Um Processo de Desenvolvimento de Componentes através da MDA para a Interoperabilidade entre Aplicações de Autoria Colaborativa

Rita Suzana Pitangueira Maciel [1,2], Bruno Carreiro [2], Daniella Rihan [2], Carlos André Guimarães Ferraz [1], Nelson Souto Rosa [1]

[1] *Universidade Federal de Pernambuco, Centro de Informática*
50732-970 Recife, Pernambuco, Brazil
e-mail: {rspm,cagf,nsr}@cin.ufpe.br
[2] *Faculdade Ruy Barbosa, Rua Theodomiro Batista 422 - Rio Vermelho, Salvador, Bahia, Brazil*
e-mail:{rsuzana, brunocs, daniella}@frb.br

Resumo. Com a proposição da MDA pela OMG, a modelagem de sistemas, no processo de desenvolvimento de aplicações distribuídas, vem se tornando cada mais vez um ponto central. Os modelos de *software* vão além da documentação do sistema. O EDOC, perfil MDA para modelagem de aplicações distribuídas baseadas em componentes, utiliza como *framework* conceitual a RM-ODP. Estes elementos, apesar de muito úteis, são insuficientes para um processo de desenvolvimento de software, pois não são acompanhados de metodologias de desenvolvimento. Neste artigo apresenta-se uma experiência de modelagem de uma arquitetura de referência de um ambiente para interoperabilidade entre aplicações de autoria colaborativa - InterDoc. Desta experiência, foram extraídos passos em direção a especificação de uma metodologia de desenvolvimento de aplicações baseadas em componentes através do EDOC.

Palavras-chave: MDA, EDOC, Desenvolvimento Baseado em Components.

Abstract. With the proposal of MDA by OMG, the modeling of systems, in development process of distributed applications, has become a central point. Therefore software models go beyond system documentation. EDOC - MDA profile for modeling distributed component-based 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 an experience of modeling a reference architecture for an environment which promotes interoperability among collaborative authoring applications - InterDoc. From this experience, steps towards a component-based applications development methodology specification have been extracted through EDOC.

Keywords: MDA, EDOC, Component-based Development.

1 Introdução

Aplicações distribuídas são inerentemente complexas, desde o seu projeto, desenvolvimento, testes e manutenção. A construção de tais aplicações, através da abordagem do desenvolvimento baseado em componentes (DBC), pode ser facilitado, uma vez que componentes já implementados e testados podem ser reutilizados em outras aplicações [1]. Contudo, o processo de decomposição de um sistema em componentes independentes e de fácil reutilização é uma tarefa de difícil execução [2].

A *Object Management Group* (OMG) vem constantemente investindo em propostas para facilitar o desenvolvimento de aplicações distribuídas. Para melhorar a sua aplicabilidade, ambientes de *middleware* incorporaram o conceito de componentes (ex. CORBA/CCM [3] e J2EE/EJB [4]) e serviços verticais, que atendem a uma determinada categoria de aplicação, estão sendo propostos. Uma das mais recentes iniciativas deste cenário é a MDA (*Model Driven Architecture)* [5] da OMG.

A MDA é um *framework* para o desenvolvimento de sistemas, baseado em fundamentos estabilizados da engenharia de *software*, que separam a especificação da funcionalidade de um sistema, da modelagem e mapeamento desta funcionalidade em uma plataforma tecnológica específica. A MDA utiliza modelos abstratos para especificar toda a lógica da aplicação, onde conceitos sobre linguagens ou plataforma são irrelevantes - modelo independente de plataforma (PIM - *Platform Independent Model*). Este modelo de alto nível é posteriormente utilizado para criar novos modelos que expressam os requisitos do sistema em uma plataforma específica - modelo específico (PSM - *Platform Specific Model*).

Estes modelos não são usados apenas para documentação de sistemas, mas também como uma ferramenta para a implementação. Cada atividade do processo de desenvolvimento requer um número de modelos de entrada que produzem outros modelos como saída. Sendo assim, o processo de construção de uma aplicação pode ser visto como um conjunto de transformações que levam ao sistema final. O sistema pode ser visto então através destes modelos que o descrevem, bem como o domínio da aplicação e seus requisitos através de diferentes visões e níveis de abstrações [6].

Para gerar os modelos, a OMG disponibiliza uma série de perfis UML. Entre eles o EDOC (*Enterprise Distributed Object Computing Specification)* [7], que é um perfil para especificar sistemas distribuídos baseados em componentes. O EDOC, que é composto de vários sub-perfis, utiliza a RM-ODP [8] como *framework* conceitual para especificar o PIM das aplicações. O EDOC possibilita que a aplicação seja vista através de componentes de alto nível desde os modelos iniciais.

O uso de técnicas e metodologias de modelagem facilita a construção de aplicações. Embora o EDOC e o *framework* RM-ODP ofereçam um conjunto de ferramentas que possibilitam a especificação de um sistema a partir de modelos de alta abstração, eles não oferecem um guia para orientar os desenvolvedores quanto à aplicação destes conceitos. Juntos eles provêem um conjunto de definições e notações, mas não um processo com passos bem definidos para serem seguidos [9].

Algumas propostas estão sendo feitas para auxiliar os desenvolvedores na construção de aplicações distribuídas através da MDA. Em [10] é proposto um processo de desenvolvimento de aplicações específico para o domínio educacional.

Esta proposta não utiliza o EDOC e conseqüentemente a abordagem de DBC. Em [9] é proposta também uma metodologia para desenvolvimento de aplicações através da MDA que, apesar de basear-se no *framework* conceitual do RM-ODP, ainda não utiliza o EDOC. A utilização dos perfis propostos pela OMG garante, de forma mais ampla, a interoperabilidade de modelos proposta pela MDA. Metodologias e ferramentas de auxílio ao desenvolvimento de aplicações, que utilizem o EDOC ou outros sub-perfis propostos pela OMG, possuirão o mesmo arcabouço conceitual e notacional. Este aspecto facilita a leitura dos modelos tanto por equipes de desenvolvimento quanto pelas ferramentas de automação do processo.

Este artigo tem como objetivo apresentar a nossa experiência na modelagem e desenvolvimento do ambiente InterDoc (Ambiente de Suporte a Interoperabilidade entre Ferramentas de Autoria Colaborativa de Documentos) [11]. O InterDoc é um ambiente que promove interoperabilidade entre tarefas assíncronas de autoria colaborativa realizadas por autores que utilizam ferramentas distintas e heterogêneas. A autoria colaborativa é uma das áreas de grande interesse em relação ao suporte ao trabalho cooperativo [12].

Para se obter um modelo independente de plataforma, a arquitetura de referência é especificada através do EDOC e o ambiente de implementação é o CORBA/CCM. O relato da experiência será baseado em passos que foram realizados para a geração dos modelos através da seqüência de execução e das regras de mapeamentos aplicados. Da experiência com o desenvolvimento do InterDoc, este guia será aplicado em outros desenvolvimentos para a especificação de uma metodologia de desenvolvimento baseada no EDOC.

O texto está dividido da seguinte forma: na próxima seção o ambiente InterDoc é apresentado. Os principais passos da especificação das visões da Empresa, Computacional, Engenharia e Tecnológica são mostrados na Seção 3. A Seção 4 conclui este texto e apresenta os próximos passos do trabalho.

2 O Ambiente InterDoc

O InterDoc permite que as diversas atividades assíncronas do processo de autoria possam ser realizadas em ferramentas distintas. O InterDoc é uma camada de serviços verticais que, posicionada entre as aplicações de autoria colaborativa e os repositórios que armazenam estes artefatos, promove a interoperabilidade entre ambientes diversos. Grupos de autores poderão realizar sessões síncronas ou assíncronas de planejamento, rascunho, revisão, edição, etc de um documento no seu ambiente preferido. Em qualquer fase do processo de autoria, o documento pode ser disponibilizado através do InterDoc para outro grupo de autores que utiliza outro ambiente realizarem uma atividade.

A Figura 1 mostra um esquema para utilização do ambiente InterDoc através de um modelo de uso. Dois Domínios (domínios 1 e 2) representam comunidades distintas. Cada domínio possui suas aplicações CWSA (*Collaborative Writting Supporting Application*), aplicações Cliente A e B, que não são prontamente interoperáveis em relação às tarefas de autoria colaborativa e que formam a Camada das Aplicações. Para se tornarem interoperáveís, neste aspecto, as aplicações devem

possuir interfaces com o InterDoc (Interfaces 1 e 2). O modelo de utilização é formado por cinco camadas: Aplicações CWSA, Interação com Aplicações, Serviços, Comunicação entre Domínios, Interação com Repositórios e por fim a camada dos Repositórios.

A camada de Interação com as Aplicações é formada por componentes que implementam as interfaces entre as aplicações e o InterDoc, para a disponibilização de documentos e informações relativos às tarefas de autoria que estão sendo realizadas. A camada de Serviço é composta pelos objetos do InterDoc que implementam os requisitos funcionais. A camada de Comunicação entre Domínios é composta por elementos que são responsáveis pela comunicação entre instâncias do InterDoc hospedados em domínios distintos. Por fim, a camada de Interação com os Repositórios possui componentes para a interface com os repositórios que são responsáveis pela validação e disponibilização das informações que devem ser armazenadas de forma persistente.

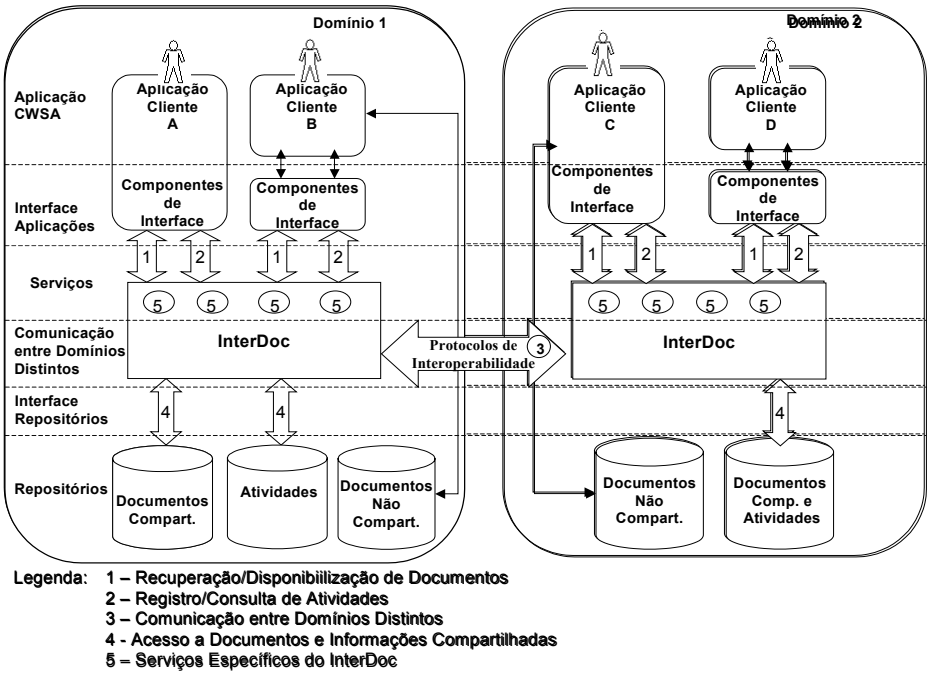

Fig. 1. Modelo de Uso do InterDoc.

