AN MDA-EDOC BASED DEVELOPMENT PROCESS FOR DISTRIBUTED APPLICATIONS

Rita Suzana Pitangueira Maciel, Bruno Carreiro da Silva
Faculdade Ruy Barbosa, Rua Theodomiro Batista, Salvador-BA, Brazil

Carlos André Guimarães Ferraz, Nelson Souto Rosa,
Centro de Informática, Universidade Federal de Pernambuco, Recife-PE, Brazil

Keywords: MDA, EDOC, Distributed Application Development

Abstract: With the proposal of MDA by OMG, the modelling of systems, in development process of distributed applications, has become a central point, therefore software models go beyond system documentation. EDOC - MDA profile for modelling distributed application - uses as conceptual framework the RM-ODP. These elements, although very useful, are insufficient for a software development process; therefore they are not followed by development methodologies. In this article is presented a MDA-based development process for distributed applications that utilize EDOC and the RM-ODP. The process is described as a sequence of steps and a set of diagrams that should be specified to provide a MDA-based system description.

1 INTRODUCTION

Distributed applications are inherently complex, from their design, development and testing through to maintenance (Bernstein, 1996). The *Object Management Group* (OMG) has been constantly investing in proposals to facilitate the development of distributed applications. To improve their applicability, middleware environments incorporating the components concept (ex. CORBA/CCM (OMG, 2002a) and J2EE/EJB (Shannon, 2003)) and specific services, which cater to a given application category, are being proposed. One of the most recent initiatives from this scenario is OMG MDA *(Model Driven Architecture)* (OMG, 2003).

MDA is a framework for systems development, based on stabilized grounds of software engineering, which separate the specification of the functionality of a system, from the modeling and mapping of this functionality in a specific technological platform. MDA uses abstract models to specify all the logic of the application, where concepts referring to languages or platform are irrelevant – platform independent model (PIM). This high level model is subsequently used to create new models that express the requirements of the system in a specific platform - platform specific model (PSM).

These models are not only used for systems documentation, but also as a tool for implementation. Each activity of the development process requires a number of input models that produce other models as an output. Accordingly, the construction process of an application can be seen as a set of transformations that lead to the final system. The system and its requirements, as well as the domain of the application, can be seen through these models that describe it through different views and levels of abstractions.

To manage the models, OMG provides a series of UML profiles. These include EDOC (*Enterprise Distributed Object Computing Specification)* (OMG, 2002b), which is a profile for specifying distributed systems based on components. EDOC, which consists of various sub-profiles, uses (*Reference Model of Open Distributed Process*) RM-ODP (ISO, 1995) as a conceptual framework to specify the PIM of applications.

The use of modeling techniques and methodologies facilitates the construction of applications. Although EDOC and the RM-ODP framework offer a set of tools that permit the

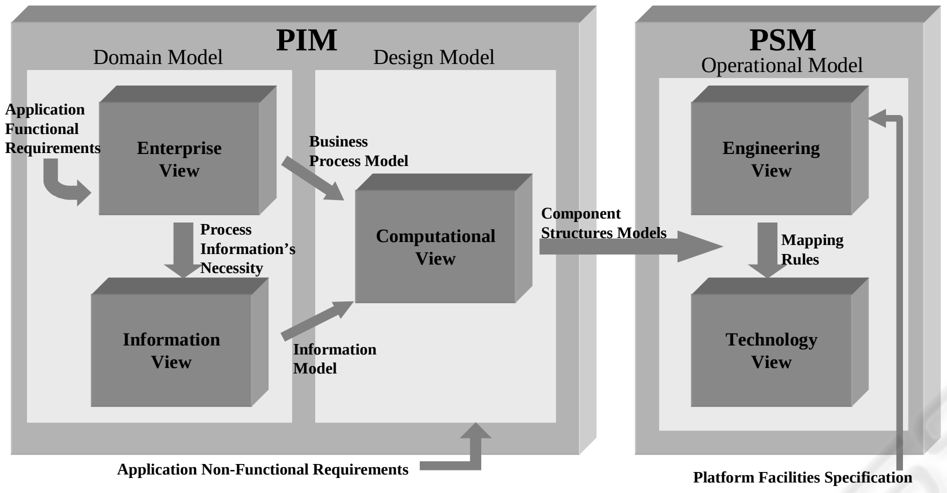


Figure 1: RM-ODP views in MDA PIM and PSM

specification of a system based on high abstraction models, they do not offer a guide to orient developers in the application of these concepts. Together they provide a set of definitions and notations, but not a process with well defined steps to be followed (Gervais, 2002).

