Fernando Silva Parreiras
Programa de Pós-Graduação em Ciência da Informação
Escola de Ciência da Informação
Universidade Federal de Minas Gerais

GERAÇÃO DE SISTEMAS DE GESTÃO DE CONTEÚDO COM SOFTWARES LIVRES

Belo Horizonte
2005

Fernando Silva Parreiras
sob a orientação do
Professor Marcello Peixoto Bax

GERAÇÃO DE SISTEMAS DE GESTÃO DE CONTEÚDO COM SOFTWARES LIVRES

Dissertação de mestrado, submetida ao Programa de Pós-Graduação em Ciência da Informação da Escola de Ciência da Informação da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Ciência da Informação.

Área de concentração: Gestão da Informação e do Conhecimento.

Belo Horizonte
Escola de Ciência da Informação da UFMG
2005

Parreiras, Fernando Silva.

P259g Geração de sistemas de gestão de conteúdo com softwares livres [manuscrito] / Fernando Silva Parreiras. – 2005.
72 f.: il.

Orientador: Prof. Dr. Marcello Peixoto Bax.
Dissertação (Mestrado) – Universidade Federal de Minas Gerais, Escola de Ciência da Informação, 2005.

1. Ciência da informação – Teses 2. Tecnologia da informação – Teses 3. Gerenciamento de recursos informacionais – Teses I. Título II. Bax, Marcelo Peixoto. III. Universidade Federal de Minas Gerais. Escola de Ciência da Informação.

CDD: 006.332
CDU:004.823:025.43

AGRADECIMENTOS

A Deus, pela minha vida;

Aos meus pais, Ataídes Luiz Parreiras e Elvira Vieira Silva Parreiras, por me mostrarem o caminho. Aos meus irmãos Tatiane Aparecida Silva Parreiras e Fábio Ataíde Silva Parreiras pelo apoio;

À minha namorada Alessandra da Silva Vaz de Melo, pelo carinho, apoio e paciência sempre presentes;

À CAPES, pelo o apoio financeiro aos meus estudos e pesquisa;

Aos meus amigos do Netic, Antonio Braz, Wladmir Brandão e Jaime Bastos, pelos períodos construtivos;

Ao professor Marcello Peixoto Bax, pela orientação. Aos professores Alberto Henrique Frade Laender, Lídia Alvarenga e Ricardo de Oliveira Duarte, por participarem da banca examinadora;

Ao professor Wilson de Pádua, pela prontidão em atender minhas solicitações. Ao Bruno Rezende pela consultoria técnica;

A Maria Goreth Gonçalves Maciel, Viviany Maria Braga de Carvalho, Sônia Jaqueline Gonçalves e Cláudia Márcia De Lucas, pelo pronto apoio durante todo o curso;

A todos os professores, colegas de curso e funcionários que ajudaram ao longo do curso, com informações, esclarecimentos e idéias.

“O Senhor é o meu pastor, nada me faltará”.

Salmos 23:1

“The more precisely the position is determined,
the less precisely the momentum is known in this instant,
and vice versa”.

Werner Heisenberg

Geração de sistemas de gestão de conteúdo com softwares livres

Fernando Silva Parreiras

Resumo

Esta dissertação trata a geração de sistemas de gestão de conteúdo, usando softwares livres. Pergunta-se se é possível, com base nas principais contribuições teóricas existentes e nos arcabouços disponíveis, implementar um processo que permita refletir um domínio de conhecimento, representado por meio de modelos, em um sistema de gestão de conteúdo. Pretende-se analisar as principais contribuições teóricas existentes acerca da modelagem e geração de sistemas de informação, estabelecer os passos necessários para implementação desta abordagem e levantar um arcabouço de desenvolvimento de sistemas de gestão de conteúdo. Foi realizada uma pesquisa experimental tendo como escopo o domínio de conhecimento de um processo de desenvolvimento software, composta de três entidades: (a) domínio de conhecimento modelado, (b) um sistema de gestão de conteúdo e (p) o processo de geração utilizado. Como resultado, tem-se o sistema de gestão de conteúdo gerado e o processo de geração, composto de cinco atividades: (1) Modelagem Independente de Plataforma; (2) Modelagem Específica de Plataforma; (3) Geração do código-fonte, (4) Manutenção Evolutiva do Código-fonte; e (5) Instalação. Este processo de geração diminui em mais de 50% o tempo estimado para desenvolvimento do sistema, mas apresenta algumas restrições. Conclui-se, entre outras questões, que: (1) a UML é uma boa candidata à linguagem de representação do conhecimento; (2) a modelagem em vários níveis é uma necessidade crucial para a evolução da área de sistemas de informação; (3) o arcabouço utilizado apresenta vantagens na prototipagem ou aplicação em domínios de conhecimento reduzidos; e (4) é possível compor um arcabouço que facilite a construção de sistemas de informação e sua manutenção.

Generating content management systems using open source softwares

Fernando Silva Parreiras

Abstract

This dissertation handles the content management systems generation using open source softwares. It asks if it is possible, based on the main existing theoretical contributions and in the available frameworks, to implement a process that allows reflecting a knowledge domain, represented by models, in a content management system. It is intended to analyze the main existing theoretical contributions concerning the modeling and generation of information systems, to establish the steps necessary for implementation of this approach and to raise framework of content management system development. It was made an experimental research using a software development process as scope, composed of three entities: (a) the knowledge domain, (b) a content management system and (p) the adopted process. As result, a content management system was generated and a process made up of five activities: (1) platform independent modeling, (2) platform specific modeling, (3) source codes generation, (4) source codes maintenance and (5) installation. This process uses 50% of the time planed for of the system development, but it presents some restrictions. I concludes that (1) the UML is a good candidate to a knowledge representation language, (2) modeling using levels of abstraction is a crucial necessity for the evolution of the area of information systems, (3) the used framework presents advantages in prototyping or application in reduced knowledge domains and (4) is possible to consider a framework that facilitates the information systems building and its maintenance.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AIE	Arquivos de Interface Externa
ALI	Arquivos Lógicos Internos
APF	Análise por Pontos de Função
CE	Consulta Externa
CF	Código-fonte
CMM	Capability Maturity Model
COLD	*Computer Output to Laser Disc*
CP	Coeficiente de Produtividade
EE	Entrada Externa
GED	Gestão Eletrônica de Documentos
IFPUG	*International Function Point User Group*
ISO	*International Standard Organization*
MCT	Ministério da Ciência e Tecnologia
MDA	*Model Driven Architecture*
MOA	Modelo de Objeto Adaptável
MOF	*Meta Object Facility*
OMG	*Object Management Group*
PFNA	Pontos de Função Não Ajustados
PIM	*Platform Independent Model*
PSM	*Platform Specific Model*
SE	Saída Externa
SEI	*Software Engineering Institute*
SGC	Sistema de Gestão de Conteúdo
UML	*Unified Modeling Language*
XMI	*XML Metadata Interchange*
XSLT	*Extensible Stylesheet Language Transformation*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, com a rápida depreciação do valor investido no código-fonte de sistemas, pesquisas sobre modelagem em camadas emergem da engenharia de software BEALE, 2002, FRANKEL, 2003), tentando separar em distintos níveis de abstração a representação de um domínio de conhecimento, de forma que o especialista de domínio seja capaz de modelar em nível diferente do analista de sistemas.

Por outro lado, a explosão das comunidades de softwares livres, a criação de repositórios de projetos e adoção deste tipo de software pelas organizações e governos cria um cenário onde o conhecimento sobre que ferramenta resolve um dado problema é valioso.

Uma terceira questão, mais explorada, é a grande quantidade de conteúdo digital gerado nas organizações. Com a necessidade de compartilhar documentos de maneira rápida e fácil utilizando a web, surgem os sistemas de Gestão de Conteúdo (SGCs). Gestão de Conteúdo é uma abordagem que surge em função da explosão de conteúdo multimídia na web e em intranets e visa a permitir a gerência de todas as etapas, desde a criação até a publicação de conteúdo (PEREIRA, BAX, 2002).

Estas três questões incentivam o desenvolvimento de pesquisas que envolvam a geração de sistemas de gestão de conteúdo baseados de softwares livres. Esta dissertação tem como problema de pesquisa verificar até que ponto é possível, com base em contribuições teóricas e em arcabouços de ferramentas disponíveis em código aberto, implementar um processo que permita refletir um domínio de conhecimento modelado em um sistema de gestão da informação. O problema em sua generalidade parece intratável, porém, reduzindo-se o escopo a um tipo específico de sistema de informação e a um determinado domínio de conhecimento, acredita-se que bons resultados possam ser alcançados atualmente com

softwares livres. Dessa forma a pesquisa focaliza-se sobre os sistemas de informação voltados à gestão de conteúdo, denominados Sistemas de Gestão de Conteúdo (SGC). Com fins de investigação prática e verificação empírica dos conceitos tratados, um estudo de caso foi realizado.

1.1 Objetivos

O objetivo geral desta pesquisa é analisar aspectos referentes à modelagem e geração de SGCs.

As seguintes questões de pesquisa são colocadas:

- Identificar e analisar as principais contribuições teóricas existentes acerca da modelagem e geração de sistemas de informação, especificamente os SGCs, sistematizando uma revisão da literatura pertinente;
- Levantar um arcabouço passível de utilização na modelagem e geração desses sistemas;
- Estabelecer os passos necessários para modelagem e geração de SGCs.

1.2 Organização da dissertação

Esta dissertação organiza-se em 7 capítulos.

O Capítulo 2 compreende os fundamentos teórico-metodológicos da pesquisa, com o objetivo de identificar as principais contribuições existentes acerca do tema de pesquisa. Esta análise não se pretende exaustiva e aborda apenas os aspectos tangentes ao estudo de caso.

O Capítulo 3 apresenta os trabalhos relacionados com esta pesquisa. São apresentadas algumas propostas de geração de sistemas de informação baseadas em MDA, propostas de geração de bibliotecas digitais e propostas de geração de portais.

O Capítulo 4 traz a metodologia adotada no estudo de caso realizado em um ambiente simulado. O Capítulo 5 descreve o estudo de caso, delimitando o escopo, discriminando o arcabouço utilizado e apresentando o processo de geração utilizado.

O Capítulo 6 discute os resultados e, finalmente, conclui-se com o Capítulo 7, onde se discutem trabalhos futuros e registram-se as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

O referencial teórico-metodológico divide-se em quatro partes e pretende orientar o estudo de caso. A primeira parte fornece uma visão sobre modelos e modelagem de domínios de conhecimento. A segunda parte conceitua os SGCs e suas especificidades. A terceira parte concentra-se na geração automática de sistemas de informação. A quarta parte descreve brevemente um processo de desenvolvimento de software, que é o domínio de conhecimento utilizado no estudo de caso.

2.1 Modelagem de domínios de conhecimento

Os modelos desempenham um papel importante como recurso metodológico. Sua aproximação sucessiva da realidade oferece aos engenheiros do conhecimento insumo para a modelagem de um dado domínio de conhecimento. A seguir, apresentam-se as características essenciais dos modelos, assim como algumas abordagens de modelagens disponíveis na literatura.

2.1.1 Modelos

Os modelos se apresentam como uma importante ferramenta no exercício da abstração. Um modelo é destinado a representar uma realidade, ou alguns dos seus aspectos, a fim de torná-los descritíveis qualitativa e quantitativamente e, algumas vezes, observáveis (SAYÃO, 2001). Um modelo também exige um modo de expressão que pode ser, entre

outros, matemático, gráfico, esquemático, físico ou discursivo (BURT, KINNUCAN, 1990).

Stachowiak (1972) apresenta três características essenciais dos modelos:

- **Mapeamento**: modelos são representações de "originais" (ou "protótipos"), naturais ou artificiais, que, por sua vez, também podem ser modelados;
- **Redução**: modelos geralmente não mapeiam todos os atributos do original que eles representam, mas somente aqueles que são relevantes para quem modela;
- **Pragmatismo**: modelos não são em si pertencentes à mesma classe que seus originais.

Os diversos tipos de modelo permitem a visualização de níveis de abstração progressivos em direção ao usuário do sistema modelado; o que leva à necessidade de trabalhos em diferentes níveis de abstração, com o objetivo de representar todo o conhecimento gerado nestes níveis. Contudo, cada modelo exige um processo de modelagem específico, conforme suas características.

2.1.2 Abordagens de modelagem

A partir de uma análise da literatura identificaram-se, pelo menos, cinco abordagens de modelagem aplicadas na Ciência da Informação: ontologias, teoria da classificação facetada, teoria geral da terminologia, teoria do conceito e modelagem orientada a objetos. Essas abordagens são apenas delineadas no texto que segue. Uma análise comparativa e mais detalhada dessas abordagens é feita por Campos (2004).

A abordagem de modelagem usando **ontologias** tem sido bastante discutida, alcançando progressos teóricos significativos, como pode ser observado em Staab e Studer (2004). Segundo Moreira, Alvarenga e Oliveira (2004), a interpretação mais popular dentro da comunidade de representação de conhecimento é a de "ontologia como uma especificação de

uma conceituação", e sua definição é apresentada por Gruber (1993) como "especificação formal e explícita de uma conceituação compartilhada". Ontologias formais são também estudadas por Cocchiarella (1991), Guarino (1995) e Sowa (2000).

A **classificação facetada** foi desenvolvida por Shiyali Ramamrita Ranganathan, na década de 1930, e, segundo Tristão, Fachin e Alarcon (2004), tem sido, atualmente, largamente discutida na academia como uma solução para a organização do conhecimento, em decorrência de suas potencialidades de acompanhar as mudanças e a evolução do conhecimento. Mais tarde, Wuester desenvolveu uma série de princípios que chamou de **teoria geral da terminologia** (WUESTER, 1981), organizando a terminologia de eletro técnica, com o objetivo de garantir comunicação precisa nesse campo da ciência. Esta experiência levou-o à criação de uma nova disciplina científica, a ciência da terminologia.

Algumas metodologias de engenharia de ontologias (FERNANDÉZ, GÓMEZ-PÉREZ, JURISTO, 1997, NOY, MCGUINNESS, 2001, GÓMEZ-PÉREZ, 2003) possuem etapas que se inspiram na teoria da classificação facetada. Isto demonstra a atualidade da teoria e sua contribuição ainda presente no desenvolvimento de novas metodologias. Além disso, a abordagem de conceituação na modelagem orientada a objetos e no desenho de bancos de dados é muito similar ao processo de conceituação no desenho de um esquema de classificação para um assunto específico (NEELAMEGHAN, 1992).

Na década de 70, Dahlberg, aluna de Wuester, traz para o campo da documentação os princípios terminológicos, e desenvolve a **teoria do conceito** no campo da terminologia (DAHLBERG, 1978), que tem por princípio básico a seguinte definição de conceito:

> *"[...] uma tríade formada por (A) um referente (qualquer objeto material ou imaterial, atividade, propriedade, dimensão, tópico, fato, etc.), (B) os enunciados (predicações) verdadeiros e essenciais sobre um referente que estabelecem as características sobre o referente e (C) o termo, que é forma externa e comunicável do referente e suas características." (tradução de MOREIRA, 2003) (DAHLBERG, 1981, p.16).*

A **modelagem orientada a objetos** é utilizada como base por grande parte dos sistemas de informação desenvolvidos nos dias de hoje. Tal paradigma surgiu com a programação de computadores no final dos anos sessenta, mas só foi utilizado na análise de sistemas no final da década de oitenta (RUMBAUGH *et al*., 1990). Objeto e Classe são conceitos básicos relacionados a esta abordagem.

Um objeto é a representação no computador de coisas físicas ou abstratas do mundo, que possuem atributos e métodos. Os atributos representam as características do objeto, enquanto os métodos são procedimentos ou funções executadas pelo objeto. Uma classe de objetos representa um conjunto de objetos com as mesmas características. Esta breve descrição destaca o caráter comportamental deste modelo, frente ao caráter predominantemente estrutural dos anteriores.

Para a representação gráfica do modelo orientado a objetos utiliza-se comumente a linguagem UML; padrão para a modelagem de software. Com ela os analistas especificam e documentam os artefatos de um sistema de informação (BOOCH, RUMBAUGH, JACOBSON, 2000). Seu vocabulário abrange três tipos de bloco de construção: itens, relacionamentos e diagramas.

Os itens são as abstrações identificadas em um modelo e podem ser de quatro tipos: itens estruturais (classes, casos de uso, componentes, etc.); itens comportamentais (mensagens e estados); itens de agrupamento (pacotes); e itens notacionais (notas).

Os relacionamentos são blocos relacionais básicos de construção da UML e podem ser de quatro tipos: dependência, associação, generalização e realização.

A representação gráfica de um conjunto de elementos é chamada de diagrama UML. Existem nove tipos de diagrama: diagrama de classes; diagrama de objetos; diagrama de casos de uso; diagrama de seqüências; diagrama de colaborações; diagrama de gráficos de estados; diagrama de atividades; diagrama de componentes; e diagrama de implantação.

Segundo Sayão (2001), um bom modelo traz, em si, na sua própria estrutura, sugestões para a sua própria extensão e generalização. A UML possui três mecanismos de extensão: estereótipos, que ampliam o vocabulário da UML; valores atribuídos, que estende as propriedades dos blocos de construção; e as restrições que ampliam a semântica dos blocos de construção da UML. Um grupo predefinido de estereótipos, valores atribuídos e restrições que, conjuntamente, especializam e configuram a UML para um determinado domínio de aplicação ou para um determinado processo de desenvolvimento denomina-se perfil UML (DESFRAY, 1999).

A UML tem se tornado padrão de fato na engenharia de software, e vem sendo utilizada com sucesso também como linguagem de representação (CRANEFIELD, 2001) e de modelagem de ontologias (CRANEFIELD, PURVIS, 1999, FELFERNIG, FRIEDRICH, JANNACH, 2000).

As abordagens apresentadas nesta seção são utilizadas na modelagem de sistemas de informação de vários tipos, inclusive sistemas de gestão da informação.

2.2 Sistemas de gestão da informação

Os sistemas de gestão da informação são considerados por autores como Davenport e Prusak (1998) e Carvalho (2000) como elementos indispensáveis em ambientes de gestão do conhecimento. A fim de definir um tipo específico de sistema de gestão da informação, isto é, o sistema de gestão de conteúdo (SGC), relaciona-se a seguir os conceitos de informação, documento e conteúdo.

Otlet (1989) considera registros gráficos e textuais como representações de idéias ou de objetos. Para o autor, objetos podem ser considerados documentos, pois informações

podem ser obtidas a partir de sua observação. Como exemplos, têm-se objetos como peças arqueológicas, modelos explicativos, peças de arte, etc. Assim, o livro, a revista, o jornal, a estampa, etc. são considerados documentos.

Esta visão de documento confunde-se com o conceito de informação como coisa, proposto por Buckland (1991), para o qual o termo "informação" também é usado para objetos que são considerados informativos. Segundo Buckland (1991), o conceito de informação "como coisa" é de interesse especial no estudo de sistemas de informação. Têm-se, desta forma, os conceitos de informação e de documento apresentados por Buckland (1991) e Otlet (1989), respectivamente, como sinônimos.

Em geral, a gestão da informação remete a um ciclo, apresentado por Dodebei (2002) como "Ciclo da Informação", ou processo de "Transferência da Informação" (FIGURA 1). Tal processo é dividido em seis fases: produção, registro, aquisição, organização, disseminação e assimilação.

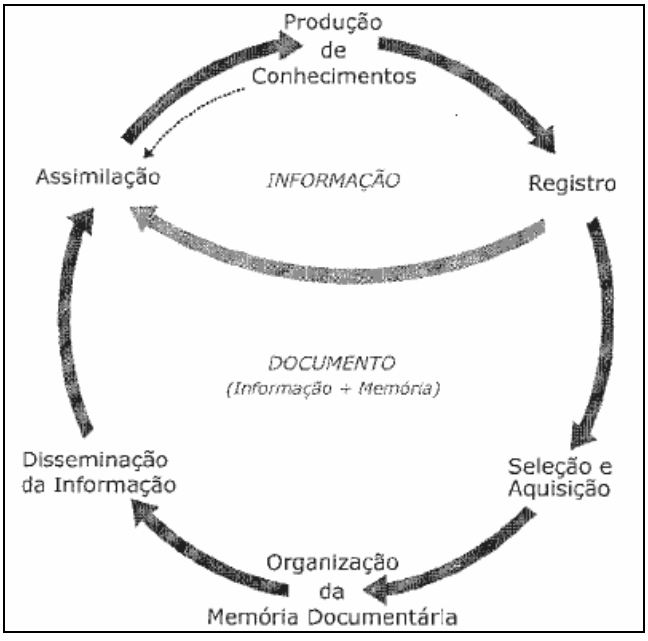

FIGURA 1 - Ciclo da Informação.

Fonte: Dodebei (2002) apud Tristão, Fachin e Alarcon (2004).

Este processo é apoiado por sistemas de gestão eletrônica de documentos, ou de conteúdo, generalizando a noção de documentos.

Para o CENADEM[1], Gestão Eletrônica de Documentos (GED) é "um conjunto de tecnologias que permite o gerenciamento de documentos de forma digital". Tais documentos podem ser das mais variadas origens e mídias, como papel, microfilme, som, imagem e mesmo arquivos já criados na forma digital.

Com a necessidade de compartilhar documentos de maneira rápida e fácil utilizando a Web, surgem os sistemas de Gestão de conteúdo. Para Pereira e Bax (2002) Gestão de conteúdo é uma abordagem que surge em função da explosão de conteúdo multimídia na Web e em *Intranets* e visa a permitir a gerência de todas as etapas, desde a criação até sua publicação. Esse conteúdo pode ser estruturado ou não, procedente de sistemas de imagem, *Computer Output to Laser Disc* (COLD), gerenciamento de documentos, sistemas legados, bancos de dados, arquivos em diretórios e de qualquer outro arquivo digital como som, vídeo etc. (CENADEM). A FIGURA 2 pretende contextualizar os SGCs perante outros tipos de sistema. Uma biblioteca digital e um SGC podem ser considerados especializações de um sistema de gestão de informação.

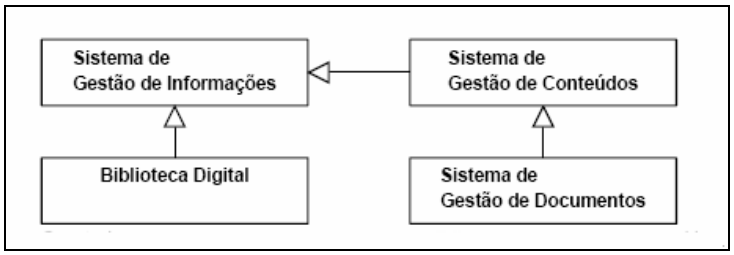

FIGURA 2 - Sistemas de gestão da informação.

[1] O CENADEM (Centro Nacional de Desenvolvimento do Gerenciamento da Informação) é uma organização que se dedica exclusivamente ao assunto GED.

Um SGC oferece acesso ao conteúdo da empresa por meio de uma interface única baseada em um navegador Web. As funcionalidades essenciais, dentre muitas outras, que caracterizam o conceito e que se desenvolvem a medida que novos produtos de mercado chegam à maturidade são (BAX, PARREIRAS, 2003): (a) Gestão de usuários e dos seus direitos (autenticação, autorização, auditoria); (b) Criação, edição e armazenamento de conteúdo em formatos diversos (*html, doc, pdf,* etc.); (c) Uso intensivo de metadados ou propriedades que descrevem o conteúdo; (d) Controle da qualidade de informação, com fluxo ou trâmite de documentos; e) Classificação, indexação e busca de conteúdo (recuperação da informação com mecanismos de busca); (f) Gestão da interface com os usuários (usabilidade, arquitetura da informação); (g) Sindicalização, publicação da informação em formatos XML visando seu agrupamento ou agregação de diferentes fontes; (h) Gestão de configuração (controle de versões); (i) Gravação das ações executadas sobre o conteúdo para efeitos de auditoria e possibilidade de desfazê-las em caso de necessidade.

2.3 Geração automática de sistemas de informação

Com o aumento do número de sistemas de informação utilizados nas organizações, a possibilidade de gerá-los automaticamente deve ser considerada. A geração automática transforma a especificação do sistema, representada por meio de modelos, num conjunto de artefatos constituintes do sistema final, levando em conta uma determinada arquitetura.

Atualmente alguns fatores contribuem para viabilizar a geração automática de sistemas. Dentre os quais se cita:

(1) A existência de uma linguagem de modelagem como a UML (BOOCH, RUMBAUGH, JACOBSON, 2000), padrão de fato, reconhecido e usado por grande parte da indústria de software.

(2) A noção de perfis UML que permite a criação de diversos níveis de abstração sucessivos durante a modelagem, separando o conhecimento em diversas camadas (ARLOW, NEUSTADT, 2003, BEALE, 2002, e FRANKEL, 2003).

(3) A existência de uma linguagem-padrão para representação de dados e *metadados*, a linguagem de marcação estendida XML (YERGEAU, 2004), e o padrão de intercâmbio de metadados, *XML Metadata Interchange* ou XMI (HEATON, 2002b).

(4) Melhor conhecimento sobre padrões e arquiteturas de software de qualidade, como resultado de anos de experiências, reflexão e reengenharia das melhores práticas, em numerosos projetos (GAMMA *et al*., 1995, BUSCHMANN *et al*., 1996, HOFMEISTER, NORD, SONI, 1999, FRANKEL, 2003).

2.3.1 Arquitetura dirigida por modelos

Em 2001, o *Object Management Group* (OMG) definiu uma abordagem de modelagem chamada Arquitetura Dirigida por Modelos – *Model Driven Architecture* (MDA). A abordagem consiste na adoção de modelos em diferentes níveis de abstração, com características distintas, a fim de isolar os aspectos computacionais do domínio de conhecimento.

Para a realização de modelos de conhecimento, é necessário um ambiente de execução computacional capaz de dar suporte à interpretação destes modelos. Consideram-se, aqui, modelos gráficos (diagramas UML) ou textuais (código-fonte). Este ambiente é chamado de plataforma (ARLOW, NEUSTADT 2003). Uma característica das plataformas é

que elas podem ser empilhadas, formando pilhas de plataformas. Isto significa que uma plataforma funciona acoplada a outra, conforme exemplificado na FIGURA 3a, pela linguagem Java.

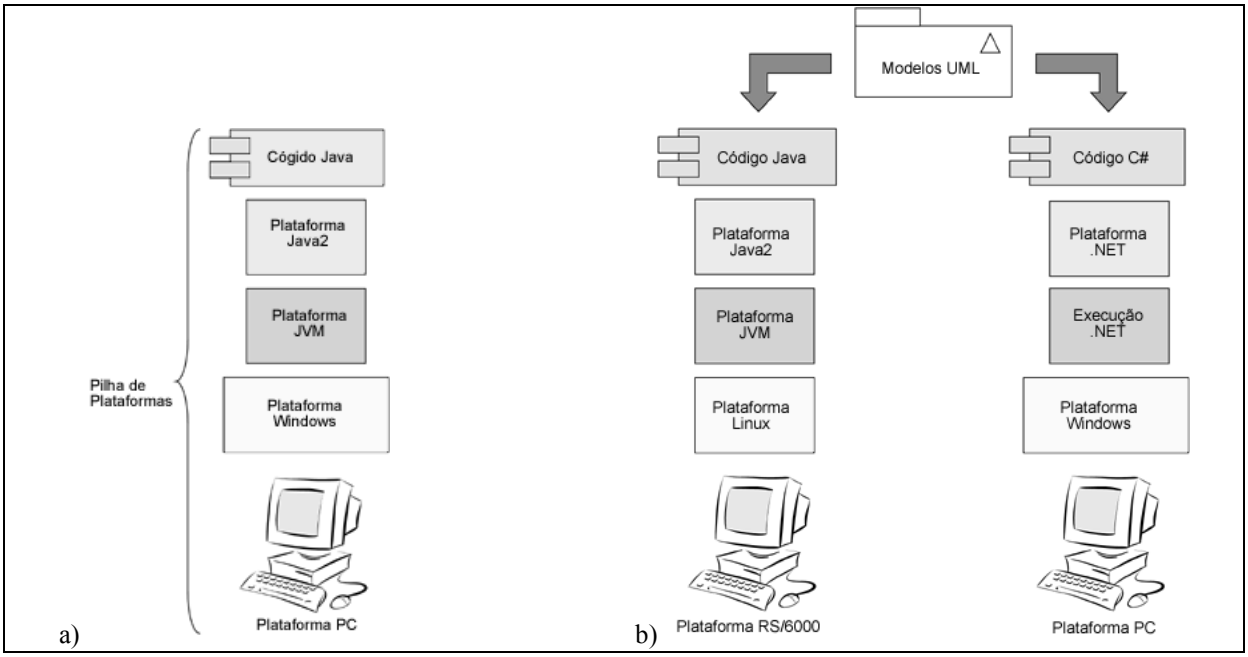

FIGURA 3 - Pilha de plataformas
a) pilha isolada b) pilhas diferentes integradas por modelos UML
Fonte: (Tradução do autor) Arlow e Neustadt (2003).

Um dos problemas com pilhas de plataformas, sob a perspectiva da engenharia de software, é a volatilidade da plataforma tecnológica. A cada dia surgem novas plataformas em substituição às já existentes. A arquitetura MDA propõe uma solução para este problema, adicionando, no topo da pilha, modelos UML de maior nível de abstração que o código-fonte do sistema de informação, que pode ser convertidos em código de outra pilha de plataforma, conforme ilustrado na FIGURA 3b.

Para Frankel (2003), a questão-chave da arquitetura MDA é a transformação de modelos (FIGURA 4). O modelo inicial é um modelo independente de plataforma – *Platform Independent Model* (PIM) – que possui as seguintes características:

- Representa o domínio de conhecimento sem a distorção de aspectos tecnológicos;
- É completamente independente da pilha de plataforma;
- É um modelo detalhado e, geralmente, representado em UML;
- É a base para o modelo específico de plataforma – *Platform Specific Model* (PSM).

O modelo PIM é transformado em modelo específico de plataforma – *Platform Specific Model* (PSM), por intermédio de mapeamentos. Um modelo PSM:

- Contém informações do domínio e da plataforma;
- É criado a partir do mapeamento de um PIM para uma plataforma específica;
- É um modelo detalhado e sempre representado em UML;
- É a base para o código-fonte.

O PSM é, então, convertido em código-fonte e outros artefatos como documentos, roteiros, etc. O código-fonte é convertido em código compilado e distribuído às estações de trabalho (FIGURA 4).

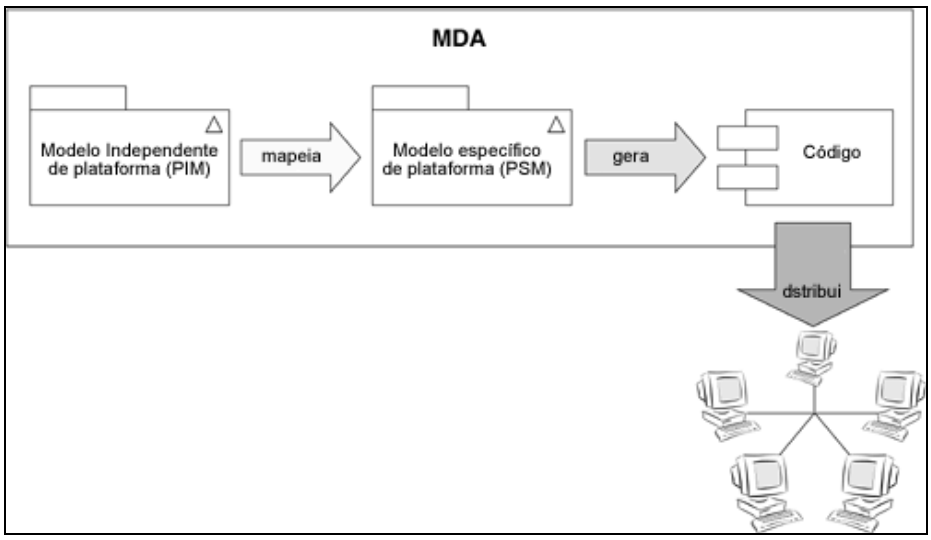

FIGURA 4 - Transformação de modelos

Fonte: (Tradução do autor) Arlow e Neustadt (2003).

O valor investido em código-fonte tende a depreciar-se rapidamente, juntamente com a linguagem de programação e plataforma ao qual ele pertence. A vantagem desta abordagem é que o domínio de conhecimento fica independente do código-fonte. Na arquitetura MDA, o valor é investido em modelos abstratos. Assim, o valor do modelo aumenta de forma diretamente proporcional ao nível de abstração, conforme a FIGURA 5. Ainda nesta abordagem, o especialista de domínio trabalha em nível de abstração diferente do analista ou programador.

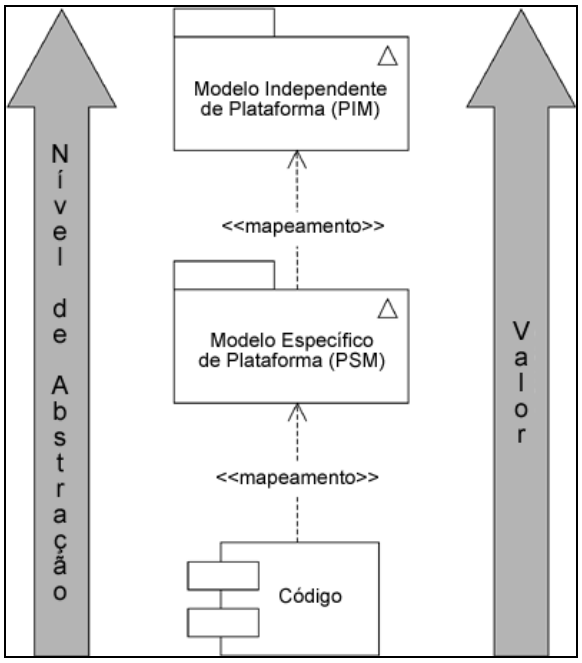

FIGURA 5 - Valor de acordo com o nível de abstração.
Fonte: (Tradução do autor) Arlow e Neustadt (2003).

A arquitetura MDA pode trazer contribuições importantes, a saber: permite a separação do conhecimento em níveis de abstração, isolando diferentes aspectos deste; habilita o especialista de domínio a lidar com ferramentas e conceitos mais próximos de sua realidade; permite ao analista ou programador se libertar da responsabilidade do conhecimento de domínio; e abre espaço para o surgimento de ferramentas de geração automática de sistemas, uma vez que, para tanto, basta concentrar esforços no mapeamento

entre os diversos níveis de abstração.

2.4 Processos de software

Detalha-se, aqui, o conceito central dos processos disponíveis atualmente, e o Praxis, processo de desenvolvimento de software utilizado no estudo de caso. O objetivo desta seção é apresentar os conceitos que cercam o domínio do estudo de caso, a fim de facilitar o entendimento do leitor sobre este. Não se pretende, aqui, realizar uma revisão da literatura sobre processos de software.

Paula-Filho (2003) conceitua processo como um conjunto de passos parcialmente ordenados, constituído por atividades, métodos, práticas e transformações, usado para atingir uma meta. Esta meta, geralmente, está associada a um ou mais resultados concretos finais, que são os produtos da execução do processo. Em engenharia de software, processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contratação de software.

A saída do processo de software é a informação, que Pressman (2002) divide em três categorias: programas de computador, tanto na forma de código-fonte quanto executável; documentos que descrevem programas de computador, voltados tanto para profissionais técnicos quanto para usuários; e dados, contidos num programa ou externos a ele. Uma grande quantidade de documentos é produzida, em diversos formatos, como documentos técnicos, memorandos, minutas de reuniões, esboços de planos, propostas, etc. Estes documentos não são necessários para futuras manutenções do sistema. No entanto, compõem o histórico do projeto, ajudando na criação de sua memória.

Um exemplo de processo de software de grande sucesso na indústria é o Processo

Unificado apresentado na próxima seção.

2.4.1 O Processo Unificado

Booch, Jacobson e Rumbaugh, além de proporem a UML como uma notação de modelagem orientada em objetos, propuseram o Processo Unificado (*Unified Process*), que utiliza a UML como notação de uma série de modelos que compõem os principais resultados das atividades do processo.

Neste processo, o ciclo de vida de um produto de software tem um modelo em espiral, em que cada projeto constitui um ciclo, capaz de entregar uma liberação do produto. Cada ciclo é dividido nas fases mostradas no QUADRO 1. Uma característica importante do Processo Unificado é que as atividades técnicas são divididas em subprocessos chamados de fluxos de trabalho (*workflows*), mostrados no QUADRO 2. Cada fluxo de trabalho tem um tema técnico específico, enquanto as fases constituem divisões gerenciais, caracterizadas por atingirem metas bem definidas.

QUADRO 1

Fases do Processo Unificado

Fase	Descrição
Concepção	Fase na qual se justifica a execução de um projeto de desenvolvimento de software, do ponto de vista do negócio do cliente.
Elaboração	Fase na qual o produto é detalhado o suficiente para permitir um planejamento acurado da fase de construção.
Construção	Fase na qual é produzida uma versão completamente operacional do produto.
Transição	Fase na qual o produto é colocado à disposição de uma comunidade de usuário

Fonte: Paula-Filho, 2003.

QUADRO 2

Fluxos do Processo Unificado

Fluxo	Descrição
Requisitos	Fluxo que visa a obter um conjunto de requisitos de um produto, acordado entre cliente e fornecedor.
Análise	Fluxo cujo objetivo é detalhar, estruturar e validar os requisitos, de forma que esses possam ser usados como base para o planejamento detalhado.
Desenho	Fluxo cujo objetivo é formular um modelo estrutural do produto que sirva de base para a implementação.
Implementação	Fluxo cujo objetivo é realizar o desenho em termos de componentes de código.
Testes	Fluxo cujo objetivo é verificar os resultados da implementação.

Fonte: Paula-Filho, 2003.

O processo unificado adota uma abordagem extensível e serve de base para a criação de outros processos de software. O exemplo de um processo de software baseado no processo unificado é o Praxis, que é descrito na próxima seção.

2.4.2 O Processo Praxis

O Praxis é um processo de desenvolvimento de software desenhado para dar suporte a projetos didáticos, em disciplinas de engenharia de software de cursos de informática e em programas de capacitação profissional em processos de software, concebido por Paula-Filho (2003). No entanto, devido ao seu sucesso, é utilizado também em organizações situadas principalmente no Estado de Minas Gerais.

O Praxis propõe um ciclo de vida composto por fases que produzem um conjunto precisamente definido de artefatos (documentos, modelos e relatórios). Para construir cada um desses artefatos, o usuário do processo precisa executar um conjunto de práticas recomendáveis. Na construção desses artefatos, o usuário do processo é guiado por padrões e auxiliado pelos modelos de documentos e exemplos constantes do material de apoio.

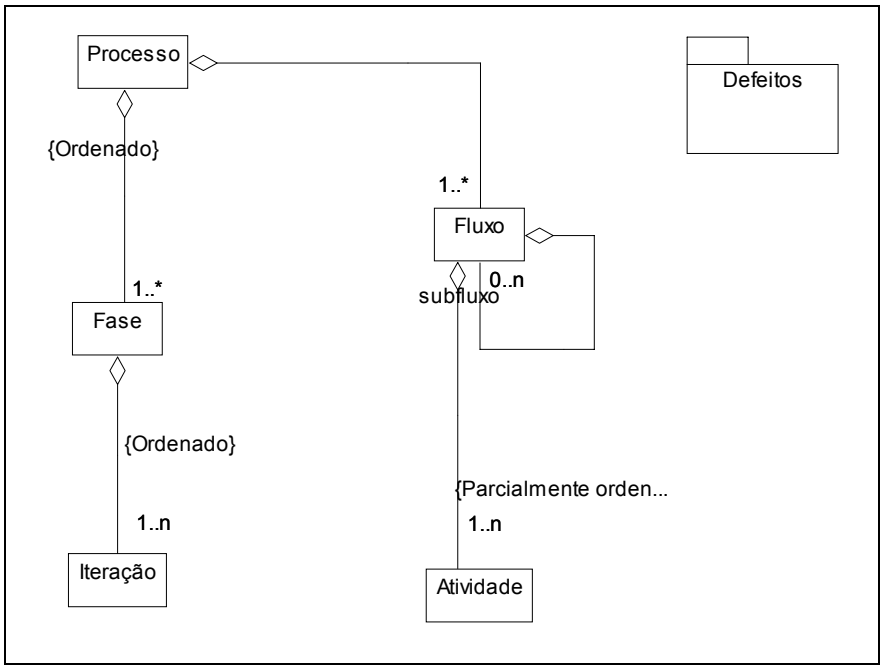

FIGURA 6 - Elementos do Praxis.
Fonte: Paula-Filho (2003).

Assim como o Processo Unificado, o Praxis abrange fases e fluxos. Uma fase é composta por uma ou mais iterações. Um fluxo é dividido em uma ou mais atividades. Iterações e atividades são exemplos de passos (FIGURA 6). Uma atividade executa um script, que possui pré-requisitos e critérios de aprovação, além de consumir e produzir artefatos (FIGURA 7).

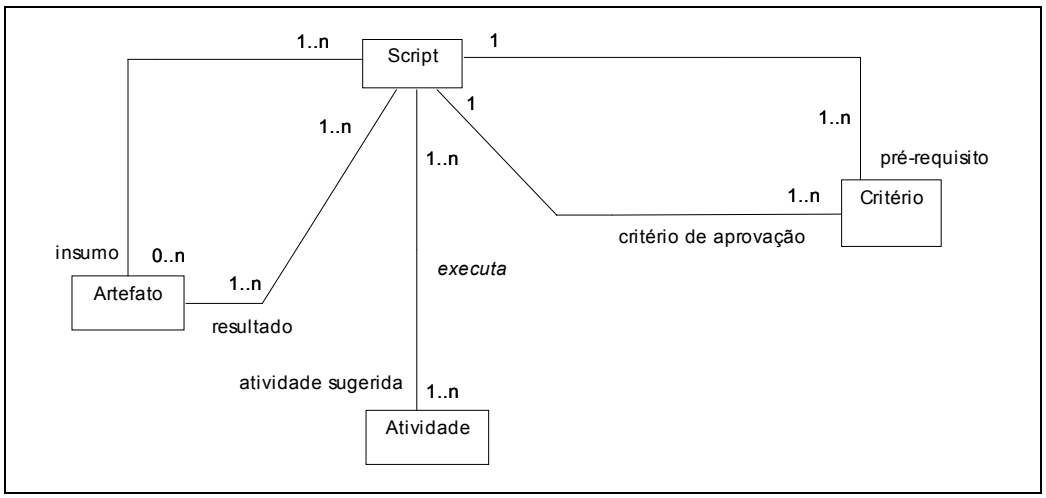

FIGURA 7 - Elementos dos scripts do Praxis.
Fonte: Paula-Filho, 2003.

Os principais elementos constituintes do Praxis são:

- Passo: Divisão formal de um processo, com pré-requisitos, entradas, critérios de aprovação e resultados definidos;
- Fase: Divisão maior de um processo, para fins gerenciais, que corresponde aos pontos principais de aceitação por parte do cliente;
- Interação: Passo constituinte de uma fase, no qual se atinge um conjunto bem definido de metas parciais de um projeto;
- Script: Conjunto de instruções que define como uma iteração deve ser executada;
- Fluxo: Subprocesso caracterizado por um tema técnico ou gerencial;
- Subfluxo: Conjunto de atividades mais estreitamente correlatas, que faz parte de um fluxo maior;
- Atividade: Passo constituinte de um fluxo;
- Técnica: Método ou prática aplicável à execução de um conjunto de atividades.

No capitulo do estudo de caso, são apresentados maiores detalhes do processo Praxis, por meio de outros diagramas de classes utilizados.

3 TRABALHOS RELACIONADOS

Este Capítulo apresenta, de forma não exaustiva, alguns trabalhos relacionados a esta dissertação. Apresentam-se propostas de geração de sistemas de informação baseadas em MDA, propostas de geração de bibliotecas digitais e propostas de geração de portais.

3.1 Propostas de geração de sistemas de informação baseadas em MDA

Utilizando uma implementação de MDA, Santos e Barros (2004) mapeiam um MOF (*Meta Object Facility*) para a linguagem Java. Um MOF (HEATON, 2002a) define uma linguagem abstrata e uma estrutura para especificação, construção e gerenciamento de metamodelos independentes de tecnologia de implementação (SANTOS, BARROS, 2004). Entretanto, esta implementação não conta com as funcionalidades pré-construídas de um ambiente de Gestão de conteúdo e demanda grande quantidade de esforço na implementação de serviços como personalização, segurança, etc.

Outro projeto que se baseia na arquitetura MDA é o projeto "XIS" (SILVA, 2003a). "XIS" (Desenvolvimento de Sistemas de Informação baseado em Modelos e Arquiteturas de Software) é um projeto de investigação e desenvolvimento realizado no contexto do Grupo de Sistemas de Informação do LAVC/INESC-ID (INESC-ID, 2005). O principal objetivo desse projeto é o estudo, desenvolvimento e avaliação de mecanismos de produção de sistemas de informação de forma mais eficiente. O projeto XIS é influenciado pelo modelo de referência MDA e baseia-se fortemente em um conjunto de práticas emergentes, seguindo uma abordagem (1) baseada em modelos, (2) centrada em arquiteturas de software, e (3) baseada em técnicas de geração automática. Contudo, este projeto também

não usufrui as características já presentes nos ambientes de Gestão de conteúdo, e nem da facilidade de incorporação de novas funcionalidades.

Yoder, Balaguer e Johnson (2001) descrevem o Modelo de Objeto Adaptável (*Adaptive Object-Model*) tratado também como arquitetura reflexiva ou meta-arquitetura. Um Modelo de Objeto Adaptável (MOA) é um sistema que representa classes, atributos e relacionamentos como *metadados*. Ele é um modelo baseado em instâncias, ao invés de classes. Usuários mudam o *metadado* (modelo de objeto) para refletir alterações no domínio. Estas alterações modificam o comportamento do sistema. MOA é uma arquitetura que pode, dinamicamente, adaptar-se, em tempo de execução, aos novos requisitos do usuário. Outras nomenclaturas para esta arquitetura são "modelo de objeto ativo" (FOOTE, YODER, 1998) e "modelo de objeto dinâmico" (RIEHLE, TILMAN, JOHNSON, 2000). Esta abordagem concentra-se no aspecto evolutivo do domínio de conhecimento. Em contrapartida, sistemas de informação que utilizam o MOA são mais difíceis de construir e de serem compreendidas pelos atores envolvidos, além de não especificar como se dá a interface com o usuário.

Uma outra implementação em camadas é realizada pelo projeto Hércules (PAIS, OLIVEIRA, LEITE, 2001). O projeto Hércules consiste em um arcabouço para desenvolvimento de sistemas de informação, visando possibilitar a geração automática de código a partir de diagramas UML, e estabelece um conjunto de descrições de alto nível de abstração, que os analistas ou programadores utilizam para especificar a apresentação, o desenvolvimento operacional e a persistência do sistema a ser construído. Este modelo é dividido em três camadas: domínio, controle e apresentação. Cada uma destas camadas possui diagramas UML associados. Embora essa abordagem envolva a geração automática de sistemas, ela não se preocupa em tratar separadamente o conhecimento de domínio do conhecimento tecnológico.

3.2 Propostas de geração de Bibliotecas Digitais

Outra abordagem para o desenvolvimento de sistemas de informação, especificamente bibliotecas digitais, é a abordagem 5S (GONÇALVES *et al*, 2004), que, a partir de 5 dimensões (*streams, structures, spaces, scenarios, societies*) e a partir de uma linguagem para especificar estas dimensões (GONÇALVES, FOX, 2002), descreve o conhecimento necessário para implementação de uma biblioteca digital. Uma das implementações desta abordagem encontra-se em Pozo (2004).

3.3 Propostas de geração de Portais

A geração de sítios Web, assim como portais, utilizando métodos formais é tratada em trabalhos como Thalheim *et al.* (2002) e Cavalcanti e Robertson (2003). Leung e Robertson (2003) também abordam a geração automática de sítios Web. A publicação disponível apresenta casos de sucesso concebidos a partir dos geradores criados no laboratório e aponta futuros esforços de pesquisa nesta direção, sem, contudo, descrever detalhadamente a arquitetura utilizada.

Outras pesquisas realizadas no Instituto de Informática Aplicada e Métodos Formais da Universidade de Karlsruhe também tratam a geração automática de portais semânticos, baseados na arquitetura SEAL (STOJANOVIC, 2001, MAEDCHE *et al.*, 2001), e no portal KAON (BOZSAK *et al*., 2002).

4 METODOLOGIA

Este capítulo descreve as etapas, as variáveis e os instrumentos utilizados na pesquisa. A pesquisa se caracteriza como estudo de caso realizado em um ambiente simulado, em que as variáveis serão manipuladas com controle metodológico.

4.1 Fases da pesquisa

Para alcançar seus objetivos, esta pesquisa foi dividida em duas fases:

Fase 1: Levantamento do arcabouço. Esta fase levanta o arcabouço utilizado no estudo de caso. Os passos executados foram:

- Identificação e análise, a partir de critérios claramente estabelecidos, das ferramentas disponíveis de código aberto passíveis de utilização no estudo de caso;
- Preparação do ambiente. Consistiu em instalar as ferramentas e realizar o teste de funcionamento destas.

Fase 2: Realização do estudo de caso. Esta fase estabelece os passos necessários para implementação. Para realizar o estudo de caso, foram executados os passos listados abaixo:

- Descrição das etapas para geração do SGC. Descreveram-se, aqui, as atividades seqüenciais realizadas para teste da hipótese, formando um processo de geração de SGCs;
- Execução do teste: Consistiu na realização do estudo de caso.

Antes de descrever o estudo de caso, contudo, faz-se necessário apresentar as variáveis, a hipótese e os instrumentos utilizados. Este é o assunto das próximas seções.

4.2 Instrumentos

Foram coletados dados de três entidades: domínio de conhecimento modelado, SGC do domínio de conhecimento e processo adotado. A partir do domínio de conhecimento modelado, foram coletados dados das seguintes propriedades:

- Número de classes modeladas;
- Número total de propriedades das classes;
- Número total de relacionamentos entre as classes.

A mensuração do domínio de conhecimento foi feita pela contagem do número de classes, propriedades e relacionamentos do domínio de conhecimento modelado. Para representar o domínio, foi utilizada a linguagem UML.

O reflexo do domínio de conhecimento é observado no SGC por meio da contagem dos pontos de função do mesmo. Foi adotado o modelo de contagem proposto por Albrecht (1979), chamado de análise por ponto de função, explicado no APÊNDICE C. Esta é uma medida de tamanho de software utilizada para mensurar a funcionalidade entregue ao usuário, independentemente da forma de implementação, plataforma, linguagem de programação, etc. (GARMUS, HERRON, 2000). Estes pontos de função são transformados em horas de desenvolvimento para estimar o esforço. Compararam-se as horas estimadas pela técnica com as horas gastas pelo autor

A terceira entidade é o do processo ou processamento adotado. Define-se, aqui, o conceito de processamento como conjunto orgânico de peças, próximo ao conceito de

máquina. Foram coletados dados a partir das seguintes categorias:

- Metodologia: etapas para implementação do processo;
- Estrutura: Arcabouço utilizado para a implementação do modelo.

Os formulários utilizados para coleta estão disponíveis no APÊNDICE B.

4.3 Arcabouço utilizado

O arcabouço utilizado é baseado em softwares de código aberto. A partir de uma busca no sítio Web sourceforge[2], buscou-se uma plataforma de gestão de conteúdo que apoiasse as atividades necessárias à realização do estudo de caso. Foram consideradas ferramentas que poderiam ser utilizadas segundo a arquitetura MDA, possuíam a maior comunidade de desenvolvedores, e se encontravam no estado de desenvolvimento estável (*5 - stable*), segundo a classificação de projetos no sourceforge (1. Planejamento; 2. Pré-Alfa; 3. Alfa; 4. Beta; 5. Produção ou estável; 6. Maduro; 7. Inativo). A lista dos projetos encontrados está disponível no APÊNDICE A, QUADRO 7.

O arcabouço é formado pela ferramenta Plone (MCKAY, 2004) e outras que colaboram com esta, apresentadas no QUADRO 3, que traz o tipo de licença, uma descrição da ferramenta e o número de pessoas envolvidas no seu desenvolvimento. Suas relações são ilustradas na FIGURA 8.

[2] O repositório de softwares de código aberto com maior número de projetos hospedados no mundo.

QUADRO 3

Arcabouço utilizado

Nome da ferramenta:	Tipo de licença	Versão	Sítio na Web	Descrição da função	No. de pessoas
Poseidon UML CE	Livre para uso não comercial.	3.0.1	www.gentleware.com	Editor UML. Utilizado para modelagem OO na notação UML.	21
Plone	GPL	2.0.4	www.plone.org	Sistema de Gestão de conteúdo.	88
Archetypes	GPL	1.3.1.	sourceforge.net/ projects/archetypes	Ferramenta de geração automática de tipos de conteúdo plone, orientada a esquemas.	65
ArchGenXML	GPL	1.1	sourceforge.net/ projects/archetypes	Utilitário de linha de comando que gera aplicações Plone baseado no framework Archetypes, a partir de um modelo UML representado em XMI ou XMLSchema.	65
strip-o-gram	GPL	1.4	sourceforge.net/ projects/squishdot	Converte a documentação HTML disponível no arquivo xmi para texto puro.	6
oi18ndude	GPL	0.2.2	sourceforge.net/ projects/plone-i18n	Permite a geração de rótulos traduzíveis.	81
Zope	GPL	2.7	www.zope.org	Servidor de aplicações, banco de dados e Web.	-

Na FIGURA 8, o especialista de domínio é responsável pela geração do Modelo Independente de Plataforma (PIM), que é mapeado em Modelo Específico de Plataforma (PSM). Este serve de entrada para o gerador de código-fonte (ArchGenXML), que tem como saída o Código Fonte (CF). A plataforma de gestão de conteúdo (Plone) recebe código-fonte e usa os componentes pré-construídos para gerar o Sistema de Gestão de Conteúdo específico para um domínio do conhecimento.

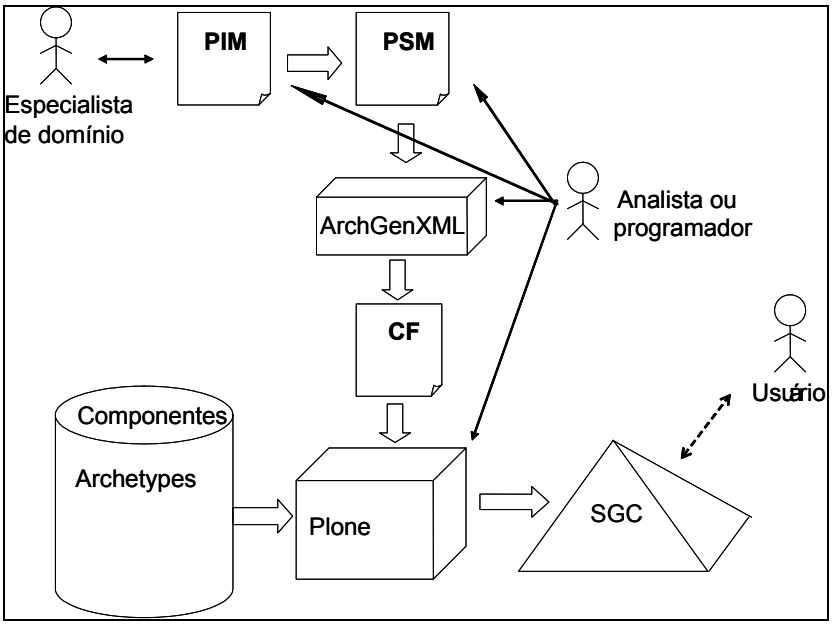

FIGURA 8 - Arcabouço utilizado

Outra forma de sistematizar as transformações realizadas pelas ferramentas do arcabouço é por intermédio da equação abaixo:

```
SGC = Plone (T3 (T2 (T1 (PIM, Perfil_UML), ArchGenXML), Manutencao))
```

Em que:

- `T1` = Transformação do `PIM` e `Perfil_UML` em `PSM`;
- `T2` = Transformação do `PSM`, a partir da ferramenta `ArchGenXML`, em `CF`;
- `T3` = Manutenção Evolutiva do Código-fonte do SGC;
- `PIM`: Modelo independente de plataforma;
- `Perfil_UML`: conjunto de mecanismos de extensão;
- `PSM`: Modelo específico de plataforma;
- `CF`: Código-fonte gerado a partir do PSM;
- `Plone`: Plataforma de gestão de conteúdo;
- `SGC`: Sistema de informação na Web. Aqui, um sistema de apoio no processo de desenvolvimento de software Praxis.

5 ESTUDO DE CASO

Este capítulo descreve a implementação de um processo para a geração de SGCs. A saída do processo é um SGC que pode ser utilizado no apoio ao desenvolvimento de software.

Trabalhos como Carvalho (2001), Liebowitz (2002) e Falbo *et al.* (2004) se dedicam, essencialmente, a explorar a utilização de sistemas de gestão de informação no apoio a processos de desenvolvimento de software. O estudo de caso relatado aqui, como dito anteriormente, trata da geração de SGC para o apoio ao processo de desenvolvimento de software Praxis (PAULA FILHO, 2003).

5.1 Escopo e justificativa

O domínio de conhecimento escolhido para o estudo de caso de geração foi o processo de desenvolvimento de software que, de acordo com o aumento do grau de maturidade, sofre alterações freqüentes.

A melhoria dos processos de software no Brasil, nos últimos anos, é constatada por Veloso *et al.* (2003). Segundo o Ministério da Ciência e Tecnologia[3] (MCT), em 2003, 214 empresas nacionais adotaram o padrão de qualidade *International Standard Organization* (ISO) 9000 (ABNT, 1996) enquanto 30 empresas adotaram um modelo especificamente voltado para software chamado CMM.

[3] http://www.mct.gov.br/.

O *Capability Maturity Model* (CMM) é um modelo criado pelo *Software Engineering Institute*[4] (SEI) da *Carnegie Mellon University*[5] e é utilizado para a criação de processos de software nas organizações (PAULK *et al*., 1993). O modelo é composto de 5 níveis de maturidade. A empresa interessada em adotar o modelo, se submete à avaliação de uma entidade certificadora. O aumento do nível de maturidade implica em mudanças no processo de desenvolvimento de software.

Das 30 empresas, hoje, no Brasil, que possuem algum nível de certificação CMM, verifica-se que 24 delas estão no nível 2 e cinco no nível 3. Há uma empresa no nível 4 e nenhuma no nível 5. WEBER (2004) constata que as empresas nacionais buscam a melhoria de seu processo de software, alterando-o e adequando-o às melhores práticas internacionais.

Este cenário demonstra o dinamismo do domínio de conhecimento dos processos de desenvolvimento de software adotados nas organizações brasileiras; o que é uma motivação para a pesquisa. Com efeito, para controlar os indicadores necessários para os níveis 2 e 3 do CMM é indispensável o uso de sistemas de apoio. Dentre outros, elegeu-se o processo de desenvolvimento de software Praxis, explicado na Seção 2.4.2, devido à facilidade de sua modelagem e do seu conhecimento prévio por parte deste autor.

5.2 Contexto e estrutura do processo

O processo de geração utilizado envolve 2 atores, especialista de domínio e analista ou programador, que realizam três grandes funções: modelagem independente de plataforma, modelagem específica de plataforma e geração do SGC (FIGURA 9). Estas

[4] http://www.sei.cmu.edu/.
[5] http://www.cmu.edu/.

funções são decompostas em atividades, detalhadas adiante.

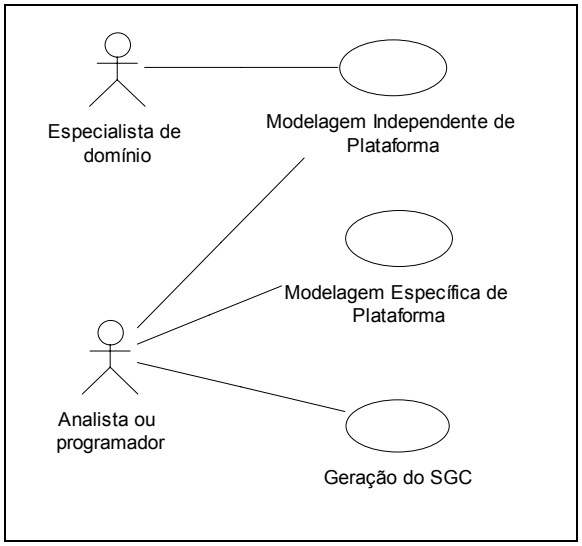

FIGURA 9 - Contexto do processo de geração utilizado.

O arcabouço utilizado é composto de ferramentas, que possuem como requisito outras ferramentas. O arcabouço é utilizado pelo processo, que é composto de atividades, que assumem compromissos. Estas atividades produzem resultados e consomem resultados produzidos nas atividades anteriores (FIGURA 10).

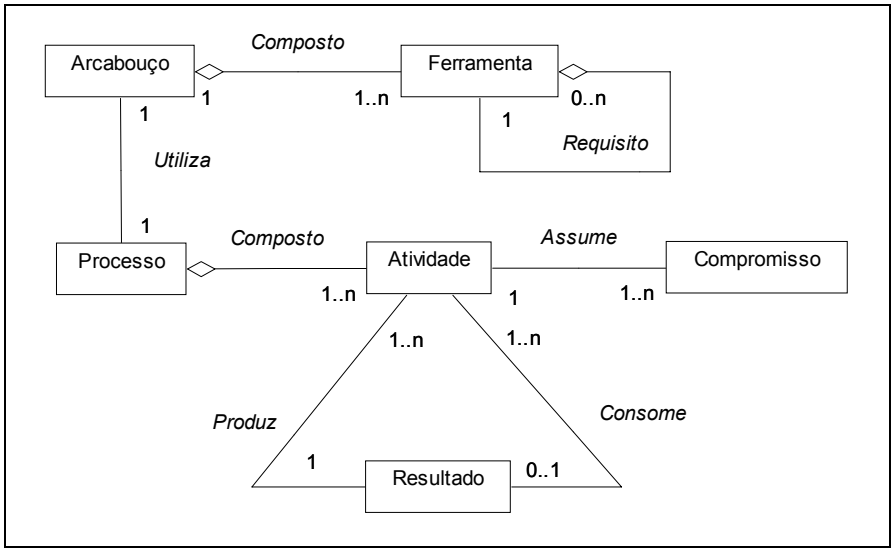

FIGURA 10 - Estrutura do processo de geração utilizado.

O QUADRO 4 apresenta as equivalências entre os modelos da arquitetura MDA e os modelos resultantes das atividades executadas no estudo de caso. Procurou-se utilizar a mesma nomenclatura da arquitetura MDA, para facilitar a equivalência.

QUADRO 4

Equivalências entre a arquitetura MDA e o processo de geração utilizado

Arquitetura MDA	Resultado do processo de geração utilizado	Atividade geradora
PIM	PIM	Modelagem independente de plataforma
PSM	PSM	Modelagem específica de plataforma
Código	Código-fonte	Geração do código-fonte
-	-	Manutenção Evolutiva do Código-fonte
-	-	Instalação

Cada resultado possui uma sigla e está associado a um responsável. Um responsável, em uma atividade, aplica uma determinada ferramenta do arcabouço (QUADRO 5).

QUADRO 5

Atividades e resultados

Resultado	Sigla	Responsável	Ferramenta aplicável	Atividade
Modelo Independente de Plataforma	PIM	Especialista de Domínio	Editor UML	Modelagem Independente de plataforma
Modelo Específico de Plataforma	PSM	Analista ou Programador	Editor UML	Modelagem Específica de Plataforma
Código-fonte	CF	Analista ou Programador	Gerador de código	Geração do código-fonte
Código-fonte Evoluído	CFE	Analista ou Programador	Editor de Código	Manutenção Evolutiva do Código-fonte
Sistema de Gestão de conteúdo	SGC	Analista ou Programador	Plataforma de gestão de conteúdo	Instalação

A FIGURA 11 apresenta as atividades do processo, relacionando-as e apresentando o resultado de cada uma. O processo começa com a Modelagem Independente

de Plataforma, realizada pelo especialista de domínio, gerando o Modelo Independente de Plataforma (PIM), utilizado pela Modelagem Específica de Plataforma, realizada pelo analista ou programador, que produz o Modelo Específico de Plataforma. Este é consumido pela Geração do código-fonte. Caso toda complexidade não tenha sido expressa, acontece a Manutenção Evolutiva do Código-fonte. Finalmente, ocorre a Instalação do SGC resultante.

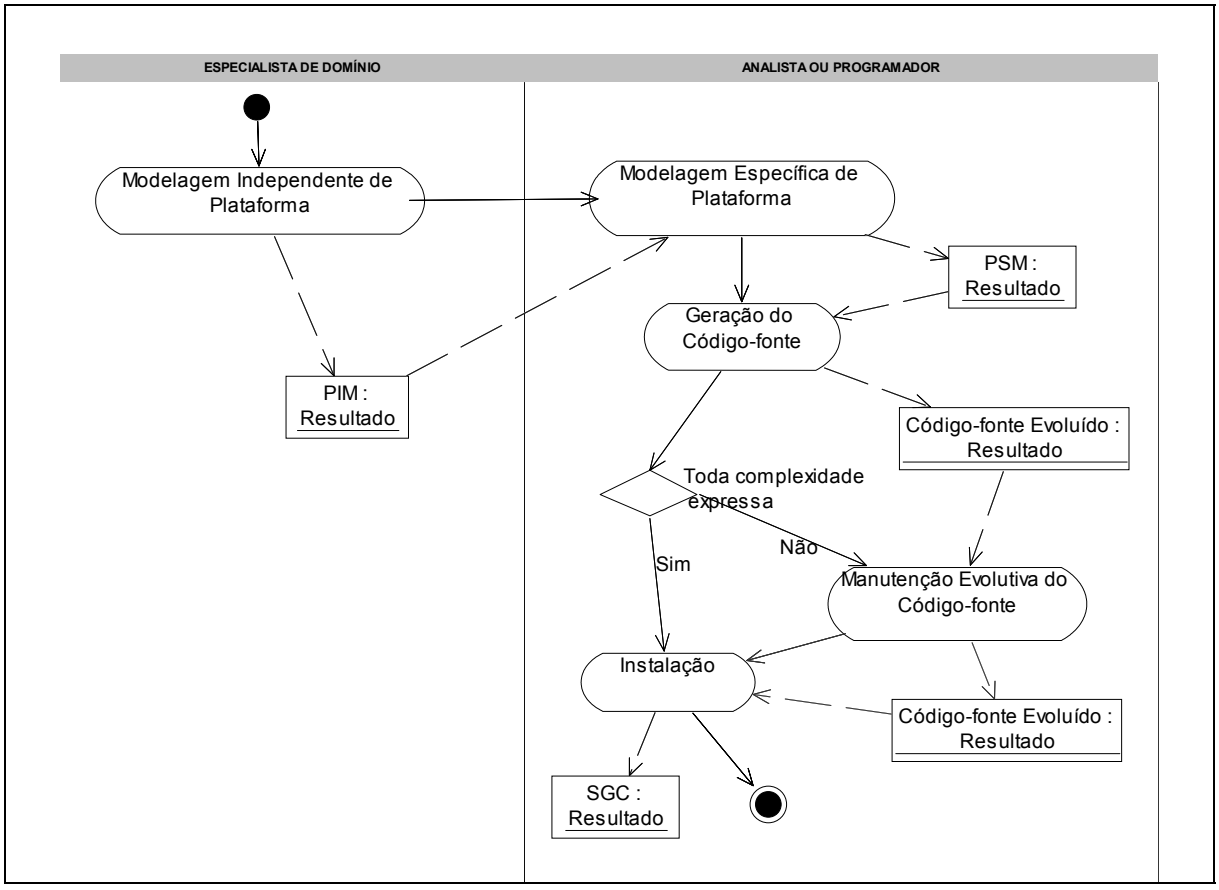

FIGURA 11 - Atividades do processo de geração utilizado.

Cada atividade assume compromissos, apresentados no QUADRO 6, em que se destaca o mapeamento entre o PIM e o PSM por meio de um roteiro de transformação. O mapeamento entre o PSM e o CF é automático, proporcionado pela geração automática. Já o mapeamento entre o CF e o CFE pode ser feito de forma não estruturada por meio de comentários no CFE.

QUADRO 6

Atividades e compromissos

Atividade	Compromisso
Modelagem Independente de plataforma	Modelo UML orientado a objetos armazenado no formato XMI.
Modelagem Específica de Plataforma	Modelo UML estendido por um Perfil UML, mapeado e armazenado em XMI.
Geração do código-fonte	Utiliza um gerador de código-fonte em linguagem Python e importa esquemas pré-construídos disponíveis no arcabouço.
Manutenção Evolutiva do Código-fonte	Mapeamento não estruturado
Instalação	-

O tempo proporcional despendido em cada atividade é apresentado na FIGURA 12. As atividades de modelagem consomem a maior parte do tempo.

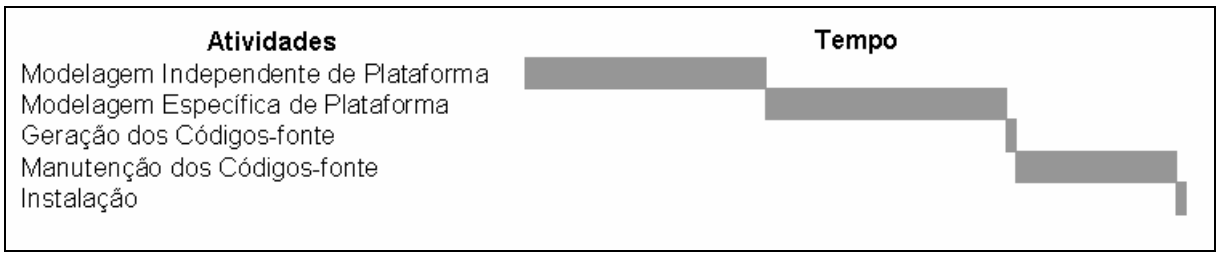

FIGURA 12 - Tempo gasto por atividade.

5.3 Atividades do processo

As cinco atividades do processo são: Modelagem Independente de Plataforma; Modelagem Específica de Plataforma; Geração do código-fonte; Manutenção Evolutiva do Código-fonte; e Instalação.

5.3.1 Modelagem independente de plataforma

Esta atividade concentra esforços na modelagem e representação de um determinado domínio de conhecimento. Ao modelar um domínio de conhecimento, Noy e McGuinness (2001) recomendam, inicialmente, a identificação de modelos já existentes. No caso do domínio de conhecimento escolhido, Paula Filho (2003) apresenta o domínio do Praxis representado em UML.

Apenas uma parte do domínio foi utilizada. Foram considerados apenas os diagramas de classe, que contemplam itens estruturais do domínio. Esta restrição se deu devido a uma das ferramentas do arcabouço utilizado, o gerador de código-fonte, que manipula somente diagramas de classe. Foram identificadas 45 classes e 49 relacionamentos em todos os diagramas de classe do Praxis. O diagrama de classes utilizado, apresentado a título de ilustração na FIGURA 13 contém 19 classes, 26 relacionamentos e 38 atributos. A parte utilizada concentrou as classes essenciais do modelo, que viabilizam o entendimento do mesmo.

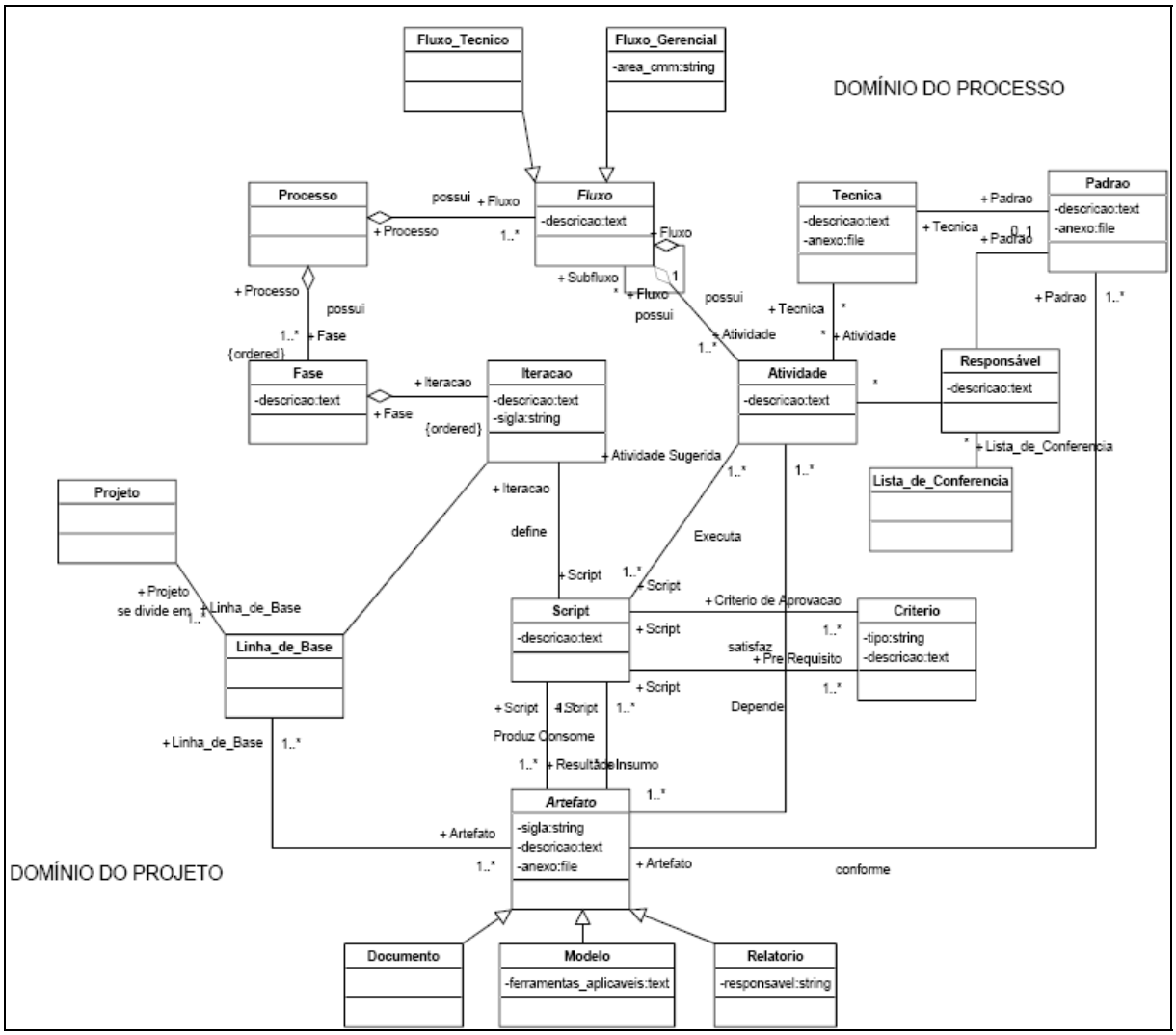

FIGURA 13 - Diagrama de classes.

Utilizou-se o editor UML Poseidon para a criação do diagrama de classes. Esse editor utiliza o formato XMI[6] (HEATON, 2002b) para armazenamento do modelo. O formato XMI é utilizado na geração automática também em Silva (2003a) e Pais, Oliveira e Leite (2001). A arquitetura MDA (FRANKEL, 2003) também recorre ao formato XMI. O resultado desta atividade é o Modelo Independente de Plataforma (PIM), representado em UML e armazenado no formato XMI.

[6] Formato aberto de armazenamento em texto estruturado baseado em XML.

5.3.2 Modelagem específica de plataforma

Os aspectos dependentes de plataforma são adicionados nesta atividade, que cria um Modelo Específico de Plataforma (PSM) com menor nível de abstração, mais próximo do código-fonte.

A partir de um perfil UML, é possível acrescentar aspectos inerentes a um determinado arcabouço. No estudo de caso, utilizou-se o perfil UML *ArchGenXML* (KLEIN, 2004), que permite a representação de ícones, rótulos de dados, máscara de entrada, etc. A adequação tecnológica ocorre nesta atividade, em que o analista ou programador incorpora aspectos inerentes ao arcabouço tecnológico utilizado.

Assim, é necessário manter um sincronismo entre o PIM e o PSM. É realizado um mapeamento entre os modelos, por meio de um roteiro de transformação. Este roteiro é escrito na linguagem XML *Extensible Stylesheet Language Transformation* (XSLT) (KAY, 2002), manualmente, pelo analista ou programador, e contém todas as transformações realizadas. Um mapeamento deste tipo também é utilizado por Silva (2003a), e outros mapeamentos semelhantes são realizados em ontologias, conforme Handschuh, Staab e Volz (2003).

O resultado desta atividade é o PSM, representado em UML, estendido segundo um perfil UML e armazenado em XMI.

5.3.3 Geração do código-fonte

Nesta atividade, o PSM é transformado em código-fonte (CF), por intermédio de um gerador de código-fonte em linguagem Python e importa esquemas pré-construídos

disponíveis no arcabouço. Este CF compõe o SGC, aqui chamado de PraxisCMF[7]. O PSM serve de entrada para o gerador de código-fonte, responsável por transformar o PSM em CF. Esta atividade resulta no CF do SGC.

5.3.4 Manutenção Evolutiva do Código-fonte

Se toda a complexidade do SGC puder ser expressa no PSM, esta atividade é opcional. Contudo, dado o atual desenvolvimento das ferramentas do arcabouço utilizado, esta ocorrência é rara. Assim, o analista ou programador deve fazer alterações no CF, a fim de atender os requisitos do sistema de informação que não puderam ser expressos no PIM ou PSM. O CF é, então, copiado e evoluído, tornando-se o resultado desta atividade, o Código-fonte Evoluído (CFE).

Esta alteração traz a mesma necessidade de sincronismo que a atividade de modelagem específica de plataforma. Entretanto, como o código-fonte é estruturado, mas não em uma linguagem de marcas, o mapeamento por meio de um roteiro de transformação fica mais difícil, e não foi considerado no estudo de caso. Neste caso, se ocorre uma alteração em uma das atividades anteriores, e existe a necessidade de manutenção evolutiva do código-fonte, esta alteração deve ser tratada manualmente. A FIGURA 14 apresenta um trecho de código-fonte python, em um editor de Código-fonte. Esta atividade resulta no CFE.

[7] O portal com o módulo PraxisCMF instalado pode ser acessado em http://netic.homeip.net/PraxisCMF.

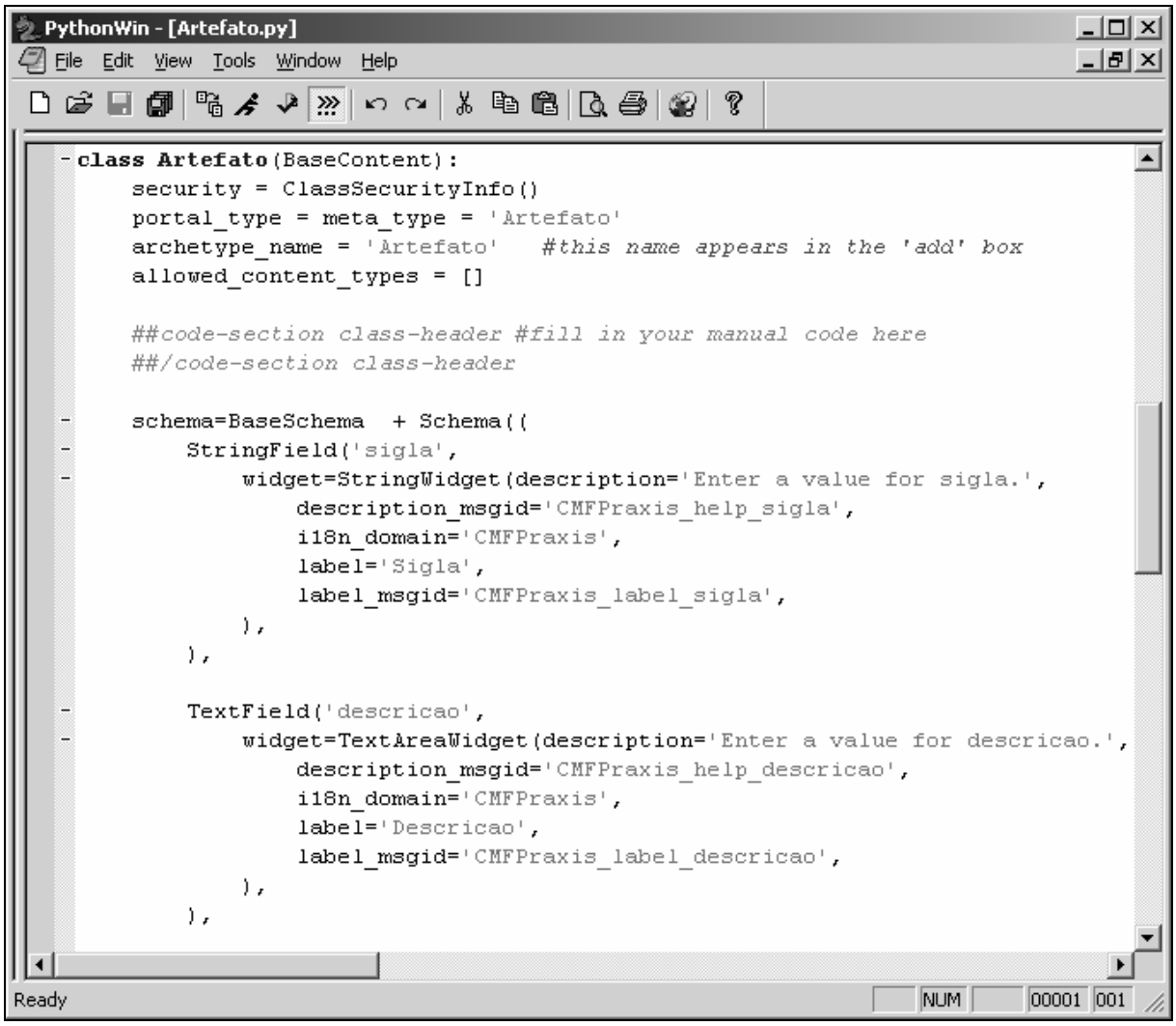

FIGURA 14 - Trecho de código-fonte da classe Artefato.

5.3.5 Instalação

Nesta atividade, o CFE é agrupado, formando o SGC. O roteiro de instalação envolve a interrupção do serviço, cópia do código-fonte e outros arquivos, inicialização do serviço e instalação do SGC na plataforma de gestão de conteúdo, conforme ilustrado na FIGURA 15 e na FIGURA 16.

```
echo Parando o serviço Zope...
net stop "Zope instance at c:\program files\plone 2\data"

echo Movendo o produto para o diretório do plone
move CMFPraxis C:\Program Files\Plone 2\Data\Products

echo Copiando os icones e imagens do produto
copy \fernando\dissertacao\imgs\*.gif CMFPraxis\skins\praxis_public

echo Iniciando o serviço Zope...
net start "Zope instance at c:\program files\plone 2\data"
```

FIGURA 15 - Roteiro de Instalação do PraxisCMF.

O resultado desta atividade é o SGC instalado, neste caso chamado de PraxisCMF.

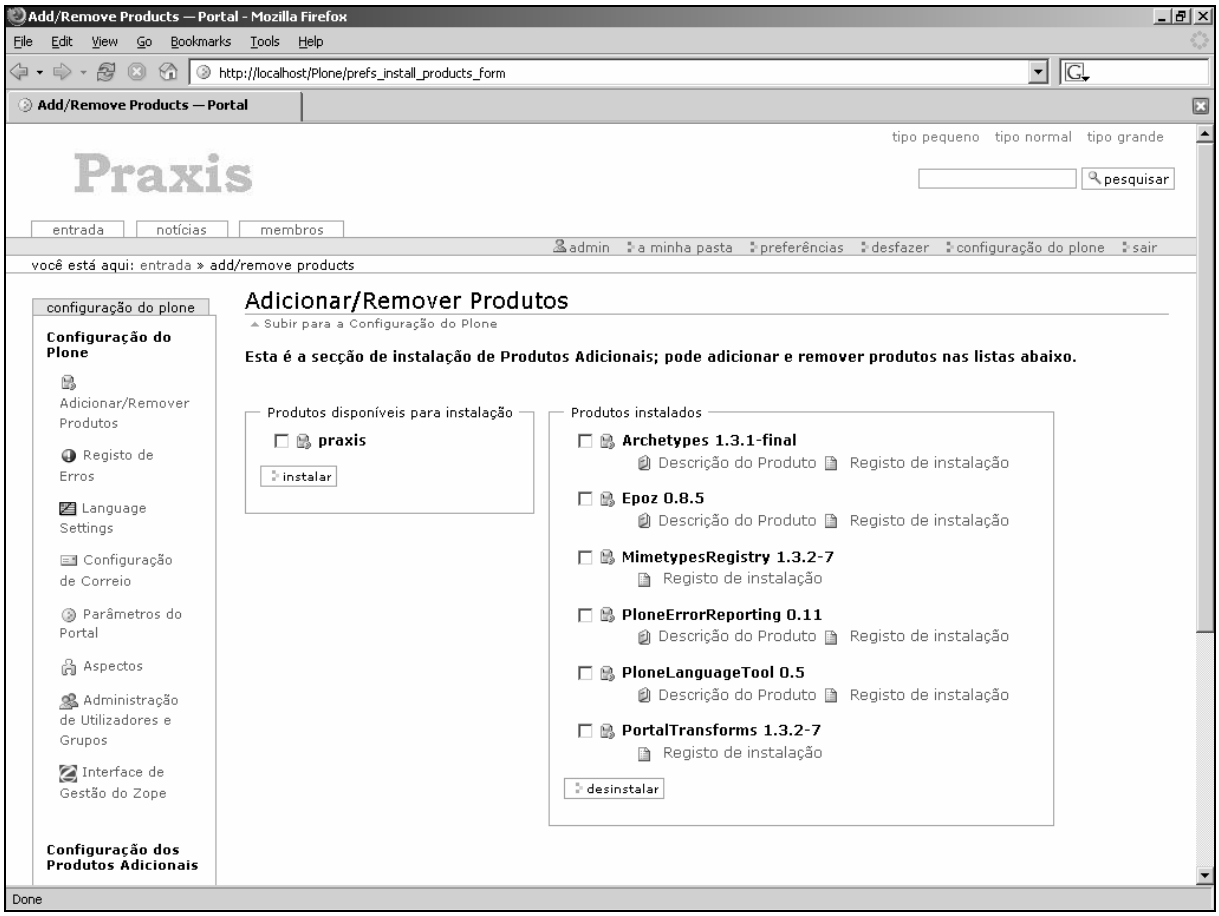

FIGURA 16 - Instalação do PraxisCMF.

6 RESULTADOS E DISCUSSÃO

Esta seção apresenta uma análise dos resultados atingidos a partir da realização do estudo de caso descrito no capítulo anterior. Analisam-se os procedimentos realizados e o domínio de conhecimento, além das limitações e os aspectos positivos e negativos da implementação.

6.1 Aspectos positivos

A medição do tamanho do SGC resulta em 247 Pontos de Função Não Ajustados (PFNA). Ajustando este valor baseado no ambiente tecnológico obtém-se 207,48 Pontos de Função Ajustados (PFA). Estes cálculos estão disponíveis no APÊNDICE B, TABELA 2 e TABELA 3. Aplicando um Coeficiente de Produtividade (CP) razoavelmente baixo em comparação com o padrão de mercado para linguagens do mesmo grupo (por exemplo, o coeficiente de produtividade de mercado da linguagem Java é 3 (três)) neste valor, ou seja, 1 (um), tem-se uma estimativa de 207,48 horas a serem gastas no desenvolvimento do SGC.

Neste estudo de caso foram gastas, aproximadamente, 66 horas para o desenvolvimento do SGC, conforme a TABELA 1. Entretanto, o estudo de caso realizado considerou um domínio de conhecimento com diagramas já disponíveis, no qual foram necessárias adaptações, a adição dos atributos e relações e a representação do modelo. Ainda assim, sendo constantes todas as outras variáveis, observa-se uma diferença significante entre os valores obtidos pelo autor e os valores estimados conforme a métrica de análise por pontos de função, salvo as possíveis falhas na metodologia utilizada. Esta diferença pode ser atribuída principalmente ao gerador de código-fonte e ao uso de componentes pré-construídos

no arcabouço. O gerador de código-fonte permite gerar interfaces CRUD[8] dos objetos, enquanto que os componentes trazem funcionalidades importantes para todo SGC, como busca, navegação, padrões de apresentação, além de outras.

TABELA 1

Tempo gasto pelo autor para implementação

Atividade	Tempo (em horas)
Modelagem independente de plataforma	24
Modelagem específica de plataforma	24
Geração do código-fonte	1
Manutenção Evolutiva do Código-fonte	16
Instalação	1
TOTAL	66

Os valores utilizados para o cálculo do esforço de desenvolvimento são estimados, devido à ausência de séries históricas de projetos desenvolvidos utilizando a mesma plataforma tecnológica, o que apresenta uma fragilidade na comparação entre o esforço estimado e o esforço realizado.

Além do uso do processo de geração para a geração do sistema definitivo, conforme apresentado acima, este pode ser utilizado para a prototipagem do software. A prototipagem consiste na elaboração de um protótipo, ou seja, versão parcial e preliminar do sistema destinada à validação com o usuário ou teste. Removendo o tempo gasto para a manutenção evolutiva do código-fonte, atividade dispensável no caso da prototipagem, o tempo gasto diminui para 50 horas. Após a atividade de modelagem, comum a qualquer processo de desenvolvimento de sistemas de informação, uma solução interessante seria utilizar esta implementação para gerar uma versão preliminar do sistema para apreciação do

[8] Acrônimo para "*Create, Retrieve, Update, Delete"*.

usuário. Em bibliotecas digitais, o uso de prototipagem pode ser visto em Pozo (2004).

A utilização de perfis UML como mecanismos de extensão apresenta uma forma interessante de adicionar aspectos da implementação ao modelo de forma isolada, sem comprometer a visão do usuário. Contribui, ainda, para a documentação do projeto, que nem sempre é concebida dentro do processo de software, ou é considerada uma atividade sacrificante por parte dos responsáveis.

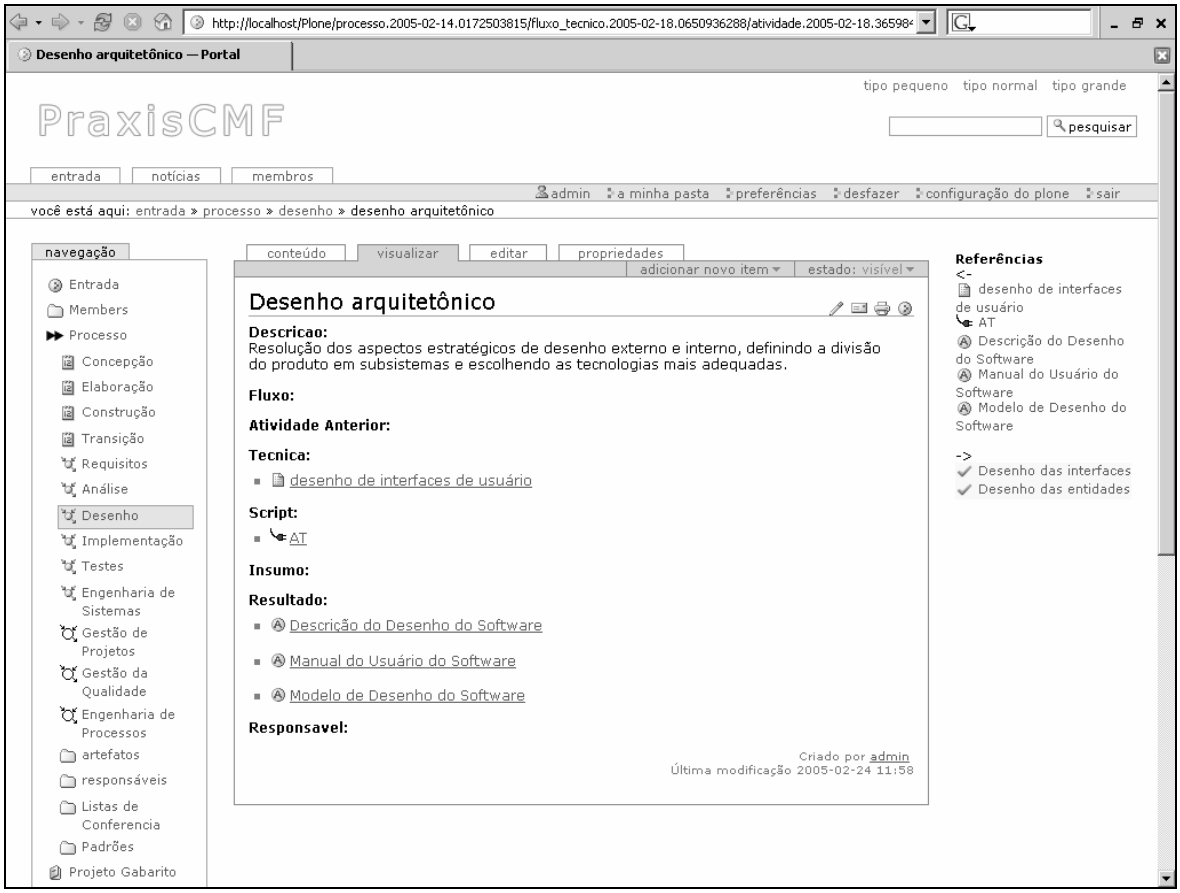

FIGURA 17 - Demonstração do SGC gerado.

Ainda que o processo de geração utilizado neste estudo de caso envolva aspectos técnicos em sua construção, este mantém certo nível de separação entre o domínio de conhecimento e o código do sistema. Desta forma, algumas alterações podem ser realizadas pelo especialista de domínio sem a necessidade do envolvimento do analista de sistemas ou programador.

Conforme Beale (2002), são necessários, no mínimo, três níveis para implementar um sistema de gestão da informação: o nível das classes, o nível dos dados pouco mutáveis e o nível dos dados mutáveis. O domínio de conhecimento escolhido para o estudo de caso permite evidenciar estes níveis. O nível das classes é simbolizado pelo diagrama de classes. O nível dos dados pouco mutáveis é representado pelo processo e suas fases, fluxos, atividades, etc. No nível dos dados mutáveis, tem-se o projeto, suas linhas de base e seus artefatos. Uma organização possui vários projetos de software, que são instâncias de um único processo de software.

Assim como observado por Leung e Robertson (2003), a automação parcial do processo de construção de SGCs pode ser alcançada por intermédio de ferramentas de conexão entre as diversas atividades, gerando código-fonte. Contudo, altos índices de automação obtêm êxito apenas em domínios de aplicação reduzidos.

Porém, ainda conforme Leung e Robertson (2003), os SGCs que não necessitam de muita manutenção no código-fonte apresentam uma chance de geração automática, gerando benefícios para sistemas de tamanho considerável.

6.2 Principais restrições

As ferramentas de código aberto disponíveis, utilizadas no arcabouço proposto, ainda não suportam todo o poder da modelagem orientada a objeto. A impossibilidade de usufruir de todos os recursos da linguagem UML, principalmente por parte da ferramenta ArchGenXML restringiu o potencial de representação utilizado na construção do modelo específico de domínio.

Um exemplo é a utilização de apenas um dentre os nove diagramas da linguagem UML pela plataforma adotada no estudo de caso, o diagrama de classes. Assim, é possível gerar o código somente de Inclusão, Recuperação, Alteração e Exclusão (*CRUD*) das classes. Diagramas com o poder de expressão comportamental como o diagrama de atividades, diagrama de estado e diagrama de seqüência ainda não são considerados pelas ferramentas disponíveis, o que restringe bastante o escopo de aplicação. Essa limitação se deve principalmente a pouca maturidade da plataforma de gestão de conteúdo utilizada no arcabouço, o Plone. Mas se deve também a dificuldade em se generalizar a construção de processos, que representam a parte comportamental de qualquer sistema. Por exemplo, a consideração de "fluxos de trabalho" modelados ainda não pode ser integrada ao processo de geração utilizado. Embora já haja algum nível de modelagem de fluxos no Plone, esta ainda é incipiente, exigindo ainda intervenção humana.

Ainda restringindo-se à utilização de diagramas de classe, outras limitações aparecem, como, por exemplo, a incapacidade inerente ao gerador utilizado de expressar o relacionamento *muitos-para-muitos*. Logo, quando ocorre este tipo de relacionamento, o código tem de ser alterado manualmente, pelo analista ou programador, na atividade de manutenção evolutiva do código-fonte. Estas questões demonstram a fragilidade do arcabouço utilizado, restringindo sua aplicação.

A usabilidade do SGC gerado apresenta algumas restrições como, por exemplo, no que diz respeito às referências cruzadas. Ao acessar um determinado objeto, tem-se uma visualização dos outros que se relacionam com ele. Entretanto, não se pode visualizar o tipo de relacionamento, conforme ilustrado na parte direta da FIGURA 17.

O mapeamento entre as camadas também é um ponto que apresenta restrições para a utilização do arcabouço proposto. Enquanto o mapeamento entre PIM e PSM deve ser escrito manualmente pelo analista ou programador, por meio de um roteiro de transformação,

o mapeamento entre o PSM e os CFs fica mais complicado, pois embora os CFs sejam estruturados, ainda não há uma forma automática de realizar este mapeamento.

Naturalmente, a implementação do processo realizada nesta dissertação por intermédio do arcabouço utilizado apresenta várias fragilidades, porém pode servir como instrumento inicial de análise de uma arquitetura dirigida por modelos.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O objetivo geral desta pesquisa foi analisar os aspectos teóricos, metodológicos e estruturais referentes à modelagem e geração de SGCs. O Capítulo 2 identificou os aspectos teóricos, enquanto o Capítulo 5 apresentou as atividades para a implementação de um processo que busca responder a pergunta de pesquisa. Restrições foram assumidas e limitações analisadas nos Capítulos 5 e 6, que apresentam o estudo de caso e os resultados.

Conclui-se que:

- É possível, com a utilização de um conjunto de ferramentas de código aberto atuais, compor um arcabouço que facilite a construção de SGCs e sua manutenção. O arcabouço permite maior agilidade de adequação aos efeitos das alterações conceituais que ocorrem no domínio de conhecimento modelado, com a conseqüente diminuição de custos associados;
- A modelagem em vários níveis é uma necessidade crucial para a evolução da área de sistemas de informação, principalmente quando considerados domínios em que o número de conceitos é muito grande e suas propriedades e relações muito dinâmicas, como no caso de aplicações no domínio da medicina ou mesmo da engenharia de software. Nesses domínios essa separação parece fundamental para a perenidade dos sistemas de informação;
- O arcabouço utilizado apresenta vantagens na prototipagem ou aplicação em domínios de conhecimento reduzidos;
- Quando um sistema utiliza o paradigma da orientação a objetos, a UML é uma boa candidata a linguagem de representação do conhecimento.

Esta pesquisa assumiu compromissos ao longo de sua execução, atribuindo restrições à implementação. Contudo, conforme Campos (2004), o compromisso ontológico feito por uma representação pode ser uma de suas mais importantes contribuições, pois somente assim pode ser lançado um olhar sobre a realidade.

Desde a ponta do usuário até a ponta do sistema de informação pode existir um número infinito de modelos que interagirem entre si. Entretanto, o poder de expressividade de cada camada de abstração deve ser identificado e conciliado, com o objetivo de aumentar o volume de conhecimento representado.

Propõe-se que este trabalho continue evoluindo nas seguintes direções:

- Continuação do desenvolvimento do arcabouço utilizado, já que as ferramentas que o constituem não param de evoluir em seus princípios teóricos;
- Aplicação do arcabouço utilizado e do processo de geração utilizado no desenvolvimento de sistemas de gestão de conteúdo em outros domínios, buscando a compreensão do problema a partir da sua consideração sob outras perspectivas;
- Investigação dos princípios teóricos que estão por traz do arcabouço utilizado, buscando atuar pró-ativamente nas comunidades de desenvolvimento dessas ferramentas;
- Avaliação por usuários do SGC gerado;
- A análise comparativa da abordagem 5S (GONÇALVES *et al*, 2004) com o processo de geração utilizado é também uma proposta para trabalho futuro já que se observa a presença de dimensões comuns em vários pontos do desenvolvimento.

O teste do SGC concebido não constituía objetivo deste trabalho, e pode também ser realizado em um trabalho futuro.

Finalmente, propõe-se que o código-fonte gerado durante esta pesquisa seja evoluído pela comunidade de programadores interessada, pois está publicado no repositório brasileiro de código livre[9] e está aberto para sugestões e alterações.

[9] O projeto de desenvolvimento do PraxisCMF encontra-se em http://codigolivre.org.br/projects/kmup/.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT. NBR ISO 9001 Sistemas da qualidade - Modelo para garantia da qualidade de projeto, desenvolvimento, produção, instalação e serviços associados. ABNT, Rio de Janeiro, 1996.

ALBRECHT, A. J. Measuring Application Development Productivity. In: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1, 1979, Monterey.

ARLOW, J.; NEUSTADT, I. **Enterprise Patterns and MDA**: Building Better Software with Archetype Patterns and UML. Boston: Addison Wesley, 2003. 528p.

BAX, M. P.; PARREIRAS, F. S. **Gestão de conteúdo com softwares livres**. In: KM BRASIL, 2003, São Paulo. *Anais (CD-ROM)...* São Paulo: [s.n.], 2003.

BEALE, T. **Archetypes**: Constraint-based domain models for future-proof information systems. In: OOPSLA, 17,2002, Seatlle. *Proceedings...* Workshop on behavioral semantics, Seatlle: ACM. 2002. Disponível em: <http://www.oceaninformatics.biz/publications/archetypes_new.pdf>. Acesso em: 03 jan. 2005.

BOOCH, G.; RUMBAUGH J.; JACOBSON I. **UML**: guia do usuário. Rio de Janeiro: Campus, 2000. 472 p.

BOZSAK, E. *et al*. **KAON**: Towards a Large Scale Semantic Web. In: EC-WEB, 3, 2002, Aix-en-Provence. *Proceedings*... Aix-en-Provence: Lecture Notes in Computer Science, Springer, 2002. p. 304-313.

BUCKLAND, M. Information as Thing. **Journal of the American Society of Information Science**, Silver Spring, v.5, n.42, p. 351-360, June 1991.

BURT, P.; KINNUCAN, M. Information models and modeling techniques for information systems. In: WILLIAMS, MARTHA E. **Annual Review of Information Science and Technology**. Kidlington: Elsevier Science Ltd, 1990. cap. II, p.175-208.

BUSCHMANN, F. *et al*. **Pattern-Oriented Software Architecture**: A system of patterns. 1. ed. Mississauga: John Wiley & Sons, 1996. 476 p.

CAMPOS, M. L. A. Modelização de domínios de conhecimento: uma investigação de princípios fundamentais. **Ciência da Informação**, Brasília, v.33, n.1, p. 22-32, jan./abril 2004.

CARVALHO, J. G. **Praxis Mentor:** Uma ferramenta de apoio à utilização de um processo de desenvolvimento de software. 2001. 127f. (Dissertação, Mestrado em Ciência da Computação) – Departamento de Ciência da Computação, UFMG, Belo Horizonte, 2001.

CARVALHO, R. B. **Aplicações de softwares de gestão do conhecimento tipologia e usos**. 144f. 2000. (Dissertação, Mestrado em Ciência da Informação) – Escola de Ciência da Informação, UFMG, Belo Horizonte, 2000.

CAVALCANTI, J. M. B.; ROBERTSON, D. Web site synthesis based on computational logic. **Knowledge and Information Systems**, Berlin, v.5, n. 3, p. 263–287, Sep. 2003.

CENADEM - Centro Nacional de Desenvolvimento do Gerenciamento da Informação. O GED. Disponível em: <http://www.cenadem.com.br/>. Acesso em: 13 mar. 2005.

COCCHIARELLA, N. B. **Formal Ontology**. In: BURKHARDT, H.; SMITH, B. (Eds.). Handbook of Metaphysics and Ontology. Munich: Philosophia Verlag, 1991.

CRANEFIELD, S. Networked Knowledge Representation and Exchange using UML and RDF. **Journal of Digital Information**, Leicestershire, v.1, Issue 8, fev. 2001. Disponível em: <http://jodi.ecs.soton.ac.uk/Articles/v01/i08/Cranefield/>. Acesso em: 18 mar. 2004.

CRANEFIELD, S.; PURVIS, M. **UML as an ontology modeling language**. In: WORKSHOP ON INTELLIGENT INFORMATION INTEGRATION, 16th, 1999, Stockholm. *Proceedings...* Stockholm: International Joint Conference on Artificial Intelligence (IJCAI-99), 1999. p.1-16.

DAHLBERG, I. Teoria do conceito. Ciência da Informação, Rio de Janeiro, v.7, n.2, p. 101-107, 1978.

_____. Conceptual definitions for INTERCONCEPT. **International Classification**, v.8, n.1, p. 16-22, 1981.

DAVENPORT, T. H; PRUSAK, L. **Conhecimento Organizacional:** como as organizações gerenciam o seu capital intelectual. Rio de Janeiro: Campus, 1998.

DESFRAY, P. **White Paper on the Profile mechanism**. Version 1.0. Needham: OMG Document ad/99-04-07, apr.1999. 13 p. Technical report. Disponível em: <http://www.omg.org/docs/ad/99-04-07.pdf>. Acesso em: 25 jan. 2005

DODEBEI, V. L. D. **Tesauro**: linguagem de representação da memória documentária. Rio de Janeiro: Interciência, 2002. 120 p.

FALBO, R. A. *et al.* **Ontologias e ambientes de desenvolvimento de software semânticos**. In: JORNADAS IBEROAMERICANAS DE INGENIERÍA DEL SOFTWARE E INGENIERÍA DEL CONOCIMIENTO, 4., Madrid. *Anais eletrônicos...* Madrid: Universidad Politécnica de Madrid, 2004. Disponível em: <http://lucio.ls.fi.upm.es/jiisic04/Papers/44.pdf>. Acesso em: 30 dez. 2004.

FELFERNIG, A.; FRIEDRICH, G. E.; JANNACH, D. UML: As domain specific language for the construction of knowledge-based configuration systems. **International Journal of Software Engineering and Knowledge Engineering**, [S.l.], v.10, n.4, p. 449-469, 2000.

FERNANDÉZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. METHONTOLOGY: From Ontological Art Toward Ontological Engineering. Spring Symposium Series on Ontological Engineering. AAAI97. Stanford. March 1997.

FOOTE, B, YODER, J W. **Metadata and Active Object-Models**. In: Collected papers from the PLoP '98 and EuroPLoP '98 Conference, 1998, Washington. *Proceedings...* Washington: Washington University. 1998.

FRANKEL, D. S. **Model Driven Architecture**: Applying MDA to Enterprise Computing. Indianapolis: Wiley Publishing, Inc., 2003. 352p.

GAMMA, E. *et al*. **Design Patterns**: Elements of Reusable Object-Oriented Software. Boston: Addison Wesley, 1995. 395p.

GARMUS, D., HERRON, D. **Function Point Analysis:** Measurement Practices for Successful Software Projects. New York: Addison-Wesley, 2000. 400p.

GENTLEWARE. Poseidon for UML. Disponível em: <http://www.gentleware.com>. Acesso em: 11 out. 2004.

GÓMEZ-PÉREZ, A. **Ontological Engineering**: A State Of The Art. Expert Update, Liverpool, p. 33-44, July 2003. Disponível em: <http://citeseer.ist.psu.edu/444416.html>. Acesso em: 11 out. 2004.

GONÇALVES, M. A. *et al.* Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries. **ACM Transactions on Information Systems**. New York: ACM Press, v.22, n.2, p.270-312, Abr. 2004.

GONÇALVES, M. A.; FOX, E. A. **5SL:** a language for declarative specification and generation of digital libraries. In: ACM/IEEE-CS JOINT CONFERENCE ON DIGITAL LIBRARIES, 2, 2002, Portland. *Proceedings...* Portland: ACM Press. 2002. p.263-272. Disponível em: <http://www.dlib.vt.edu/projects/5S-Model/p117-goncalves.pdf>. Acesso em: 20 out. 2004.

GRUBER, T. R. **Toward Principles for the Design of Ontologies Used for Knowledge Sharing**. In: FORMAL ONTOLOGY IN CONCEPTUAL ANALYSIS AND KNOWLEDGE REPRESENTATION, 1993. Padova. *Proceedings...* Padova: Kluwer Academic Publishers, in press. 1993. p.01-23.

GUARINO, N. Formal Ontology, Conceptual Analysis and Knowledge Representation. **International Journal of Human-Computer Studies**, Duluth, v.43, n.5/6, p.625-640, nov./dez. 1995. Disponível em: <http://dx.doi.org/10.1006/ijhc.1995.1066>. Acesso em: 11 out. 2004.

HANDSCHUH, S.; STAAB, S.; VOLZ, R. **On deep annotation**. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE (WWW), 12, 2003, Budapest. *Proceedings*... Budapest: ACM Press, 2003.

HEATON, L. **Meta Object Facility (MOF) Specification**. Versão 1.4. Needham: OMG, apr. 2002a. 358 p. Technical report. Disponível em <http://www.omg.org/docs/formal/02-04-03.pdf>. Acesso em 25 jan. 2005.

HEATON, L. **XML Metadata Interchange (XMI) Specification**. Versão 1.2. Needham: OMG, jan. 2002b. 268 p. Technical report. Disponível em <http://www.omg.org/docs/formal/02-01-01.pdf>. Acesso em 25 jan. 2005.

HOFMEISTER, C.; NORD, R.; SONI, D. **Applied Software Architecture**. Boston: Addison Wesley, 1999. 397p.

INESC-ID - Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa. Disponível em: <http://berlin.inesc-id.pt>. Acesso em: 25 mar. 2005.

INTERNATIONAL FUNCTION POINT USERS GROUP - IFPUG. **Function Point Counting Practices Manual**. Princeton. Versão 4.1.1. 2000. Disponível em: <http://www.ifpug.org>. Acesso em: 13 jun. 2004.

JACOBSON, I.; RUMBAUGH, J.; BOOCH, G. **Unified Software Development Process**. Reading - MA: Addison-Wesley, 1999.

KAY, M. **XSLT** referência do programador. 2. ed. Rio de Janeiro: Alta Books, 2002. 667p.

KLEIN, J. W. **ArchGenXML Manual** - generating Archetypes using UML. 2004. Disponível em: <http://plone.org/documentation/archetypes/archgenxml-manual>. Acesso em: 11 out. 2004.

LATTEIER, A.; PELLETIER, M. **The Zope Book**. Berkeley: Pearson Education, 2001. 384p.

LEUNG, S.; ROBERTSON, D. **Automated Website Synthesis**. In: LINUX 2003: CONFERENCE AND TUTORIALS, 2003, Edinburgh. *Proceedings*... Edinburgh: UKUUG, 2003. Disponível em: <http://www.ukuug.org/events/linux2003/papers/leung.pdf>. Acesso em: 20 out. 2004.

LIEBOWITZ, J. A look at NASA Goddard Space Flight Center's knowledge management initiatives. **IEEE Software**, v.19, n.3, p. 40-42, May/Jun 2002.

MAEDCHE, A. *et al.* **SEAL**: A Framework for Developing SEmantic Web PortALs. In: BRITISH NATIONAL CONFERENCE ON DATABASES, 18, 2001, London. *Proceedings*... London: Springer-Verlag, 2001, p.1-22. Disponível em: <Disponível em: <http://portal.acm.org/citation.cfm?id=646103.681199#>. Acesso em: 20 out. 2004.>. Acesso em: 20 out. 2004.

MCKAY, A. **The Definitive Guide to Plone**. Berkeley: Apress, 2004. 584p.

MOREIRA, A. **Tesauros e Ontologias**: estudo de definições presentes na literatura das áreas das Ciências da Computação e da Informação, utilizando-se o Método Analítico-Sintético. 2003. 150f. (Dissertação, Mestrado em Ciência da Informação) – Escola de Ciência da Informação, UFMG, Belo Horizonte, 2003.

MOREIRA, A.; ALVARENGA, L.; OLIVEIRA, A. O nível do conhecimento e os instrumentos de representação: tesauros e ontologias. DataGramaZero on-line, v.5, n.6, dez. 2004. Disponível em: <http://datagramazero.org.br/dez04/F_I_art.htm>. Acesso em: 30 dez. 2004.

NEELAMEGHAN, A. Application of Ranganathan's General Theory of Knowledge Classification in Designing Specialized Databases. **Libri**, Copenhague, v.42, n.3, p. 202-226, Jul/Sep 1992.

NOY, N. F.; MCGUINNESS, D. L. **Ontology Development 101**: A Guide to Creating Your First Ontology. Stanford: Stanford Knowledge Systems Laboratory Technical Report KSL-

01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001. 25 p. Report.

OTLET, P. **Traité de documentation**: le livre sur le livre, théorie et pratique. Bruxelles: Editions Mundaneum, 1934. Reprint, Liège: Centre de Lecture Publique de la Communauté française. 1989.

PAIS, A.P.V.; OLIVEIRA, C.E.T.; LEITE, P.H.P.M. Robustness Diagram: A Bridge Between Business Modeling And System Design . In: Proceedings of VII International Conference on Object-Oriented Information Systems - OOIS'01. Calgary, Canadá: Springer-Verlag. 2001, v. 1, p. 530-539.

PAULA FILHO, Wilson de Pádua. **Engenharia de software**: fundamentos, métodos e padrões. 2. ed. Rio de Janeiro: LTC, 2003. 602p.

PAULK, M. C. *et al.* **Capability Maturity Model for Software**, Version 1.1. Pittsburgh: Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, Feb. 1993. 82 p. Technical report. Disponível em <http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf>. Acesso em 25 jan. 2005.

PEREIRA, J.C.L.; BAX, M. P. Introdução à Gestão de Conteúdos. In: KM BRASIL, 2002, São Paulo. Anais (CD-ROM)... São Paulo: [s.n.], 2002.

POZO, D. P. V. Especificação e Prototipagem de uma Biblioteca Digital Utilizando a Linguagem 5SL. 2004. (Dissertação, Mestrado em Ciência da Computação) – Departamento de Ciência da Computação, UFMG, Belo Horizonte, 2004.

PRESSMAN, R. S. **Engenharia de Software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002. 843p.

RANGANATHAN, S. R. **Prolegomena to library classification**. Bombay: Asia Publishing House, 1967. 640p.

RIEHLE, D.; TILMAN, M.; JOHNSON, R. **Dynamic Object Model**. In: PLoP2000, 7, 2000, Monticello. *Proceedings...* Monticello: Technical Report #wucs- 00-29, Dept. of Computer Science, Washington University, 2000. p. 1-13. Disponível em: <http://jerry.cs.uiuc.edu/~plop/plop2k/ proceedings/Riehle/Riehle.pdf>. Acesso em: 30 jul. 2004.

RUMBAUGH, J. *et al*. **Object-Oriented Modeling and Design**. New Jersey: Prentice Hall, 1990. 500p.

SANTOS, H. L.; BARROS, R. S. M. **Utilizando o MOF na construção de metamodelos em um ambiente MDA**. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 18, 2004, Brasília. *Anais...* Brasília: Sociedade Brasileira de Computação (SBC), 2004. CD-ROM.

SAYÃO, L. F. Modelos teóricos em Ciência da Informação abstração e método científico. **Ciência da Informação**, Brasília, v.30, n.1, p. 82-91, jan./abr. 2001.

SILVA, A. R. **Abordagem XIS ao Desenvolvimento de Sistemas de Informação**. In: Conferência da Associação Portuguesa de Sistemas de Informação, 4, 2003, Porto. Anais... Porto: Universidade Portucalense. 2003a.

SILVA, S. **Archetypes**: An Introduction. In: ZopeMag.com. Kurfürstendamm: beehive KG. 2003b. Disponível em: <http://www.zopemag.com/Issue006/Section_Articles/article_IntroToArchteypes.html>. Acesso em: 05 jun. 2004.

SOURCEFORGE.NET. OSTG Open Source Technology Group. Disponível em <http://www.sourceforge.net>. Acesso em: 25 jan. 2005.

SOWA, J. F. **Knowledge representation**: logical, philosophical, and computational foundations. Pacific Grove: Brooks-Cole, 2000. 594p.

STAAB, S.; STUDER, R. (eds.). **Handbook on Ontologies**. Berlin: Springer, 2004. 660p.

STACHOWIAK, H. Models. In: MOSTOWSKI, A. **Scientific Thought**: concepts, methods and procedures. Paris: Unesco, 1972. p. 145-166.

STOJANOVIC, N. **SEAL - A Framework for Developing SEmantic PortALs**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON KNOWLEDGE CAPTURE, 2001, Canada. Proceedings… Canada: ACM Press, 2001.

THALHEIM, B. *et al*. Website modeling and website generation. In: International Conference on Web Engineering, 2004, Munich. Proceedings… Berlin: Springer, 2004.

TRISTÃO, A. M. D.; FACHIN, G. R. B.; ALARCON, O. E. Sistema de classificação facetada e tesauros: instrumentos para organização do conhecimento. **Ciência da Informação**, Brasília, v.33, n.2, p. 161-171, maio/ago. 2004.

VELOSO, F., *et al***. Slicing the Knowledge - based Economy in Brazil, China and India**: A Tale of 3 Software Industries. Cambridge: Massachussetts Institute of Technology (MIT), set. 2003. 42 p. Report.

WEBER, K. *et al.* **Modelo de Referência para Melhoria de Processo de Software**: uma abordagem brasileira. In: XXX CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA (CLEI2004), 30, 2004, Arequipa. *Anais*...Arequipa: [s.n], 2004.

WUESTER, E. L'étude scientifique générale de la terminologie, zone frontalière entre la linguistique, la logique, l'ontologie, l'informatique et les sciences des choses. In: RONDEAU, G.; FELBER, F. (Org.). **Textes choisis de terminologie**. I. Fondéments théoriques de la terminologie. Québec : GIRSTERM, 1981. p. 57-114.

YERGEAU, F. *et al*. **Extensible Markup Language (XML) 1.0** (Third Edition). W3C Recommendation, February 2004. Disponível em: <http://www.w3.org/TR/2004/REC-xml/>. Acesso em: 10 fev. 2005.

YODER, J. W.; BALAGUER, F.; JOHNSON, R. **Architecture and Design of Adaptive Object Models**. In: INTRIGUING TECHNOLOGY PAPER OOPSLA, 2001, Tampa Bay. *Proceedings*... Tampa Bay: ACM SIGPLAN Notices, ACM Press, Dez. 2001. Disponível em:

<www.adaptiveobjectmodel.com/ OOPSLA2001/OOPSLA2001BusinessRules.pdf>. Acesso em: 18 nov. 2004.

APÊNDICES

APÊNDICE A - Softwares pesquisados no sourceforge.

QUADRO 7

Descritor "*content management system*"

Nome do Projeto	Descrição (em inglês)	Atividade %	Maturidade	No. de desenvolvedores
XOOPS Dynamic Web CMS	XOOPS is a dynamic web content management system written in PHP for the MySQL database. It's object orientation makes it an ideal tool for developing small to large community websites, intra company and corporate portals, weblogs and much more.	99.41 %	6	94
Plone	**Plone is a content management and publishing system, sharing the same qualities as Teamsite, Livelink and Documentum. OODBMS, RDBMS, WEBDAV, FTP and XMLRPC integration out-of-the-box. Plone is built with Python and Zope. Please see http://plone.org**	**99.20 %**	**5**	**88**
RainbowPortal	Rainbow is a content management system (CMS) based on MS IBUYSPY portal. We have extended it by implementing multi-language; multiple portals; different themes. This is only the beginning and many more features are planned.	98.29 %	5	42
FSL - OpenUSS	Freestyle Learning (FSL) and Open University Support System (OpenUSS) are specifications for Learning Content System (LCS) and Learning Management System (LMS). They provide J2SE, J2ME and J2EE reference implementations on those specifications.	99.46 %	5	17
myPHPNuke	myPHPNuke is a content management system written in PHP.	99.63 %	5	13
phpCMS Content Management System	phpCMS is a highly flexible flat file, no SQL, Web CMS with complete content/logic separation, featuring e.g.: powerful menu and template system, plug-in capability, scripting (even non-PHP), search engine, statistics, e-mail address cloaking, fast cache	98.46 %	5	12
Bricolage	Bricolage is a full-featured open source content management and publishing system. Its features include intuitive yet highly configurable administration, workflow, permissions, templating, server-neutral output, distribution, and document management.	99.01 %	5	9
Cofax - Content Management - News Media	Cofax is a Web-based text and multimedia publication system. It was designed to simplify the presentation of newspapers on the Web and to expedite real-time Web publication. At Knight Ridder, it is used to manage and serve content for 30+ newspapers.	98.94 %	5	9
GeekLog - The Ultimate Weblog System	GeekLog is a web content management system suitable for running full-featured community sites. It supports article posting, threaded comments, event scheduling, and link management and is built around a design philosophy that emphasizes ease of use.	99.29 %	5	7
Plume CMS	Plume CMS is a fully functional Content Management System in	97.59	5	6

	PHP on top of MySQL. Including articles, news, file management and all of the general functionalities of a CMS. It is completely accessible and very easy to use on a daily basis.	%		
phpWebLog	A complete web news management system written in PHP. All the content control is configurable with an web based administration section. Features include story moderation, threaded comments, templating/themes, polls, multi-language translations, RDF impo	99.56 %	5	5
ATutor (Learning Management System)	ATutor is an Open Source Web-based Learning Content Management System (LCMS), designed with accessibility and adaptability in mind. Interoperable content packaging for creating and reusing learning objects.	98.23 %	5	5
Pagetool	Pagetool is a CMS (content management system) that allows people with limited technical skills to contribute to a web site via a web browser while still giving maximum flexibility to web designers. Please use the pagetool-user list for support.	99.08 %	5	4
SSR Technology	SSR technology is a freely available Web application development and publishing engine (Content Management System, CMS). It's provide powerful API and core function: Very flexible role based user access; Session management; pages and menu management; Ea	99.58 %	5	3
Envolution	Next Generation Dynamic Content Management System, Cutting edge technology, advanced compatibility, fully templated, completely modular and API powered.	97.88 %	5	3
Mambo Open Source Project	Mambo Open Source is an award winning dynamic web content management system (CMS). All Mambo Open Source development has now moved to http://mamboforge.net/	99.61 %	5	1
RUNCMS / E-Xoops	A comprehensive content management system (CMS) where ease of use, speed, & flexibility are the main development keypoints. E-Xoops 1.05r3 was the last version with the E-XooPS name. Change off name to RUNCMS will lead Us into the next generation!.	98.65 %	5	1

APÊNDICE B - Instrumentos de coleta

QUADRO 8

Domínio de conhecimento

Domínio de conhecimento escolhido	Descrição do domínio	Linguagem de representação	No. de classes	No. de propriedades	No. de relacionamentos
Praxis – Processo de Desenvolvimento de Software	O processo Praxis é um processo de desenvolvimento de software. Ele tem enfoque educacional, com o objetivo de dar suporte ao treinamento em Engenharia de Software e à implantação de processos em organizações que desenvolvem, mantêm ou contratam software.	UML	19	26	38

TABELA 2

Medição do tamanho do Sistema de Gestão de Conteúdo

Complexidade	Funções	Qtde	Valores	Total
	ALI	17	7	119
	AIE	0	5	0
Simples	EE	3	3	9
	SE	0	4	0
	CE	17	3	51
	ALI	0	10	0
	AIE	0	7	0
Média	EE	8	4	32
	SE	0	5	0
	CE	0	4	0
	ALI	0	15	0
	AIE	0	10	0
Complexa	EE	6	6	36
	SE	0	7	0
	CE	0	6	0
Pontos de Função Não Ajustados = PFNA				247

TABELA 3

Ajuste dos pontos de função

Níveis de Influências (NI)	**Peso**
Comunicação de Dados	0
Funções Distribuídas	0
Performance	2
Configuração do Equipamento	4
Volume de Transações	0
Entrada de Dados On-Line	0
Interface com o usuário	5
Atualização On-Line	0
Processamento Complexo	5
Reusabilidade	0
Facilidade de implantação	0
Facilidade Operacional	3
Múltiplos Locais	0
Facilidade de mudanças	0
Total dos Pesos (TP)	19
Fator de Ajuste (FA) = (TP * 0,01) + 0,65)	0,84
Pontos de Função Ajustados = PFNA * FA	207,48

APÊNDICE C - Análise por pontos de função

De acordo com a técnica de Análise por Pontos de Função (APF), uma aplicação de software, vista sob a ótica do usuário, é um conjunto de funções ou atividades do negócio que o beneficiam na realização de suas tarefas. O manual do Grupo Internacional de Usuários de Ponto de Função – *International Function Point User Group* (IFPUG) – classifica os seguintes tipos de elementos funcionais (IFPUG, 2000):

- Entrada Externa (EE): transações lógicas que mantém dados internos, nas quais dados entram na aplicação;
- Saída Externa (SE): transações lógicas em que dados saem da aplicação para fornecer informações para usuários da aplicação;
- Consulta Externa (CE): transações lógicas em que uma entrada solicita uma resposta da aplicação;
- Arquivos Lógicos Internos (ALI): grupo lógico de dados mantido pela aplicação;
- Arquivos de Interface Externa (AIE): grupo lógico de dados referenciado pela aplicação, mas mantido por outra aplicação.

O manual do IFPUG fornece tabelas e diretrizes para determinar a complexidade de cada elemento funcional. A complexidade dos ALIs e AIEs é baseada no número de registros lógicos e no número de itens de dados referenciados. A complexidade das transações é baseada no número de arquivos referenciados e no número de itens de dados referenciados.

Para dimensionar a quantidade de horas que serão utilizadas para o desenvolvimento dos programas, aplica-se o coeficiente Horas por Ponto de Função (HPF). Esse coeficiente é baseado em avaliações de produtividade de analistas nas diversas linguagens e ferramentas de desenvolvimento.

Esse coeficiente leva em conta ainda fatores de produtividade da equipe, nível dos

analistas envolvidos, conhecimentos adquiridos, tendo com isso variações para mais ou para menos no coeficiente de produtividade da linguagem. Definem-se coeficientes padrões reconhecidos por diversos sistemas de medição, e posteriormente ajustam-se estes coeficientes.