Para o InterDoc a autoria colaborativa está dividida em duas grandes fases: Planejamento e Escrita. A fase de Planejamento é o momento em que um grupo deseja construir um documento e disponibilizar as informações básicas para a descrição do projeto. A fase de escrita é o momento em que autores disponibilizam versões do documento para outros autores realizarem atividades.

3 O Processo de Desenvolvimento do InterDoc

A especificação do InterDoc utilizou o conceito de visões do RM-ODP recomendado pelo EDOC. O PIM do InterDoc é composto das visões da Empresa, Informação e Computacional. O PSM é composto pelas visões da Engenharia e da Tecnologia. As visões podem ser especificadas de diversas formas. O processo de desenvolvimento do InterDoc foi iniciado pela especificação da visão da Empresa seguido da visão da Informação. A visão da Empresa fornece a necessidade inicial de informação dos processos de negócio. Para completar a especificação do PIM, modelou-se a visão Computacional. Esta visão tem como base a divisão dos processos da visão da Empresa e o modelo de dados da Visão da Informação para especificação dos componentes. O PIM fornece o modelo e a estrutura de componentes da aplicação para visão da Engenharia do PSM. Esta visão define as regras de mapeamento para a plataforma de implementação (figura 2). As retro-alimentações entre as diversas visões, por questões de escopo, não serão mostradas neste texto.

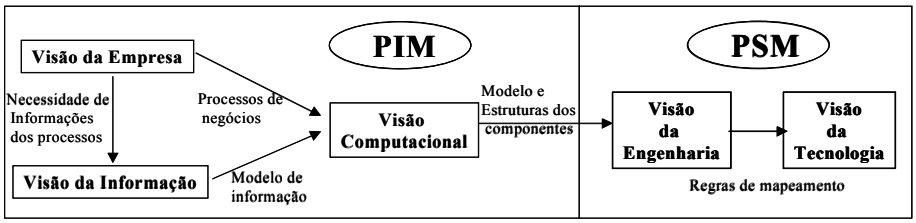

Fig. 2. Visão Geral da Especificação do PIM e PSM.

3.1 Visão da Empresa

A visão da Empresa modela a estrutura e o comportamento da aplicação no ambiente em que está inserido. Nesta visão devem ser identificados as comunidades, seus objetivos, processos de negócios e as principais restrições.

Passo 1 - Identificação dos objetivos e restrições: O Interdoc possui apenas um tipo de comunidade: a comunidade de Autoria. Uma comunidade de Autoria envolve os autores e as aplicações que os mesmos utilizam para elaborar os seus documentos.

O EDOC não especifica uma notação para identificar os objetivos de uma comunidade. Diagramas de Caso de Uso foram utilizados, pois, por serem de conhecimento de todos os desenvolvedores envolvidos, facilitaram a identificação dos objetivos. Dez casos de usos foram identificados: Define Projeto de Autoria, Define Grupo, Define Autor, Define Papéis, Define Atividades, Disponibiliza Documento para Realização de Atividade, Registra Atividade Realizada, Notifica Autores sobre Atividades, Recupera ou Salva Documento e Comunicação com outros Domínios.

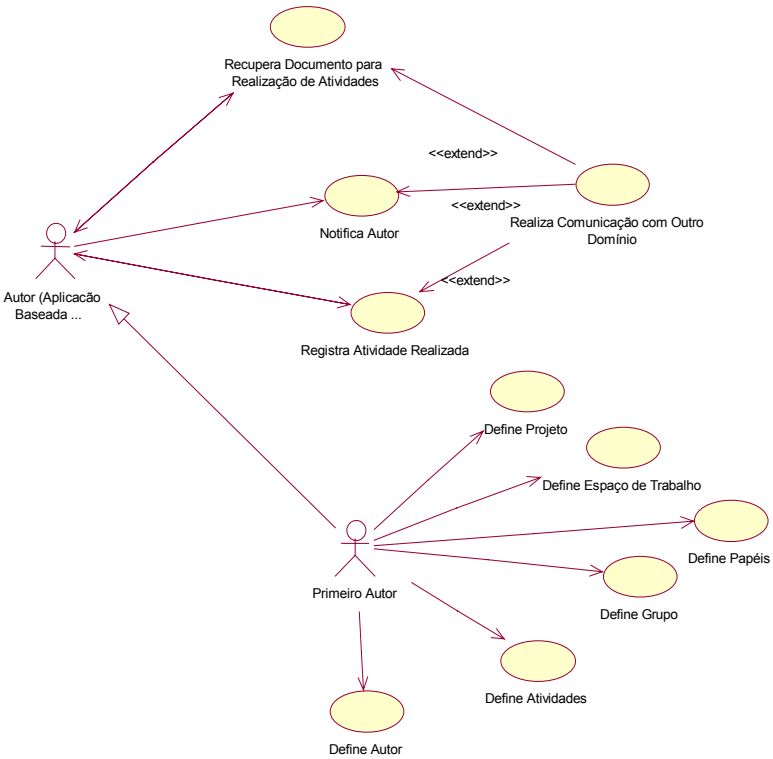

Fig. 3. Diagrama de Casos de Uso do InterDoc – Definição dos Objetivos.

Passo 2 - Identificação dos Processos de Negócio e Atividades: Após a definição dos objetivos, para identificar o comportamento da comunidade, o sub-perfil do EDOC, *Business Process Profile*, foi utilizado para identificar os processos de negócios (*Business Process – BP*), portas de comunicação e as atividades que são realizadas em cada um destes processos. Os casos de usos foram divididos em dois BP: *Design a Project* e *Writing*, que estão relacionados às fases da autoria colaborativa que o InterDoc provê suporte (Figura 3).

O BP *Design a Project* está subdividido em duas atividades: *Form a Group* e *Register a Project.* O BP *Writing* está subdividido em três atividades: *Delegate an Activity*, *Register an Activity* e *Notify.* As atividades foram por sua vez subdivididas em outras atividades até chegar a granularidade dos Casos de Uso identificados. Os dados necessários para a execução das atividades determinam as portas de entrada e saída dos BP e atividades, bem como a seqüência de execução das mesmas.

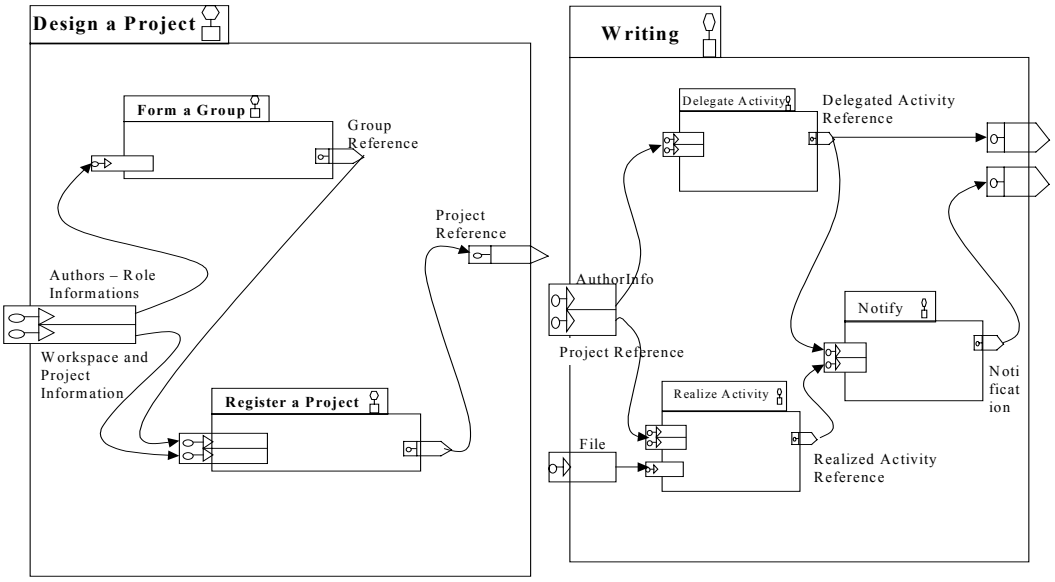

Fig. 4. Business Process *Design a Project* e *Writing*.

A Figura 5 mostra o diagrama da atividade *Register a Project*. Nestes diagramas são vistas as atividades que são pré condições e pós condições para a execução de uma determinada atividade. Para cada atividade deve ser determinado qual (ou quais) o componente (*Process Component)* que será responsável pela execução da atividade (*Performer*), qual serão os dados manipulados por esta atividade e quem será a parte responsável *(Artifact),* e que entidade (computacional ou não) será responsável por esta atividade (*ResponsabilityParty*).

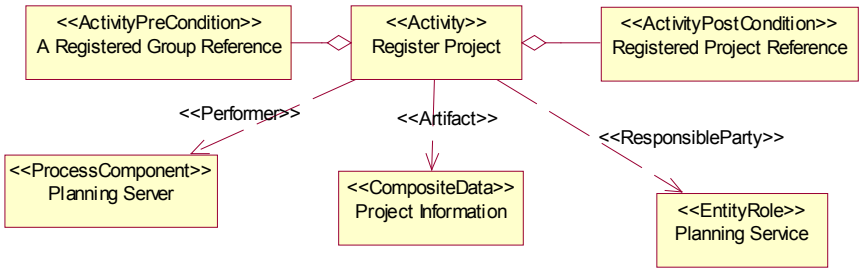

Fig. 5. Atividade Register Project retirada do Business Process Design a Project

Nesta fase da modelagem ainda não é possíveis determinar os componentes, dados e todos os responsáveis pela execução de uma atividade, pois os mesmos podem ser serviços. Isto só será possível no passo 1 da Visão Computacional. Portanto é indicado que se faça este diagrama apenas com as pré e pós condições, e que depois ele seja acrescido destas informações.

3.2 Visão da Informação

A visão da Informação modela entidades de dados e seus relacionamentos. Entidades (*Entity*) representam conceitos do domínio do problema. O modelo resultante define essencialmente o vocabulário utilizado na discussão do domínio do problema, que representa a estrutura dos objetos utilizados para representar os conceitos do negócio no ambiente computacional. Esta visão utilizou o sub perfil EDOC, *Entities Profile*. O *Entities Profile* permite que duas visões sejam dadas sobre os dados. Um conceito central do *Entities Profile* é uma *Entity Data*. Uma *Entity Data* é uma estrutura de dados que representa um determinado conceito do domínio do problema, ou seja, uma *Entity*. Uma *Entity Data* é equivalente a uma entidade ou a uma relação no modelo relacional.

Passo1 - Idendificação dos Entity Data e Composite Data: Os *Entitydata* e seus relacionamentos devem ser especificados através de um diagrama de classes (figura 6).

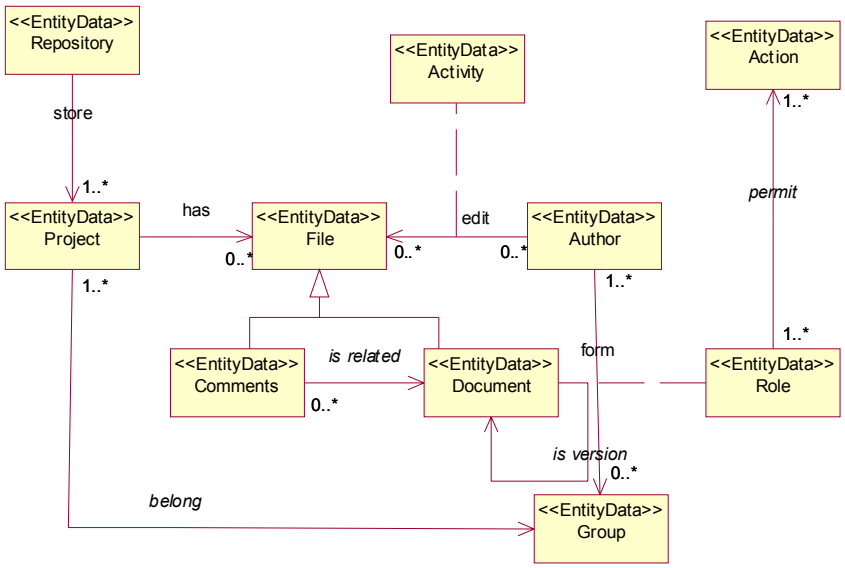

Fig. 6. Diagrama da Visão da Informação.

Para iniciar qualquer atividade de autoria colaborativa é necessário registrar um projeto no Interdoc. Um projeto possui um repositório de arquivos associado, pertencente a um determinado grupo de autores. Estes autores podem desempenhar papéis diferentes em diferentes projetos. Por exemplo, um autor pode ser líder em um determinado projeto e revisor em outro. Os papéis possuem associadas a eles determinadas ações. Ações são resultados que serão registrados no ambiente. Após revisar um documento, um autor pode salvar no repositório uma nova versão do mesmo. Ações são pré-determinadas pelo ambiente pois dependem de configurações

específicas. Uma ação de salvar um arquivo em um sistema de arquivo é diferente de salvar em um banco de dados.

Tabela 1. Descrição de cada *Entity Data* da Visão da Informação.

Entity Data	**Descrição**
Repository	Entidade responsável por armazenar, de forma persistente, todos os documentos de cada um dos projetos. Contém informações que indicam sua localização, tipo, etc
Project	Entidade responsável por conter as informações de cada um dos projetos que são criados, como, por exemplo, dados do criador, a data da criação do projeto e o nome do projeto.
File	Entidade abstrata que contém as características comuns às classes *Document* e *Comment*. Possui informações do conteúdo, data de criação, autor, projeto pertencente, etc. Enfim, essa classe é responsável por guardar as informações do arquivo a partir do momento em que ele é criado ou editado. Arquivos com anotações pertencem à classe *Document*.
Document	*Document* é a entidade que representa os documentos produzidos pelos autores em um determinado projeto. Além das informações do documento, o texto propriamente dito está associado a esta entidade.
Coment	*Comment* é a entidade que armazena os comentários que os autores realizam sobre determinado documento.
Group	Entidade que contém informações sobre os grupo que se formam para realizar um projeto. Possui informações sobre o nome do grupo, data de criação, autores que farão parte do grupo, etc.
Author	Entidade que contem informações sobre os autores (nome, endereço eletrônico, *login*, senha, etc) que elaboram documentos
Role	Entidade que associa para cada autor de um grupo, em cada projeto, um papel relativo à autoria colaborativa (primeiro autor, co-autor, revisor, comentarista, etc).
Activity	Entidade que contém as informações das atividades (delegadas e registradas) que estão sendo realizadas pelos autores em um documento em um determinado projeto.
Ação	Entidade que contém informações sobre as ações que um autor pode realizar em determinado arquivo (salvar, apagar, sobrescrever, salvar uma nova versão, bloquear, etc). Ações possuem sua semântica pré-determinadas pelo ambiente.

Autores revisam documentos produzindo comentários (ou anotações) ou versões destes documentos. Portanto um documento pode ser versão de outro, e um comentário está associado a um documento.

Uma vez definida as *EntityData* de um projeto, foram derivadas alguns *CompositeData*, como por exemplo: *GroupReference* e *ProjectReference* que são usados nas portas do BP *Design a Project*. Outros *CompositeData* podem ser acrescentados no momento da especificação da Visão Computacional.

3.3 Visão Computacional

A visão Computacional descreve a implementação da aplicação e dos seus componentes. Nesta visão são modelados os elementos que são responsáveis pela realização dos processos identificados na visão da Empresa. A modelagem da visão Computacional é realizada em uma decomposição funcional do sistema em termos de componentes que interagem através de interfaces. A estrutura dos componentes bem como os dados que os mesmos manipulam são mostrados nesta visão. O *Component*

Collaboration Architecture (CCA) foi o sub-perfil utilizado para a modelagem desta visão.

Passo1 - Identificação dos Componentes: Cada elemento do modelo de uso do InterDoc (aplicação, Interdoc e repositório), e conseqüentemente do modelo de composição da comunidade de Autoria (não mostrado neste texto), foram modelados como um ou mais *Process Component (PC)* (Figura 7). Um *PC* é uma unidade de processamento ativa que possui um conjunto de portas para interagir com outros *PC*. Um PC pode ser dividido em outros PC. Os serviços do InterDoc foram agrupados em seis PC*: DocumentSharingService* (gerencia o acesso aos repositórios), *PlanningService* (gerencia as informações sobre os projetos), *AuthoringActivitiesService (*registro e delegação das atividades), *ActivitiesNotificationService (*notificação das atividades delegadas e registradas)*; AuthorGroupService* (gerencia as informações de autores e dos grupos formados para realizar os projetos) e *CommunicationInterDomainService* (formatação de mensagens para comunicação entre InterDoc de domínios distintos).

Cada PC do InterDoc realiza uma ou mais atividades identificadas na visão da Empresa. Um PC realiza uma ou mais atividades com informações (portas) correlatas que definem uma seqüência de chamadas. Nesta fase da modelagem foram inseridos componentes para realização de requisitos não funcionais do InterDoc (*CommunicationInterDomainService e DocumentSharingService)*. A Figura 7 mostra o diagrama da visão geral da estrutura de componentes do InterDoc.

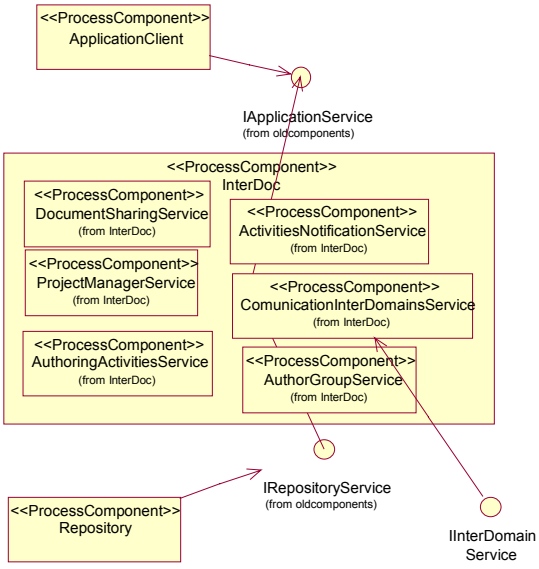

Fig. 7. Visão Geral da Estrutura de Componentes do InterDoc.

A arquitetura de referência do InterDoc também abrange a especificação de componentes da aplicação CSWA e do repositório de documentos, (respectivamente *Application* e *Repository*),que se comunicam com o mesmo. A comunicação do

InterDoc com estes componentes é especificada através de Interfaces *(IApplicationService e IRepositoryService)*.

Passo 2 – Identificação dos Protocolos de Comunicação e Modelagem da Estrutura dos Componentes: Parte da especificação de um PC é a modelagem do conjunto de protocolos que o mesmo implementa. Um protocolo especifica a seqüência de interações entre os componentes. Interações complexas entre componentes foram especificadas através de *Protocol Ports* (PP). PP definem uma conversação completa entre componentes. Cada interação entre atividades dos BP da visão da Empresa foi mapeada como um protocolo. A Figura 7 mostra o diagrama de estrutura do protoloco *RegisterProjectProtocol*, que define a visão computacional da atividade de registro de um projeto de autoria. Este protocolo especifica a interação entre os componentes *PlanningServer* e *ApplicationClient* para a realização da atividade *Register a Project* do BP *Design a Project*. Neste tipo de diagrama são especificados que (quais) componente inicia o protocolo e qual (quais) componente responde. A partir deste momento é possível completar o diagrama de atividades (Figura 5, seção 3.1) com a informação *Performer*. O PC *Planning Server* foi então definido como o responsável pela execução desta atividade.

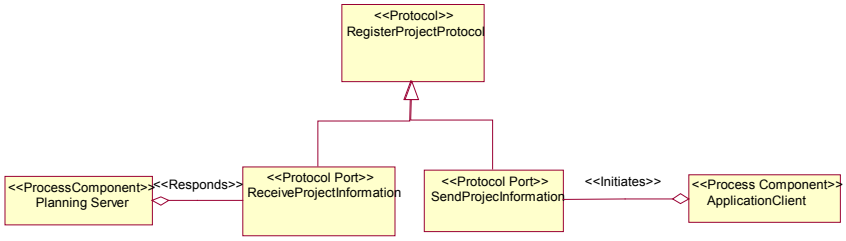

Fig. 8. Diagrama de Estrutura do Protocolo RegisterProjectProtocol

Após a modelagem de todos os protocolos, é possível definir a estrutura de cada componente. A estrutura de cada componente é composta das portas de entrada e saída mapeadas através dos estereótipos <<Responds>> e <<Initiates>>. A Figura 9 mostra o diagrama de estrutura do PC *Planning Server* com apenas os protocolos que o mesmo responde. O *Planning Server* responde a sete protocolos iniciados pelos componentes da aplicação de autoria e inicia a outros oito protocolos de comunicação como o *CommunicationInterDomainService, DocumentSharingServic e RepositoryClient.* Por exemplo: a PP *RegisterProjectInformation* é iniciada pelo PC *ApplicationClient* e a resposta é enviada pelo PC *Planning Server.* No InterDoc, pela organização dos componentes ser em camadas, a maior parte dos seus componentes internos (segunda camada), respondem a chamadas do componente *ApplicationClient* e iniciam chamadas para 'solicitação de serviços do componente *RepositoryClient*.

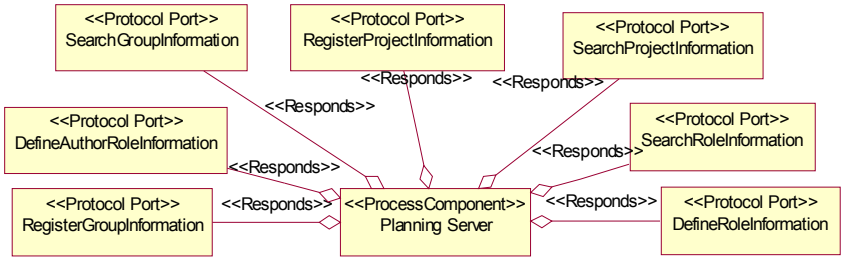

Fig. 9. Diagrama de Estrutura do Componente *Planning Server*

Passo 3 – Detalhamento da Modelagem dos Protocolos de Conversação entre Componentes: Para cada PP deve ser atribuído um protocolo de conversação entre os componentes. Um protocolo é então modelado em termos de uma ou mais portas que realizam a conversação entre os componentes. A Figura 10 apresenta o protocolo *ResgisterProjectProtocol*. Este protocolo utiliza portas do tipo *OperationPort.* Uma *OperationPort* representa o padrão de operação do tipo *Request/Reply.* Para cada porta de comunicação são indicados *CompositeData* (CD). CD mostram os parâmetros de entrada ou saída de cada porta. Os CD são extraídos dos elementos modelados na visão da Informação. A seqüência da execução das mensagens é mostrada através de diagramas de atividades.

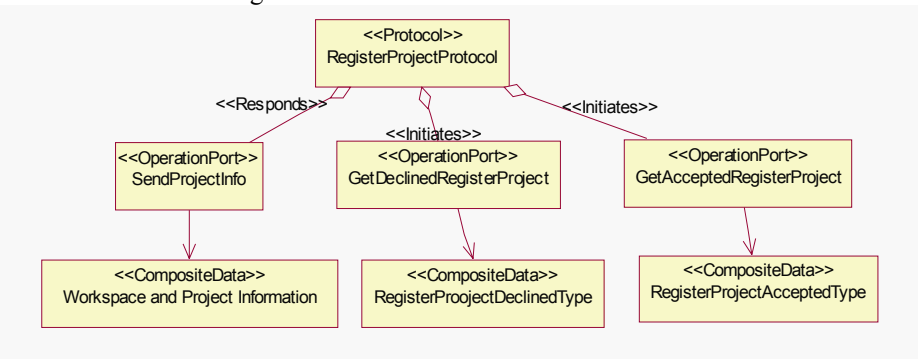

Fig. 10. Diagrama de Estrutura do Protocolo *ResgisterProjectProtocol*

3.4 Visão da Engenharia e Tecnologia

A especificação da visão da Engenharia é derivada da visão computacional e define regras de mapeamentos para a especificação dos componentes na tecnologia escolhida para a implementação da aplicação. A visão tecnológica especifica a estrutura dos componentes na tecnologia escolhida.

Passo 1- Definição das Regras de Mapeamento Para a Tecnologia de Implementação: A MDA disponibiliza alguns perfis para a modelagem dos PSM. A tabela 2 apresenta as regras de mapeamentos que foram utilizadas para os conceitos apresentados neste texto. Para cada tipo de elemento da visão computacional, que não

foi decomposto em outros, define-se uma regra de mapeamento. O perfil UML para a tecnologia CORBA/CCM, até o momento da escrita deste texto, está no processo de revisão final para se tornar uma especificação da OMG. Os estereótipos CCM foram retirados da proposta deste perfil.

Tabela 2. Regras de Mapeamento para a plataforma CCM.

EDOC	Estereótipo CCM	Descrição do Mapeamento
ProcessComponent	<<CORBAComponent>>	Cada ProcessComponent pode ser mapeado para um ou mais Componentes CORBA.
ProtocolPort	<<CORBAInterface>>	Para comunicação síncrona. Pode ser usada em *two-way direction*.
	<<CORBAEvent>>	Para comunicação assíncrona. Apenas *one-way direction*.
OperationPort	<<CORBAProvides>>	Faceta do CCM. Indica uma interface provida que responde (*responds*) à comunicação.
	<<CORBAUses>>	Receptáculo do CCM. Inicia (*initiates*) a comunicação através de uma conexão com uma interface provida por outro componente.

Passo 2 - Modelagem e Escrita dos Componentes na Tecnologia de Implementação: Cada componente identificado deve ser modelado para a devida apresentação dos serviços que o mesmo disponibiliza para os demais componentes. A Figura 11 demonstra uma das relações existentes entre os componentes *ApplicationClient* e *PlanningServer* através do diagrama de classes. O componente *ApplicationClient* usa a interface que é provida pelo componente *PlanningServer*. Desse modo, e de acordo com a tabela 2, as *OperationPort* são mapeadas para CORBAUses (lado aplicação - cliente) e CORBAProvides (lado servidor). O Protocolo *RegisterProject* torna-se então uma interface CORBA (CORBAInterface da tabela 2).

Um componente CCM é composto de portas. As portas podem ser: facetas, receptáculos, consumidor de eventos, e receptor de eventos. A Figura 11 mostra o código IDL3 (*Interface Definition Language*) que define: a Interface CORBA *RegisterProject* e a sua assinatura; o componente *PlanninServer*, bem como a sua faceta (interface *RegisterProject* provida); e o componente *ApplicationClient* que utiliza a interface como um receptáculo através do comando *uses*.

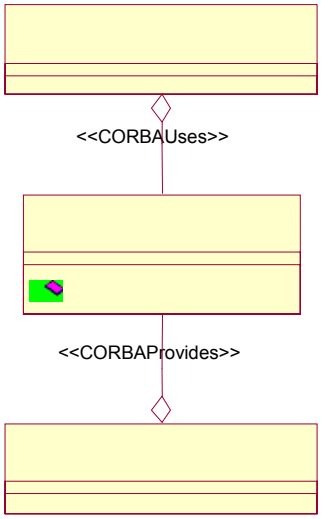

Fig. 11. Diagrama de classes da interação entre os componentes *PlanningServer* e *ApplicationClient* através de uma interface CORBA.

O *home* dos componentes declarados no código abaixo é um recurso do CCM que tem a funcionalidade de gerenciar instâncias de componentes. Por isso ele deve ser obrigatoriamente declarado, mesmo que de forma simples, nas plataformas de desenvolvimento em CCM, não cabendo a sua explicação no contexto deste texto.

```
Module InterDoc
{
      interface RegisterProject
        {
            boolean register_project(in string id_project, in string name, in
string description, in string dataCriacao, in string dataTermino, in string
id_owner, in string extensions, in string id_group);
         };

      component PlanningServer
      {
          provides RegisterProject facetRegisterProject;
      };

      home PlanningServerHome manages PlanningServer
      {
      };

      component ApplicationClient
      {
          uses RegisterProject ReceptacleRegisterProject;
      };
      home ApplicationHome manages Application
      {
      };
```

Fig. 12. Código IDL3 dos Componentes *PlanningServer* e *ApplicationClient*

4 Conclusão e Trabalhos Futuros

Este artigo apresentou o relato do processo de desenvolvimento do ambiente InterDoc. Para se obter um modelo independente de plataforma, a arquitetura de referência do InterDoc foi especificada através da MDA, utilizando o perfil EDOC.

A abordagem do desenvolvimento de aplicações, através da MDA, tem como atividade principal a produção de modelos de sistemas através de perfis UML. Os perfis MDA disponibilizam elementos apenas para a especificação das aplicações, que não acompanham metodologias para o desenvolvimento de um *software*. Metodologias auxiliam a produtividade na medida que processos de desenvolvimento semelhantes podem ser aplicados a vários sistemas. Desenvolvedores necessitam de metodologias para facilitar o seu trabalho, deixando-os livres para focarem atenção nos requisitos funcionais da aplicação.

A utilização do EDOC como sub-perfil para a produção de modelos auxiliou a abordagem do DBC, pois o InterDoc foi concebido como um conjunto de componentes desde as fases iniciais do projeto. Sendo assim, o mapeamento dos componentes para a plataforma CCM foi facilitado devido à utilização da mesma abordagem.

O relato do processo de desenvolvimento, através de uma seqüência de passos e guias para o mapeamento de conceitos, é uma etapa para a especificação de uma metodologia de desenvolvimento de aplicações baseadas em componentes que utilizem como *framework* conceitual o EDOC. Estes passos e guias estão sendo utilizados em outras aplicações para revisão e consolidação da metodologia de desenvolvimento. Após a consolidação, a metodologia será aplicada no desenvolvimento de aplicações utilizando ferramentas MDA, que apóiam automação do processo de desenvolvimento de aplicações.

5 Referências

1. BERNSTEIN, Philip A. Middleware: A Model for Distributed System Services. *Communication of the ACM*. New York. v 39. n 2. p 86-98. February 1996.
2. CHEESMAN, J.; DANIELS, J. UML Components. A simple Process for Specifying Component-based Software. Addison-Wesley, 2000.
3. OBJECT MANAGEMENT GROUP. *CORBA Component Model v3.0 full specification.* OMG Adopted Specification (formal/02-06-05).2002.
4. SHANNON, Bill. *Java 2 Platform Enterprise Edition Specification v 1.4*. [s.l.]. Sun Microsystems, 2003.
5. OBJECT MANAGEMENT GROUP. *MDA Guide Version 1.0*. 2003
6. GUELFI, Nicolas; RIES, Benoît; STERGES, Paul. *MEDAL: A Case Tool Extension for Model-Driven Software Engineering*. In: IEEE International Conference on Software – Science, Technology & Engineering *(SwSTE'03),* 2003, Israel. Proceedings… November 2003 p.33-42.
7. OBJECT MANAGEMENT GROUP. *UML Profile for Enterprise Distributed Object Computing Specification.* OMG Adopted Specification (ptc/02-02-05). 2002.

8. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO. *Basic Reference Model of Open Distributed Process, ISO/IECIS* 10746. Partes 1-4, 1995.
9. GERVAIS, Marie-Pierre. *Towards MDA-Oriented Methodology*. In: Annual International Computer Software and Applications Conference, 2002, England, Proceedings …,August 2002, p 265-270.
10. WANG H.; ZHANG, D. MDA-based Development of E-Learning System, In: 27th International Computer Software and Applications Conference, 2003, IEEE Press, Nov., p. 684-689.
11. MACIEL, Rita. FERRAZ, Carlos. ROSA, Nelson. A Model-Driven Architecture for Interoperable Collaborative Writing Environments. In: X International Workshop on Groupware. Doctoral Colloquium. Costa Rica. September 2004.
12. CERRATTO, Teresa. *Instrumenting Collaborative Writing and Its Cognitive Tools*. In: Human Centered Process Conference, 1999, France. Proceedings… September. 1999. p.41-147.