Some proposals are being drawn up to assist developers in the construction of distributed applications through MDA. In (Wang, 2003) a specific application development process is proposed for the educational domain. This proposal does not use EDOC. In (Gervais, 2002) the ODAC methodology is proposed for the development of applications through MDA. Although ODAC is based on the conceptual framework of RM-ODP, it does not yet use EDOC. The use of the profiles proposed by OMG guarantees, in a more extensive fashion, the interoperability of models proposed by MDA. Methodologies and tools that assist in the development of applications and use EDOC or other sub-profiles proposed by OMG, will have the same conceptual and notational framework. This aspect facilitates the reading of models both by development teams and by process automation tools, besides which EDOC allows the application to be seen through high level components as of the initial models, facilitating the application decomposition from the perspective of components.The objective of this article is to present a development process proposal for distributed applications through MDA and the EDOC profile. This process was applied to the development of the InterDoc ((Environment for Supporting the Interoperability among Collaborative Document Authoring Tools) (Maciel, 2004). Based on the experience with the development of InterDoc, the process is being applied in other environments for the specification of a complete development methodology based on EDOC.

The rest of this text is organized as followed: the next section presents the proposed process overview. Section 3 presents the main steps for PIM specification. In the conclusion, we summarize the contribution of the work presented in this paper.

2 THE DEVELOPMENT PROCESS OVERVIEW

The process is based on the concept of views proposed by RM-ODP and its notation originates from EDOC. A viewpoint is a subdivision of a complex system specification. It corresponds to a particular perspective, allowing the system to be "viewed" from a particular angle, focusing on specific concerns. Views permit the structuring of modeling activities for an application. RM-ODP is an architectural framework and not a methodology, i.e. it is not prescriptive enough and does not provide tools such as a notation or a sequence of steps to specify the viewpoints that would help the software designers to build the system specification. Three categories of model should be specified: Domain, Project and Operational (fig 1).

The Domain model corresponds to the specification of the Enterprise and Information's Views. The scope and responsibilities of the application are defined in this model via the functional requirements initially established for the application. The services to be offers and the information handled by the application should be defined in this model. The Project Model describes the Computational view for the specified Domain model. This model identifies the components that will fulfill the requirements established in the domain model, independently of the platform. The non-functional requirements of the application should be observed on this occasion to identify elements for the performance thereof. The Domain and Project models form PIM as described in MDA. The Operational Model describes the application execution environment on a specific platform and corresponds to the specification of Engineering and Technology Views. The characteristics of the implementation platform chosen are considered in this model to reflect the real execution environment. The Operational Model form PSM as described in MDA.

Table 1 presents a suggestion of a set of diagrams that should be made to specify the RM-ODP views and consequently the proposed models. The EDOC presents several UML sub profiles; however our process just uses the Business, Entity and CCA profiles. The application's objectives, politics and restrictions specification, that is part of the Engineering view concepts, should be specified, respectively, through use case diagrams and a simple text. These mechanisms are not part of the EDOC proposal, but they are suggested in the EDOC specification annexes (OMG, 2002c).

Table 1: Process Diagrams

Engineering View	Standard UML and *Business Profile*
Objective Definition	Use Case Diagram
Main Policies and Restriction	Text
Business Process Identification	Collaboration Diagram
Activity Definition	Class Diagram
Information View	Entity Profile
Entities Data Definition	Class Diagram
Composite Data Definition	Class Diagram
Computational View	CCA Profile
Architecture Structure	Collaboration Diagram
Component Structure	Class Diagram
Protocol Specification	Class Diagram
Protocol Structure	Class Diagram
Protocol Description	Activity Diagram
Protocol Choreography	Activity Diagram

3 THE PROCESS GUIDELINES FOR PIM

The development process was applied to describe reference architecture and development of the InterDoc environment. InterDOC should allow the various asynchronous activities of the authoring process to be executed in different tools. InterDOC should be made available as middleware domain-specific service layer, which when positioned between collaborative authoring supporting applications (CASA) and their repositories, promotes interoperability among these environments. Groups of authors can hold synchronous or asynchronous sessions for the planning, drafting, revision and editing of a document in their favorite environment. In any phase of the authoring process, the document can be made available through InterDOC to another group of authors that uses another environment, or even an individual work tool, to perform an activity. For InterDOC, collaborative authoring is divided into two large phases: Planning and Writing. The Planning phase is when an author from the group provides the basic information for the description of the authoring project of a document: group of authors, roles of authors in the group, location of the repository of shared information etc. The Writing phase is when authors make versions of the document available for other authors to execute activities.

We will use InterDoc, more precisely the planning phase, to illustrate how the methodology can be applied. The example focus will be the PIM specification

3.1 Enterprise View

In this view the communities must be identified, their business-oriented objectives, processes and the main constraints. The Enterprise view describes interactions among the different InterDocs and interactions between the applications, repositories and the InterDocs.

Step 1. Objective and Constraints Definition. InterDoc has just one community type: the Authorship community. The Authorship community involves the authors and the applications used for writing documents. The main InterDoc's constraints refer to the community that registers a project to be the same that hosts the repository of documents for the group.

The EDOC does not specify a notation to identify the objectives of a community. One use case diagram was used and ten use cases were identified: Design a Project, Define a Group, Define Author, Define Role, Define Activity, Delegate an Activity, Register an Activity, Notifies Authors, Retrieve Documents, Communication with other Domains.

Step 2. Business Process and Activity Identification. In this step should be identified and detailed the processes that accomplish the objectives of the application defined in the step 1.

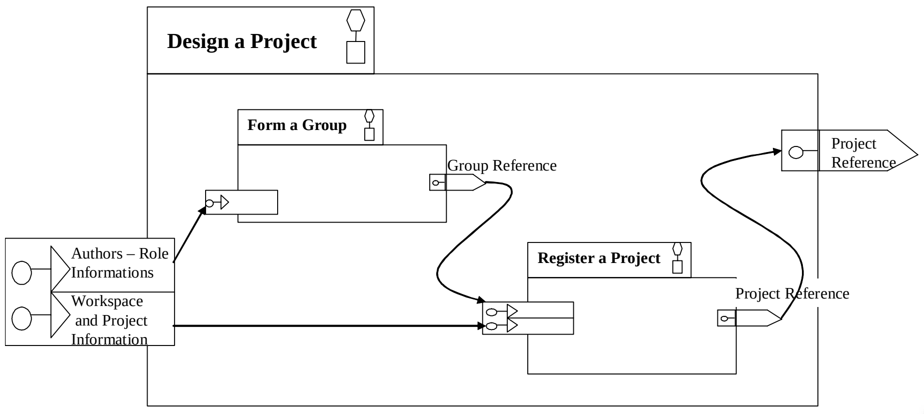


Figure 2: Business Process *Design a Project*

The Business Process Profile was the EDOC sub-profile used to model the Enterprise view. This sub-profile permits design models that present the structure and behavior of the application in the environment where it is inserted. The functionalities that InterDoc realized are described in terms of Business Process (BP) that specifies a complete business task. A BP may contain Activities, which are the pieces of work required to complete a task. Data flows connect different BPs, and Activities in a BP, defining temporal and data dependencies between this elements. These data are modeled as communication ports. InterDOC has two major BPs: *Design a Project* and *Writing* that specify the situations described in the beginning of this section. Figure 2 presents the diagram of *Design a Project*.

