OdysseyShare: an Environment for Collaborative Component-Based Development

Cláudia Werner¹, Marco Mangan¹ ², Leonardo Murta¹, Robson Pinheiro¹, Marta Mattoso³, Regina Braga³, Marcos Borges⁴
COPPE/UFRJ – Federal University of Rio de Janeiro, Brazil ¹
FACIN/PUCRS – Catholic University of Rio Grande do Sul, Brazil ²
CTU/UFJF – Federal University of Juiz de Fora, Brazil ³
IM/NCE/UFRJ – Federal University of Rio de Janeiro, Brazil ⁴
{werner, mangan, murta, pinheiro, marta}@cos.ufrj.br, regina@acessa.com, mborges@nce.ufrj.br

Abstract - Automated support such as the one provided by Software Development Environments (SDEs) is a key requirement for the systematization of large-scale component-based software development. However, to provide a Component-Based SDE, adequate software development process, methods and tools that consider Component-Based Development (CBD) activities must be previously defined. Moreover, CBD can be a highly distributed and collaborative activity that needs group interaction support. In this paper we describe OdysseyShare Environment, a Collaborative Component-Based SDE under development at the Computer Science Department of COPPE/UFRJ. It supports activities involved in modeling, construction, reuse and group interaction by providing an integrated set of tools and a repository of reusable components.

Keywords: Software Development Environment, Reuse-Oriented Development, Component-Based Software Engineering, Collaborative Software Development.

1 Introduction

Software development processes are remarkably collaborative. Software engineers and software stakeholders daily routine requires the coordination of activities and tasks, continuous information sharing about project state and domain knowledge, and working on tightly coupled tasks such as editing of documents and programs. This collaborative nature has motivated the proposal and construction of a variety of tools and environments for supporting collaboration in this context [1] [2] [8] [10].

In the last decades, many organizations have been adopting remotely located facilities and outsourcing in software production. Component-based Software Development (CBD) is the leading reuse paradigm that allows the work of teams dispersed over time and physically distant. In this context, we suggest the need of a collaborative development environment that considers CBD processes and techniques. Odyssey [5] [30] is a research project that aims at the construction of an environment for CBD support. Recently, the collaborative aspects of software development are being considered within Odyssey Software Development Environment (Odyssey SDE) by the definition of tools for group interaction support. This collaborative version of Odyssey SDE is now being called OdysseyShare [31].

This paper describes the main characteristics of OdysseyShare SDE. Its aim is similar to many other environments such as Serendipity [12], Gossip [11], and MILOS [22]. These environments are motivated by the need of distributed, or virtual, development teams, typical of global software development. The requirements are to support process enactment for distributed teams, to encourage communication and socialization, and to support the activities of individuals or small groups. They aim to provide an efficient and effective software development and, above all, a better satisfaction of individuals who participate in the process. The design of OdysseyShare SDE adopts relevant features of the previously referred environments with emphasis on supporting scenarios of collaboration, which are commonplace in CBD. Therefore, we are able to provide support that is best fitted to processes, activities, roles, artifacts, and goals of a CBD community of developers.

In OdysseyShare SDE, components are developed according to a specific Domain Engineering (DE) process, and are made available in “packages” that include diagrams, source code, textual descriptions, and binary code, among other artifacts. These artifacts are further reused in all phases of an application development, according to an Application Engineering process (AE). Both DE and AE processes are initially modeled, using a workflow engine, as presented in Section 3. This engine provides support to decision-making and workflow management of software engineering teams involved in DE and AE activities. By establishing these processes, the sequence of activities, tools to be used, roles of people involved, and consumed and produced artifacts are defined. These artifacts can be retrieved from local or
remote (i.e. on the Internet) component repositories, through mediators and ontologies, which are capable of integrating information stored in distributed repositories (Section 4).

Once the common artifacts are made available, members of different teams can interact in a collaborative manner, as for instance within an editing session. To attend some collaborative editing requirements, a set of awareness widgets was implemented in OdysseyShare SDE for supporting remote and synchronous interactions. The widgets were built as software components so that any environment tool could use them (Section 5).

Next section provides an overall view of the environment. Some related works are presented in Section 6. Finally, we conclude the paper in Section 7.

2 Environment Overview

OdysseyShare Project aims at constructing a collaborative software development environment to support Component-Based Development (CBD). CBD is an approach in software engineering that aims at software reuse. CBD can be defined as an approach to software development in which all software artifacts – from executable code to interface specifications, architectures, and business models – can be built by assembling, adapting, and wiring together existing components in a variety of configurations [9].

A CBD method can rely on domain-specific information, i.e., concepts, processes, and tasks that are specific to an organization or business area, in order to build component specifications [9]. In this context, a framework (Figure 1) can be conceived, where conceptual models, software architectures, and implementation models are specified within previously selected application domains.

![Figure 1. Domain models framework](image)

Conceptual models are represented by a domain pattern system, and object-oriented (OO) models, using an extended UML notation. Architectural and Design pattern systems, in addition to OO models, are used to represent architectural models. The implementation model is represented by the reuse components set.

Considering this framework, a domain-specific component may depend on information that is beyond the knowledge of a software engineer. On the other hand, domain experts usually do not have knowledge about software development techniques but they have a deep insight of specific domains such as health care, telecommunications, finances, or law.

In this scenario, a collaborative software development environment aimed to support CBD, relying on domain models, must focus on issues related to coordination, cooperation, sharing of documents, and, especially, in the establishment of a community of domain experts and software engineers.

CBD tools provided by OdysseyShare SDE include support to workflow management, component retrieval, requirement elicitation, pattern and components’ management, model criticism, and reverse engineering. In the next sections, the workflow engine, the component publishing and retrieval architecture, and awareness components are described in detail.

3 Workflow Engine

In the context of OdysseyShare SDE, a workflow engine, called Charon, was built to provide coordination over collaborative development. In this sense, a workflow modeling environment is provided. It is based on UML activity diagram, which allows the creation of composed processes (i.e. a workflow that defines how its sub-processes are related) and primitive processes (i.e. definition of its execution script, tools that are to be used, roles of authorized developers, estimated execution time, and consumed and produced artifacts).

After the process modeling, processes are instantiated for a given project. This instantiation is composed of two stages. The first one consists of process simulation, aiming at the detection of modeling errors and generation of the expected execution time of composed processes. The simulation techniques are: Event based simulation and Monte Carlo simulation. The second stage involves the association between developers and roles for each existing sub-process.

Internally, the instantiation consists of mapping the modeled process to a Prolog representation, which populates the project’s knowledge base. This knowledge base contains all information about process execution. This information is further used and modified by the existing agents.
Once the processes are instantiated, developers have access to a list of pending activities. In order to perform a given activity, the associated tools must be used. It is also possible that decisions are taken, changing the course of the project development due to unexpected events.

During development, the manager has access to high-level information about the process execution. A colored and overlapping notation presents this information (color can not be perceived in this black-and-white printing) (Figure 2). Red color indicates that activity duration was longer than initially estimated, green indicates activity duration within the estimated time, and yellow indicates activities still under execution.

**Figure 2. A process under execution**

The overlapping notation indicates how many times a certain activity was executed by repeating the activity drawing in an overlapping manner. The most recent execution presented on top of the others.

The workflow engine flexibility is obtained by an agent-based architecture. Initially, four agents were constructed:

**Simulation Agent:** Responsible for verifying the correctness of the modeled process, allowing its execution or not, and calculating its estimated execution time;

**Execution Agent:** Responsible for verifying the knowledge base state, searching for finalized activities or decisions to be taken, allowing process execution;

**Management Agent:** Responsible for interacting with the developer, to indicate pending activities and decisions to be taken. It also allows the finalization of these activities or undertaking a decision;

**Backtracking Agent:** Responsible for backtracking the knowledge base for a given moment in the past, in order to recover from errors occurred during tool usage.

New requirements are modeled by adding new agents to the workflow engine. Each new agent can act in a reactive manner, motivated by events coming from the environment or other agents, or in a pro-active manner, searching for its goals even if no external event occurs.

### 4 Component Publishing and Retrieval from Heterogeneous and Distributed Repositories

As mentioned before, in the context of collaborative component development, a team may be composed of developers coming from all over the world, producing and storing components in repositories of different kinds and independent from each other (i.e., varying storage formats, access means, query capacities, etc). For this reason, searching and retrieving components, specially the ones available on the Internet, is a complex task.

For solving this problem, an architecture, called ComPublish [27], was defined. It aims to support OdysseyShare developers in publishing and retrieving software artifacts that are available on the Internet, such as models, diagrams, source code or any other kind of artifact used or produced at the different stages of software development. Based on the concept of mediators [34], ComPublish provides a logical view of various published components, both local or remotely, within a given application domain. The main services provided are:

**Component Description and Publishing:** publishing components in remote repositories is done by Le Select server [20], which provides mechanisms for the needed heterogeneity abstraction of the different component repositories on the Internet. By using Le Select metadata publishing services, components can be described according to XML standard. A pre-defined DTD is provided for each component category, such as code, documents, services, and provided and required interfaces. To register a component, the publisher must provide information such as: application domain, author, language, services, and interfaces that this component provides and requires, among other information. When published components are stored in a remote repository, they are referenced by a specific mediator within the integration layer, via metadata for each component exported by the publishing module. In this way, each component becomes accessible to queries submitted by users at the application layer. Once found and selected, the publishing layer provides services to transfer the remote component to the local user engine.

