quality attributes may be added to each DA method and steps can be improved to achieve these.

References

[1] Rubén Prieto-Díaz, “Domain Analysis An Introduction”, The Contel Technology Center, Software Engineering Notes, Vol. 15, Page 47, Apt 1990.

[2] Eduardo Santana de Almeida, Alexandre Alvaro, Daniel Lucrédio, Vinicius Cardoso Garcia, Silvio Romero de Lemos Meira1, “A Survey on Software Reuse Processes”, Federal University of Pernambuco - Recife Center for Advanced Studies and Systems, Federal University of São Carlos, Brazil, IEEE, 2005.

[3] By Carma McClure, “Software Reuse (A standard- Based Guide)”, John Wiley & Sons Publishing, ISBN: 076950874X.

[4] Andrea Valedo, Giancado Succi §, Massimo Fenarolit, “Domain Analysis and Framework-based Software Development”, Software Production Engineering Lab (LIPS), Department of Communication, Computer and System Sciences, University of Genova, Genova (GE), Italy.

[5] Kelly, W. Lam and B.R. Whittle, “A domain analysis case-study from avionics”, Diary of a domain analyst Rolls-Royce University Technology Centre University of York, YO1 5DD, UK.

[6] X. Ferré S.Vegas Facultad de Informátic Universidad Politécnica de Madrid Campus de Montegancedo, “An Evaluation of Domain Analysis Methods”, s/n, 28660 Madrid.

[7] Edward R. Comer, “Domain Analysis: A System Approach to Software Reuse”, IEEE 1990.

[8] Hjørland & Albrechtsen, “Domain-analysis as a new paradigm for Library and Information Science (LIS)”, 1995.

[9] Dipl.-Inf. Krzysztof Czarnecki, “Generative Programming Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models”, October 1998.

[10] Eduardo Santana de Almeida, Jorge Cláudio Cordeiro Pires Mascena, Ana Paula Carvalho Cavalcanti, Alexandre Alvaro, Vinicius Cardoso Garcia, Silvio Romero de Lemos Meira1, Daniel Lucrédio, “The Domain Analysis Concept Revisited: A Practical Approach”.

[11] Maarit Harsu, “A survey on domain engineering”, Institute of Software Systems Tampere University of Technology.

[12] Pietu Pohjalainen, “Feature Oriented Domain Analysis Expressions”, Department of Computer Science, University of Helsinki.

[13] Hafed Mili , Ali Mili, Edward Addy “Reuse based software engineering”, Wiley Publishing, ISBN: 0-471-39819-5, 2001.

[14] CS 578 “Software Architecture--Domain-Specific Software Architecture (DSSA)”, February 2003.

[15] Robert Holibaug, “Joint IntegratedAvionics Working Group (JIAWG) Object-Oriented Domain Analysis”, Version 3.1, November 1993.

[16] Mark A. Simos-Organon Motives, “Organization Domain Modeling Domain Modeling Life Cycle”, 1995.

[17] Boerstler J., “Feature Oriented Classification for Software Reuse,” In Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering, SEKE ’95, KSI Knowledge Systems Institute, page 204–211, 1995.

[18] Kyo C. Kang,Sholom G. Cohen,James A. Hess,William E. Novak, A. Spencer Peterson, “Feature-Oriented Domain Analysis(FODA) Feasibility Study”, November 1990.

[19] Prieto-Diaz R., and Freeman, P., “Classifying software for reusability”, IEEE Software Vol. 4, pp 6-16, 1987.

[20] Leire Etxeberria, Goiuria Sagardui, Lorea Belategi, “Modelling Variation in Quality Attributes”, Computer Science Department University of Mondragon, Spain.

[21] Dr. Linda H. Rosenberg, Unisys Government Systems Goddard Space Flight Center, Lawrence E. Hyatt Software Assurance Technology Center, Goddard Space Flight Center, “Software Quality Metrics for Object-Oriented Environments”.

[22] Mônica Zopelari Roseti, Cláudia Maria Lima Werner, “A Knowledge Acquisition Systematic within the Domain Analysis Context”, COPPE/UFRJ – Computer Science Department Federal University of Rio de Janeiro RJ – Brazil, 1991.

[23] Mark Simos—Organon Motives, “Where the Rubber Meets the Road: Applying Organization Domain Modeling (ODM) on the STARS Army/Unisys Demonstration Project, 1995.

[24] Dr. Rubén Prieto-Díaz, Dr. Bill Frakes, Dr. Christopher Fox, “DARE A Domain Analysis and Reuse Environment”, Defense Advanced Research Projects Agency Defense Small Business Innovation Research Program ARPA, July 31, 1995.

[25] Paul C. Clements, “From Domain Models to Architectures”, Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213.

[26] Will Tracz, Lou Conglianese, Patrick young, “A Domain Specific Software Architecture Process outline”, IBM Corporation, Federal Systems Company, NY.

[27] James W. Armitage, “Process Guide for the DSSA Process Life Cycle”, CMU/SEI-93-SR-21 December 1993.

[28] Matthias Riebisch, Kai Böllert, Detlef Streitferdt, Ilka Philippow, “Extending Feature Diagrams With UML Multiplicities”, Ilmenau Technical University, Integrated Design and Process Technology, IDPT-2002**,** Printed in the United States of America, Society for Design and Process Science, June, 2002.

[29] M. Simos, “Organization Domain Modeling and OO Analysis and Design: Distinctions, Integration, New Directions”, Pages 119-123, ISBN: 1790-5117, 2008.

[30] S. Jarzabek, B. Yang and S. Yoeun, “Addressing quality attributes in domain analysis for product lines”, ISSN 0104-6500, Mar. 2008.

[31] Elena Alaña, Ana Isabel Rodríguez, “Domain Engineering Methodologies Survey” GMV Aerospace And Defence S.A, Isaac Newton 11, July 2007.

[32] Robert W. Krut, Jr., “Integrating 001 Tool Support into the Feature-Oriented Domain Analysis Methodology”, Technical Report CMU/SEI-93-TR-11 May 1993.

[33] Dipl.-Inf. Krzysztof Czarnecki, “Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models”, Department of Computer Science and Automation Technical University of Ilmenau, October 1998.

[34] Maarit Harsu, “A survey on domain engineering”, Institute of Software Systems Tampere University of Technology.

[35] Arango, G., R. Prieto-Diaz: “Domain Analysis Concepts and Research Directions”, IEEE Computer Society Press, 1991

List of Publications

Aman Jatain, Shivani Goel, “Comparisons of Domain Analysis Methods in Software Reuse”, accepted to publish in the International Journal of Information Technology & Knowledge Management Vol-II, Issue-II of Dec. 2009.