Design a Project has two Activities: *Form a Group* and *Register a Project*. To initiate this process, first it is necessary to define a group with the authors' information and the roles they will play in the project. The Port Authors and Role Information fulfill these data. This process is finished when a group reference, that identifies the group uniquely, is generated. To initiate the Register of a Project, it is necessary the group reference and the information about the workspace that will store the shared information. The activities are decomposed up all the objectives specified in the Enterprise View (use cases) were reached.

3.2 Information View

The Information view models data (Entity) and their relationships. Entities represent concepts of the problem domain. The resulting models present the structure of used objects that represents the concepts of the business in the computational environment. The description of objects that InterDoc will handle internally was provided.

The Entities Profile was the EDOC sub-profile used. A central concept of the Entities Profile is an EntityData. An EntityData is a structure of data that represents one definitive concept of the problem domain, or either an Entity. An EntityData is equivalent to an entity or a relation in the relational model. CompositeData are derived from the EntityData to group data used in the component's ports.

Step1 – Entity and Composite Data Identification. EntityData and their relationships should be specified though a class diagram (Figure 3).For the InterDoc was defined the following Entitydata: Project - a certain authorship project; Repository - storage of the project information; File - file stored in the repository; Document and Comments - specializations of File to represent documents and their respective comments; Activity - activities that the authors accomplish in a document; Author – user-author; Group - authors' group and Role - the authors' role in a group.

A project has a repository of associated files and belonging to a certain group of authors. These authors can play different roles in different projects. The role has associated to them certain actions. Actions will be registered in the environment. After revising a document, an author can save in the repository a new version of these documents. Authors revise documents producing comments (or annotations) or versions of these documents. Therefore, a document can be a version of another one, and a comment is associated to a document.

3.3 Computational View

The Computational view is a viewpoint on the system and its environments that enable distribution through functional decomposition of the system into components that interact at interfaces. Component Collaboration Architecture (CCA) was the EDOC sub-profile used for modeling this view.

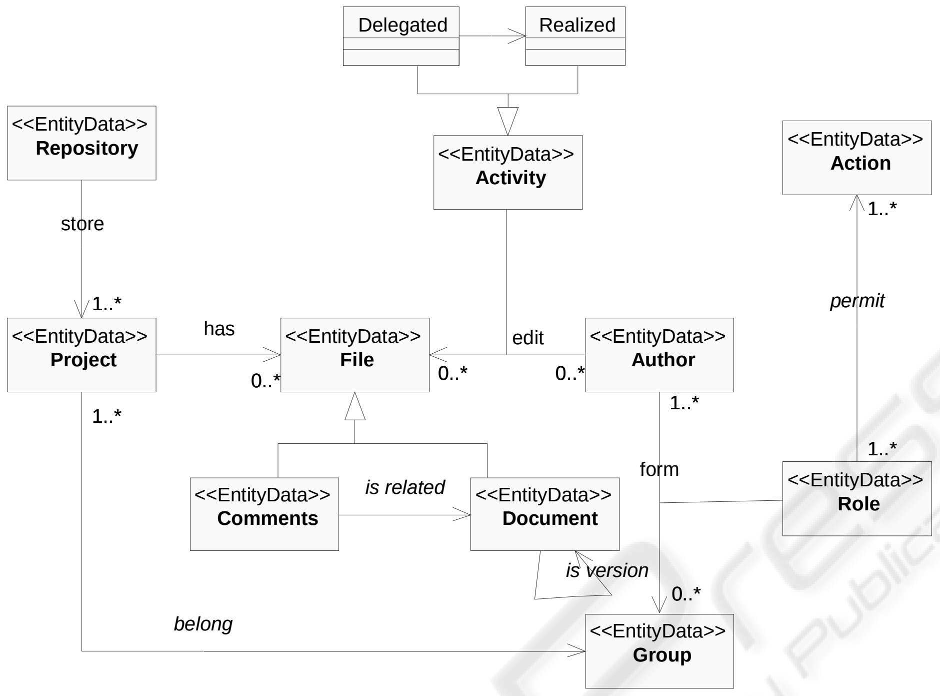


Figure 3: InterDoc Information Model

Step 1. Arquitecture Definition (Component Overview Definition). The structure of the component in terms of communication that will form InterDOC, as well the interfaces and the data that these components handle, are described in the models of this view. Figure 4 shows an overview of the InterDoc's component structure model in terms of Process Components (PC). A PC represents an active processing unit that has a group of ports to interact with another PC. Each PC will execute one or more Activities identified in the Enterprise View. Activities with correlate information were grouped into only one PC. InterDOC has six major PCs:

- *DocumentSharingService*: manages the access to the information that is stored in the repositories;
- *ProjectManagerService*: manages the access to the project's information.
- *AuthoringActivitiesService*: manages the delegation and registration of authoring activities;
- *ActivitiesNotificationService:* performs the notification of an author's activity to the members of a group;
- *AuthorGroupService:* manages access to authors and groups information.
- *InterDomainCommunicationService*: is a wrapper service component that formats messages for the interoperability protocol used by the domain.

The InterDOC reference architecture also describes two client-side components that use its services: the CASA components and the Repository component. These components are, respectively, *ApplicationClient* and *RepositoryClient ProcessComponent*.

The interfaces are conceived through the Facade design pattern. The *IApplicationService* is a generalization of all the communication interfaces among *ApplicationClient* the InterDoc. The same design is applied to *IRepositoryService* that generalized the communication between the InterDoc and repositories; and *IInterDomainService* that generalized the communication among different InterDoc.

Step 2. Protocol and Component Structure Specification. A protocol specifies what messages the component sends and receives when it collaborates with another component. For each activity that a PC performs, a protocol was defined. Figure 5 describes the *RegisterProjectProtocol*.

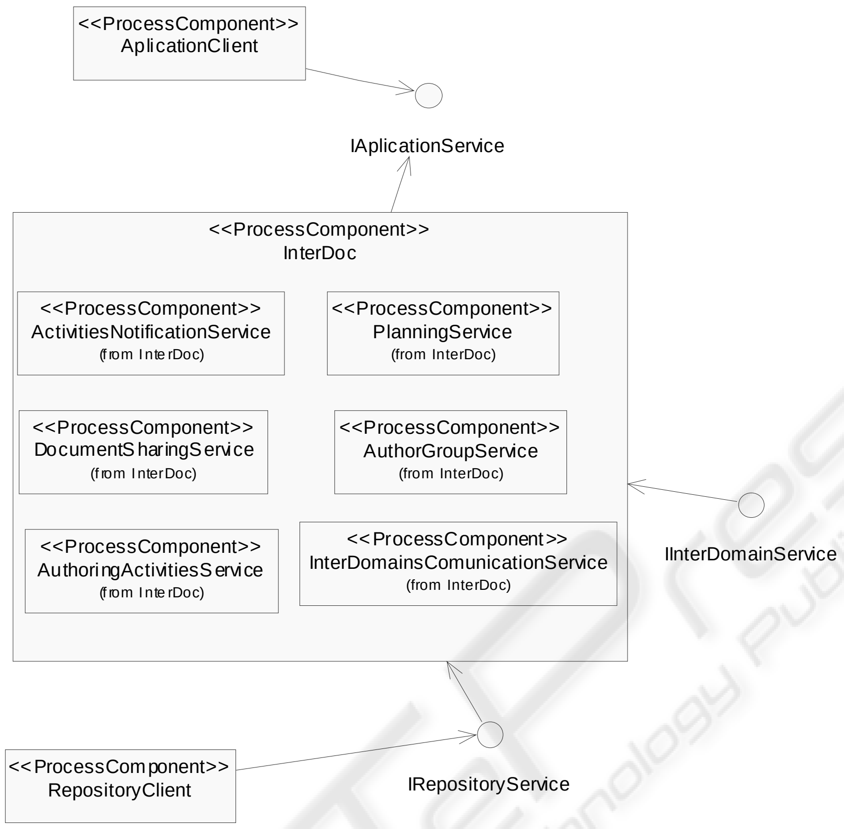

<<ProcessComponent>> AplicationClient
IAplicationService
<<ProcessComponent>> InterDoc
<<ProcessComponent>> ActivitiesNotificationService (from InterDoc)
<<ProcessComponent>> PlanningService (from InterDoc)
<<ProcessComponent>> DocumentSharingService (from InterDoc)
<<ProcessComponent>> AuthorGroupService (from InterDoc)
<<ProcessComponent>> AuthoringActivitiesService (from InterDoc)
<<ProcessComponent>> InterDomainsComunicationService (from InterDoc)
IInterDomainService
<<ProcessComponent>> RepositoryClient
IRepositoryService

Figure 4: InterDoc Components Structure Overview.

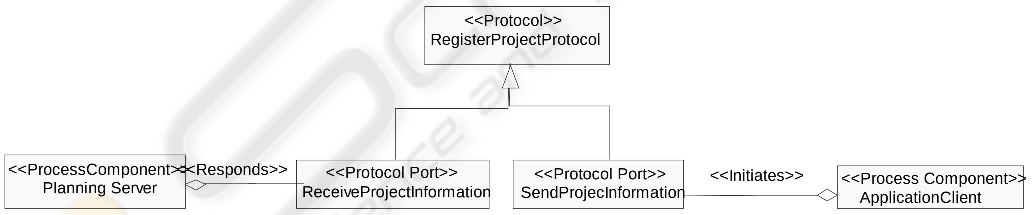
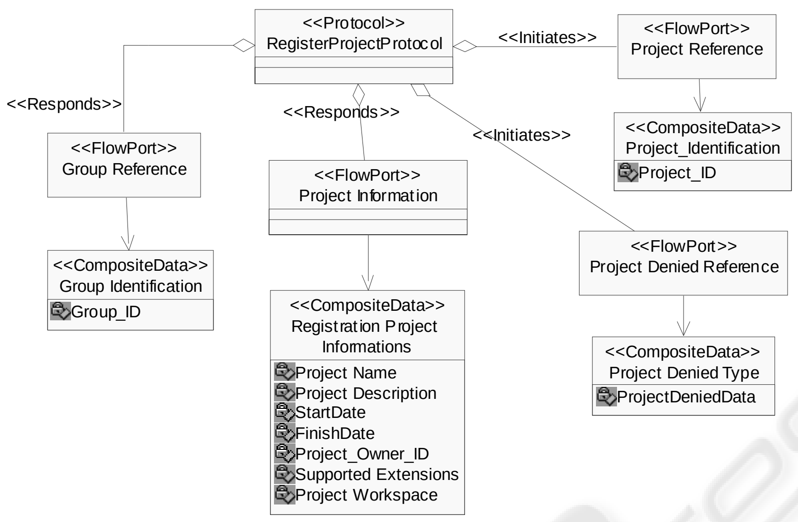

<<Protocol>> RegisterProjectProtocol
<<ProcessComponent>> Planning Server
<<Responds>>
<<Protocol Port>> ReceiveProjectInformation
<<Protocol Port>> SendProjecInformation
<<Initiates>>
<<Process Component>> ApplicationClient

Figure 5: Register Project Protocol Specification (external view).

This protocol specifies the interaction among the components *PlanningServer* and *ApplicationClient* for the accomplishment of the activity *Register a Project* of BP *Design a Project*. In this diagram type it is specified which component initiates *(<<initiates>>)* the protocol and which component answers *(<< responds>>)* though Protocols Ports. So this diagram describes the protocol's external view.

The Protocol structured diagram details the protocol's internal view. It describes the data involved in a Protocol and the ports type. Each port has an association to a Composite Data derived from the Entities Data defined in the Information View. Figure 6 shows the protocol structure for the *RegisterProjectProtocol*. The Protocol Port *ReceiveProjectInformation* (fig 5) is composed of two Flows Ports: *GroupReference* and *Project Information* (fig 6). *SendProjectInfo* Protocol Port is composed of *ProjectReference* and *ProjectDeclinedType* Flow Ports. A Flow Port is port that which defines a data flow in or out of a port on behalf of the owning component or protocol.



<<Protocol>> RegisterProjectProtocol
<<Initiates>>
<<FlowPort>> Project Reference
<<Responds>>
<<Responds>>
<<Initiates>>
<<CompositeData>> Project_Identification
Project_ID
<<FlowPort>> Group Reference
<<FlowPort>> Project Information
<<FlowPort>> Project Denied Reference
<<CompositeData>> Group Identification
Group_ID
<<CompositeData>> Registration Project Informations
Project Name
Project Description
StartDate
FinishDate
Project_Owner_ID
Supported Extensions
Project Workspace
<<CompositeData>> Project Denied Type
ProjectDeniedData

Figure 6: Register Project Protocol Structured (internal view)

Each component is detailed in terms of ports and a set of protocols it implements. Figure 7 describes the *PlanningService* component structure. This component answers to seven protocols through the specified interfaces. These interfaces are *IApplication Interface* specializations (fig 4). An Interface is a protocol (protocol stereotype) constrained to match the capabilities of the typical object interfaces. *PlanningService* begins two other protocols: *Notify Information* and *Comunication InterDomain*. The protocol *Notify Information* establishes a communication with the component ActivitiesNotification for notification of delegated and realized activities in a text. The protocol *Comunication InterDomain* links two *InterComunicationDomain* components for communication establishment with different InterDocs.

Two activity diagrams describe the dynamic aspects of a protocol (not shown in this text). One of this activity diagrams describes the protocol choreography to sequence the actions of the ports within one component. Another one describes the messages sequence between two or more components.

4 CONCLUSIONS AND FUTURE WORKS

This article presented a proposal for the distributed application development process through the MDA approach. The process was applied to the development of the InterDoc environment that is implemented in CORBA/CCM. The main activity of the application development approach through MDA is the production of system models through UML profiles. The MDA profiles only make elements available for the specification of applications, which do not track methodologies for the software. Methodologies leverage productivity as similar development processes can be applied to various systems. Developers need methodologies to facilitate their work, leaving them free to focus their attention on the functional requirements of the application.

The use of EDOC as a sub-profile for the production of models proved helpful in the process of breaking down a system into standalone components, since InterDoc was designed as a set of components from the initial phases of the project. Accordingly, the mapping of components for the CCM platform was facilitated due to the use of the same approach. At the same time EDOC provides sundry sub-profiles that can be combined in different ways. The combination of the profiles used, as well as the diagrams chosen to comprise the process, resulted in a smooth process that does not overburden the developer.

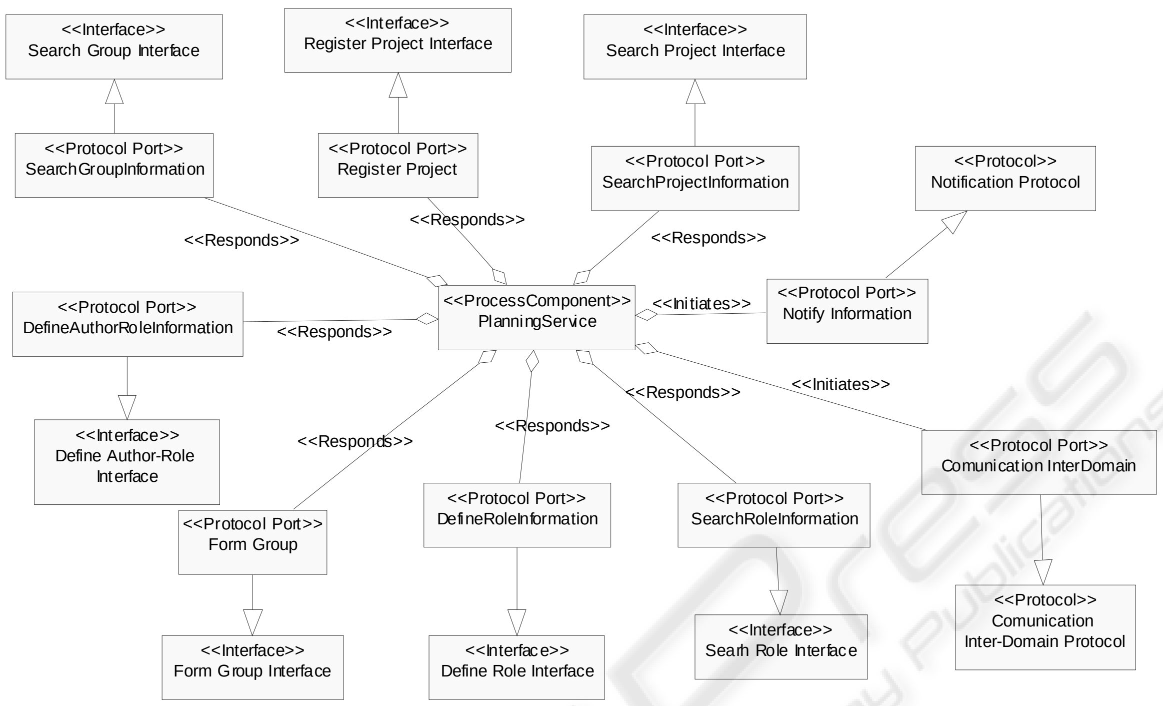

<<Interface>> Search Group Interface
<<Interface>> Register Project Interface
<<Interface>> Search Project Interface
<<Protocol Port>> SearchGroupInformation
<<Protocol Port>> Register Project
<<Protocol Port>> SearchProjectInformation
<<Protocol>> Notification Protocol
<<Responds>>
<<Responds>>
<<Responds>>
<<Protocol Port>> DefineAuthorRoleInformation
<<Responds>>
<<ProcessComponent>> PlanningService
<<Initiates>>
<<Protocol Port>> Notify Information
<<Interface>> Define Author-Role Interface
<<Responds>>
<<Responds>>
<<Responds>>
<<Initiates>>
<<Protocol Port>> Comunication InterDomain
<<Protocol Port>> Form Group
<<Protocol Port>> DefineRoleInformation
<<Protocol Port>> SearchRoleInformation
<<Protocol>> Comunication Inter-Domain Protocol
<<Interface>> Form Group Interface
<<Interface>> Define Role Interface
<<Interface>> Searh Role Interface

Figure 7: PlanningService Component Structure.

The report on the development process, through a sequence of steps and guides to the mapping of concepts, is a stage for the specification of an application development methodology based on components that utilize EDOC as a conceptual framework. These steps and guides are being used in other applications for review and consolidation of the development methodology. After consolidation, the methodology will be applied in the development of applications using MDA tools that support the automation of the application development process.

REFERENCES

Bernstein, P., 1996. Middleware: A Model for Distributed System Services. *Communication of the ACM.* New York. v 39. n 2. p 86-98. February.

ISO, 1995. Basic Reference Model of Open Distributed Process, *ISO/IECIS 10746.* Partes 1-4.

Gervais, M., 2002. Towards MDA-Oriented Methodology. In: *Annual International Computer Software and Applications Conference*, England, August, p 265-270.

Maciel, R.; Ferraz, C.; Rosa, N., 2004. INTERDOC: Interoperable Services in Collaborative Writing Environments. In: 8 th IASTED International Conference on Software Engineering and Application - SEA, Cambridge,Ma -USA.

OMG, 2002. CORBA Component Model v3.0 full specification. *OMG Adopted Specification* (formal/02-06-05).

OMG, 2002. UML Profile for Enterprise Distributed Object Computing Specification. *OMG Adopted Specification* (ptc/02-02-05).

OMG, 2002. UML Profile for Enterprise Distributed Object Computing Specification – Part II. Supporting Annexes. *OMG Document (ad/2001-08-20).*

OMG, 2003. MDA Guide Version 1.0. *OMG Document*.

Shannon, B., 2003. Java 2 Platform Enterprise Edition Specification v 1.4. [s.l.]. Sun Microsystems.

Wang, H.; Zhang, D., 2003. MDA-based Development of E-Learning System, In: *27th International Computer Software and Applications Conference*, IEEE Press, Nov., p. 684-689.