**Mediation Services for Metadata integration:** the integration layer consists of a set of mediators and translators that communicate with each other through a CORBA bus. Translators encapsulate a Le Select API
client that provides the necessary services for communication with Le Select remote servers. Mediators are responsible for organizing the published component information for a given application domain. GOA object server [29] provides the storage and query services to metadata and local components. All component information exported by Le Select servers are converted and stored in GOA, according to an object-oriented model. One special mediator, called Service Manager (SM), manages the related information to other mediators of this layer. SM is responsible for directing the submitted queries done by the application layer to the adequate mediators, and performing intra and inter-domain ontological links through GOA Ontology Services module (GOS).

Ontology Management and Domain Terms Integration: GOS is the module responsible for performing intra and inter-domain ontological links. This module performs the necessary mappings between terms of the same domain, or different ones, and maintains domain information up-to-date. GOS receives from SM ontological service solicitations, such as, an ontology creation and inclusion of ontological terms, discovery of component information related to a given term, and establishment of intra and inter-ontological relationships. GOS uses a forward chaining algorithm and some rules (stored in GOA) with the objective to generate new relationships from existing ones. With this tool, the search for components in a given and related domain is improved, since the search is done using ontological terms, adding more semantic to this search.

5 Awareness Components

Software engineers and domain experts spend most of their time in the collaborative authoring of documents while describing a domain, whether they are textual or graphical descriptions. Thus, collaborative authoring was identified as their primary activity within OdysseyShare SDE. Initially, our strategy was to provide components that implement an adequate set of small possible actions needed to perform the editing in a shared workspace [15].

Collaborative editing of documents is normally carried out in a face-to-face style. Participants expose their ideas in turns and when they reach a consensus they pass the result to the paper. When using a collaborative editor, however, participants present their ideas in parallel, sometimes in different parts of the documents. Although this may bring the benefit of parallelism, it may also cause confusion and conflicts because one participant may not be aware of other people’s work. Therefore, awareness mechanism becomes essential when using collaborative editors.

In a real-time collaborative editing session, awareness information is defined as the information that increases the awareness of each participant in relation to the actions of other users. Many collaboration mechanisms are described in Computer Supported Cooperative Work (CSCW) literature, aiming at increasing both the quantity and quality of awareness information available to participants [14].

These mechanisms explore some kind of graphical representation that allows transferring information about the activity of a participant to all the others. The awareness information completes the explicit communication offered by an available voice channel or chat during the document editing.

Two classic widgets for real-time collaborative authoring were implemented as software components (Figure 3): a telepointer and a radar view component, both commonly available in groupware frameworks and kits [4][24]. OdysseyShare components were implemented using the Java Platform [19].

![Figure 3. Radar view (small window at left hand side) and telepointers (cursor hands at center) in the OdysseyShare SDE Collaborative Editor.](image)

The radar view is an awareness widget commonly used in graphical and textual editors to help in determining current editing position in a large document. For example, Acrobat Reader, Aladdin Ghost View, and Microsoft PowerPoint provide radar views. A radar view component, in the context of groupware, displays information about the local user as well as remote users’ positioning. The user can scroll an editing window by clicking within the radar view window.

The telepointer presents the position of the remote user’s mouse, or another pointing device, cursor in the document. It requires no additional user interaction; mouse pointer information is captured automatically.
Telepointers are popular in teleconferencing, but are seldom seen in non-groupware applications.

Although these mechanisms generate positive effects over participants’ performance, their usage is restricted to applications developed by CSCW research groups [14]. The adoption of these or other available mechanisms in a practical way is currently one of the biggest CSCW challenges. By adopting DBC techniques [28] to awareness mechanisms, it is possible to reuse them in various applications. Each component performs a separation of concern, a vertical segmentation that completely performs the mechanism, with less modification to the integrated application.

During an editing session, two or more execution instances of an application form a unique collaborative session through interface event sharing. These events can be obtained at a windowing system. This implementation technique is known as flexible collaboration transparency [4]. In this way, components can be used in any application that uses a windowing system.

The awareness component proposal fulfills two objectives: (a) to provide awareness information for collaborative editing of documents within OdysseyShare SDE, and (b) to present an alternative adoption of CSCW mechanisms that can be integrated to the design of a new editor or to the implementation of existing ones. The current implementation of components uses the window services provided by Java Foundation Classes and the facilities for distributed programming using Remote Method Invocation.

The development of software components in OdysseyShare SDE follows an iterative and incremental process. Before adding more functionality or greater reliability to these components, we are concerned about how these widgets can be effectively used within OdysseyShare SDE for supporting synchronous and online peer reviews. Preliminary investigations are in progress [21].

6 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 OdysseyShare SDE. None of them treat with the same emphasis each one of the several activities and technologies that are important for the development of Collaborative Component-based Development, as we do in our work. Regarding each of the described features in this paper, we make some considerations in the following.

Automated approaches to software process can be divided into three groups [17]. State machine or Petri net-based engines, such as SPADE [3] and Memphis [32], emphasize the graphical modeling of processes, but provide poor model semantics for modeling activity dependencies. The process execution usually happens at the workflow, which increases the coupling between process data and its execution. Agent, rules or scripts based approaches, such as HyperCode [23] and CAGIS [33], provide the necessary flexibility for the construction of any workflow engine. However its design is complex, since no graphical modeling environment is available and the workflow is usually embedded in code, making process creation and modification hard. Hybrid approaches combine graphical notation with rules, as for example EPOS [16] and Merlin [18], provide a modeling environment and a mechanism for transforming the modeled workflow in event-condition-action rules, which are easily handled by computers at workflow execution. However, a pro-active software process machine is needed to improve collaboration between developers, because some collaborative opportunities are made available through execution process states that only the software process machine knows. None of the studied approaches that deal with software process, except Charon, provides this feature.

Regarding the component publishing and retrieval mechanism, the work presented by Seacord, Hissan and Wallnau [26] describes a search engine for the retrieval of reusable code components, such as JavaBeans and CORBA components, in the Agora system. The Agora system uses an introspection mechanism for registering code components, through its interface. Agora search is Web based and only searches on component interfaces, covering solely the component connectiveness problem. The Agora system only deals with code components whereas our approach is concerned with domain information in all abstraction levels, including code components. Moreover, new information is always associated to domain terms within a given domain ontology [7], improving its accessibility and reuse in our architecture.

Ye and Fischer [35] present an approach that provides an active repository for components. Their work focuses on active delivery of reuse information, helping to reuse components that developers were not aware of. In this aspect, their work is similar to ours. Our component retrieval system provides similar functionality, since it accesses components from other domains based on semantic similarity. The active repository functionality is presented in a previous work [6]. One aspect that is different in our proposal is the retrieval of distributed information (documents), which is not mentioned in the work of Ye and Fischer.

Our work also aims to support the retrieval of information resources (people). Collaborative support may
help to find work partners in problem solving scenarios and to share knowledge not yet documented. Future work aims at recording an editing session for review and reuse of experience based knowledge.

Recently, some software development environments have been proposed to support Global Software Development or Collaborative Software Development [1][12][13][11][22][25]. These environments aim to combine software development task support with some sort of teamwork support. Altmann & Pomberger’s [1] ample survey of cooperative software development practices lead to a proposal of an environment aimed at large software projects organizational support. Distributed process coordination is present at Serendipity [12][13] component-based environment. Farshchian [11] proposes the Gossip environment based on improved product awareness information over artifact changes. Real-time, collaborative editing support is applied on distributed extreme programming in both MILOS [22] and Tukan [25] environments.

OdysseyShare SDE offers teamwork support for tasks and activities provided in these earlier collaborative software environments. Additionally, the application of reuse and component-based software development techniques help distributed teams and organizations to deal with the complexity of large-scale, distributed software development.

7 Conclusion

The aim of this paper was to describe OdysseyShare SDE - a collaborative component-based software development environment. A workflow engine, a component publishing and retrieval mechanism from heterogeneous and distributed repositories, and a set of awareness components were defined and built as a first step towards a strategy for providing OdysseyShare SDE with adequate support for the interaction among the environment users – software engineers and domain experts – while editing software engineering documents.

The requirements to support process enactment for distributed teams, to incentive communication and socialization, and to support the activities of individuals or small groups, aiming at providing an efficient and effective component-based software development, makes the construction of such environment a very complex task.

Apart from the work described in this paper, many others are under development in the context of OdysseyShare Project, among them: an asynchronous awareness support within databases, concurrent XML document access services, and support to configuration management activities, which will be described in future papers.

Acknowledgements

The authors would like to thank CAPES and CNPq, for financial support, and all participants of OdysseyShare Project, who direct or indirectly supported this work.

References


