Odyssey: A Reuse Environment based on Domain Models[1]

Regina M. M. Braga Cláudia M. L. Werner Marta Mattoso
{regina, werner, marta}@cos.ufrj.br
COPPE/UFRJ - Computer Science Department
Federal University of Rio de Janeiro
Caixa Postal 68511 – CEP. 21945-970
Rio de Janeiro – Brazil

Abstract

This paper presents a Reuse based Software Development Environment that provides support to Component-Based software Development (CBD) within certain domains, named Odyssey. Object-oriented frameworks, software architectures, artificial intelligence techniques, domain engineering, , and mediators are some of the technologies used by Odyssey.

Keywords*: Reuse based Software Development Environments, Component-based Development, Domain Engineering, Object-Orientation, Software Architecture, Patterns, Mediators.*

1. Introduction

Reuse is a promising way to help improving software development. One of the most encouraging reuse techniques available is Component-Based software Development (CBD). CBD exploits interrelations between preexisting components and reuse of components that have been exhaustively tested to reduce complexity and costs of software development [12].

The success of CBD depends on the application of reuse techniques in all phases of the development process. Therefore, domain concepts that are considered to be reusable in initial development phases must be closely related to code components that will be used during application implementation. A reuse environment based on domain concepts can help in the effective application of reuse during software development, since it provides methods, tools, and procedures that are adequate for the specification of domain models and applications. There is no environment to our knowledge that is capable of addressing all these aspects together. Works found in the technical literature [9], [10], [11], [15], generally concentrate on one aspect or another (see section 7 for details).

The Odyssey Environment provides support to the whole development cycle from conceptual models to component implementation, including the following elements [20]: *Domain models* that are representations of different domain aspects found in various abstraction levels; *Domain Engineering* (DE); Adoption of a *method* to systematize the DE process; and *Mediation layer*, providing a direct representation, storage and management of domain model information.

In this paper, we present the main aspects involved in the specification of Odyssey. In section 2, we provide an overview of the environment. In section 3, we describe the representation of the different levels of domain models (i.e., conceptual, architectural, and implementation). Tools that permit users to specify and use domain models are presented in section 4. A brief description of a DE method, and the use of a mediation layer are presented in sections 5 and 6, respectively. Section 7 discusses some related works and, finally, in section 8, we conclude our work.

2. An Overview of Odyssey

Odyssey has been conceived as a framework where *conceptual models*, *architectural models* and *implementation models* are specified for previously

[1]This work is a result of two research projects of COPPE/UFRJ, sponsored by the CNPq agency. One involves research on Domain Oriented Software Development Environments, and the other deals with distribution and parallelism in OO databases.

selected application domains, as shown in Figure 1. *Conceptual models* are represented by *use cases*, *feature models*, *domain and analysis pattern systems*, and *OO models* (using a UML notation), as presented in section 3. *Architectural and Design pattern systems*, in conjunction to *OO models*, are used to represent architectural models. The *implementation model* is represented by the *reuse components set*. These models are specified and further modified according to activities of a DE method, based on a DE process (see section 5). To accomplish this, *domain tools* are available to support the specification and evolution of domain knowledge. Also, all stored domain models are preserved in a distributed and heterogeneous way using the mediation technology. The mediation layer is a key feature of the environment, since it provides a uniform representation and manipulation mechanism. The main objective of this layer is to provide the integration of information from various domains that are stored in heterogeneous and distributed data sources, in a way that the user has access to it in a transparent and uniform way.

The main users of Odyssey are domain engineers, domain specialists, and software engineers who are responsible for the development of applications within that domain. Domain engineers and specialists use the environment mainly to specify and enlarge domain concepts. Software engineers use it to gain an understanding of the application domain and reuse this knowledge during the specification of his/her application.

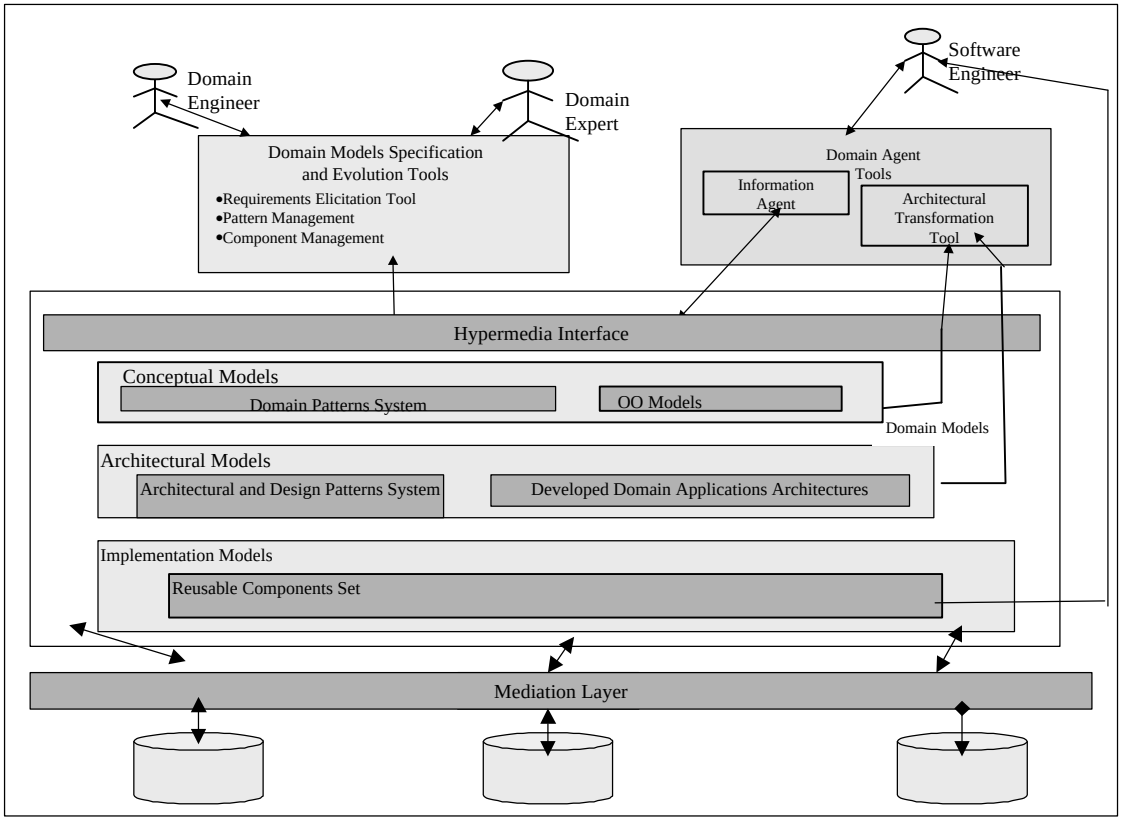

Figure 1 - Reuse Environment Diagrammatic View

3. Representation of Domain Models

As mentioned before, Odyssey considers three different kinds of models: *conceptual models*, *architectural models*, and *implementation models*.

Regarding the representation of *conceptual models*, it is important to pay special attention to the understanding and recognition of domain concepts and functionalities by Odyssey users. Thus, the means for expressing them is very important. Functionality is important for the creation of reusable components [22], [12], [27]. Domain concepts are equally important since they provide users with an understanding of the domain as a whole, moreover facilitating the comprehension of interactions among use case internal

types[2]. The main conceptual models used by Odyssey are:

- *Domain Use Cases and related OO models*: Use Cases are mainly used to capture basic functionalities of the domain in a way that it should be possible to derive other OO models. The use case model is the most used model to elicit domain information [12]. In fact, the information is elicited using scenarios that instantiate use case templates. Based on these domain use cases, it is possible to derive the next model that is essential for a complete understanding of the domain.
- *Feature Model:* It presents, in an abstract level, relationships between main concepts of the domain. This model is based on the feature diagram of FODA [3], [28]. It can be seen as an evolution of the original FODA's feature diagram, having added some concepts used in ontology for information systems research [24]. By doing so, at the conceptual level, the Odyssey Feature model provides two main views: the domain concepts model (represents main domain concepts and relationships among them) and the functional model (presents, in an abstract level, relationships among functionalities of the domain, represented in a more detailed level by domain use cases).

The domain concepts model tries to describe the meanings of main domain concepts and relationships that facilitate the understanding of the domain as a whole. However, for a complete understanding of domain concepts, its synonyms, restrictions, among others, domain concepts models alone are not enough. This complete understanding is an essential characteristic to the development of domain applications. In Odyssey, a structured template that describes domain concepts in more detail is used. This structure is denominated *Domain Pattern*, similar to HotDraw framework documentation patterns [31].

It is important to point out at this moment that all Odyssey models are connected via a trace relationship, as shown in Figure 2. It means that, if the user is examining a certain model, he can look at other related models by following connections to other models. Odyssey provides automated support for this trace capability.

Based on domain use cases and on the domain concepts model, it is possible to derive collaboration type models for each domain use case. These collaboration models represent, in a conceptual level, reusable domain components that will be further refined in more concrete models in later phases. Odyssey uses interaction diagrams (UML notation) for the representation of collaborations among types in a use case.

An interesting feature of Odyssey is that in the process of specifying models, it is possible to consult a group of *support analysis patterns* [7]. These patterns help in the abstraction process of types, also providing advice on the best modeling practices within the domain.

The concept of patterns is also used in the representation of *architectural models*. Generic architectures – architectural styles that are relevant for the domain – are represented by structures similar to Bushmann's [2] architectural patterns and also to Gamma's design patterns [8]. Based on conceptual domain use cases, OO models, and on the advice given by architectural and design patterns, architectural models are composed. The main characteristic of these models is that it is partitioned by components. Each component specifies a domain task, and how this task can be designed. Connections between components are also described. Therefore, main architectural models are: *Service (interface) Model of components* (presents each component as a type that provides some services that are visible to other components; *Architectural Collaboration Model* (mainly concerned with the definition of a global architecture of the domain; *Classes model and state diagram for each type*(the definition of internal component design).

Implementation models are formed by a set of code components that are connected via CORBA protocol. Components are generic, but they can be specialized by using techniques such as parameterization, class specialization, etc. The CORBA protocol is used for the interoperability between components, regarding two strategies: Codification of components in an OO programming language; Use of legacy components.

4. Specification and Use of Domain Models (Domain Engineering)

Odyssey provides tools that enable users to specify and use domain models. In this sense, tools for eliciting requirements, patterns and reusable components' management, among others are provided:

- *Requirements Elicitation Tool*: This tool is responsible for the acquisition of domain information, which can be in the form of domain expert's knowledge, documentation and applications. The acquisition process is directed to the construction of use case models.

[2] We use the type notion, as proposed by OMG and ODMG models, meaning a reference to an object or class of an OO model. Type is an object in a high level of abstraction.

- *Pattern and Components Management System*: It is responsible for the creation, deletion and modification of patterns and components.
- *Information Agent Tool*: This tool guides the search for domain specific information which is interrelated as a hypermedia web. As soon as it notifies some user's interest about a certain concept, it seeks for related concepts and other associated information such as use cases, OO models, etc .
- *Architectural Transformation Tool*: This tool, supported by the advice of architectural and design patterns, helps in the transformation of conceptual models to architectural components models.

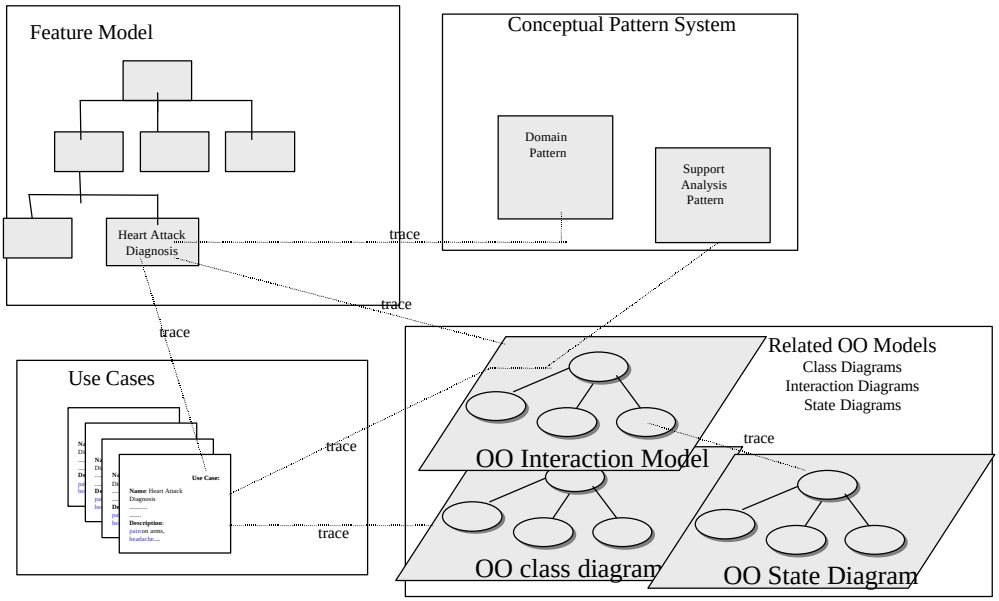

Figure 2 – Traceability between conceptual models

5. Odyssey-DE Method

Surveying main DE methods found in the technical literature [3], [13], [19], [10], [27], [28] in the light of Odyssey requirements, it was concluded that none of these were completely adequate. Therefore, it was necessary to specify a method that could retain the best features of the main current DE methods and at the same time avoid their shortcomings.

Some deficiencies found in preexisting methods, regarding Odyssey requirements, are: i) none of these methods covers every phase of DE, with equal emphasis; ii) none of these methods provides a domain viability analysis phase. The concept of viability analysis is important, since the development of domain models for a certain application domain can be costly; iii) concerning the transition from conceptual models to architectural models, some DE methods propose a set of heuristics to accomplish this task [13]. Nevertheless, no systematic approach has been proposed.

Therefore, a DE process was defined, named Odyssey-DE, that is composed of four stages: *domain viability analysis*, *domain analysis*, *domain design*, and *domain implementation (Figure 3)***.**

The main objective of the *domain viability analysis* stage is the analysis of the viability of the domain to provide a group of reusable components. This stage is important for the process as a whole, because it is based in the result of this analysis that domain engineers and specialists decide whether it is worthwhile or not to implement a base of interconnected components for the domain. This viability analysis adopts selection criteria for the domain, attributing weights for each criterion. These criteria are defined in a way that contemplates organization needs [18]. The *domain analysis* consists in the definition of main domain concepts, standing out similarities and differences among these concepts in a high abstraction level. Models are divided based on main characteristics (i.e., features) of the domain.

The main objective of the process is the creation of components that can be reused in domain applications in all levels of application development. Besides, the creation of reusable components is done in an incremental and evolutionary process, i.e., as new information are aggregated to the domain, reusable components are developed or adapted based on the

acquired information.

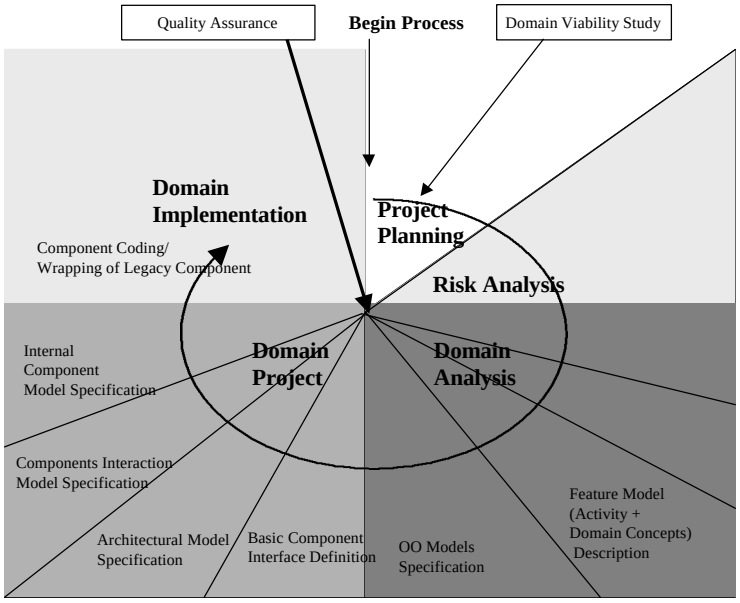

Figure 3 - The Odyssey-DE main stages

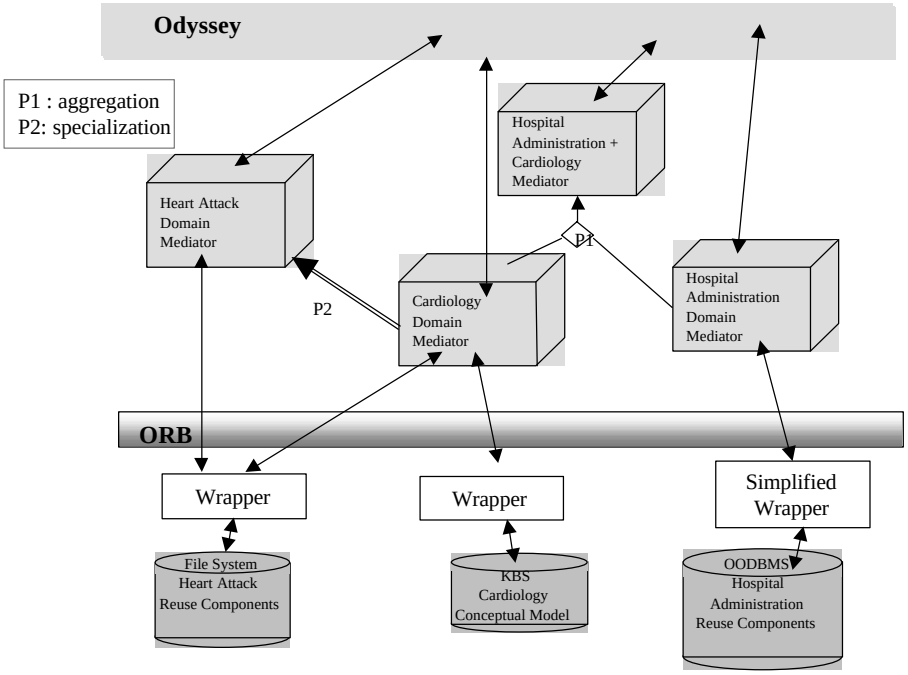

Figure 4 - An example of a mediation layer for a reuse environment

6. Using a Mediation Layer to Store Domain Models

One of the key issues in our project is to accomplish the management of domain models, according to activities defined in the DE method, in an integrated and efficient way. For the effective implementation of technologies involved in the specification of Odyssey, we need a component that provides the integration of concepts, preserving semantics. However, in general, domain information is stored in a great variety of data sources, using different

data models, access mechanisms, and platforms. Further, typically, the domain information is geographically distant, resulting in a complex manipulation.

Thus, a possible solution to access domain information is to use a software layer that provides the integration of different domain databases (distributed and possibly heterogeneous), called *mediation layer* [29]. Mediators are programs that make the connection between distributed databases, heterogeneous data models, and users of this data, providing the information in an adequate format to the user.

In the context of Odyssey, where the access to domain information is an essential requirement, the use of mediators permit that this access to information be carried out independent of the format and the operational platform where it is stored. Moreover, mediators enable the aggregation of information already stored in legacy databases, without having to perform transformations in the original database format.

Figure 4 presents an example of a mediation structure for a specific domain of application, i.e., the cardiology domain. Some mediators are presented in the context of Cardiology and Hospital Administration domains [16]. There are several mediators, each responsible for managing a specific domain, i.e.; there exists the Cardiology Domain Mediator, which was specialized (P2) in a mediator that provides detailed information about a cardiology sub-domain, the domain of Heart Attack. The Hospital Administration Mediator is aggregated (P1) to the Cardiology Mediator, generating a more powerful mediator, which combines these two domains. The latter can be used in cases where information concerning these two domains is necessary.

7. Related Works

Related works can be found in the technical literature, which have something in common with ours. Nonetheless, most of them deal with only a few aspects of Odyssey. None of them treats with the same emphasis each one of the several activities and technologies that are important for the development of component-based software, as we do in our work.

Regarding the specification of environments to support component-based software development, there are some interesting approaches. Domain modeling in [6] uses AI techniques. By comparing this work with Odyssey, we observe that it is mostly concerned with the representation of conceptual models. However, no attention is paid to the description of a detailed method for structuring its knowledge base. Moreover, all design environments reported until now are specific to predefined domains, and cannot be used to store knowledge about other application domains.

The work of Gomaa et al [10] focus on the creation of a reuse environment based on the automation of its DE method, the EDLC (*Evolutionary Domain Life Cycle*), thus creating a generic environment named KBSEE (*Knowledge Based Software Engineering Environment*). KBSEE has some points in common with our work, such as: the adoption of a method to systematize domain engineering; and the specification of domain models in various abstraction levels, using mainly the object-oriented paradigm. However, when we consider the aspect of data storage, KBSEE requires transformations between different representations and databases. This results in redundant information storage and overhead. Moreover, the semantic gap is increased. In this aspect, our proposal differs from Gomaa's, since it is based on the use of a generic and standard model to store domain models. This standard is compatible with UML, using the structure of a mediation layer, which leads to a better performance.

The work presented in [12] specifies a CBD method based on OO techniques. Although this method is adequate for an application development perspective (development with reuse), it lacks detailed information about a domain engineering perspective (development for reuse). There are some guidelines for a domain engineering perspective, but a detailed description of a DE process is not the main concern of the work. Our domain engineering method differs from the Catalysis method [36] in the same way as it differs from the work presented in [12].

In [22], it is presented a work that is considered an evolution of the method presented in [12]. The FODACom method considers use cases and feature diagrams as adequate structures to deal with domain complexities. The difference between our proposal and this one is that we divide the feature model in multiple views and multiple levels. In FODACom method, conceptual, architectural and implementational features are related in the same feature diagram. In our approach, we divide conceptual, architectural, and implementation features in three views, respectively. With this approach, architectural and implementational issues do not appear at conceptual level. It enables the developer to concentrate on the structure of the domain, preventing him from worrying about architectural and implementational details too soon. Moreover, our feature model has some concepts derived from

ontological research in information systems, proposed in [24].

A work that is quite similar to ours is presented in [27]. The OODE method is a DE method that uses techniques like use cases, pattern systems, and OO models as does our work. The main difference between our proposals lies in the level of abstraction that structures are shown. We consider that OODE method is mainly concerned with the architectural and implementational level and our proposal concentrates also at the conceptual level. We believe that the complete and detailed understanding of domain concepts is crucial for future component reuse. Though, we are also concerned with high levels of abstraction.

Works that are more closely connected to the specification of *DSSA* can also be found in the technical literature, such as NASA's space project, named AMPHION environment [11], a domain-oriented environment based on formal specifications and on deductive synthesis of domain applications. The main advantage of AMPHION lies in its formal specification, which provides a non-ambiguous abstract representation of the user's requirements. Garlan and Shaw's group [9] has also developed an environment for the specification of architectural descriptions, named AESOP. AESOP is an architectural environment in the sense that it provides support for the representation, customization, and analysis of software architectures. Our work is similar to the ones reported in [9] and [11], since it specifies software architectures for specific domains. However, both projects base their architectural descriptions on formal approaches, using ADLs (*Architectural Description Languages*) – in the case of AESOP [9] – and mathematical theorems – in the case of AMPHION [14]. The architectural description using ADLs has many followers. Nevertheless, some authors, such as Bosh [1], believe that the use of constructs such as patterns eases the transition from architectural constructs to structures coded in programming languages, mainly OO programming languages. In our work, we use architectural constructs that have already been largely used in object-oriented specifications, such as patterns [2] and frameworks [5].

Therefore, in spite of intersections with works cited above, our work presents a differential that includes: the specification of patterns based on both architectural styles and specific information from the application domain [7]; and the specification of a complete reuse environment which defines software architectures and conceptual model representations of a high level of abstraction. Moreover, our proposal is also concerned with the specification of operations that will be performed on models, as well as with the systematization of these operations, in order to provide the consistent creation of models.

8. Conclusion

In this work, we presented some requirements to support CBD, through the specification of a reuse environment based on domain models (Odyssey).

The implementation of the Odyssey environment involves several research topics and nine persons (i.e., two senior researchers, two Ph.D. students, three master students, and two undergraduate students). Currently we have an operational prototype of the environment that provides some basic functionalities, such as an OO diagram editor, designed specifically to deal with Odyssey models and their traceability, and a tool that helps to configure the DE generic process to specific projects. The mediation layer and information agent tool are under development.

Odyssey environment brings some interesting contributions, mainly in the following points:

- Identification of technologies and specification of components capable of addressing various stages involved in CBD;
- A DE method to support all phases of the process, including a viability analysis stage, which aims at validating the viability of applying model oriented reuse in that domain;
- Systematic use of high-level OO constructs, such as patterns, and its insertion into a DE method;

Approaches that are similar to ours [6] [10] [9] [11], although presenting concrete results, do not specify all aspects addressed by Odyssey when supporting CBD. Such proposals present results in certain aspects of CBD, but these are isolated from a wider context. The innovative approach of Odyssey reduces the semantic gap between the specification and software construction. Therefore, with the help of Odyssey, the software developer is able to apply reuse techniques from early stages of the application development process. Other approaches instead, generally put emphasis on just one of the phases of the process.

References

[1] Bosh, J. - Reusable Specification of Architectural Fragments, University of Karlskrona/Ronneby, Sweden, 1997- at http://www.pt.hk-r.se/~bosh

[2] Buschmann F. et al - Pattern-Oriented Software Architecture : A system of patterns, John Wiley,

1996

[3] Cohen, S - Feature-Oriented Domain Analysis: Domain Modeling, Tutorial Notes; 3rd Int. Conference on Software Reuse, Rio de Janeiro, November 1994.

[5] Fayad, F., Douglas Schmidt - Object-Oriented Application Frameworks, Communications of the ACM, Special Issue on Object-Oriented Application Frameworks, Vol. 40, No. 10, October 1997.

[6] Fischer, G. - Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments, In: IFIP WG 8.1/13.2 Joint Working Conference, A. Sutcliffe, D. Benyon and F. van Assche (eds): Domain Knowledge for Interactive System Design, Chapman & Hall, pp 1-16, May 1996.

[7] Fowler, M. - Analysis Patterns : Reusable Object Models, Addison Wesley, 1997

[8] Gamma E. et al - Design Patterns: Reuse of Object Oriented Design, Addison Wesley, 1994

[9] Garlan, D.,Kompanek, A., Melton R., and Monroe R. - Architectural Style: An Object-Oriented Approach, February, 1996 at http://www.cs.cmu.edu/afs/cs/project/able/www/able/papers_bib.html

[10] Gomaa, H et al - A Knowledge-Based Software Engineering Environment for Reusable Software Requirements and Architectures, Automated Software Engineering 3(3/4): 285-307, August 1996

[11] Lowry, M., Van Baalen J. - Meta-Amphion: Synthesis of Efficient Domain-Specific Program Synthesis Systems, Automated Software Engineering, 4, pp. 199-241, 1997.

[12] Jacobson, I.; Griss, M.; Jonsson, P. - Software Reuse: Architecture, Process and Organization for Business Success, Addison Wesley Longman, May 1997

[13] Klingler, C. D.,Schwarting, D. - A Practical Approach to Process Definition, Proceedings of the Seventh Annual Software Technology Conference, Utah, April 1995

[15] Euzenat J. - Corporate Memory through cooperative creation of knowledge based and hyper-documents, In: Proceedings of KAW'96, 1996

[16] Braga, R.; Mattoso, Marta; Werner, Claudia - Data Integration in a Reuse Environment, Submitted to publication, 1999

[17] Studer, R.; Angele J. Fensel D. - Domain and Task Modeling in MIKE, In: A. Sutcliffe, D. Benyon, F. van Assche (Eds.) Domain Knowledge for Interactive System Design, Proceedings of IFIP 8.1/13.2 Joint Working Conference, Geneva, May 1996.

[18] QSORT Domain Engineering Team - Domain Selection Report for NASA Domains, April 1995 at http://sort.ivv.nasa.gov/related.htm

[19] Tracz, W., Batory, D. David McAllester, Lou Coglianese - Domain Modeling in Engineering of Computer-Based Systems, In: Proceedings of the 1995 International Symposium and Workshop on Systems Engineering of Computer Based Systems, Tucson, Arizona, February 1995.

[20] Braga, Regina; Werner, Claudia; Mattoso, Marta – A Reuse Infrastructure Based on Domain Models, In: Proceedings of ICCI'98, Canada, June, 1998

[22] Griss, Martin; Favaro, John; Alessandro, Massimo – Integrating Feature Modeling with the RSEB, In: Proceedings of the Fifth International Conference on Software Reuse, Canada, 1998

[24] Guarino, Nicola, Formal Ontology and Information Systems In: N. Guarino (ed.) Formal Ontology in Information Systems. IO Press, Italy 1998

[27] Chan, S; Lammers, T. – Reusing a Distributed Object Domain Framework, In: Proceedings of the Fifth International Conference on Software Reuse, Canada, 1998

[28] Cohen, S.; Northrop, L – Object-Oriented Technology and Domain Analysis, In: Proceedings of the Fifth International Conference on Software Reuse, Canada, 1998

[29] Gio Wiederhold and Michael Genesereth - The Conceptual Basis for Mediation Services, IEEE Expert, Vol.12 No.5, Sep-Oct 1997.

[31] Johnson, R - Documenting Frameworks, In: Proceedings of OOPSLA'93, 1993