Um *Framework* para Gerência de Reuso em Projetos de Sistemas de Informação

Aluizio Haendchen Filho[1], Hércules A. Prado[2], Joner C. Worm[3]

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bairro Agronomia
Porto Alegre / RS - Brasil
Caixa Postal 15064 - CEP 91.501-970
Fone +55(051)316-6804
Fax +55(051)319-1576
{aluizio, prado, simao}@inf.ufrgs.br

Resumo. O desenvolvimento de *software* baseado em reuso possui dois lados: o do fornecimento e o da demanda. O lado do fornecimento deve prover produtos de uso genérico e um *framework* para captura dos componentes reusáveis e sua manutenção em um repositório para um dado domínio. Novos produtos são continuamente incorporados neste repositório à medida em que surgem componentes potencialmente reusáveis durante o processo de desenvolvimento de sistemas. O lado da demanda recupera os componentes do repositório do domínio, adaptando-os às aplicações específicas. Este artigo apresenta um projeto de uma ferramenta CASE (*Computer-aided Software Engineering)* repositório que permite classificar, catalogar e recuperar componentes potencialmente reusáveis para que possam ser utilizados em outras aplicações. O *framework* foi implementado e um exemplo da sua aplicação, juntamente com a descrição da ferramenta, são mostrados.

Palavras-chave: Ferramenta CASE, Repositório de componentes, Componentes reusáveis, Reuso sistemático de *software*.

1 Introdução

O reuso de código é uma prática utilizada desde os primórdios da programação de *software.* A proposta de bibliotecas de componentes compartilhados, apresentada em 1968 por Dog McIlroy, marcou o início de uma era na Engenharia de *Software* em que o reuso de código passou a ser reconhecido como uma atrativa idéia para reduzir custo e tempo de desenvolvimento. Hoje é largamente difundida a idéia de que construir sistemas de *software* com componentes de qualidade, previamente desenvolvidos, certamente reduz o custo e tempo gasto em trabalho redundante e melhora a qualidade dos sistemas [JAC 98]. Por outro lado, importantes autores, em diferentes momentos ([KAN 87] e [SOD 98]), enfatizam que a prática tem sido principalmente *ad hoc*, ou seja, a aplicação de reuso sem haver um processo de planejamento ou uma abordagem formal estruturada. É desejável que o processo de reuso seja mais formal do que um simples ato de rememorar.

---

[1] Professor e pesquisador da Universidade Luterana do Brasil e consultor independente
[2] Professor da Universidade Católica de Brasília e pesquisador da Empresa Brasileira de Pesquisa Agropecuária
[3] Analista de Sistemas da Justiça Federal do Brasil

A aplicação de uma estratégia para reusabilidade amplifica as capacidades dos desenvolvedores de *software*, podendo, através da utilização de modelos e metodologias, reduzir drasticamente os custos e melhorar a qualidade do *software* desenvolvido [BIG 89].

Este artigo apresenta um projeto de uma ferramenta CASE repositório que permite classificar, catalogar e recuperar componentes potencialmente reusáveis para que possam ser utilizados em outras aplicações. A idéia é validada através da implementação de um protótipo e um exemplo de aplicação para o desenvolvimento de sistemas de informação. Esta ferramenta torna disponível um repositório que, através de uma interface, permite classificar, ordenar, armazenar e recuperar e os mais variados componentes de um projeto, como documentos, diagramas, especificações, código, e outros, facilitando a sua adaptação. Estes componentes são criados e alterados com ferramentas do sistema operacional hospedeiro, sendo mantidos em dois repositórios lógicos: repositório de componentes do domínio e repositório de componentes da aplicação.

No Capítulo 2 descrevemos uma abordagem destilada das propostas por Moore [MOO 91] e Sodhi [SOD 98] para a Engenharia de *Software* dirigida ao reuso. No Capítulo 3 discutimos os trabalhos mais relevantes relacionados com a nossa proposta. O Capítulo 4 descreve a ferramenta ao mesmo tempo em que exemplifica a sua aplicação para um caso real. As contribuições deste trabalho são revistas no Capítulo 5. No decorrer do artigo usaremos o termo *artefato* para designar genericamente qualquer produto do processo de desenvolvimento de um sistema de informação, podendo incluir documentos, diagramas, cronogramas, e mesmo *software*. O termo *componente* é mais restrito, referindo-se a um artefato que foi preparado para ser reusado, conforme [JAC 98] p. 85.

2 Um processo para reuso sistemático de *software*

O desenvolvimento de *software* baseado em reuso consiste de dois lados: o do fornecimento e o da demanda. O lado do fornecimento deve prover produtos de uso genérico e um *framework* para captura dos componentes reusáveis e sua manutenção em um repositório para um dado domínio [MOO 91]. O lado da demanda recupera os componentes do repositório do domínio, adaptando-os às aplicações específicas.

Utilizando o paradigma do reuso sistemático de *software*, o processo de reusabilidade captura os pontos comuns e as diferenças existentes entre os modelos de sistemas. