AgileOOHDM

João Felipe Santos Condack
condack@rdc.puc-rio.br

Daniel Schwabe
dschwabe@inf.puc-rio.br

PUC-RioInf. MCC46/04 December, 2004

Abstract: This technical report describes an initial proposal of AgileOOHDM. It is an adaptation of a traditional software engineering based method, OOHDM, considering the Agile approach, more specifically its reflexes in Extreme Programming. This initial proposal was born from a study involving conceptual models of development process, commented in this work too. It is also intended to contribute with the debate between lightweight versus heavyweight methods as the proposal matures.

Keywords: Software development process, OOHDM, agile methods.

Resumo: Este relatório técnico descreve a proposta inicial do AgileOOHDM. Trata-se da adaptação de um método tradicional de engenharia de software, o OOHDM, considerando uma abordagem Ágil, mais especificamente seus reflexos em *Extreme Programming*. Esta proposta inicial nasceu de um estudo envolvendo modelos conceituais de processos de desenvolvimento, também comentado neste trabalho. É também pretendido contribuir com a discussão entre métodos leves *versus* pesados conforme a proposta for evoluindo.

Palavras-chave: Processos de desenvolvimento de software, OOHDM, métodos ágeis.

Sumário

1 Introdução

Este relatório técnico é o resultado de um estudo realizado durante o primeiro semestre de 2003, pelo aluno de mestrado em informática João Condack, sob a orientação do Professor Daniel Schwabe. Neste estudo foram abordadas questões relativas a Agile Manifesto[1] e OOHDM. [2] no sentido de saber como o método OOHDM se comportava com relação as premissas do manifesto. Basicamente trata-se de uma reflexão entre abordagens de desenvolvimento de software.

Para realização deste estudo, inicialmente foram modelados dois processos de desenvolvimento de software: OOHDM e XP. O OOHDM - Object-Oriented Hypermedia Design Method - é um baseado na engenharia de software tradicional enquanto o XP - Extreme Programming tem suas raízes no Agile Manifesto. O objetivo desta modelagem foi ter uma visão clara, não ambígua e estruturada de métodos para poder traçar comparações e fornecer uma base comum mais exata para discussão.

Em seguida, uma proposta inicial para um novo método de desenvolvimento foi concebida. Ela resume-se em uma adaptação de um método de engenharia de software tradicional – OOHDM – considerado a abordagem Agile, mais especificamente seus reflexos no Extreme Progamming.

Além das próprias conclusões tiradas durante a elaboração do novo método em si, dois casos testes foram executados para avaliar a viabilidade do novo método. Estes experimentos permitiram uma melhor visualização e compreensão do processo de desenvolvimento que resultou em sua análise final e conclusão deste estudo.

Ainda na introdução será apresentada a motivação deste estudo. No capitulo 2 serão vistas as modelagens dos dois métodos de desenvolvimento, OOHDM e XP. O capítulo 3 mostra a elaboração da proposta inicial do AgileOOHDM e em seguida no capítulo 4 resume a aplicação do novo método em dois casos teste. Finalmente no capítulo 5 são apresentadas as conclusões deste relatório técnico.

1.1 Motivação

Generalizando o que foi dito no International Workshop on Time-Constrained Requirements Engineering em 2002 na Alemanha, até o momento, a engenharia de software tradicional e os métodos Agile não parecem estar conectados. Algumas das questões que permanecem sem resposta incluem: quando e qual abordagem deve ser usada? Como ambas as abordagens podem ser combinadas? Nós esperamos colaborar na resposta para estas perguntas.

A engenharia de software tradicional tem trabalhado com a idéia de que num ciclo de desenvolvimento de software um sistema pode ser especificado e desenhado antes da implementação começar. Não obedecer a um planejamento rigoroso definido por um

processo de desenvolvimento pesado pode causar problemas no orçamento, atrasos e outras falhas de projeto. Devido a falta de tempo e a grande pressão para entrega de resultados ao cliente, é comum que este conhecimento, já amadurecido, seja deixado para traz e caminhos alternativos sejam seguidos. Um destes caminhos é o de soluções ad hocs, porém um segundo meio que vem ganhando importância e notoriedade na indústria e na academia são as abordagens ágeis como Extreme Programming.

1.1.1 Engenharia de Software Tradicional

Por volta dos anos 60 o mundo da ciência da computação começou a ficar alerta com a eminência de uma crise de software. Em 68 o comitê de ciência da OTAN patrocinou uma conferência para discutir as questões desta crise e a necessidade de foco no desenvolvimento de software. Naquele momento foi cunhado o termo “engenharia de software”. De lá até hoje ele evoluiu para uma definição mais formal como "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software" de acordo com a IEEE. Como dito em [3] "the basic objective of software engineering s to: Develop methods and procedures for software development that can scale up for large systems ant that can be used to consistently produce high-quality software at low cost and with a small cycle time".

Desde então novos métodos e procedimentos foram construídos e testados, mas a maioria deles são processos pesados que desencorajam a criatividade e levam a soluções sobrecarregadas. Isto provavelmente é um efeito de uma crença comum entre programadores, cientistas de computação e capitães da industria de que o custo de uma mudança cresce exponencialmente de acordo com a fase de desenvolvimento [4]. Portanto metodologistas têm desenhado métodos altamente focados nos requisitos e design e menos preocupados com aspectos de codificação e teste na esperança de encontrar os problemas o mais cedo possível.

1.1.2 OOHDM

O Object-Oriented Hypermedia Design Method – OOHDM – é um método proposto por Gustavo Rossi e Daniel Schwabe[2] que se serve bem para sistemas de aplicações baseados na web. Este tipo de software envolve complexas estruturas hipermídia que manejam dados multimídia. Apesar de ter princípios orientados a objeto, OOHDM não restringe a implementação em termos de hardware e software. OOHDM é claramente um processo de desenvolvimento que segue a engenharia de software tradicional como será mostrado mais adiante no capítulo 2. O processo de desenvolvimento define as seguintes fases: Planning and Requirements Gathering; Conceptual Modeling; Navigational Modeling; Interface Modeling; Implementation; Evaluation and Maintenance.

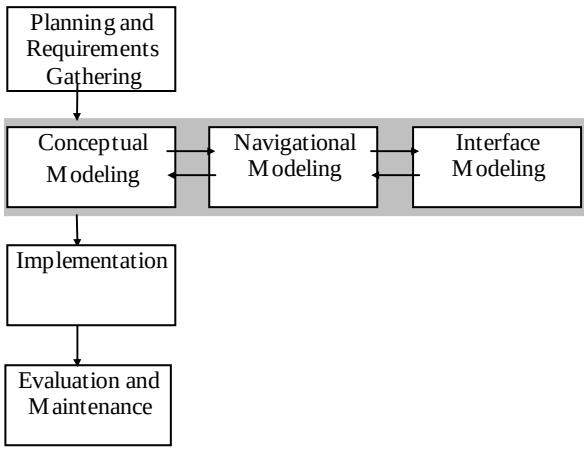

Figura 1 - Fases do OOHDM

1.1.3 O Manifesto Agile

Infelizmente ao passo que engenharia de software tradicional evoluiu nenhuma saída para a crise foi encontrada. De acordo com o *Uniform Computer Information Transactions Act - UCITA* - um comentário oficial da *USA national conference of commissioners on uniform state laws,* "It is often literally impossible or commercially unreasonable to guarantee that software of any complexity contains no errors that might cause unexpected behavior or intermittent malfunctions, so-called *bugs.*"[5]. Este anúncio foi corroborado por pesquisas como o *The CHAOS Report* [6] do *The Standish Group*.

No intuito de mudar o presente cenário, foi publicado o manifesto Agile, que é altamente focado em "Indivíduos e interações ao invés de processos e ferramentas; Software rodando ao invés de documentação compreensível; Colaboração do cliente ao invés de negociação de contratos; resposta a mudanças ao invés de seguir um plano". Alguns princípios básicos que contrastam com o senso comum da engenharia de software são: a boa aceitação da alteração de requisitos em qualquer fase de desenvolvimento; a entrega freqüente de pedaços de softwares rodando ao invés do sistema inteiro de uma vez; arte de maximizar o total de trabalho a não se fazer em nome da simplicidade; e um consenso de que o processo de desenvolvimento deve ser orientado as pessoas.

1.1.4. Extreme Programming

Extreme Programming é um processo de desenvolvimento ágil idealizado por Kent Beck [7] que visa pequenas equipes de TI em um ambiente de constante alteração de requisitos. Ele é baseado em quatro valores: comunicação, simplicidade, feedback e coragem. A partir destes valores uma série de princípios é derivada, como por exemplo: Feedback rápido, simplicidade assumida, mudança incremental e trabalho de qualidade. Estes princípios são instanciados através de várias melhores práticas e um ciclo de vida de projeto definido.

Um ciclo de vida de um projeto XP é guiado pelo *planning game*. Este, por sua vez, é composto de três fases seqüenciais: *Exploration*, *Commitment* e *Steering*. É na fase *Steering* que ocorre a codificação necessária para o sistema. Ela tem um processo chamado *Interaction* que obedece a um planejamento. O planejamento da interação é também composto pelas três fases - *Exploration*, *Commitment* e *Steering* – porém contendo as atividade apropriadas para *Interaction*

1.1.5 Reunindo Idéias

De acordo com o *The CHAOS Report* [6] a probabilidade de sucesso do desenvolvimento aumenta devido a intervalos menores entre os requisitos e o programa efetivamente rodando, além da entrega de componentes freqüentemente e cedo. Isto é uma mudança de "desenvolvimento" para "maturação" de um software. Adicionando-se a necessidade de métodos orientados a pessoas nos temos uma base potencial para construção de novos processos de desenvolvimento[8]. Talvez com nestes novos métodos seja possível atingir o "sweet spot in the middle"[9] para equilibrar as forças da geração de software e e então chegar mais perto de uma bala de prata para a crise de software[10].

Concordando que estamos desenvolvendo software em ambientes de constante mudança, o que claramente afeta seus requisitos, e sendo estes novos métodos acima uma possível solução nós decidimos procurar por um novo caminho também.

2 Modelando Métodos de Desenvolvimento de Software - Alguns Estudos de Caso

Dois estudos de casos foram feitos no intuito de ter uma visão mais clara e estruturada dos métodos OOHDM e XP. O propósito também foi construí-los sobre uma base canônica para a partir daí traçar comparações e fazer modificações. Inicialmente a motivação era descobrir como os processos tratavam a teoria do desenvolvimento baseado em componentes. Porém o avançar dos estudos mostrou uma utilidade potencial caso os métodos fossem combinados. Assim nasceu a idéia do AgileOOHDM.

Para obter uma descrição comum do OOHDM e do XP foi decidido que se devia modelá-los em algum modelo conceitual. Então alguns modelos foram pesquisados, tais como PROC [11], IMPACT [12] e um estudo sobre modelos conceituais para processos de desenvolvimento encontrados em [13]. A partir das pesquisas feitas, escolheu-se ter três tipos de diagramas baseados nas definições dadas pelo trabalho de Olsina. São eles: Diagrama de Agentes e Papeis, Diagrama de Artefatos, Diagramas de Fases e Processos.

O Diagrama de Agentes e Papeis mostra os agentes que realizam processos como pessoas, ferramentas ou sistemas de software. Para fazê-lo eles necessitam incorporar um papel que distingue as características e habilidades necessárias para executar um processo.

O Diagrama de Artefatos apresenta os artefatos como produtos de entrada e saída de tarefas. Um artefato é criado, evoluído, mantido ou destruído durante um processo de desenvolvimento.

Os Diagramas de Fases e Processos mostram as relações entre fases e processos onde: fase é um agrupamento de processos de software fortemente relacionados conduzidos em algum tipo de ordem; e processo é um conjunto parcialmente ordenado de subprocessos que tem recursos, agentes, condições, artefatos e construtores associados com a finalidade de produzir resultados de acordo com alguma meta.

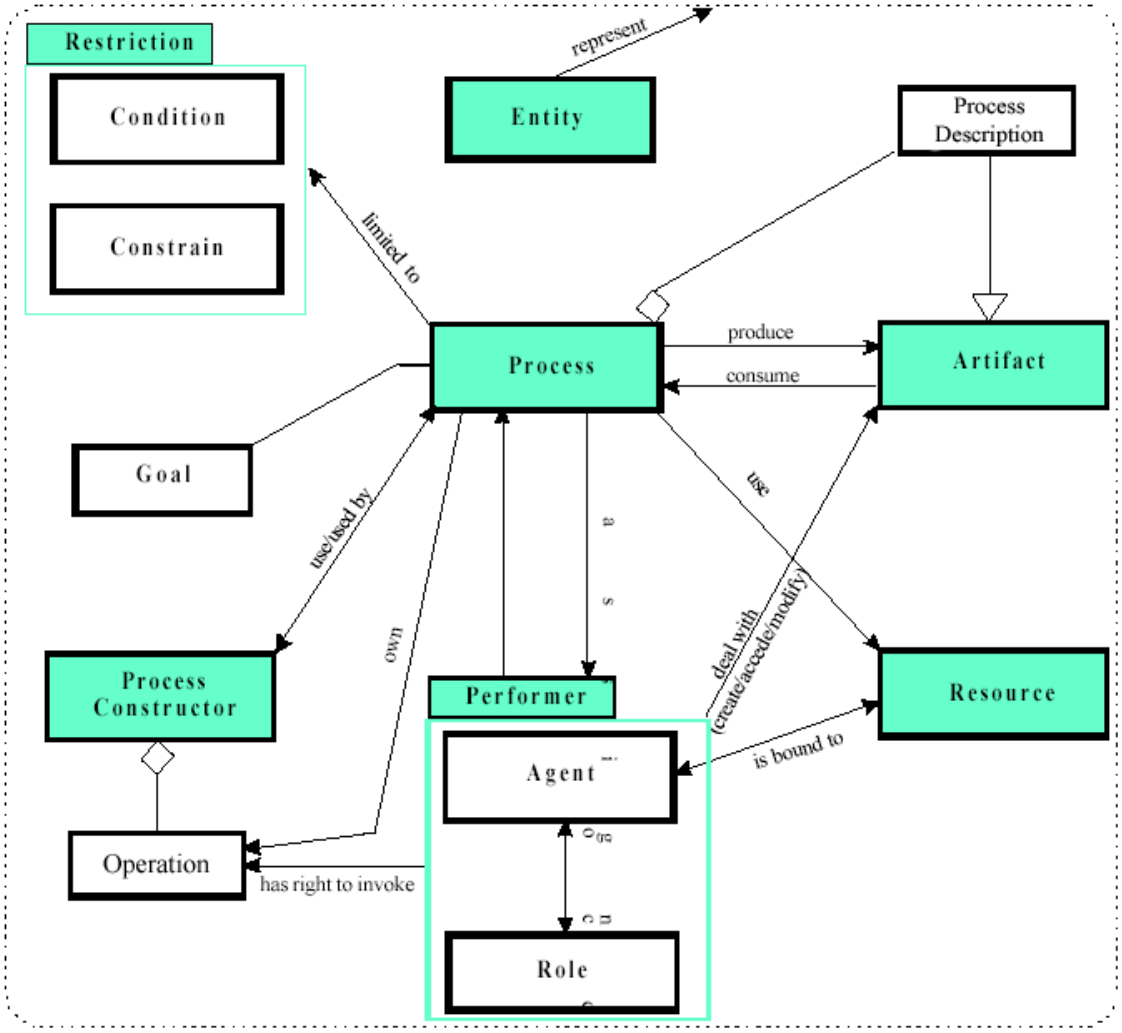

Figura 2 – Diagrama de Classes Principal para um Modelo de Processos Genérico [13]

2.1 Estudo de caso: XP

2.1.1 Um modelo conceitual para XP

Este modelo foi gerado com base em interpretações pessoais do método XP feitas sobre [7], [14], [15] e [16]. Em alguns casos, quando necessário, a rationale do processo de análise será exposta. A seguir os seguintes diagramas serão apresentados:

- Diagrama de Agentes e Papeis
- Diagrama de Artefatos
- Diagramas de Fases e Processos

2.1.2 Diagrama de Agentes e Papeis

No diagrama abaixo estaremos apresentando os agentes e papeis humanos definidos para XP. Os demais tipos de agentes não serão abordados uma vez que o método não faz menção a ferramentas ou softwares em sua concepção, sendo portanto uma escolha no momento de instanciação do mesmo.

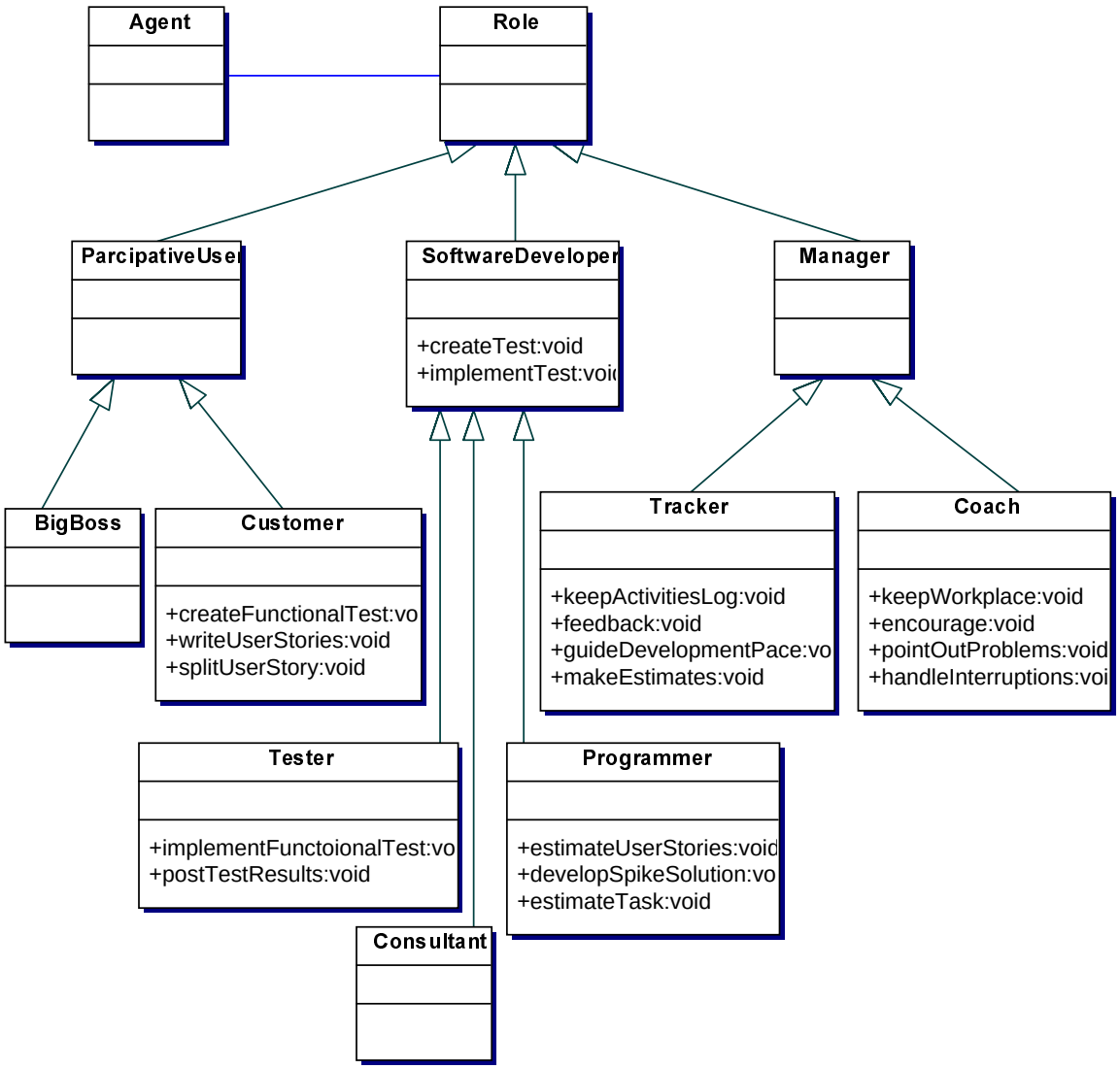

Figura 3 - Diagrama de Agentes e Papeis

Resumo das Classes	
Agent	Agent is the executer of a process.
BigBoss	Usually is the financial owner of the system.
Coach	Responsible for the process as a whole.
Consultant	Called when the team needs deep technical knowledge.
Customer	Responsible for telling programmers what to program.
Manager	Superclass for management staff.
ParcipativeUser	Superclass for users.
Programmer	The heart of XP.
Role	Role is a set of characteristics that is associated to an agent when accomplishing a specific task.
SoftwareDeveloper	Superclass for technical/development personnel.
Tester	Focused on the customer, helps him to choose and write functional tests.
Tracker	Is the team historian.

2.1.3 Diagrama de Artefatos

Abaixo apresentamos os artefatos presentes no extreme programming.

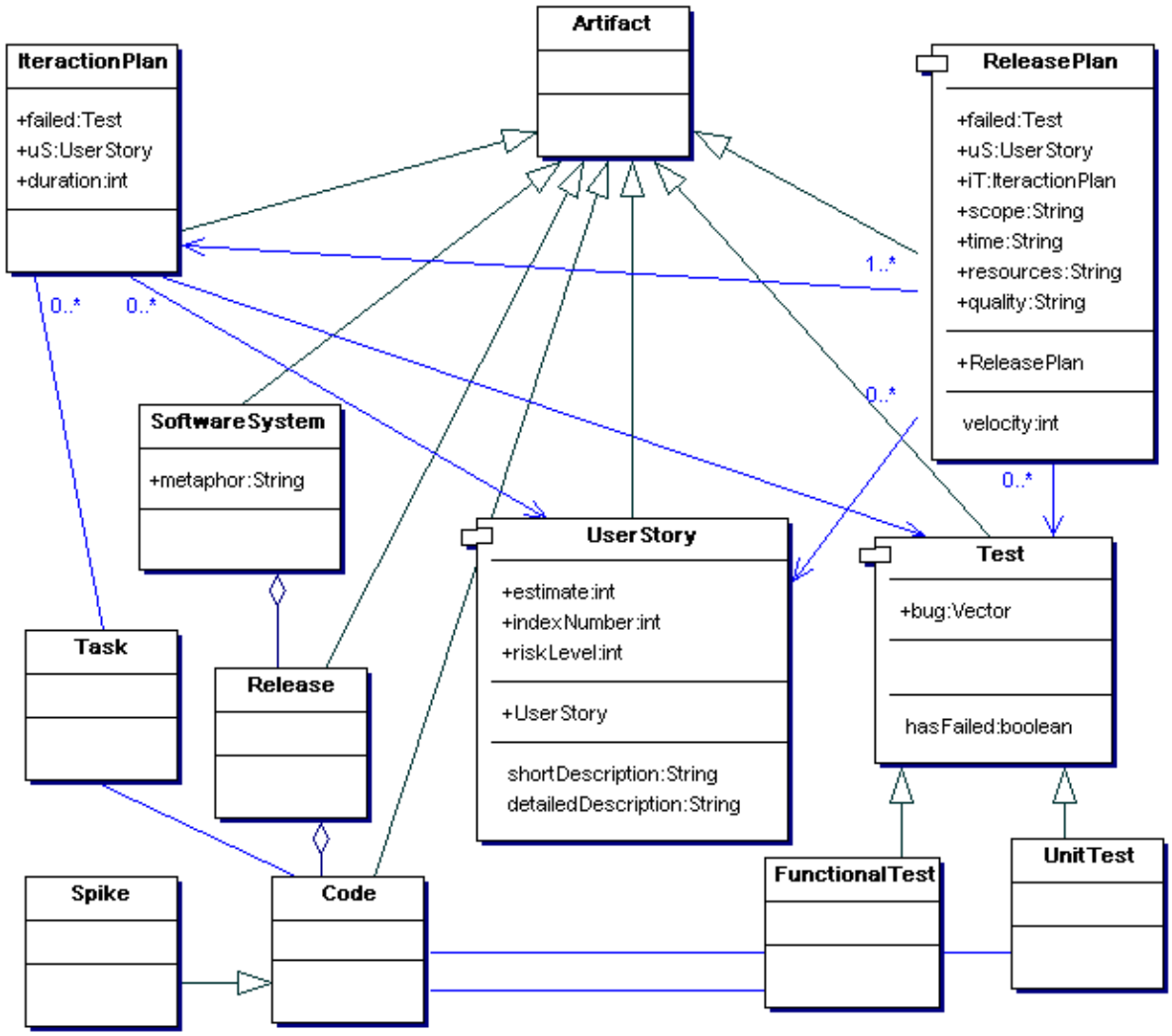

Figura 4 - Diagrama de Artefatos

Resumo das Classes	
Artifact	Artifact is an output of performing some task.
Code	Code is system's data and operations translated in some programming language.
FunctionalTest	Functional tests are created from user stories.
IteractionPlan	A set of stories and its derived programming tasks that should be done in one interaction.
Release	A pile of stories that make a business sense converted in running code.
ReleasePlan	The release plan specifies exactly which user stories are going to be implemented for each system release and dates for those releases.
SoftwareSystem	Final product of a software development process.
Spike	Spike is a simple try of some technical issue.
Task	Task is an indivisible implementation step.
Test	Test is a statement, rule or convention which a program or part of it must be conformed.
UnitTest	Unit test is a test written from the perspective of the programmer.
UserStory	User Stories are written by the customers as things that the system needs to do for them.

2.1.4 Diagramas de Fases e Processos

O decorrer do desenvolvimento XP é guiado pelo jogo do planejamento ou *planning game*. Ele é composto por três fases consecutivas: *Exploration*, *Commitment* e *Steering*. Na fase de *Steering* ocorrem as codificações necessárias ao sistema. Esta fase possui um processo denominado *Interaction* que obedece a um *interaction planning* que também é composto pelas três fases *Exploration*, *Commitment* e *Steering*, porém, com os processos apropriados a *Interaction*.

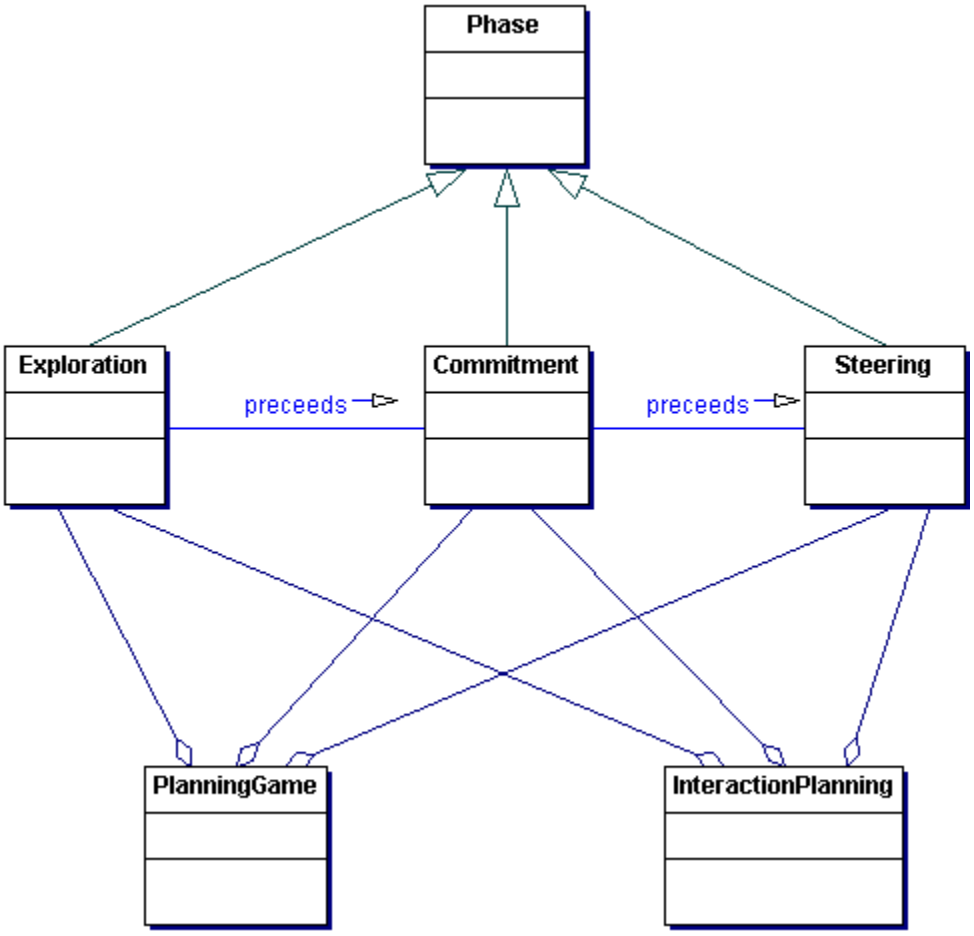

Figura 5 – Fases de XP e suas relações com PlanningGame e InteracionPlanning

No intuito de facilitar a compreensão das fases e processos de *extreme programming*, estes serão divididos, de acordo com o exposto acima, nos seguintes diagramas:

- Diagrama de Fases e Processos 1 – Fase *Exploration* do *planning game*.
- Diagrama de Fases e Processos 2 – Fase *Commitment* do *planning game*.
- Diagrama de Fases e Processos 3 – Fase *Steering* do *planning game*.
- Diagrama de Fases e Processos 4 – Fase *Exploration* do *interaction planning*.
- Diagrama de Fases e Processos 5 – Fase *Commitment* do *interaction planning*.
- Diagrama de Fases e Processos 6 – Fase *Steering* do *interaction planning*.

2.1.4.1 Diagrama de Fases e Processos 1 – Fase *Exploration* do *planning game*.

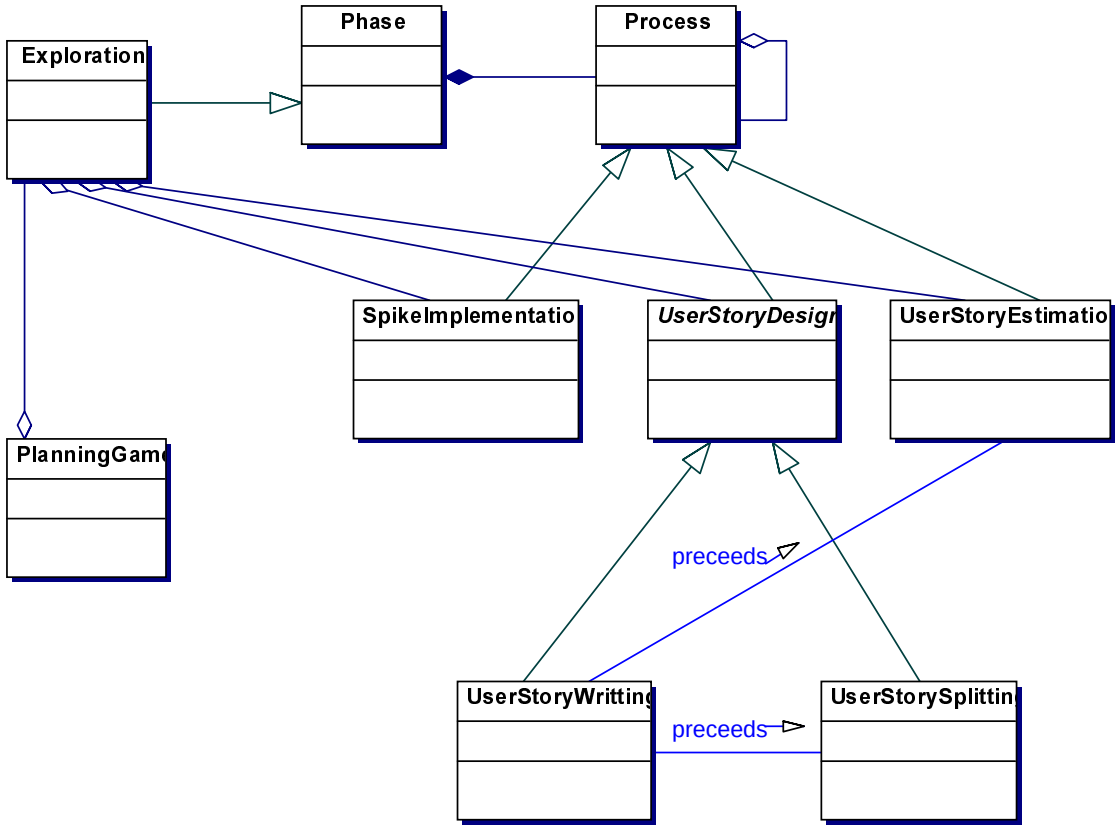

Figura 6 - Diagrama de Fases e Processos 1 – Fase *Exploration* do *planning game*.

Resumo das Classes	
Exploration	The phase of development when the customer and programmers start to deal with the problem.
Phase	Phase is group of software processes tightly related which is conduced in some kind of order.
PlanningGame	Planning Game is the XP planning process. It is where customers get to specify what the system needs to do and developers specify how much each feature costs.
Process	A software process is a partially ordered set of subprocess which has resources, agents, conditions, artifacts and constructors associated in order to produce derivables according to some goal.
SpikeImplementation	A spike solution is a very simple program to explore potential solutions.

UserStoryDesign	Customers and developers manipulate stories. Abstract class.
UserStoryEstimation	Developers estimate how long the stories might take to implement.
UserStorySplitting	If a story can't be estimated due to its size or it contains more than one important subject it should be splited by developers or customers.
UserStoryWritting	User Stories are written by the customers as things that the system needs to do for them.

2.1.4.2 Diagrama de Fases e Processos 2 – Fase *Commitment* do *planning game*.

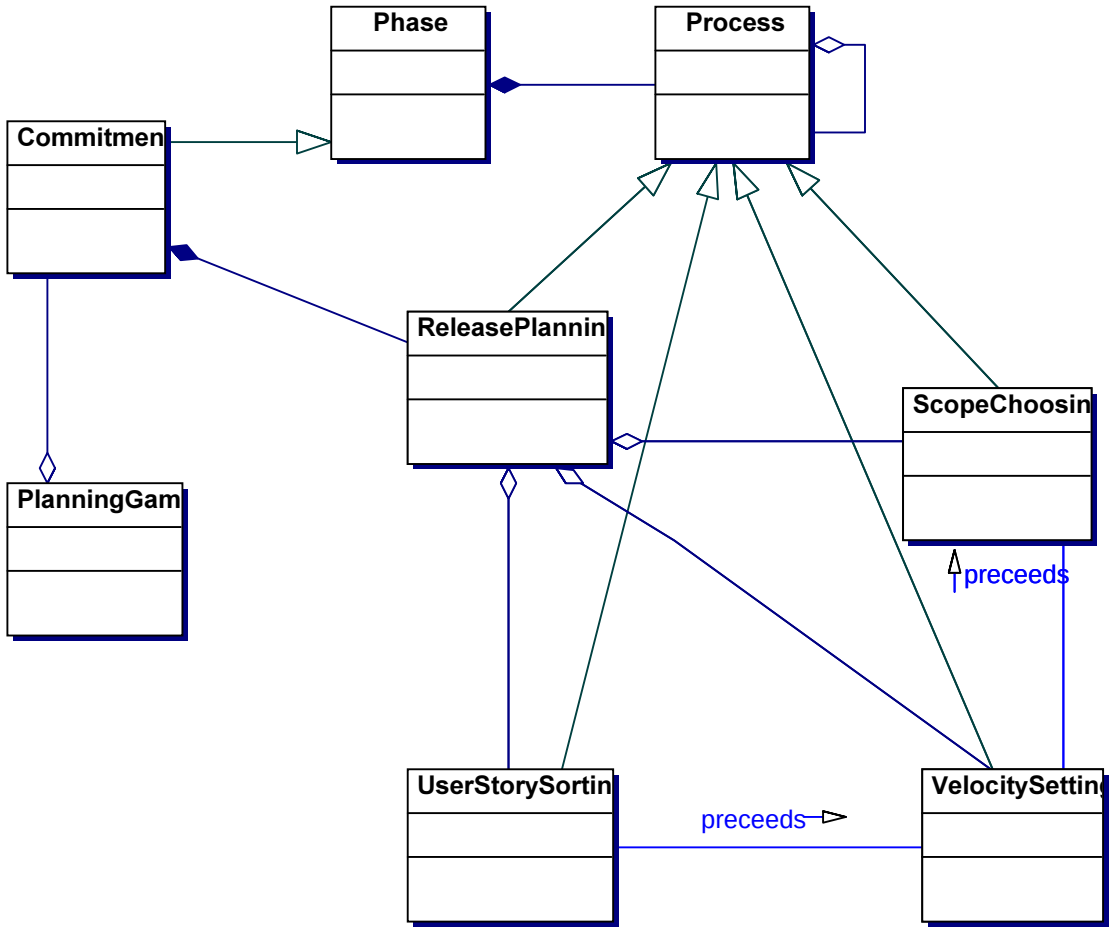

Figura 7 - Diagrama de Fases e Processos 2 - Fase Commitment do planning game

Resumo das Classes	
Commitment	Phase where players accept their responsibilities.
Phase	Phase is group of software processes tightly related which is conduced in some kind of order.
PlanningGame	Planning Game is the XP planning process.
Process	A software process is a partially ordered set of subprocess which has resources, agents, conditions, artifacts and constructors associated in order to produce derivables according to some goal.
ReleasePlanning	A release planning meeting is used to create a release plan, which lays out the overall project.
ScopeChoosing	Customer chooses the set of cards in the release, either by setting a date for engineering to be complete and choosing cards or choosing cards and calculating the date.
UserStorySorting	The customer sorts the stories by value or by risk in order to achieve a development sequence.
VelocitySetting	Developer tells customers how fast the team can program.

2.1.4.3 Diagrama de Fases e Processos 3 – Fase *Steering* do *planning game*.

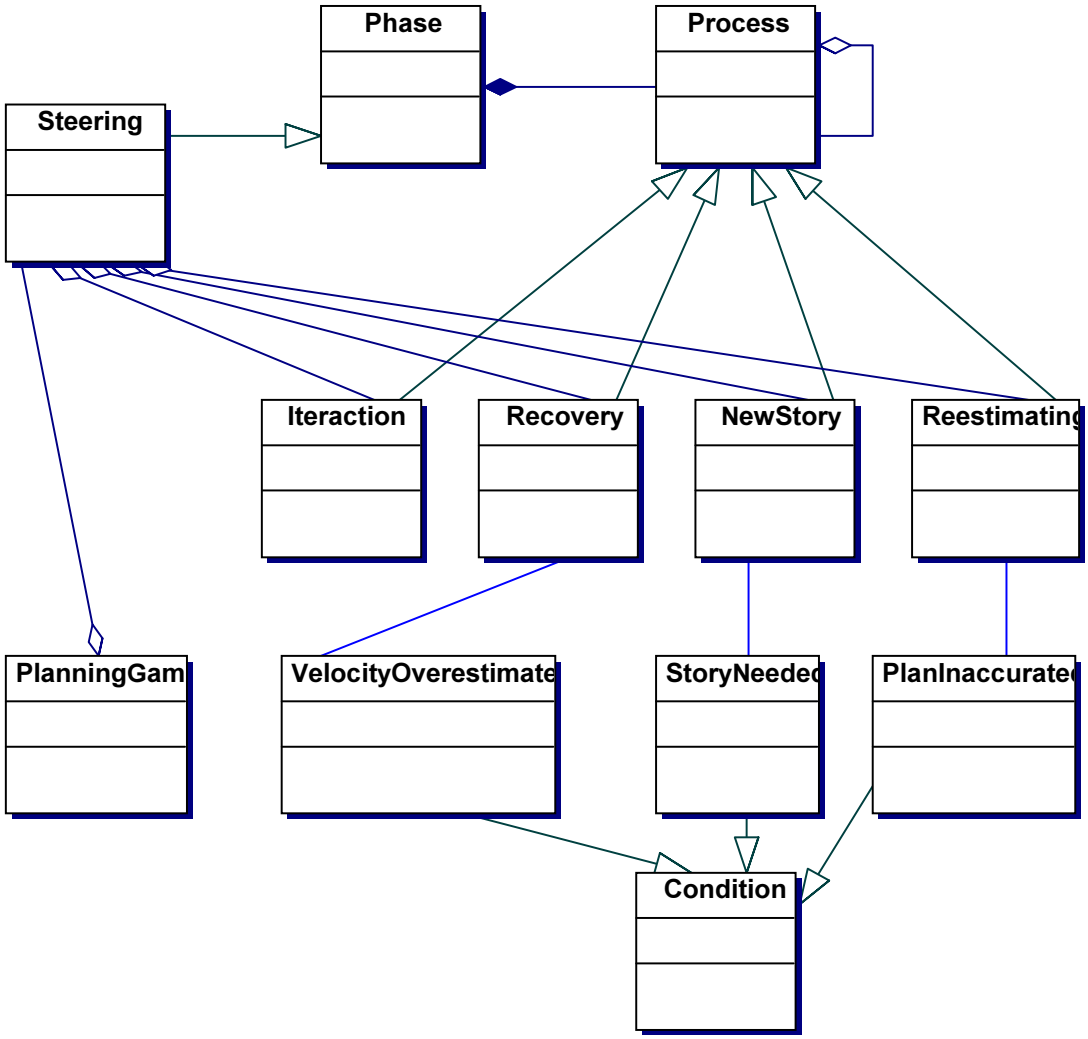

Figura 8 - Diagrama de Fases e Processos 3 – Fase *Steering* do *planning game*

Resumo das Classes	
Condition	Condition is a statement that must happen at the beginning, during execution and at the termination of a process.
Iteraction	A one- to four-weed period where at the beginning the customer chooses the stories to be implemented and in the end it's possible to run functional tests to see if the interaction succeeded.
NewStory	If customers realize it needs a new story, they can write the story, developers estimate it and then customers sort the stories again.
Phase	Phase is group of software processes tightly related which is conduced in some kind of order.
PlanInaccurated	A condition to Reestimating process.
PlanningGame	Planning Game is the XP planning process.
Process	A software process is a partially ordered set of subprocess which has resources, agents, conditions, artifacts and constructors associated in order to produce derivables according to some goal.
Recovery	If it is realized that velocity is overestimated, developers can ask customers what is the most valuable set of stories to retain in the current release.
Reestimating	If the development team feels that the plan no longer provides an accurate map of development, it cam reestimate all of the remaining stories and set velocity again.
Steering	Phase devoted to program, check estimatives and update plans.
StoryNeeded	A condition to NewStory process.
VelocityOverestimated	A condition to Recovery process.

2.1.4.4 Diagrama de Fases e Processos 4 – Fase *Exploration* do *interaction planning*.

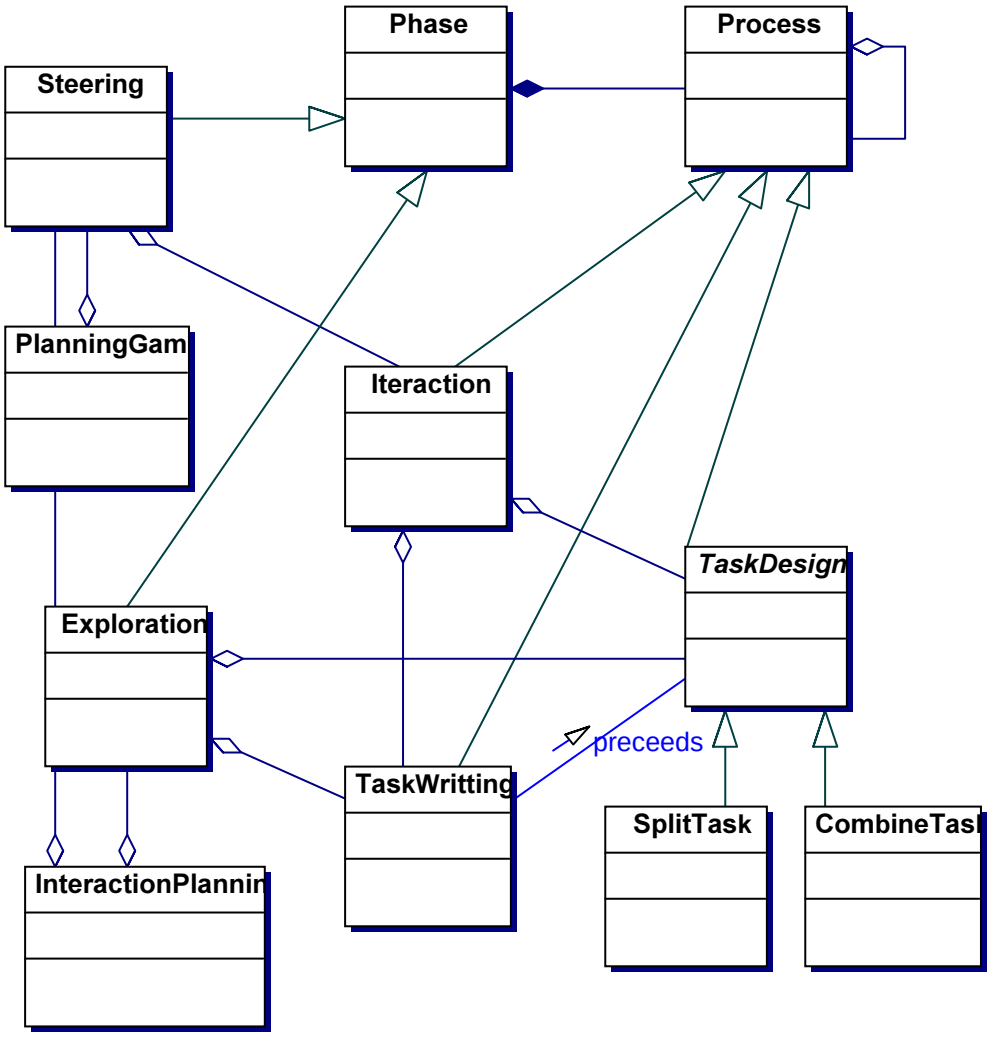

Figura 9 - Diagrama de Fases e Processos 4 – Fase *Exploration* do *interaction planning*

Resumo das Classes	
CombineTask	Tasks that are too small should be combined.
Exploration	The phase of development when the customer and programmers start to deal with the problem.
InteractionPlanning	Similar to planning game but processes are accomplished by developers in order to achieve an Interaction process.
Iteraction	A one- to four-weed period where at the beginning the customer chooses the stories to be implemented and in the end it's possible to run functional tests to see if the interaction succeeded.
Phase	Phase is group of software processes tightly related which is conduced in some kind of order.
PlanningGame	Planning Game is the XP planning process.
Process	A software process is a partially ordered set of subprocess which has resources, agents, conditions, artifacts and constructors associated in order to produce derivables according to some goal.
SplitTask	Tasks that can't be estimated should be splitted.
Steering	Phase devoted to program, check estimatives and update plans.
TaskDesign	Task manipulation; abstract class.
TaskWritting	Stories are converted in engineering tasks.

2.1.4.5 Diagrama de Fases e Processos 5 – Fase *Commitment* do *interaction planning*.

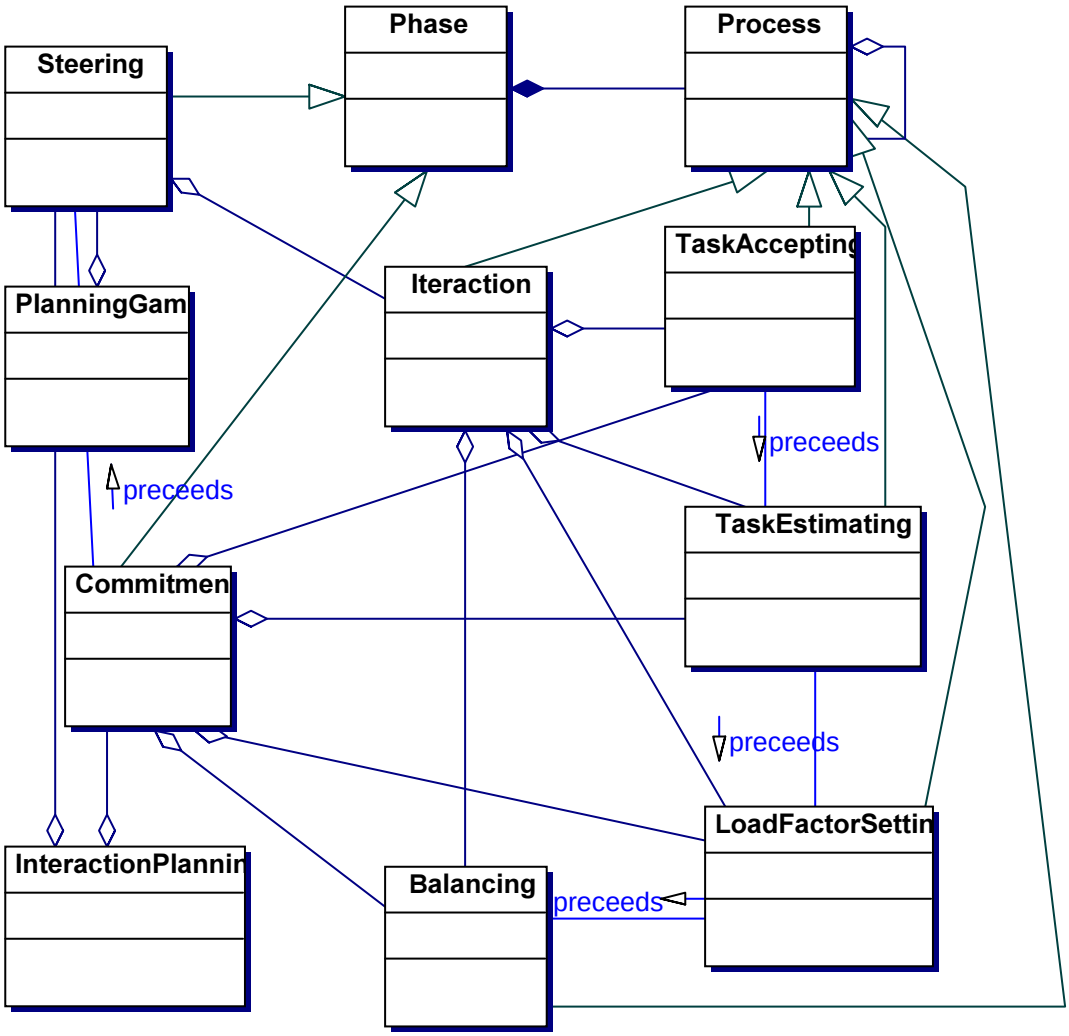

Figura 10 - Diagrama de Fases e Processos 5 – Fase *Commitment* do *interaction planning*

Resumo das Classes	
Balancing	Each developer adds up their task estimates and multiplies by their load factor.
Commitment	Phase where players accept their responsibilities.
InteractionPlanning	Similar to planning game but processes are accomplished by developers in order to achieve an Interaction process.
Iteraction	A one- to four-weed period where at the beginning the customer chooses the stories to be implemented and in the end it's possible to run functional tests to see if the interaction succeeded.
LoadFactorSetting	Each developer chooses their load factor for the interaction - the percentage of time they will spend actually developing.
Phase	Phase is group of software processes tightly related which is conduced in some kind of order.
PlanningGame	Planning Game is the XP planning process.
Process	A software process is a partially ordered set of subprocess which has resources, agents, conditions, artifacts and constructors associated in order to produce derivables according to some goal.
Steering	Phase devoted to program, check estimatives and update plans.
TaskAccepting	A developer accepts responsibility for a task.
TaskEstimating	The responsible developer estimates the number of ideal engineering days to implement the task.

2.1.4.6 Diagrama de Fases e Processos 6 – Fase *Steering* do *interaction planning*.

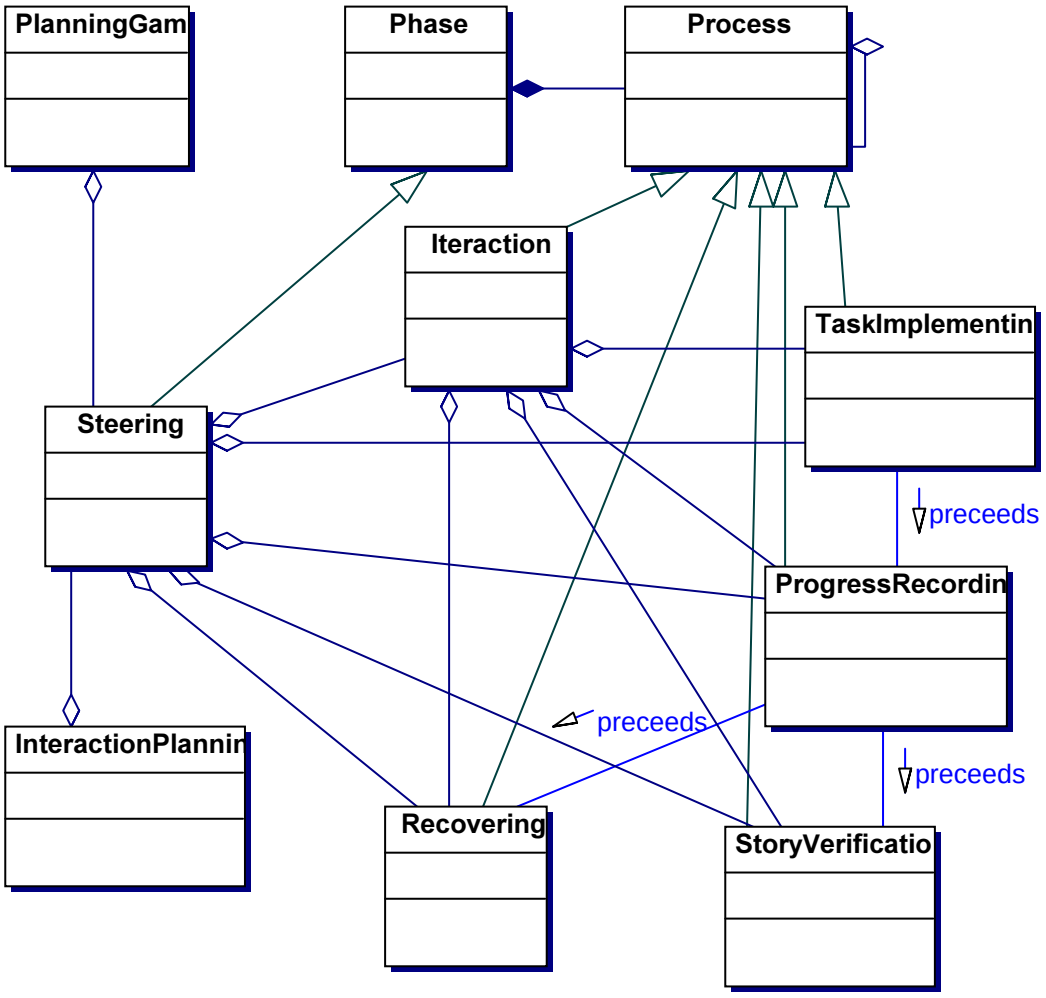

Figura 11 - Diagrama de Fases e Processos 6 – Fase *Steering* do *interaction planning*

Resumo das Classes	
InteractionPlanning	Similar to planning game but processes are accomplished by developers in order to achieve an Interaction process.
Iteraction	A one- to four-weed period where at the beginning the customer chooses the stories to be implemented and in the end it's possible to run functional tests to see if the interaction succeeded.
Phase	Phase is group of software processes tightly related which is conduced in some kind of order.
PlanningGame	Planning Game is the XP planning process.
Process	A software process is a partially ordered set of subprocess which has resources, agents, conditions, artifacts and constructors associated in order to produce derivables according to some goal.
ProgressRecording	Record of task implementing progress.
Recovering	Developers who turn out to be overcommitted ask for help.
Steering	Phase devoted to program, check estimatives and update plans.
StoryVerification	As soon as the functional tests are ready and the tasks for a story are complete, the functional tests are run to verify that the story works.
TaskImplementing	A developer takes a task card, finds a partner, writes the test cases for the task, makes all work, then integrates and releases the new code when the universal test suit runs.

2.2 Estudo de caso: OOHDM

2.2.1 Um modelo conceitual para OOHDM

Este modelo foi gerado com base em interpretações pessoais do método OOHDM feitas sobre [2], [17], [18] e [19]. Em alguns casos, quando necessário, a rationale do processo de análise e citações das referências anteriores serão expostas. A seguir os seguintes diagramas serão apresentados:

- Diagrama de Agentes e Papeis
- Diagrama de Artefatos
- Diagramas de Fases e Processos

2.2.2 Diagrama de Agentes e Papeis

No diagrama abaixo estaremos apresentando os agentes e papeis humanos definidos para OOHDM. Os demais tipos de agentes não serão abordados uma vez que o método não faz menção a ferramentas ou softwares em sua concepção, sendo portanto uma escolha no momento de instanciação do mesmo.

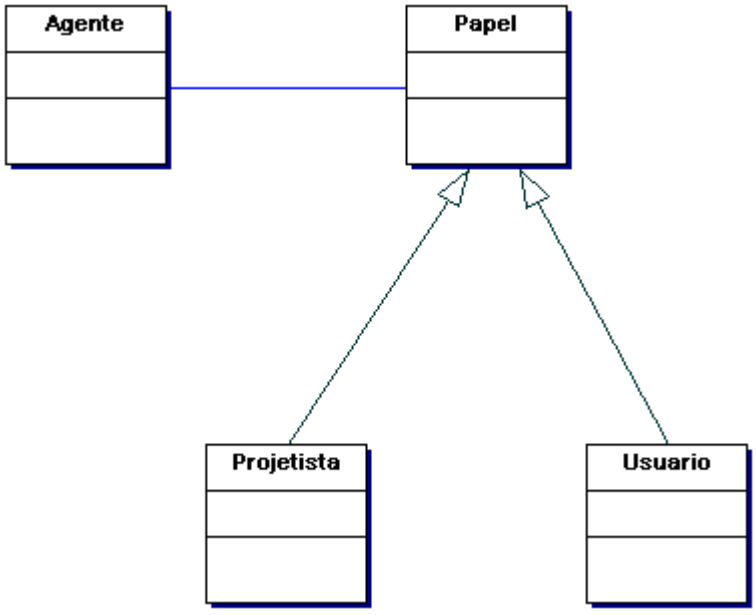

Figura 12 - Diagrama de Agentes e Papeis

Resumo das Classes	
Agente	Agente é o executor de um processo.
Papel	Papel é um conjunto de características associadas a um agente quando realiza uma tarefa específica.
Projetista	Responsável pela transformação da necessidade do usuário em uma aplicação.
Usuario	Detentor do problema a ser resolvido, é sujeito que operará a aplicação a ser construída.

2.2.3 Diagrama de Artefatos

No intuito de facilitar a compreensão dos artefatos do OOHDM, estes serão divididos nos seguintes diagramas de acordo com a fase em que são criados:

- Artefatos construídos durante a fase de Levantamento de Requisitos.
- Artefatos construídos durante a fase de Modelagem Conceitual.
- Artefatos construídos durante a fase de Modelagem Navegacional
- Artefatos construídos durante a fase de Projeto da Interface Abstrata.
- Artefatos construídos durante a fase de Implementação.
- Artefatos construídos para manter o rastreamento entre os elementos definidos em cada fase.

2.2.3.1 - Artefatos construídos durante a fase de Levantamento de Requisitos.

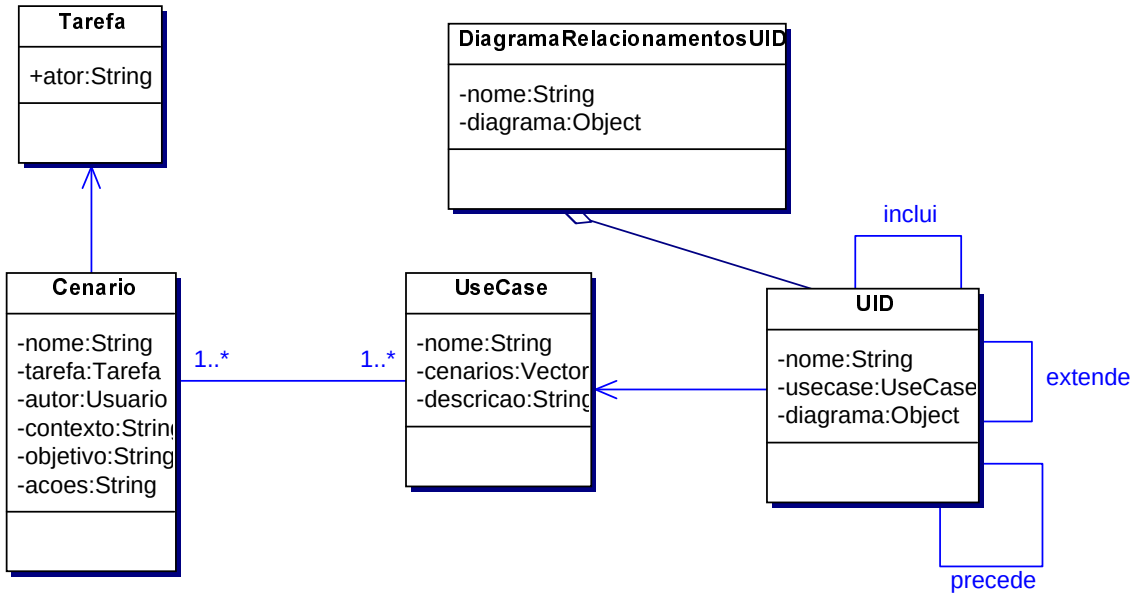

Figura 13 - Artefatos construídos durante a fase de Levantamento de Requisitos.

Resumo das Classes	
Cenario	Cenários são descrições narrativas em linguagem natural de como a aplicação pode ser usada; é descrito por um usuário representando um ator do sistema.
DiagramaRelacionamentosUIDS	O diagrama de relacionamento entre UIDs apresenta os relacionamentos mais significativos entre os UIDs; este diagrama serve para salientar alguns relacionamentos que são descritos nos use cases, mas que não são representados nos UIDs, além de facilitar a visualização de como os UIDs interagem para dar suporte a tarefas maiores; os atores que estão associados aos UIDs também podem ser especificados no diagrama de relacionamento entre os UIDs.
Tarefa	Tarefa é o objetivo que o usuário deseja alcançar usando a aplicação incorporando um ator.
UID	Um UID (User Interaction Diagram) representa a interação entre o usuário e a aplicação que foi descrita textualmente em um use case, sem entrar em detalhes relativos à interface com o usuário.
UseCase	Os use cases tratam somente da interação do usuário com o sistema ou das informações visíveis para o usuário, não abordando o funcionamento interno do sistema; o projetista especifica os use cases a partir dos cenários.

2.2.3.2 - Artefatos construídos durante a fase de Modelagem Conceitual.

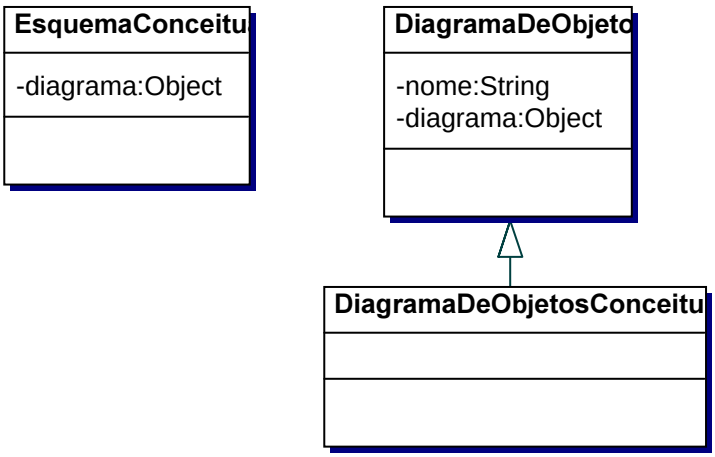

Figura 14 - Artefatos construídos durante a fase de Modelagem Conceitual

Resumo das Classes	
DiagramaDeObjetos	Um diagrama de objetos mostra um conjunto de instâncias de classes.
DiagramaDeObjetosConceituais	Diagrama que apresenta um conjunto de instâncias de classes de acordo com o diagrama de classes conceituais definido.
EsquemaConceitual	O esquema conceitual é um diagrama contendo as classes representativas da aplicação, os relacionamentos existentes entre essas classes e os subsistemas que agrupam as classes; a notação é semelhante a UML porém possui algumas primitivas próprias como as perspectivas de atributos.

2.2.3.3 - Artefatos construídos durante a fase de Modelagem Navegacional

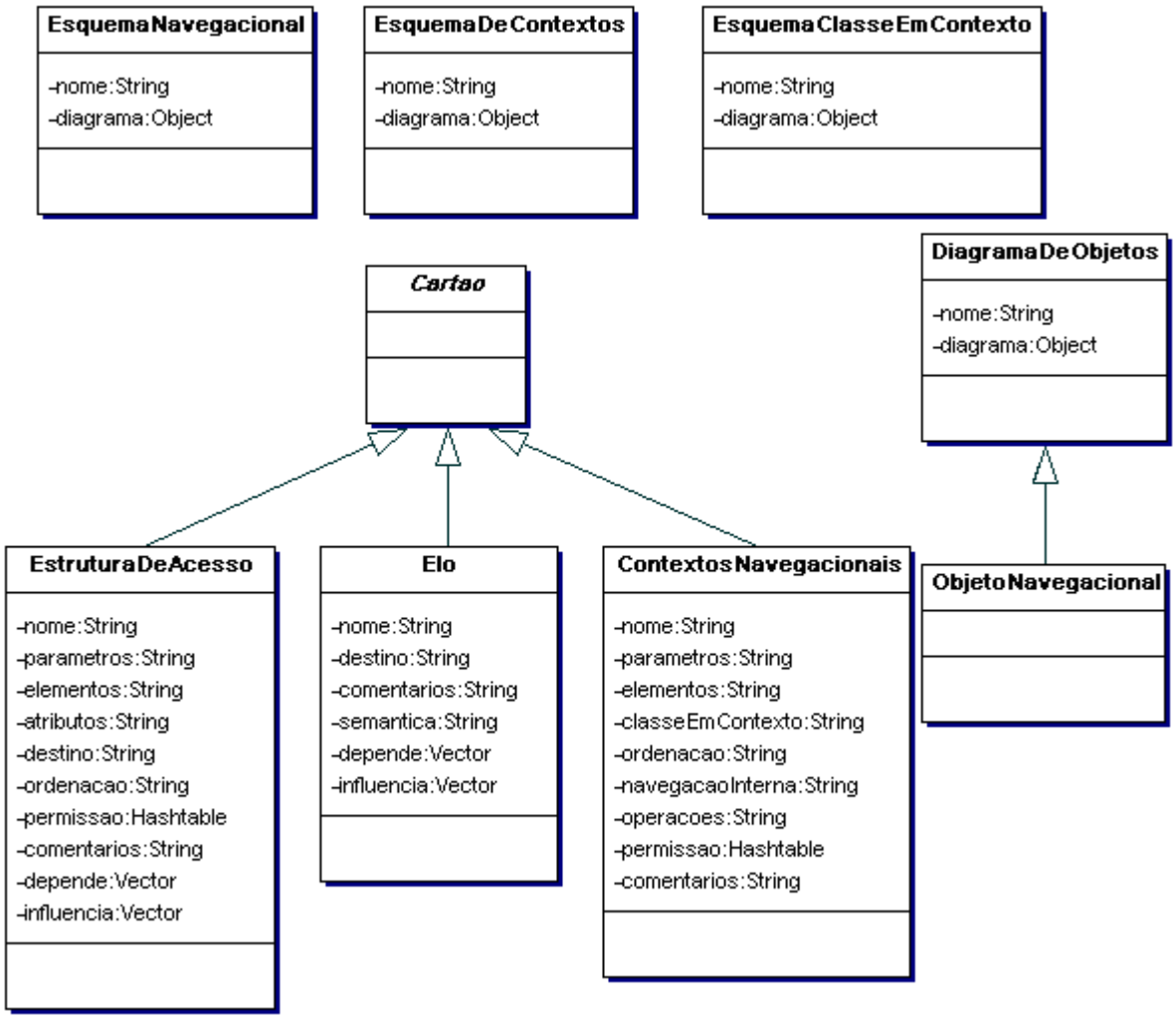

Figura 15 - Artefatos construídos durante a fase de Modelagem Navegacional

Resumo das Classes	
Cartao	Define propriedades de elementos navegacionais.
ContextosNavegacionais	Cartão que especifica um contexto ou grupo de contexto de navegação
DiagramaDeObjetos	Um diagrama de objetos mostra um conjunto de instâncias de classes.
Elo	Cartão que especifica um elo.
EsquemaClasseEmContexto	Define as classes em contexto juntamente com seus nós de origem; como os nós podem apresentar características diferentes de acordo com o contexto no qual eles aparecem, são criadas classes em contexto para definir a aparência e as âncoras de cada nó em cada contexto ao qual ele pertence, e também outras informações importantes ao contexto; uma classe em contexto só é necessária se o nó tem uma aparência diferente e âncoras distintas em diferentes contextos.
EsquemaDeContextos	Define em quais contextos será permitida a navegação entre as informações e quais informações serão apresentadas.
EsquemaNavegacional	O esquema navegacional define o conjunto de nós e elos (oriundos das classes e relacionamentos que compõem o esquema conceitual) que fazem parte de uma visão navegacional da aplicação; uma aplicação pode ter um ou mais esquemas navegacionais de acordo com as visões existentes da aplicação.
EstruturaDeAcesso	Cartão que especifica uma estrutura de acesso.
ObjetoNavegacional	Diagrama que apresenta um conjunto de instâncias de nós de acordo com o diagrama de classes navegacionais definido.

2.2.3.4 - Artefatos construídos durante a fase de Projeto da Interface Abstrata.

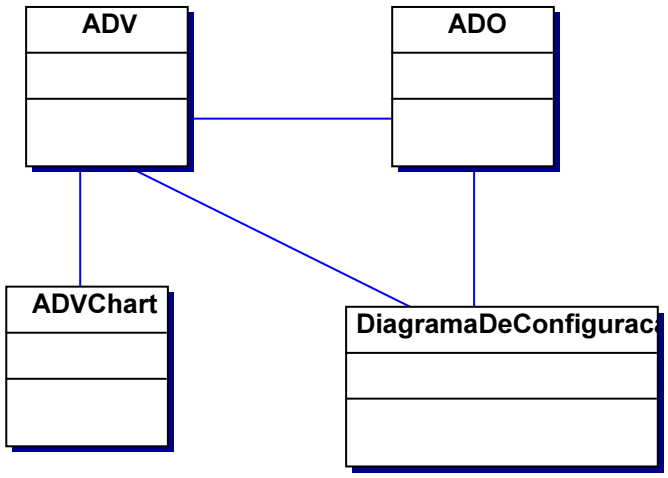

Figura 16 - Artefatos construídos durante a fase de Projeto da Interface Abstrata

Resumo das Classes	
ADO	Existem outros objetos que modelam a interface mas que não suportam eventos externos, somente interagem com as estruturas de dados e de controle da aplicação.
ADV	Os ADVs são objetos de interface usados para especificar a aparência e interface dos objetos da aplicação.
ADVChart	Os ADVcharts apresentam as transformações ocorridas nos ADVs, responsáveis por mudanças na interface com o usuário e nos objetos navegacionais; cada ADVchart está relacionado com uma tela da interface, assim toda a interface da aplicação é mostrada através de ADVcharts [17].
DiagramaDeConfiguracao	Os diagramas de configuração são usados para representar os relacionamentos entre os objetos de interface (ADV´s) e os objetos navegacionais (nós, elos, classes em contexto e estruturas de dados); especificam também outras características: os eventos externos iniciados pelo usuário e que serão manipulados pelos ADVs, os relacionamentos estruturais entre os ADVs, os relacionamentos entre os ADVs e os ADOs.

2.2.3.5 - Artefatos construídos durante a fase de Implementação.

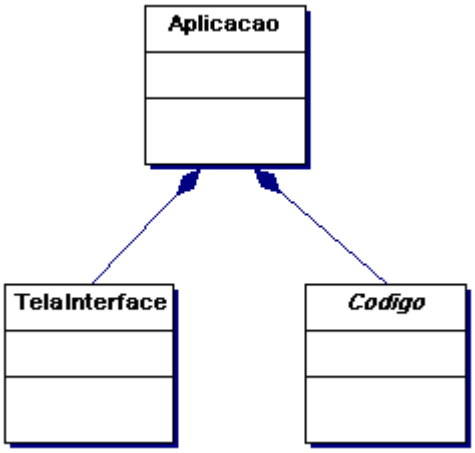

Figura 17 - Artefatos construídos durante a fase de Implementação

Resumo das Classes	
Aplicacao	Objetivo final do processo de produção de software.
Codigo	Generalização dos elementos diretamente necessários para se construir uma aplicação; não há referência explícita a este artefato no OOHDM.
TelaInterface	Artefatos obtidos através da instanciação de ADVCharts.

2.2.3.6 - Artefatos construídos para manter o rastreamento entre os elementos definidos em cada fase.

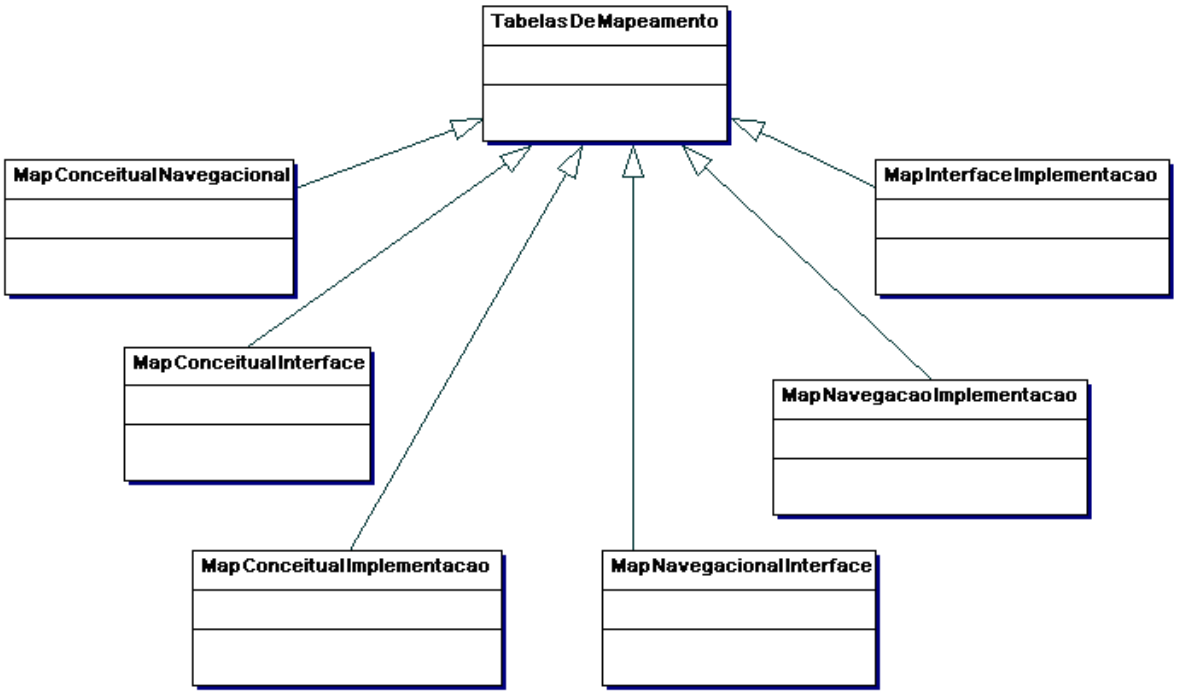

Figura 18 - Artefatos construídos para manter o rastreamento entre os elementos definidos em cada fase

Resumo das Classes	
MapConceitualImplementacao	"Tabela de Mapeamento dos Objetos Conceituais para Objetos de Implementação; esta tabela é construída após a conclusão da atividade de implementação e apresenta o mapeamento dos objetos conceituais para os objetos de implementação" [17].
MapConceitualInterface	"Tabela de Mapeamento dos Objetos Conceituais para Objetos de Interface; esta tabela é construída após a conclusão da atividade de projeto de interface abstrata e apresenta o mapeamento dos objetos conceituais para os objetos de interface" [17].
MapConceitualNavegacional	"Tabela de Mapeamento dos Objetos Conceituais para Objetos Navegacionais; esta tabela é construída após a conclusão da atividade de modelagem navegacional e apresenta o mapeamento dos objetos conceituais para os objetos navegacionais" [17].

MapInterfaceImplementacao	Tabela de Mapeamento dos Objetos de Interface para Objetos de Implementação; esta tabela é construída após a conclusão da fase de implementação.
MapNavegacaoImplementacao	Tabela de Mapeamento dos Objetos de Navegação para Objetos de Implementação; esta tabela é construída após a conclusão da fase de implementação.
MapNavegacionalInterface	Tabela de Mapeamento dos Objetos de Navegação para Objetos de Interface; esta tabela é construída após a conclusão da fase de Projeto da Interface Abstrata.
TabelasDeMapeamento	As tabelas de mapeamento apresentam o mapeamento dos objetos conceituais, especificamente as classes, atributos, relacionamentos e classes de relacionamento, em objetos navegacionais, de interface e de implementação; também ocorre o mapeamento entre os objetos de interface com os objetos navegacionais e de implementação - Após o término de cada atividade (modelagem navegacional, projeto de interface abstrata e implementação) são construídas as tabelas específicas.

2.2.4 Diagramas de Fases e Processos

De acordo com [17] e [18] um conjunto de atividades deve ser realizada para chegar até a implementação - "Essas atividades não seguem o modelo de desenvolvimento cascata, mais sim, uma mistura de estilos iterativos, incrementais e de prototipação rápida; em cada passo, um modelo é construído ou enriquecido e ao final do projeto, a aplicação hipermídia é implementada utilizando-se as informações obtidas durante todo o processo" [18] - Este comportamento foi modelado através das classes Modelagem e Implementação e a relação de precedência entre elas.

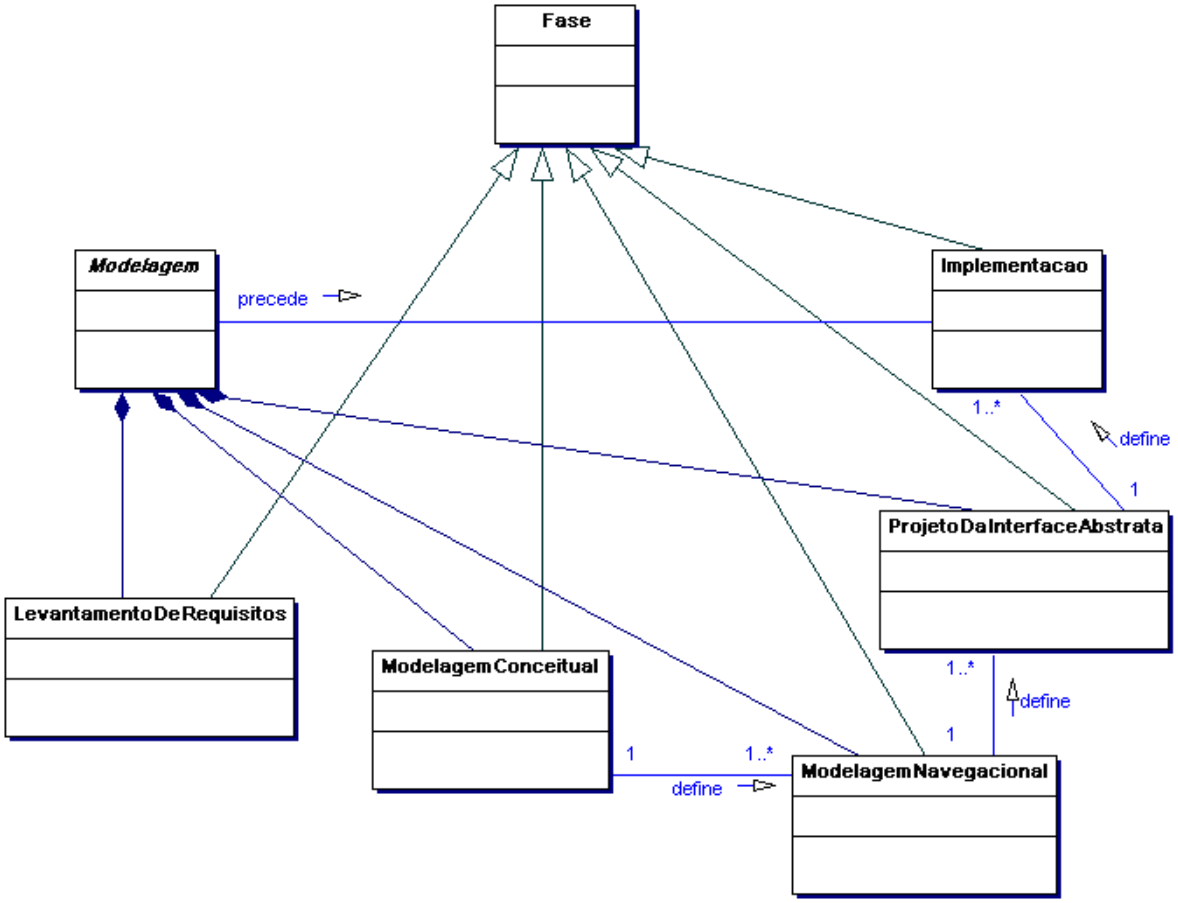

Figura 19 - Fases do OOHDM

Resumo das Classes	
Fase	OOHDM propõe 5 atividades (fases) durante a construção de uma aplicação hipermídia: levantamento de requisitos, modelagem conceitual, modelagem navegacional, projeto da interface abstrata e implementação; as quatro primeiras atividades são desenvolvidas iterativamente, enquanto a atividade de implementação geralmente é desenvolvida após o término destas.
Implementacao	"A etapa de implementação é a última etapa do OOHDM e é responsável pela tradução do projeto navegacional e do projeto de interface para um ambiente de implementação; nesta etapa deve ser definido o ambiente de implementação que será utilizado e como é feito o mapeamento dos objetos navegacionais e de interface para esse ambiente" [17].
LevantamentoDeRequisitos	A atividade de levantamento de requisitos define quais são os atores da aplicação que será desenvolvida e as tarefas que deverão ser apoiadas.
Modelagem	A classe Modelagem foi criada para reunir uma série fases que necessariamente ocorrem antes da implementação.
ModelagemConceitual	"A modelagem conceitual é a atividade responsável pela análise do domínio da aplicação, ou seja, engloba todo o universo de informações relevantes para a aplicação em questão, mesmo que apenas um subconjunto dessas informações venha a ser considerado posteriormente na sua implementação.... uma mesma modelagem conceitual podem ser definidas mais de uma modelagem navegacional, e conseqüentemente, mais de um projeto de interface abstrata e implementação" [17].
ModelagemNavegacional	"A modelagem navegacional define as informações que serão apresentadas e a possível navegação entre elas; a modelagem navegacional é definida a partir de um modelo conceitual...e produz as seguintes saídas: um esquema de classes navegacionais contendo a definição dos nós e elos; um esquema de contextos identificando os contextos navegacionais e as estruturas de acesso; um esquema de classes em contexto; cartões especificando todos os objetos criados na modelagem navegacional (objetos navegacionais e contextos navegacionais)" [17].

ProjetoDaInterfaceAbstrata	O projeto de interface abstrata define como serão os objetos de interface (objetos navegacionais e outros auxiliares, ex: barras de menus, botões de controle, ...), as suas propriedades e transformações, além de promover a independência de diálogo e o reuso desses objetos; o projeto de interface é desenvolvido antes de iniciar a implementação e de forma independente do ambiente de implementação, entretanto, deve também considerar algumas características do ambiente para que possa ser implementado.

No intuito de facilitar a compreensão dos processos relativo as fases do OOHDM, estes serão divididos, de acordo com o exposto acima, nos seguintes diagramas:

- Diagrama de Fases e Processos 1 – Fase Levantamento de Requisitos
- Diagrama de Fases e Processos 2 – Fases de Modelagem Conceitual, Modelagem Navegacional e Projeto da Interface Abstrata.
- Diagrama de Fases e Processos 3 – Fase Implementação

2.2.4.1 - Diagrama de Fases e Processos 1 – Fase Levantamento de Requisitos

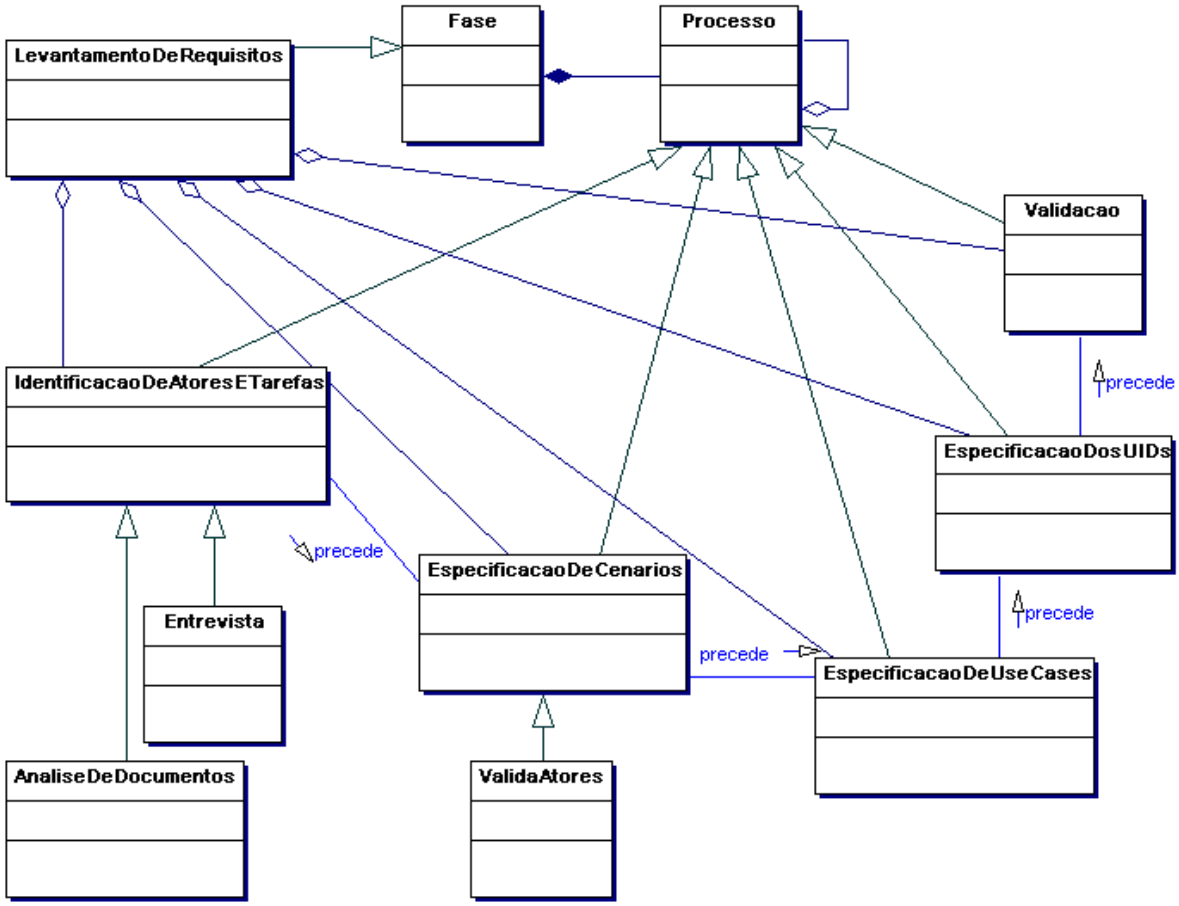

Figura 20 - Diagrama de Fases e Processos 1 – Fase Levantamento de Requisitos

Resumo das Classes	
AnaliseDeDocumentos	Análise de documentos pertinentes ao domínio.
Entrevista	Entrevistas como os usuários.
EspecificacaoDeCenarios	Na fase de especificação de cenários, cada usuário especifica (textualmente ou verbalmente) cenários que descrevem as suas tarefas como atores do sistema.
EspecificacaoDeUseCases	Na fase de especificação de use cases, o projetista especifica os use cases a partir dos cenários.
EspecificacaoDosUIDs	Na fase de especificação dos diagramas de interação do usuário (UIDs), os UIDs que representam os use cases são especificados.
Fase	OOHDM propõe 5 atividades (fases) durante a construção de uma aplicação hipermídia: levantamento de requisitos, modelagem conceitual, modelagem navegacional, projeto da interface abstrata e implementação; as quatro primeiras atividades são desenvolvidas iterativamente, enquanto a atividade de implementação geralmente é desenvolvida após o término destas.
IdentificacaoDeAtoresETarefas	Na fase de identificação de atores e tarefas, o projetista interage com o domínio da aplicação para identificar os atores e as tarefas; esta interação é alcançada através da análise de documentos disponíveis e entrevistas com os usuários; o principal objetivo é perceber e capturar as necessidades dos usuários.
LevantamentoDeRequisitos	A atividade de levantamento de requisitos define quais são os atores da aplicação que será desenvolvida e as tarefas que deverão ser apoiadas.
Processo	Conjunto parcialmente ordenado de subprocesos que tem como fim alcançar alguma meta estabelecida.
ValidaAtores	Os atores identificados devem ser confirmados com os usuários na etapa seguinte a sua identificação, ou seja, durante a especificação dos cenários.
Validacao	Validação dos use cases e UIDs; o projetista interage com cada usuário para validar os use cases e UID´s

2.2.4.2 - Diagrama de Fases e Processos 2 – Fases de Modelagem Conceitual, Modelagem Navegacional e Projeto da Interface Abstrata.

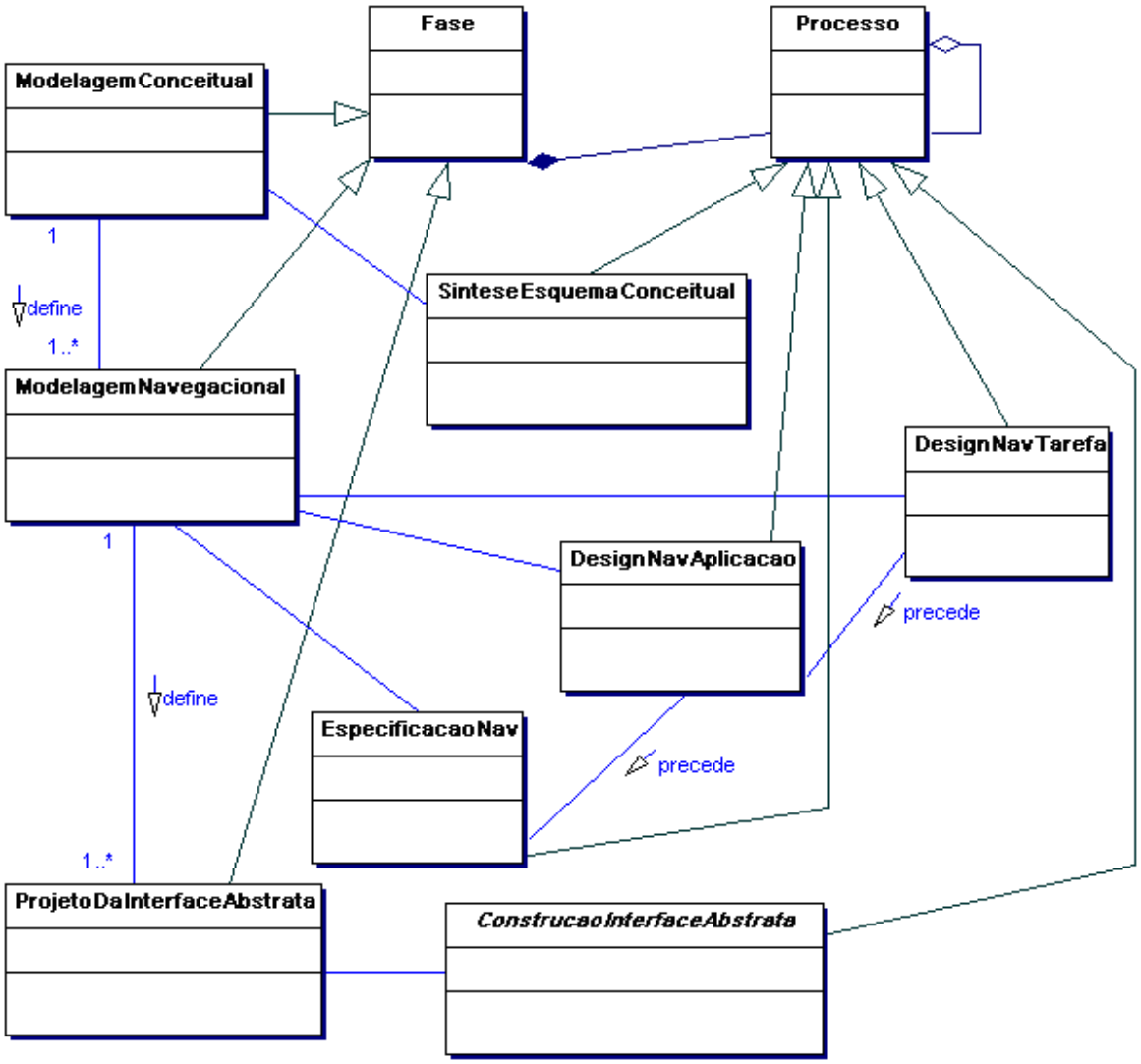

Figura 21 - Diagrama de Fases e Processos 2 – Fases de Modelagem Conceitual, Modelagem Navegacional e Projeto da Interface Abstrata

Resumo das Classes	
ConstrucaoInterfaceAbstrata	Os artefatos da interface abstrata são construído baseados no esquema conceitual e nos artefatos navegacionais.
DesignNavAplicacao	O Design Navegacional da Aplicação se dará através da união dos designs das tarefas.
DesignNavTarefa	Design Navegacional das Tarefas: baseado em sua experiência própria, nos UIDs e nos cenários, o projetista define a sequencia de navegação mais apropriada para o usuário para cada tarefa.
EspecificacaoNav	A Especificação Navegacional da Aplicação será a reunião e organização de todos os artefatos gerados durante os processos de design navegacional.
Fase	OOHDM propõe 5 atividades (fases) durante a construção de uma aplicação hipermídia: levantamento de requisitos, modelagem conceitual, modelagem navegacional, projeto da interface abstrata e implementação; as quatro primeiras atividades são desenvolvidas iterativamente, enquanto a atividade de implementação geralmente é desenvolvida após o término destas.
ModelagemConceitual	"A modelagem conceitual é a atividade responsável pela análise do domínio da aplicação, ou seja, engloba todo o universo de informações relevantes para a aplicação em questão, mesmo que apenas um subconjunto dessas informações venha a ser considerado posteriormente na sua implementação.... uma mesma modelagem conceitual podem ser definidas mais de uma modelagem navegacional, e conseqüentemente, mais de um projeto de interface abstrata e implementação" [17].
ModelagemNavegacional	"A modelagem navegacional define as informações que serão apresentadas e a possível navegação entre elas; a modelagem navegacional é definida a partir de um modelo conceitual...e produz as seguintes saídas: um esquema de classes navegacionais contendo a definição dos nós e elos; um esquema de contextos identificando os contextos navegacionais e as estruturas de acesso; um esquema de classes em contexto; cartões especificando todos os objetos criados na modelagem navegacional (objetos navegacionais e contextos navegacionais)" [17].
Processo	Conjunto parcialmente ordenado de subprocesos que tem como fim alcançar alguma meta estabelecida.

ProjetoDaInterfaceAbstrata	O projeto de interface abstrata define como serão os objetos de interface (objetos navegacionais e outros auxiliares, ex: barras de menus, botões de controle, ...), as suas propriedades e transformações, além de promover a independência de diálogo e o reuso desses objetos; o projeto de interface é desenvolvido antes de iniciar a implementação e de forma independente do ambiente de implementação, entretanto, deve também considerar algumas características do ambiente para que possa ser implementado.
SinteseEsquemaConceitual	O esquema conceitual é construído através da aplicação de algumas regras sobre as descrições obtidas nos UID´s.

2.2.4.3 - Diagrama de Fases e Processos 3 – Fase Implementação

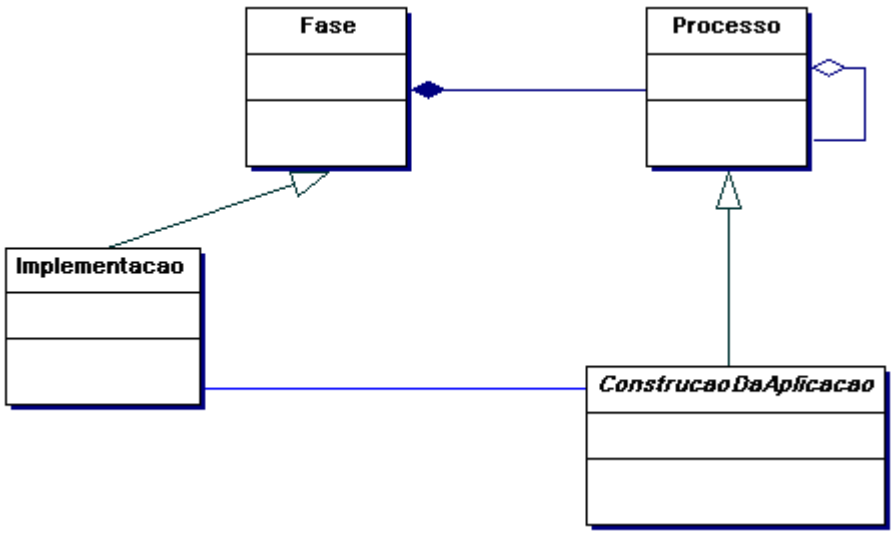

Figura 22 - Diagrama de Fases e Processos 3 – Fase Implementação

Resumo das Classes	
ConstrucaoDaAplicacao	O método OOHDM não especifica como a aplicação deve ser construída, existindo contudo algumas propostas de implementação como em [18] e [20].
Fase	OOHDM propõe 5 atividades (fases) durante a construção de uma aplicação hipermídia: levantamento de requisitos, modelagem conceitual, modelagem navegacional, projeto da interface abstrata e implementação; as quatro primeiras atividades são desenvolvidas iterativamente, enquanto a atividade de implementação geralmente é desenvolvida após o término destas.
Implementacao	"A etapa de implementação é a última etapa do OOHDM e é responsável pela tradução do projeto navegacional e do projeto de interface para um ambiente de implementação; nesta etapa deve ser definido o ambiente de implementação que será utilizado e como é feito o mapeamento dos objetos navegacionais e de interface para esse ambiente" [17].
Processo	Conjunto parcialmente ordenado de subprocesos que tem como fim alcançar alguma meta estabelecida.

2.3 Fatos extraídos dos estudos de caso

2.3.1 Com relação aos diagramas de atores e papéis

Fato 1 - O diagrama de atores e papéis do XP tem mais classes que o OOHDM. São dez classes contra duas do OOHDM

Fato 2 - Os papéis representados no XP são mais específicos em termos de divisão de tarefas. Em especial, existe um nível a mais na hierarquia de subclasses que por sua vez possui classes com várias operações definidas.

Fato 3 - Não há papeis no OOHDM que incorporem especificamente habilidades gerenciais do processo de desenvolvimento.

2.3.2 Com relação aos diagramas de artefatos

Fato 4 - Existem seis diagramas de artefatos para OOHDM e um para o XP. Isto se deu porque foi possível dividir os artefatos OOHDM de acordo com a fase em são criados.

Fato 5 - Os diagramas de artefatos do OOHDM tem mais classes ao todo que o diagrama equivalente XP. São onze contra trinta.

Fato 6 - Não há no OOHDM artefatos relacionados ao planejamento do processo de desenvolvimento. No XP há dois: Release Plan e Interaction Plan

Fato 7 - O número de artefatos voltados diretamente para a implementação é maior em XP. São seis contra três. O OOHDM não possui artefatos específicos relacionados a teste ou versão como há no XP.

Fato 8 - O número de artefatos voltados para concepção do problema e design da solução é maior no OOHDM. São vinte contra quatro. No XP não há artefatos relacionados com a interação com o usuário ou com a conceitualização do domínio em termos mais abstratos e formais.

Fato 9 - No XP não há artefatos de rastreamento como as Tabelas de Mapeamento do OOHDM.

2.3.3 Com relação aos diagramas de fases e processos

Fato 10 - O número de fases do OOHDM e do XP são próximos, eles têm cinco e seis fases respectivamente.

Fato 11 - O número de processos do OOHDM e menor do que no XP, eles têm doze e vinte e seis processos respectivamente.

Fato 12 - No OOHDM, o processo relativo à fase de Implementação está definido abstratamente.

Fato 13 - As fases do XP são agregadas em dois grupos planning game e interaction planning, sendo que o segundo ocorre repetidamente dentro de um processo do primeiro.

Fato 14 - As fases do OOHDM, exceto implementação, compõem a modelagem do software que necessariamente precede a fase de implementação.

2.4 Análise dos fatos

O fato 10 aliado a explicação dada para o ciclo de desenvolvimento OOHDM em [18] ("Essas atividades não seguem o modelo de desenvolvimento cascata, mais sim, uma mistura de estilos iterativos, incrementais e de prototipação rápida; em cada passo, um modelo é construído ou enriquecido e ao final do projeto, a aplicação hipermídia é implementada utilizando-se as informações obtidas durante todo o processo"), pode levar a uma suspeita de que os ciclos de desenvolvimento são semelhantes porém o XP têm uma

grande ênfase no processo em espiral – fato 13. No XP ocorrem várias etapas sucessivas de implementação ordenadas de acordo com a importância das funcionalidades a serem desenvolvidas. Já no OOHDM toda implementação deve ser executada em uma fase apenas – fato 14.

A justificativa para tais abordagens descritas anteriormente se baseia na ideologia dominante da concepção de cada método. O OOHDM está alinhado com as idéias da engenharia de software tradicional, resultando num método que busca uma grande compreensão do problema e em seguida várias etapas de refinamento para obter uma solução e então a implementação da mesma. Temos então um processo de desenvolvimento que expõe a rationale da obtenção da solução e que minimiza possíveis falhas de interpretação, reduzindo assim a probabilidade de erros após a implantação do software. Desta maneira porém, é esperado que o processo seja mais pesado, pouco focado na implementação – fato 12 - e averso a mudanças, principalmente nos requisitos. Por sua vez, o Extreme Programming participa do movimento Agile. Isto indica que o método é fortemente direcionado as pessoas envolvidas e para a solução concreta de problemas num ambiente em que os requisitos são incertos ou voláteis. Apresenta-se como vantagem, um desenvolvimento mais leve, pronto para manutenção e evolução do software, que fornece resultados rapidamente e cuja atividade chave é a programação. Em contrapartida o processo de desenvolvimento fica pouco reproduzível e muito dependente dos programadores como indivíduos.

Esta dualidade de abordagens vem sendo discutida tanto na comunidade acadêmica quanto na indústria. Nomes como Grady Booch, Alan Brown, Ivar Jacobson, Kent Beck, Erich Gamma, Martin Fowler, Roy Miller, entre outros, revelam suas opiniões sobre o fato. Com relação ao desenvolvimento de software para web os extremos ficam ainda mais evidentes, chegando a se discutir se aplicações desta natureza podem ser construídas por um processo de engenharia [21].

Nosso intuito é dar continuidade a este debate. Dado os novos desafios no processo de desenvolvimento de sistemas, esperamos que soluções sejam encontradas a fim de evoluir a área e responder as novas necessidades. Sendo assim, este trabalho proporá a frente a adaptação de um método baseado na engenharia de software tradicional, o OOHDM, levando em consideração a abordagem AGILE, mais especificamente os seus reflexos em Extreme Programing. Pretende-se com isso buscar um novo equilíbrio para as forças que agem no ambiente e no ciclo de vida do desenvolvimento de aplicações.

3 AgileOOHDM – Uma Proposta Inicial

Nesta seção será descrita uma proposta inicial para o AgileOOHDM, que pretende ser uma adaptação de um método tradicional de engenharia de software, OOHDM, considerando a abordagem Agile. Esta proposta nasceu do estudo apresentado no capítulo anterior sobre modelagem conceitual de processos de desenvolvimento. É também propósito deste trabalho contribuir para o debate entre métodos leves e pesados conforme seu amadurecimento.

Algumas modificações foram feitas no ciclo de desenvolvimento do OOHDM com a intenção de transformá-lo em um método ágil. Fases e processos foram rearranjados, artefatos foram simplificados ou eliminados, descrição de processos foram adaptadas e a execução do processo em si foi visto sobre um outro ponto de vista. É esperado desta maneira realizar implementações progressivas com um método mais leve, ajustando-o para um ambiente em constante evolução.

A seguir será apresentado o detalhamento da proposta de acordo com os diagramas desenhados e as mudanças no OOHDM. É importante esclarecer que o desenvolvimento do método e seus estudos apesar de terem sido suspensos para a elaboração deste relatório técnico ainda deixaram margem para melhorias.

3.1 Diagrama de Fases e Processos – somente fases

Não existe mais uma etapa de modelagem precedendo toda a implementação com se tem no OOHDM. A fase *Planning and Requirements Gathering* deve ser executada e então uma serie de *Building Interactions* irá progressivamente construindo o sistema. O objetivo de uma *Building Interaction* é modelar e implementar uma tarefa por vez, onde uma tarefa é expressada através de um UID. Estas interações também fornecem feedback para validar os requisitos, sendo possível mudar ou criar novas *Building Interactions* para atualizar as tarefas.

As fases de *Conceptual* e *Navigational Modeling* foram reunidas. Os pontos chave geraram então a fase denominada *Concept-Navigational Modeling*.

A fase *Interface Modeling* foi absorvida pela fase *Application Project*. Sua meta definir o sistema no mais baixo nível antes da codificação.

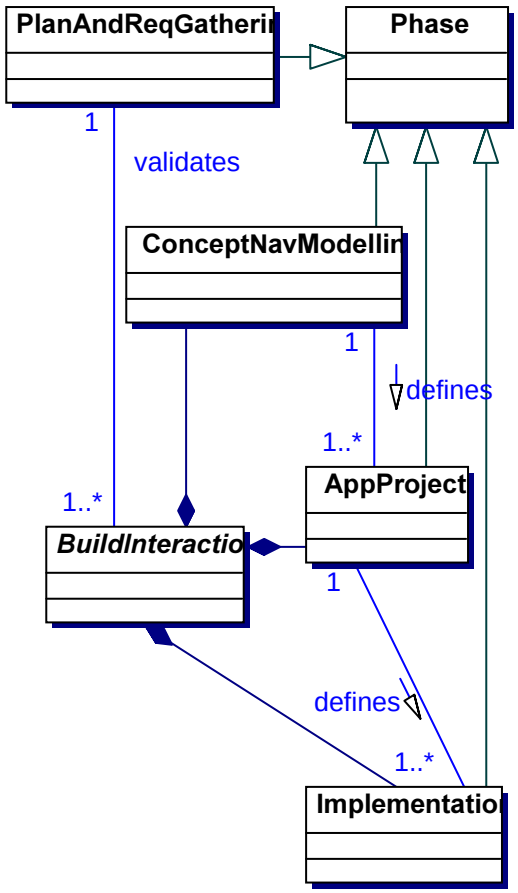

Figura 23 - AgileOOHDM – Diagrama de Fases

3.2 Diagrama de Fases e Processos – Fase de Planning and Requirements Gathering

Como os UID´s e os use cases são muito similares semanticamente, o processo *Use Case Specification* e seus artefatos foram extintos. A escolha em favor dos UID´s se deve a maior comunicabilidade do artefato em reuniões de validação com o cliente.

Não houve alteração no processo Scenarios Specification, contudo, um cenário deve conter apenas uma tarefa e o analista deve estar ciente disto durante o processo *Actors and Tasks Identification*.

No processo *UID´s Diagragrams Specification* os cenários para uma mesma tarefa devem ser reunidos e o resultante será a base para um UID.

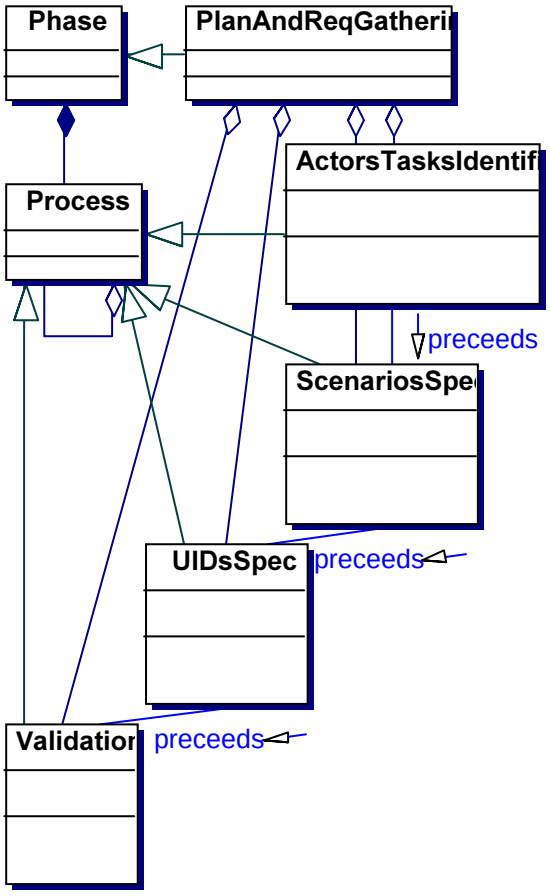

Figura 24 - AgileOOHDM – Processos da Fase de Planning and Requirements Gathering

3.3 Diagrama de Fases e Processos – Fases de Concept-Navigational Modeling e Application Project

O processo *Conceptual Model Synthesis* foi absorvido pelo *Task Design* e pelo *Application Specification*, sendo este último também responsável por rearranjar o design das tarefas, que era feito no processo *Navigational Design* no OOHDM e foi excluído.

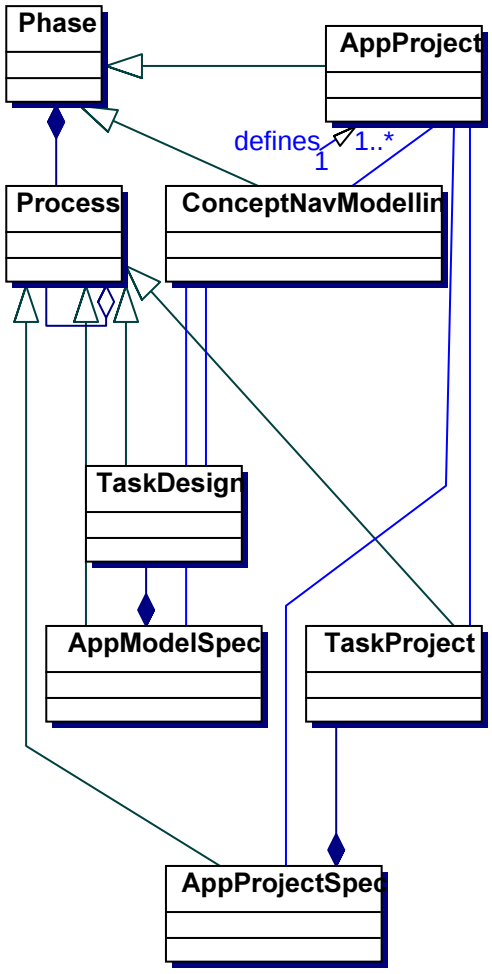

Figura 25 - AgileOOHDM – Processos das fases Concept-Navigational Modeling e Application Project

3.4 Diagrama de Fases e Processos – Fase de Implementation

Uma série de processos foi criada para manejar a implementação.

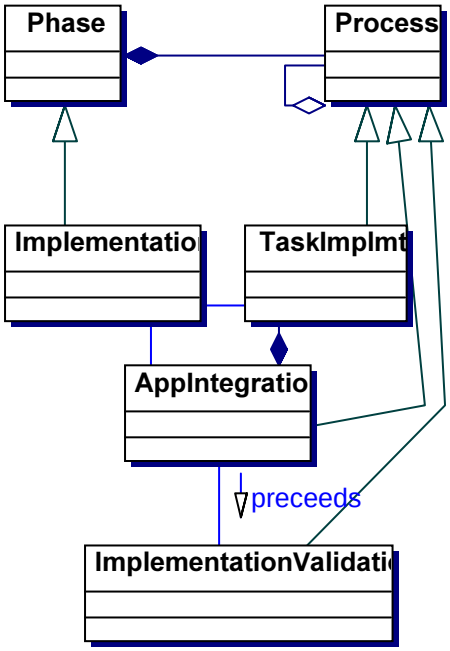

Figura 26 - AgileOOHDM- Processos da fase de Implementation

3.5 Diagrama de Artefatos

Devido a exclusão de seu respectivos processos, os artefatos *Use Case* e *Conceptual Model* também foram excluídos.

As *Mapping Tabels* foram eliminadas.

Os *Object Diagrams*, que eram opcionais no OOHDM foram descartados no AgileOOHDM.

Ainda não há resolução quanto aos *Cards*.

CDV´s e CDO´s foram criados para implementar os ADV´s and ADO´s.

3.6 Análise da Proposta e Próximos Passos

Esta proposta conforme já foi dito é inicial. Existe uma falta natural e esperada de precisão em sua definição e, sobretudo, há a necessidade de ser testada. É importante também considerar que alguns processos de XP e OOHDM não foram modelados no capítulo anterior devido a um muito baixo nível de abstração. Um dos próximos passos neste sentido é modelá-los e estudar com maior detenimento como estes processos afetam o novo método. Quanto ao teste, dois pequenos casos serão apresentados no próximo capítulo.

Com relação ao *Concept-Navigational Modeling* existe uma necessidade inerente e avaliação da possibilidade de juntar o *Conceptual Model* e o *Navigational Model*. Esta avaliação deve responder como eles devem ser reunidos e como evidenciar as diferenças de maneira a refleti-las na implementação. Provavelmente uma mudança na notação será requerida. Não obstante ferramentas específicas para criar e manter os novos artefatos, e os não tão novos como os UID´s, devem ser desenvolvidas para tornar mais rápido e produtivo o seu uso.

Sobre os artefatos é importante lembrar que apesar das *Mapping Tabels* terem sido extintas ainda é necessário decidir se e como o rastreamento entre os artefatos irá ocorrer. Resta também uma dúvida com relação ao destino dos artefatos *Cards*, eles poderiam ser eliminados e sua informação seria espalhada através de outros artefatos como possivelmente os *ADVs* e *ADOs*.

Nós acreditamos que ter estas questões resolvidas é crucial para obtermos um método leve mas elas não são a única mudança significante. Outro ponto de vital importância a ser alcançado e que ainda não foi tocado é a questão de um enfoque mais voltado a pessoas. Os processos precisam ser explicados com base nas necessidades e desejos dos agentes. O processo de validação na fase *Planning and Requirements Gathering* é um exemplo fortemente baseado na interação entre pessoas. A necessidade de comunicação clara e efetiva é evidente. O diagrama de agentes e papéis precisa ser desenhado com muito cuidado.

Outro quesito que se deseja estudar mais a fundo é como este novo método, com implementação incremental, pode facilitar a componentização de código e a elaboração de sistemas em arquitetura baseada em serviços.

4 Utilizando AgileOOHDM

Este capítulo apresentará o seguinte exercício: Descrever através de um exemplo como seria o desenvolvimento de uma aplicação utilizando o AgileOOHDM.

Estão sendo considerados alguns artefatos mencionados no *Anexo C: Projeto OOHDM de uma Loja de CDs* [22], parte integrante da disciplina *Análise e Projeto na Web* do curso de *Desenvolvimento de Aplicações para Web* da CCE PUC-Rio.

4.1 Realizando a fase de Implementation sem reuso inicial ou suporte de framewoks.

1. Inicial Contact

1.1 Phase: Planning and Requirements Gathering Phase (PlanAndReqGathering)

1.1.1 Process: Actors and Tasks Identification (ActorsTasksIdentific)
Idêntico a "Identificação dos Atores e Tarefas" presente em [22]

1.1.2 Process: Scenarios Specification (ScenariosSpec)
Idêntico a "Especificação dos Cenários" presente em [22]

Por motivos de concisão deste exemplo seguiremos apenas com dois cenários explicitados a seguir. Todo o restante do processo de desenvolvimento seguirá como se apenas houvesse estes dois cenários.

Cenário 2: Consulta dos CDs a partir de uma palavra Quero consultar os CDs a partir de uma palavra, já que geralmente eu não decoro o título dos CDs. Gostaria de ter a opção de indicar se desejo que a palavra seja encontrada no título dos CDs ou no nome da gravadora. Como resultado desta consulta eu gostaria de ver uma lista de CDs contendo o título, o ano de lançamento do CD, imagem da capa do CD, disponibilidade em estoque e preço unitário. Posso querer efetuar a comprar de um dos CDs da lista ou ainda ver informações mais detalhadas de algum CD. Nesse detalhamento, além das informações anteriores, aparecem os nomes das músicas, tempo de cada música, os nomes dos artistas, gênero e possibilidade de ouvir um trecho das músicas.

Cenário 3: Consulta sobre as informações de um artista/conjunto.
Digito o nome do artista ou parte dele. Aguardo a visualização da lista de artistas com o nome que eu digitei. Seleciono o artista procurado e aguardo a visualização das informações do artista tais como data de nascimento, uma descrição e uma foto. Poderia aparecer também a lista de CDs gravados pelo artista escolhido contendo ano de lançamento, título e capa. Existe a possibilidade de consulta mais detalhada sobre algum desses CDs; no detalhamento, aparecem ano do CD, disponibilidade em estoque, preço unitário, descrição, procedência (nacional/internacional), os nomes das músicas, tempo de cada música, artista e possibilidade de ouvir um trecho das músicas.

1.1.3 Process: UIDs Specification (UIDsSpec)
A partir dos cenários descritos em 1.1.2 os seguintes UIDs são especificados:

UID a partir do Cenário 2: Consultar CDs a partir de uma palavra.
Note que o UID não descreve exatamente o que foi dito no cenário 2, ele leva em consideração o que está em outros cenários relativos. Neste caso, a inclusão da "descrição" do CD na busca e na exibição. A análise que revela esta diferença é uma tarefa realizada pelo projetista neste processo.

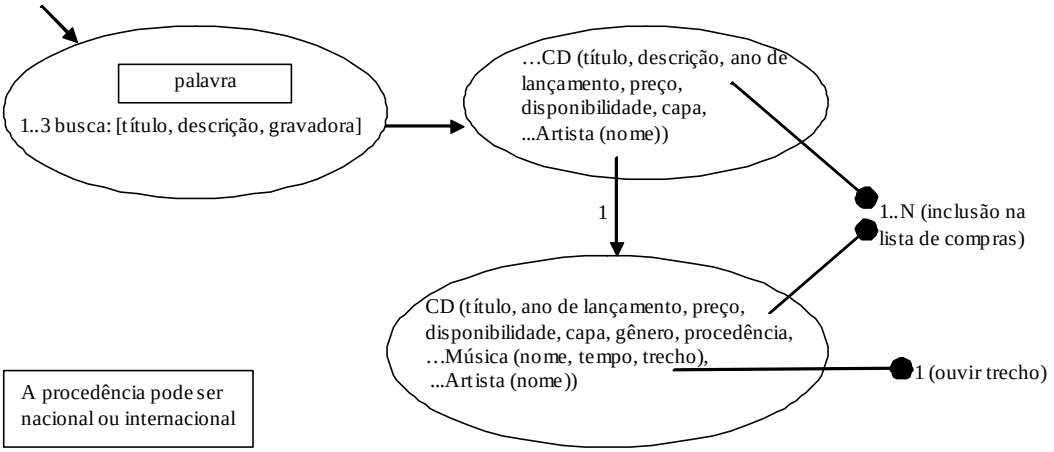

Figura 27 - UID do Cenário 2: Consultar CDs a partir de uma palavra

UID a partir do Cenário 3: Pesquisar informações de um artista.

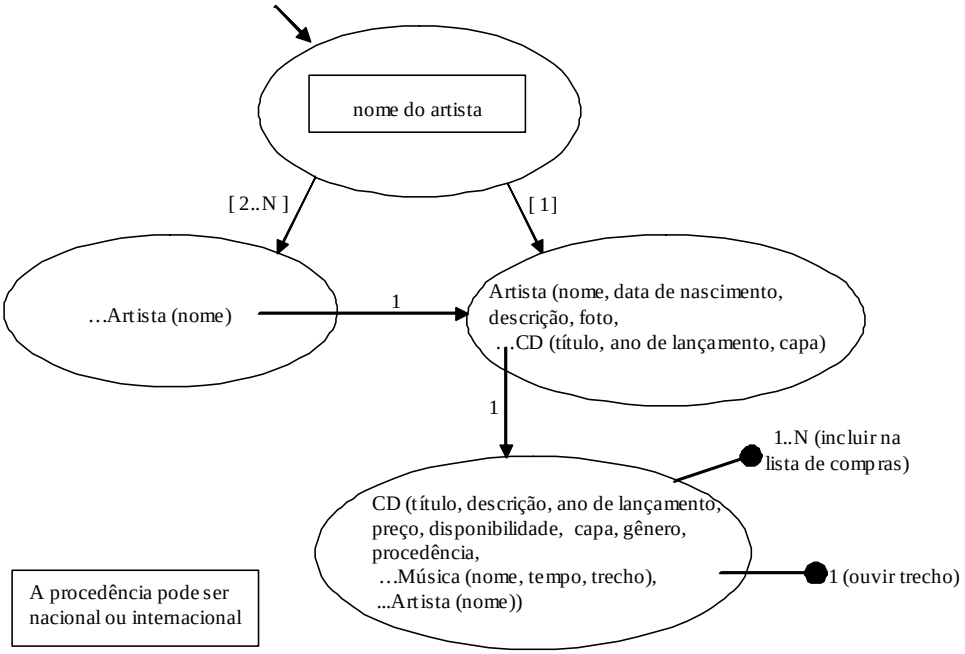

Figura 28 - UID do Cenário 3: Pesquisar informações de um artista

1.1.4 Process: Requirements Validation (Validation)

O usuário viu e aprovou os UIDs, inclusive a inclusão da "descrição" do CD.

2. First Build Interaction (BuildInteraction)
Conforme o usuário solicitou a tarefa de "Consultar CDs a partir de uma palavra" será implementada primeiro, enquanto a tarefa "Pesquisar informações de um artista" será implementada na próxima rodada.

2.1 Phase: Concept-Navigational Modeling (ConceptNavModeling)

2.1.1 Process: Task Design (TaskDesign)

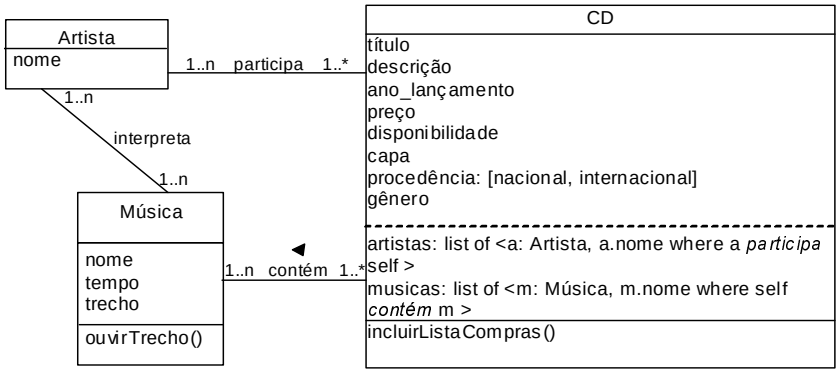

Figura 29 - Diagrama Conceito-Navegacional da Consulta de CDs a partir de uma palavra

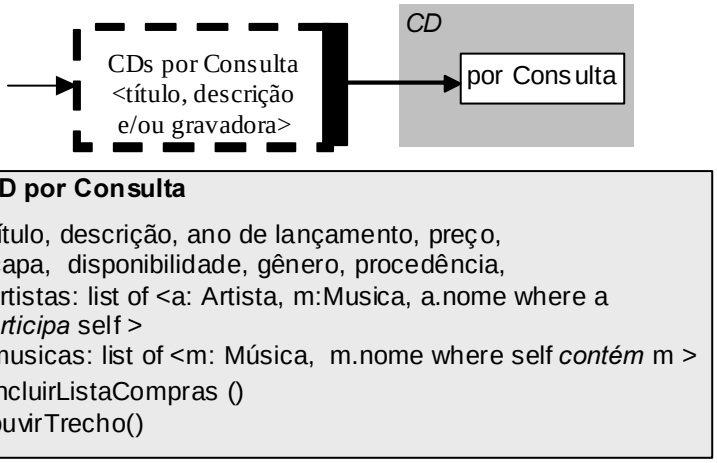

Figura 30 - Diagrama de Contextos da Consulta de CDs a partir de uma palavra

2.1.2 Process: Application Model Specification (AppModelSpec)
Uma vez que esta é a primeira Build Interacion não há necessidade de integrar os modelos descritos no processo anterior. Eles constituem o Modelo da Aplicação.

2.2 Phase: Application Project (AppProject)

2.2.1 Process: Task Project (TaskProject)

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
	Apresentação do Site e suas funções	1
Nome da Tela:	**Contexto:**	
CDAppT001 - Tela de Abertura		
Conteúdo	**Nav. Global:** CDAppT002 – Form CDs por Consulta	
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx CDs por consulta	Formulário de busca de CDs por palavras chave	2

Nome da Tela:	**Contexto:**	
CDAppT002- Form CDs por Consulta	Por Consulta	
Conteúdo Busque aqui o CD que você procura: palavras chave: input text buscar: submit button	**Nav. Global:**	
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx CDs por consulta	Resultado da busca por palavras chave	3

Nome da Tela:	**Contexto:**	
CDAppT003 - CDs por Consulta	por Consulta	
Conteúdo 250 CDs encontrados com a palavra chave “palavras chave” **CD.nome** CD.descrição CD.Ano de lançamento CD.Preço CD.Capa CD.Disponibilidade CD.Artistas do CD ...	**Nav. Intra-contextual:** Exibindo itens de 1 a 25 **Páginas 1 2 3 4 5 ...** Indicação dos CDs com a palavra chave “palavras chave” **Nav. Inter-contextual:**	**Nav. Referências Cruzadas:** **Operações:** incluirListaCompras()
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Ctx CDs por consulta	Tela de “Detalhe”do CD por Consulta	4

Nome da Tela:	**Contexto:**	
CDAppT004 – Detalhe CDs por Consulta	por Consulta = “palavras chave”	
Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**

DESCRIÇÃO DO CD: CD.Título CD.Descrição CD.Ano de lançamento CD.Preço CD.Capa CD.Disponibilidade CD.Gênero CD.Procedência CD.Músicas do CD CD.Artistas do CD	Retornar para Índice CDs Consulta “palavras chave” **Nav. Inter-contextual:**	**Operações:** incluirListaCompras() Musica.ouvirTrecho()
Observações:		

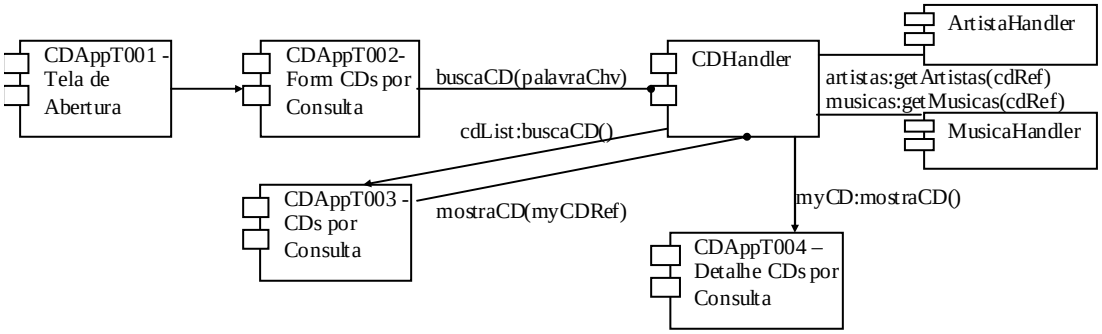

Figura 31 - Diagrama de Componentes de Implementação

2.2.2 Process: Application Project Specification (AppProjectSpec)
Uma vez que esta é a primeira Build Interacion não há necessidade de integrar os modelos descritos no processo anterior. Eles constituem o Projeto da Aplicação.

2.3 Phase: Implementation (Implementation)

2.3.1 Process: Task Implementation (TaskImplmt)
Foi criado um arquivo, CDApEx1_v231.zip, que contém os códigos fonte deste passo.

2.3.2 Process: Application Integration (AppIntegration)
Uma vez que esta é a primeira Build Interacion não há necessidade de integrar os códigos descritos no processo anterior. Eles constituem o Projeto da Aplicação.

2.3.3 Process: Implementation Validation (ImplementationValidation)
Usuario aprovou a tarefa de Consultar CDs a partir de uma palavra.

3. Second Build Interaction (BuildInteraction)
Interação responsável pela tarefa “Pesquisar informações de um artista”.

3.1 Phase: Concept-Navigational Modeling (ConceptNavModeling)

3.1.1 Process: Task Design (TaskDesign)

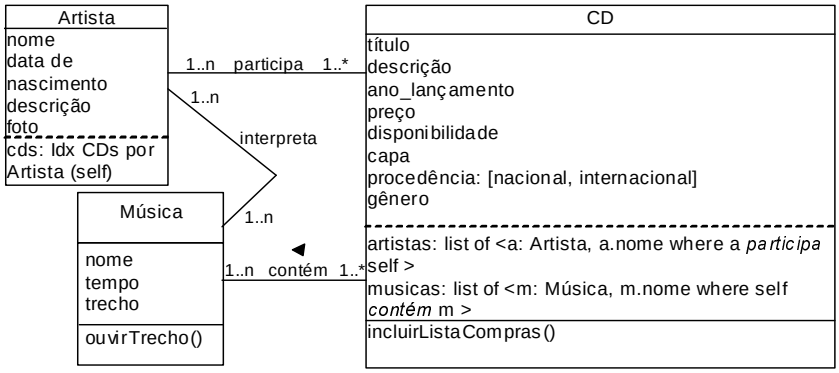

Figura 32 - Diagrama Conceito-Navegacional da Pesquisa de informações de um artista

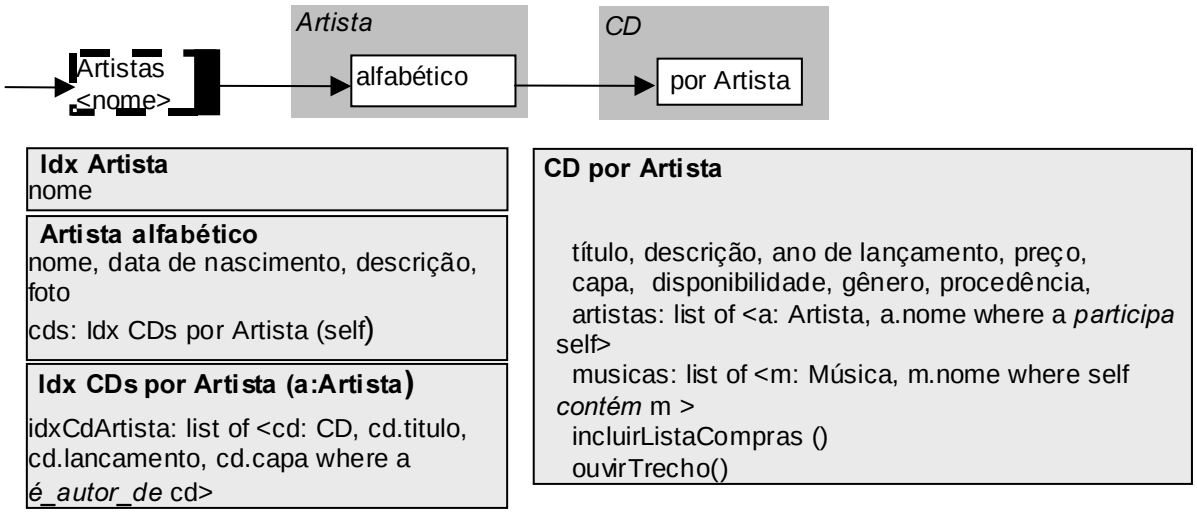

Figura 33 -Diagrama de Contextos da Pesquisa de informações de um artista

3.1.2 Process: Application Model Specification (AppModelSpec)

O Diagrama Conceito-Navegacional da Pesquisa de informações de um artista foi construído adicionando-se atributos a classe artista do Diagrama de Contextos da Consulta de CDs a partir de uma palavra, já estando desta forma integrado para compor o modelo da especificação.

Como os objetos navegacionais dos diagramas de contexto da Consulta de CDs a partir de uma palavra e da Pesquisa de informações de um artista são

distintos, o novo diagrama de contextos do modelo da aplicação é a concatenação dos dois.

3.2 Phase: Application Project (AppProject)

3.2.1 Process: Task Project (TaskProject)

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
	Apresentação do Site e suas funções	1

Nome da Tela:	**Contexto:**
CDAppT001 - Tela de Abertura	

Conteúdo	**Nav. Global:**
	CDAppT002 – Form CDs por Consulta CDAppT005 – Form Aritstas alfabético
Observações:	

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx Artista	Formulário de busca de Artistas por nome	5

Nome da Tela:	**Contexto:**
CDAppT005 – Form Aritstas alfabético	alfabético

Conteúdo	**Nav. Global:**
Busque aqui o Artista que você procura: nome: input text buscar: submit button	
Observações:	

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**

Idx Artista	Resultado da busca por nome de artista	6

Nome da Tela:	**Contexto:**	
CDAppT006 - Artista alfabético	alfabético	
Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**
250 Artistas encontrados o nome "nome" Artista.nome ...	Exibindo itens de 1 a 25 **Páginas 1 2 3 4 5 ...** Indicação dos Artistas com o nome "nome"	
	Nav. Inter-contextual:	**Operações:**
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Ctx Artistas alfabético	Tela de "Detalhe"do Artista alfabético	7

Nome da Tela:	**Contexto:**	
CDAppT007 – Detalhe Artista alfabético	alfabético = "nome"	
Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**
INFORMAÇÕES DO ARTISTA: **Artista.nome** Artista.data de nascimento Artista.descrição Artista.foto Índice dos CDS do artista cds: Idx CDs por Artista (Artista)	Retornar para Índice Artistas alfabético "nome"	CDs do Artista (vai para CD por Artista)
	Nav. Inter-contextual:	**Operações:**
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Ctx CDs por Artista	Tela de "Detalhe"do CD por Artista	8

Nome da Tela:	**Contexto:**	
CDAppT008 – Detalhe CDs por Artista	por Artista	
Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**

DESCRIÇÃO DO CD: CD.Título CD.Descrição CD.Ano de lançamento CD.Preço CD.Capa CD.Disponibilidade CD.Gênero CD.Procedência CD.Músicas do CD CD.Artistas do CD	Retornar para Artista alfabético **Nav. Inter-contextual:** CDs Próximo e Anterior do artista do contexto	**Operações:** incluirListaCompras() Musica.ouvirTrecho()
Observações:		

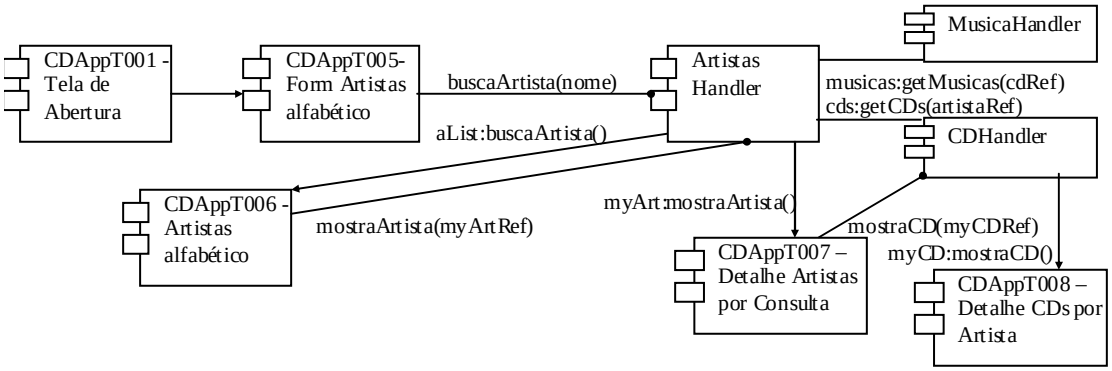

Figura 34 - Diagrama de Componentes de Implementação

3.2.2 Process: Application Project Specification (AppProjectSpec)
As telas de interface geradas no processo descrito no item 3.2.1 são distintas das geradas no mesmo processo porém no item 2.2.1, apenas a primeira tela é a mesma e foi construída apenas adicionando a opção de acesso desta tarefa, portanto, a integração das telas é a penas a concatenação.

3.3 Phase: Implementation (Implementation)

3.3.1 Process: Task Implementation (TaskImplmt)
Foi criado um arquivo, CDApEx1_v331.zip, que contém os códigos fonte deste passo.

3.3.2 Process: Application Integration (AppIntegration)
Foi feita juntamente no processo anterior.

3.3.3 Process: Implementation Validation (ImplementationValidation)
O usuário gostou mas pediu que a partir dos nomes dos artistas no contexto CD por consulta fosse possível navegar para informações de determinado artista conforme visto nesta interação.

4 Change Request
Por engenharia reversa, é possível detectar que a requisição do usuário altera o UID do cenário 2 ficando da seguinte maneira:

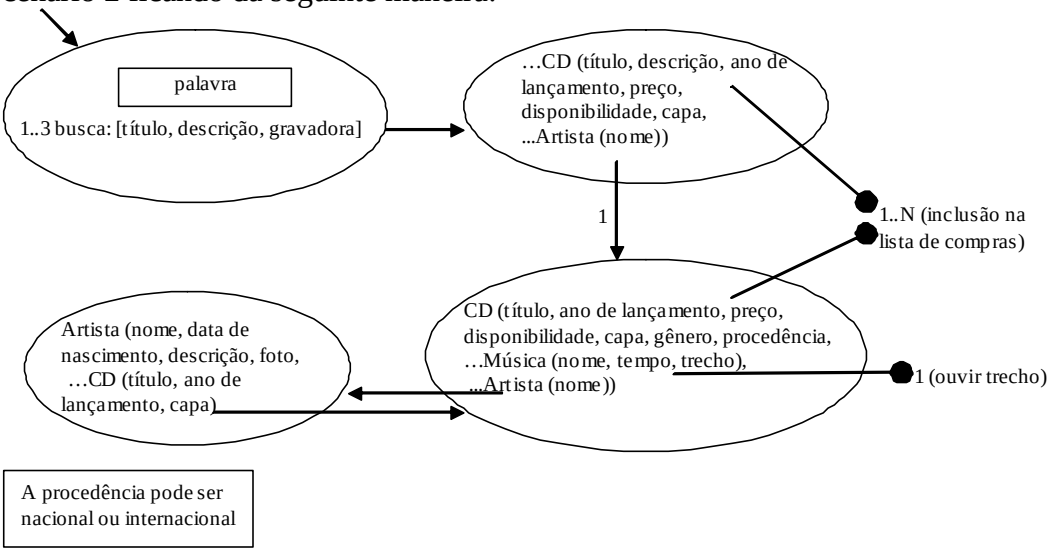

Figura 35 - UID de "Consultar CDs a partir de uma palavra" após requisição de mudança do usuário

Este novo UID alterará a tarefa de "Consultar CDs a partir de uma palavra"

5. Third Build Interaction (BuildInteraction)
Interação para adaptar o sistema ao novo requisito do usuário: a partir dos nomes dos artistas no contexto CD por consulta ser possível navegar para informações de determinado artista.

5.1 Phase: Concept-Navigational Modeling (ConceptNavModeling)

5.1.1 Process: Task Design (TaskDesign)
Este processo foi realizado juntamente com o seguinte para construir os artefatos incrementalmente sobre os já existentes.

5.1.2 Process: Application Model Specification (AppModelSpec)
No modelo conceito-navegacional, a lista de artistas se transforma em um índice para os artistas:

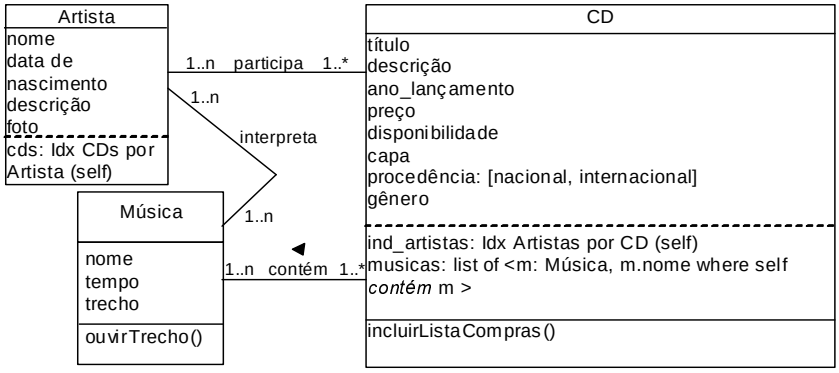

Figura 36 - Novo Diagrama Conceito-Navegacional após requisição de mudança do usuário

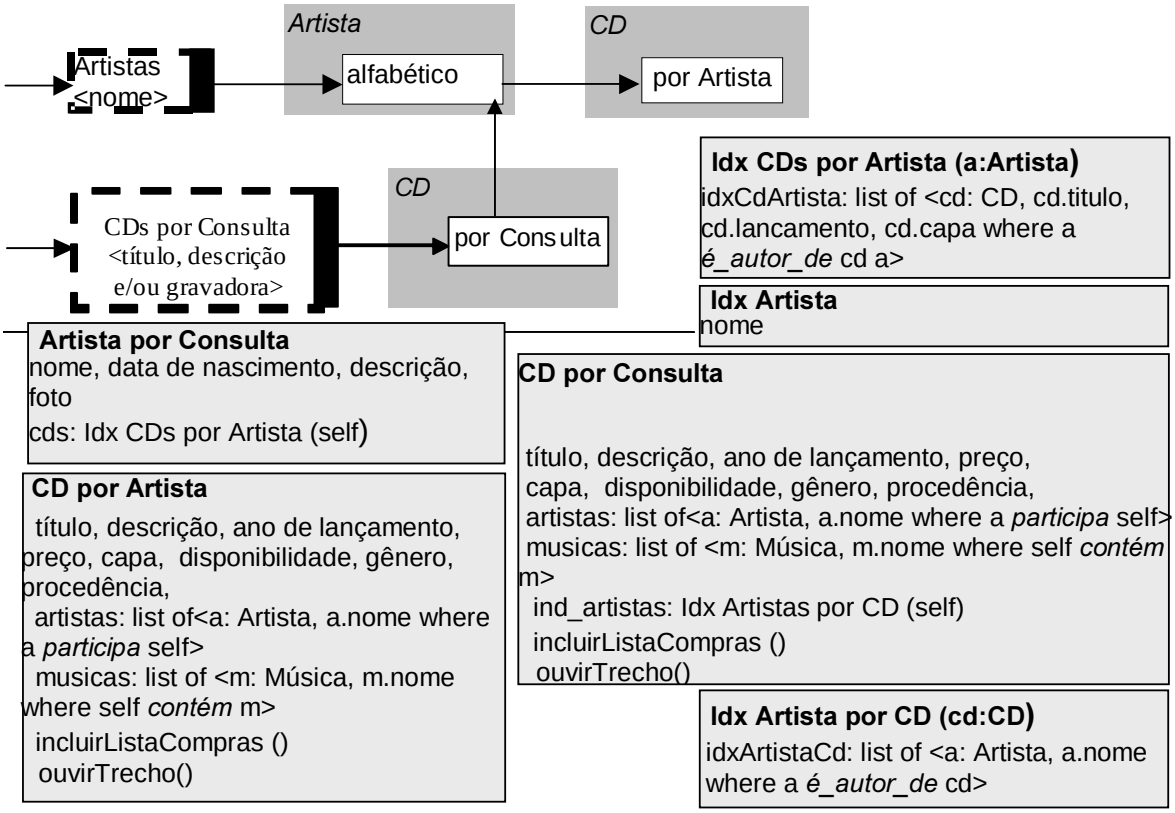

Figura 37 - Diagrama de Contextos após requisição de mudança do usuário

5.2 Phase: Application Project (AppProject)

5.2.1 Process: Task Project (TaskProject)

Este processo foi realizado juntamente com o seguinte para construir os artefatos incrementalmente sobre os já existentes.

5.2.2 Process: Application Project Specification (AppProjectSpec)

Alteração realizada devido a solicitação do usuário: foi alterado o campo Nav. Referências Cruzadas da tela No 4

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
	Apresentação do Site e suas funções	1
Nome da Tela:	**Contexto:**	
CDAppT001 - Tela de Abertura		
Conteúdo	**Nav. Global:** CDAppT002 – Form CDs por Consulta CDAppT005 – Form Aritstas alfabético	
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx CDs por consulta	Formulário de busca de CDs por palavras chave	2
Nome da Tela:	**Contexto:**	
CDAppT002- Form CDs por Consulta	Por Consulta	
Conteúdo Busque aqui o CD que você procura: palavras chave: input text buscar: submit button	**Nav. Global:**	
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx CDs por consulta	Resultado da busca por palavras chave	3
Nome da Tela:	**Contexto:**	
CDAppT003 - CDs por Consulta	por Consulta	

Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**
250 CDs encontrados com a palavra chave “palavras chave” **CD.nome** CD.descrição CD.Ano de lançamento CD.Preço CD.Capa CD.Disponibilidade CD.Artistas do CD ...	Exibindo itens de 1 a 25 **Páginas 1 2 3 4 5 ...** Indicação dos CDs com a palavra chave “palavras chave”	
Observações:	**Nav. Inter-contextual:**	**Operações:** incluirListaCompras()

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Ctx CDs por consulta	Tela de “Detalhe”do CD por Consulta	4

Nome da Tela:	**Contexto:**
CDAppT004 – Detalhe CDs por Consulta	por Consulta = “palavras chave”

Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**
DESCRIÇÃO DO CD: CD.Título CD.Descrição CD.Ano de lançamento CD.Preço CD.Capa CD.Disponibilidade CD.Gênero CD.Procedência CD.Músicas do CD CD.Artistas do CD	Retornar para Índice CDs Consulta “palavras chave”	Informações do artista Vai para Artista alfabético
Observações:	**Nav. Inter-contextual:**	**Operações:** incluirListaCompras() Musica.ouvirTrecho()

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx Artista	Formulário de busca de Artistas por nome	5

Nome da Tela:	**Contexto:**

CDAppT005 – Form Aritstas alfabético	alfabético	
Conteúdo Busque aqui o Artista que você procura: nome: input text buscar: submit button	**Nav. Global:**	
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Idx Artista	Resultado da busca por nome de artista	6
Nome da Tela:	**Contexto:**	
CDAppT006 - Artista alfabético	alfabético	
Conteúdo 250 Artistas encontrados o nome “nome” Artista.nome ...	**Nav. Intra-contextual:** Exibindo itens de 1 a 25 **Páginas 1 2 3 4 5 ...** Indicação dos Artistas com o nome “nome”	**Nav. Referências Cruzadas:**
	Nav. Inter-contextual:	**Operações:**
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Ctx Artistas alfabético	Tela de “Detalhe”do Artista alfabético	7
Nome da Tela:	**Contexto:**	
CDAppT007 – Detalhe Artista alfabético	alfabético = “nome”	
Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**

INFORMAÇÕES DO ARTISTA: **Artista.nome** Artista.data de nascimento Artista.descrição Artista.foto Índice dos CDS do artista cds: Idx CDs por Artista (Artista)	Retornar para Índice Artistas alfabético "nome" **Nav. Inter-contextual:**	CDs do Artista (vai para CD por Artista) **Operações:**
Observações:		

Origem Conceito-Navegacional:	**Descrição da Tela:**	**Nº:**
Ctx CDs por Artista	Tela de "Detalhe"do CD por Artista	8

Nome da Tela:	**Contexto:**
CDAppT008 – Detalhe CDs por Artista	por Artista

Conteúdo	**Nav. Intra-contextual:**	**Nav. Referências Cruzadas:**
DESCRIÇÃO DO CD: CD.Título CD.Descrição CD.Ano de lançamento CD.Preço CD.Capa CD.Disponibilidade CD.Gênero CD.Procedência CD.Músicas do CD CD.Artistas do CD	Retornar para Artista alfabético **Nav. Inter-contextual:** CDs Próximo e Anterior do artista do contexto	**Operações:** incluirListaCompras() Musica.ouvirTrecho()
Observações:		

Componentes do passo 2.2.1 Process: Task Project (TaskProject) da 1a Build Interaction

Alteração realizada devido a solicitação do usuário: os componentes alterados estão na área da elipse vermelha (que só foi desenhada por motivos ilustrativos, não pertencendo ao diagrama)

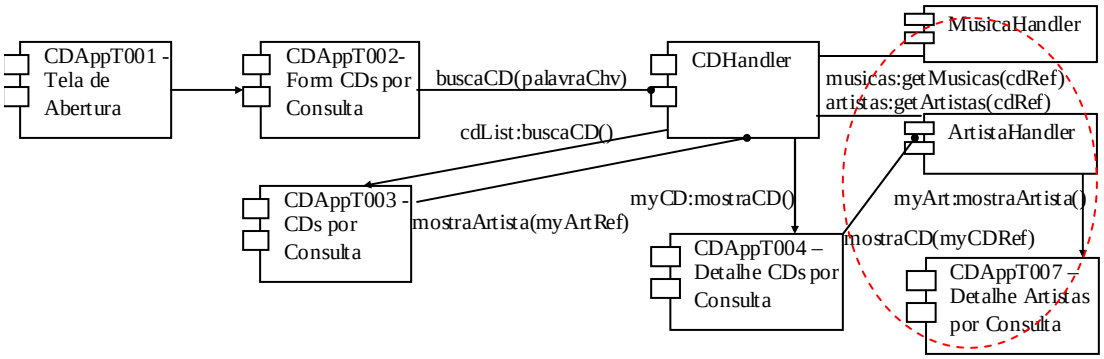

Figura 38 - Diagrama de Componentes de Implementação

Componentes do passo 3.2.1 Process: Task Project (TaskProject) da 2a Build Interaction
inalterado

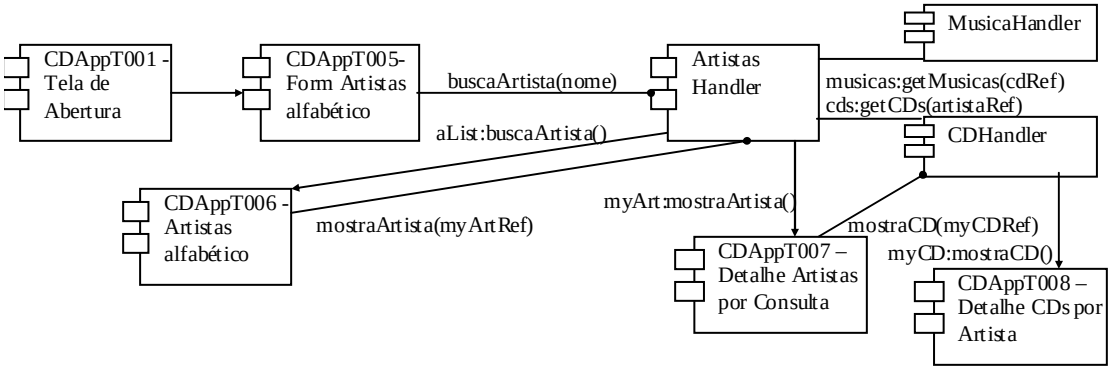

Figura 39 - Diagrama de Componentes de Implementação

5.3 Phase: Implementation (Implementation)

5.3.1 Process: Task Implementation (TaskImplmt)

Este processo foi realizado juntamente com o seguinte para construir os artefatos incrementalmente sobre os já existentes.

5.3.2 Process: Application Integration (AppIntegration)

Foi criado um arquivo, CDApEx1_v532.zip, que contém os códigos fonte deste passo.

5.3.3 Process: Implementation Validation (ImplementationValidation)

O usuário aprovou o sistema

4.2 Realizando a fase de Implementation utilizando o framewok OOHDMJava2.

As fases de *Implementation* das 3 *Build Interactions* apresentadas na seção anterior serão refeitas utilizando o framework OOHDMJava2 [20].

4.2.1 Passos para instanciar o Framework OOHDMJava2 visando implementar o exemplo de site de CDs utilizando AgileOOHDM

[1]"fazer um modelo computacional do domínio a ser implementado"
Ok, estamos aproveitando o modelo da 1ª parte de acordo com as suas *build interactions*.

4.2.1.1 Instanciação Do Módulo Transacional

"como será feita a persistência dos dados"
Resposta: SGBD, mais especificamente IBM UDB DB2 v.7.2 com auxílio dos recursos disponíveis no IBM WSAD 5.0

"criar tabelas que segundo o modelo conceitual OOHDM"
1ª Build Interaction
Ok. Criadas as tabelas segundo o modelo conceito-navegacional da 1ª *build interaction* – figura 1 - Diagrama Conceito-Navegacional da Consulta de CDs a partir de uma palavra.
Tabelas: artista, musica, cd, cd_musica e cd_artista.
2ª Build Interaction
Incluir os campos nascimento, descrição e foto na tabela do artista.

"definir a camada(s) de EJBs... Tendo criado o esquema do banco de dados, o próximo passo é decidir quais os objetos serão modelados como *entity beans*"
1ª Build Interaction
Modelar entity beans: ArtistaBean, MusicaBean e CDBean de espelhando o banco de dados criado anteriormente.
2ª Build Interaction
Alterar ArtistaBean de acordo com as alterações feitas no banco de dados (campos nascimento, descrição e foto).

"Também devemos definir para cada *entity bean*, o respectivo *State Object* que encapsula o estado *entity bean*"
1ª Build Interaction
Criar : ArtistaSO, MusicaSO e CDSO

[1] Os textos entre "aspas" correspondem a citações [20]

2ª Build Interaction
Alterar ArtistaSO de acordo com as alterações feitas no banco de dados (campos nascimento, descrição e foto).

"Se utilizarmos BMP..."
Não, estaremos utilizando CMP logo não criaremos DAOs associados aos entity beans.

"criar os *statefull* e *stateless session beans*"
1ª Build Interaction
Como tudo o que foi criado consiste basicamente de navegação, existe apenas a necessidade de criar stateless session beans para fazer a consultas a artistas, músicas e de cds . Dicas que levaram a escolha de stateless session beans: Modelar objetos reutilizáveis provedores de serviços; Operar sobre múltiplas linhas de uma vez; Prover visão procedural de dados.
Artefatos gerados: ArtistaHandlerBean, MusicaHandlerBean e CDHandlerBean
2ª Build Interaction
Alterar ArtistaHandlerBean, a consulta de artista retorna mais dados.

"Também devemos criar para cada *statefull session bean* que contenha estado o respectivo *State Object*. Além dos EJBs, devem ser criados os *State Objects* para os objetos conceituais que não deram origem a EJBs"
1ª Build Interaction
Não é necessário porque não existem statefull session beans e não sobraram objetos conceituais.
2ª Build Interaction
Mantido igual a 1ª Build Interaction

"criar os *Web Objects* para os EJBs cujo estado pode ser "espelhado" nos escopos de sessão ou aplicação da camada *web*."
1ª Build Interaction
De acordo com os diagramas de componentes do processo Task Project (TaskProject) no item 2.2.1, apenas CDHandler troca informações com telas de interface do sistema PORTANTO o web object será CDHandlerWO.
2ª Build Interaction
De acordo com os diagramas de componentes processo Task Project (TaskProject) no item 3.2.1, apenas ArtistaHandler e CDHandler trocam informações com telas de interface do sistema PORTANTO os web objects serão **ArtistaHandlerWO** e CDHandlerWO(já existente porém alterado para receber uma chamada de consulta de cd por artista).

"definir quais os objetos anteriores (EJBs, *State Objects*, *Web Objects*) devem implementar a interface *NavigationalBlackBoard*"
1ª Build Interaction

Pela interpretação da definição de *NavigationalBlackBoard*: os seguintes objetos o devem implementar: ArtistaSO, MusicaSO e CDSO; ArtistaHandlerBean, MusicaHandlerBean e CDHandlerBean; CDHandlerWO.

2ª Build Interaction

Pela interpretação da definição de *NavigationalBlackBoard*: os seguintes objetos o devem implementar: ArtistaHandlerWO.

"Devemos criar o *EJB Controller* específico da aplicação"

1ª Build Interaction

Criar o CDApEx2EJBController.

2ª Build Interaction

Mantido igual a 1ª Build Interaction

"Uma vez criado *EJB Controller* específico da aplicação, devemos criar o seu *proxy* na camada *web*, ou seja, o *Web Controller* específico da aplicação."

1ª Build Interaction

Criar o CDApEx2WebController e redefinir o método getEJBController que retorna o *EJB Controller* da aplicação.

2ª Build Interaction

Mantido igual a 1ª Build Interaction

"analisar os casos de uso e cenários para neles identificar os possíveis eventos de negócio da aplicação"

1ª Build Interaction

Neste caso será analisado o apenas o cenário e o UID da tarefa *Consultar CDs a partir de uma palavra* referente a 1ª build interaction. Eventos identificados: buscar cd por título gravadora e descrição; incluir 1 a n cds na lista de compras; e ouvir trecho de música.

2ª Build Interaction

Neste caso será analisado o apenas o cenário e o UID da tarefa *Pesquisar informações de um artista* referente a 2ª build interaction. Novos eventos identificados: buscar artista por nome.

"Depois de identificar eventos candidatos, devemos rever nossa análise para criar os eventos definitivos."

1ª Build Interaction

Esta tarefa foi facilitada pois o escopo de uma build interaction é bem menor.

Análise: o evento *incluir 1 a n cds na lista de compras* será apenas superficialmente implementado pois ainda não há tarefas modeladas ou implementadas sobre o objetos *lista de compras*; e *ouvir trecho de música* será implementado na interface apenas como um link para um arquivo de áudio, não sendo necessário criar um evento de negócio.

2ª Build Interaction

Análise: nada a declarar

"definir a estrutura de cada objeto evento e criar os objetos do evento. Para isso devemos identificar quais os EJBs afetados pelo evento. A estrutura do evento deve conter todos os

dados necessários para sua execução por seu respectivo *Event Handler*, dados estes retirados da requisição http pelo respectivo *Request Handler*."

1ª Build Interaction

Criar BuscarCDEvt, BuscarCDEH, BuscarCDRH e os esqueletos de IncluirCDnaListaEvt, IncluirCDnaListaEH e IncluirCDnaListaRH.

"Request Handler e *Event Handler* são responsáveis por sua instanciação e execução sobre o *model*, respectivamente. No caso do *Request Handler*, devemos ver quais os parâmetros necessários para instanciar o evento, traduzir os parâmetros http neles, e instanciar e retornar o evento. Já no caso do *Event Handler*, devemos identificar, para cada EJB, quais os métodos que devem ser invocados e em que ordem. O *Event Handler* recupera as informações necessárias trazidas pelo evento, invoca os EJBs pertinentes e retorna a lista com os nomes JNDI da interface *Home* de cada EJB que teve seu estado alterado, para que objetos observadores da camada *web* se atualizem."

2ª Build Interaction

Criar BuscarArtistaEvt, BuscarArtistaEH, BuscarArtistaRH

"criar objetos de exceção de evento (*Event Exception*) para cada evento, bem como os respectivos *Event Exception Handlers*."

1ª Build Interaction

Não há exceptions a criar a principio.

2ª Build Interaction

Não há exceptions a criar a principio.

4.2.1.2 Instanciação Do Módulo Navegacional

"criar para cada contexto o objeto *Navigational Context* ... consiste em obter os parâmetros do contexto a partir da requisição http, em seguida, acessar o *Navigational BlackBoard* adequado para recuperar os IDs dos nós do contexto, e retornar este IDs (*NodeIDs*)."

1ª Build Interaction

Criar CDCtx e CDporConsultaCtx

2ª Build Interaction

Criar ArtistaCtx, ArtistaporConsultaCtx e CDporArtistaCtx

3ª Build Interaction

Alterar CDporConsultaCtx para navegar para ArtistaporConsultaCtx

"definir para cada índice da aplicação o correspondente *Index Creator*. A implementação do *Index Creator* consiste, basicamente, em obter os parâmetros do índice a partir da requisição http, em seguida, acessar o *Navigational BlackBoard* adequado para recuperar a lista de objetos que contém as informações necessárias para montar o índice, e por fim, para cada objeto desta lista, criar uma entrada de índice (componente *Index Entry*) e adicionar ao índice. Para criar as âncoras de cada entrada do índice devemos acessar o *Navigational Manager* para obter informações de URLs de contextos e outros índices. Depois de montado o índice, devemos retorná-lo."

1ª Build Interaction

Criar CDporConsultaIdx

2ª Build Interaction

Criar ArtistaIdx, CDporArtistaIdx
3ª Build Interaction
Criar ArtistaporCDIdx

"criar para cada nó base (com os atributos comuns a todos os contextos) e nó em contexto o correspondente *Node Creator*. A implementação do *Node Creator* do nó base consiste, basicamente, em obter o ID do nó da requisição http, em seguida, acessar o *Navigational BlackBoard* correspondente para obter o objeto (*State Object*) que contém os dados do nó. Com isso, devemos crias os atributos, âncoras, listas e índices do nó. Novamente, aqui também o *Navigational Manager* é acessado para obter informações contextos, de índices e para obter índices prontos. Quando o nó estiver pronto devemos retorná-lo. Já o *Node Creator* do nó em contexto consiste em, primeiramente, invocar o *Node Creator* do nó base para obter o nó base pronto, e acrescentar os novos atributos ao nó base. Para isso, o *Node Creator* recupera os parâmetros da requisição http (ID do nó, parâmetros de grupos de contexto) e faz seu trabalho similarmente ao *Node Creator* do nó. Atributos muito comuns acrescentados são âncoras próximo e anterior do contexto corrente ou de outros contextos (navegação inter-contextual)."
1ª Build Interaction
Criar CDCC, CDporConsultaCC
2ª Build Interaction
ArtistaCC, ArtistaporConsultaCC e CDporArtistaCC
3ª Build Interaction
Criar ArtistaporCDCC

4.2.1.3 Definição Dos Arquivos Xml

"definir todos os arquivos XML da aplicação e instalar aplicação num servidor de aplicações para executá-la."
Arquivos XML de Configuração do módulo transacional
urlmappings.xml
interfaces.xml
eventmappings.xml
exceptionmappings.xml

Arquivos XML de Configuração do módulo navegacional
contextmappings.xml
indexmappings.xml
1ª Build Interaction
Criar todos os arquivos xml
2ª Build Interaction
Alterar os arquivos xml necessários
3ª Build Interaction
Alterar os arquivos xml necessários

4.3 Análise dos Casos de Teste

Com relação aos casos de teste as seguintes questões foram levantadas:

- no passo 1.1.4 Process: Requirements Validation (Validation) – o usuário poderia não ter aprovado os UID´s, ficou então a dúvida da necessidade de mais um processo para atualização de UID´s (*UID Updating*). Esta hipótese foi descartada pois o OOHDM já considera estas etapas de revisão no ciclo de validação dos UID´s, portanto não sendo preciso sua criação.

- os seguintes passos extendem um princípio do XP chamado *Continuous Integration*:
  2.1.2 Process: Application Model Specification (AppModelSpec)
  2.2.2 Process: Application Project Specification (AppProjectSpec)
  2.3.2 Process: Application Integration (AppIntegration)
  No caso do Extreme Programming este princípio se aplica apenas a codificação, aqui estamos usando em etapas de design. De um certo modo, e ainda a ser mais bem aprofundado e avaliado, poderia se dizer que estamos tentando fazer com o AgileOOHDM um XP de design.

- Talvez o melhor lugar para as informações sobre as sobre as telas do sistema fosse nos respectivos códigos fonte e não nos artefatos apresentados em:
  2.2.1 Process: Task Project (TaskProject)
  3.2.1 Process: Task Project (TaskProject)
  5.2.2 Process: Task Project (TaskProject)

- no passo 2.3.3 Process: Implementation Validation (ImplementationValidation): Esta fase em alguns casos pode ser pulada de acordo com entendimento entre o usuário e o projetista. *Build Interactions* podem ser agrupadas e então validadas de uma vez só. Quando ocorreria este entendimento? Novo processo: *Build Interaction Planning* - seria o último processo da fase de *Planning and Requirements Gathering* e definiria ordem e grupos de *Build Interactions*

- percebeu-se no passo 3.3.2 Process: Application Integration (AppIntegration) que a navegação intra e intercontextual torna-se cada vez mais difícil fazendo “na mão” no caso de não estar se utilizando o framework.

- Uma vez que é mais pratico trabalhar nos artefatos já integrados da aplicação ao invés de se atualizar a tarefa e depois integrá-la, evitando assim o retrabalho, os seguintes passos foram pulados:
  5.1.1 Process: Task Design (TaskDesign)
  5.2.1 Process: Task Project (TaskProject)
  5.3.1 Process: Task Implementation (TaskImplmt)

- Talvez nas etapas de *Implementation Validation* poderia haver algum sub processo de teste funcional como ocorre no XP.

- Na implementação utilizando o framework OOHDMJava2 a requisição de alteração do usuário não afetou, a princípio, o módulo transacional.

- A implementação utilizando o framework OOHDMJava2 mostrou-se mais restritiva porém mais rápida de se implementar ao passo que a implementação sem o mesmo, apesar de ter mais liberdade de codificação, era necessário em vários momentos ficar “reinventando a roda” para implementar conceitos básicos de aplicações web, o que tornou esta opção mais lenta.

- Em uma posterior análise foi percebido que a utilização do framework OOHDMJava2 no caso de teste poderia ser simplificada tornando a implementação mais simples.

5 Conclusões

Uma das questões mais pertinentes se refere a granularidade da interação com o cliente. Conforme proposto inicialmente pelo AgileOOHDM, cada cenário seria desenhado em um UID que representaria então uma tarefa. Esta tarefa geraria uma *Build Interaction*. Com os casos teste do capítulo anterior foi possível observar que a requisição de mudança feita pelo cliente não foi complexa o suficiente para alterar o módulo transacional. Além disto, de acordo com o OOHDM o projetista poderia prever solicitações como estas e sugeri-las de antemão caso as duas *Build Interacions* fossem reunidas. Também foi exposto que a fase de validação, em alguns casos, pode ser pulada de acordo com entendimento entre o usuário e o projetista. *Build Interactions* podem ser agrupadas e então validadas de uma vez só.

O outro extremo da granularidade, não abordada pelo AgileOOHDM, é a construção do sistema como um todo e então passar por vária fases de validação ou manutenção. Este procedimento já foi desconsiderado por nós frente à evidência de que estamos num ambiente em constante mudanças levando-nos portanto a necessidade de um método que propague uma requisição de mudança até a sua implementação o mais rápido possível.

Isto posto, deve-se considerar as *Build Interactions* como um conjunto de UID´s/Tarefas onde ideal seria regular estes conjuntos para fornecer ao cliente informação de qualidade e quantidade ótimas para que o mesmo pudesse ter uma visão concisa e não dispersante, diminuindo as solicitações de mudança e conseqüentemente as relações entre as *Build Interactions* também. Assim sendo, tem-se a seguinte visão de um processo de desenvolvimento:

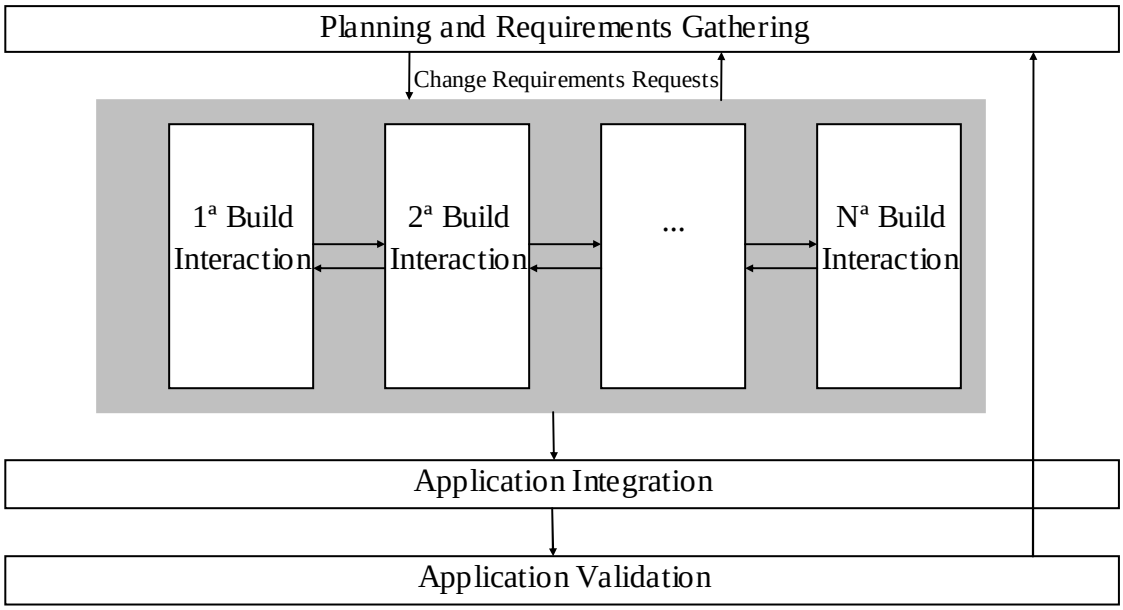

Figura 40 – Nova Visão do Processo de Desenvolvimento

Onde deve-se tentar minimizar as interações entre as *Build Interactions* através do agrupamento ótimo de UID´s/Tarefas e sendo que este processo de desenvolvimento também pode ser alcançado com o OOHDM utilizando-se o recurso de subsistema.

Prosseguindo sobre a proposta inicial do AgileOOHDM é importante relembrar que os processos não estão precisamente definidos. Ainda existe uma necessidade de avaliação da possibilidade de juntar o *Conceptual Model* e o *Navigational Model* na fase *Concept-Navigational Modeling* pois o seu diagrama ainda está notadamente confuso.

Um passo futuro, além do aprimoramento da proposta do AgileOOHDM, seria encaixar ou pelo menos estudar a integração de um processo de IHC como o MoLIC [23] para que as questões da interação do usuário com a aplicação seja mais bem trabalhada.

Uma última constatação é que trabalhar com métodos de desenvolvimento de software é uma tarefa naturalmente complicada e de acordo com Berry [25] até "dolorosa". Esta afirmação se fundamenta no fato de que computadores digitais e sistemas de software são uma das coisas mais complexas que o ser humano construiu[24], portanto o seu desenvolvimento não haveria de ser simples. Frameworks, como o proposto por Daflon [26], poderiam ajudar na tarefa de definição, manutenção e utilização de processos de desenvolvimento. Outra ferramenta que poderia ser de grande utilidade seria um "book of knowledge" dos métodos existentes, pois conforme foi visto no capítulo 2, quando da modelagem do OOHDM e XP, diversas fontes tiveram que ser consultadas.

6 Referências Bibliográficas

[1] - BECK, K. et al. “Manifesto for Agile Development” http://www.agilemanifesto.org/ acessado em Junho de 2003

[2] - ROSSI, G.; SCHWABE, D.;“Um método orientado a objetos para o projeto de Aplicações Hipermídia”, Tese de Doutorado, Departamento de Informática PUC-Rio, Brasil. 1996.

[3] - JALOTE, P. “An integrated approach to software engineering” 2nd. ed. New York: Springer, c1997.

[4] - PRESSMAN, Roger S. “Software engineering :: a practitioner's approach” 2nd. ed. - New York : McGraw-Hill, 1987.

[5] - National Conference of Commissioners on Uniform State Laws. UCITA, “Final Act With Comments”, 23 de agosto de 2001

[6] - The Standish Group. “The CHAOS Report” http://www.standishgroup.com/sample_research/chaos_1994_1.php acessado em Junho de 2003

[7] - BECK, K. “Extreme Programming Explained Embrace Change” Addison Wesley Longman, Inc. 2000.

[8] - FOWLER, M. “The New Methodology” www.martinfowler.com/articles/newMethodology.html acessado em Junho de 2003

[9] - BOOCH, G. “The Illusion of Simplicity” http://www.sdmagazine.com/documents/s=825/sdm0102m/ acessado em Junho de 2003

[10] - GIBBS, W. “Software's chronic crisis" Scientific American. 1994

[11] - FONTOURA, M., "An Environment for Process Modeling and Execution", Dissertação de Mestrado. Departmento de Informática, PUC-Rio, Brasil. 1997.

[12] - LOWE, D., WEBBY, R.; "The Impact Process Modelling Project", 1st International Workshop on Hypermedia Development, at ACM Hypertext 98, Pittsburgh, 1998

[13] - OLSINA, L.A. "Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web", Tese de Doutorado, 1999.

[14] COLLINS, C., Miller, R., “XP Distilled”, http://www.rolemodelsoftware.com/moreAboutUs/publications/xpDistilled.php acessado em janeiro de 2003

[15] www.extremeprogramming.org acessado em fevereiro de 2003

[16] MILLER, R., “Desmystifying Extreme Programming” http://ibm.com/ acessado em janeiro de 2003

[17] VILAIN, P. “Notação do Método OOHDM Versão 2.0”

[18] MEDEIROS, A. P. “Especificação Declarativa e Implementação de Aplicações Hipermídia na Web”. Dissertação de Mestrado. Departmento de Informática, PUC-Rio, Brasil. 2001.

[19] GÜELL, N. “Modeling Interactions and Navigation in Web Applications”. Proceedings of the World Wild Web and Conceptual Modeling'00 Workshop, ER'00 Conference, Springer, Salt Lake City, 2000.

[20] JACYNTHO, M. D. de A. “OOHDM-Java2 Um Framework e uma arquitetura para implementação de aplicações hipermídia na web”, Dissertação de mestrado, Departamento de Informática PUC-Rio, 2001.

[21] PRESSMAN, R. Can Internet-Based Applications Be Engineered? IEEE Software (Sep/Oct 98).

[22] *Anexo C: Projeto OOHDM de uma Loja de CDs*, parte integrante da disciplina *Análise e Projeto na Web* do curso de *Desenvolvimento de Aplicações para Web* da CCE PUC-Rio.

[23] PAULA, M.G. “Projeto da Interação Humano-Computador Baseado em Modelos Fundamentados na Engenharia Semiótica: Construção de um Modelo de Interação” Dissertação de Mestrado. Departmento de Informática, PUC-Rio, Brasil. 2003.

[24] BROOKS, F., "No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Comp. 20,4 (Apr 87); pp.10-19

[25] BERRY, D. M. “The Inevitable Pain of Software Development: Why There Is No Silver Bullet”

[26] DAFLON, L. “Um Framework para a Definição e Análise de Processos de Desenvolvimento de Software” Dissertação de Mestrado. Departmento de Informática, PUC-Rio, Brasil. 2003.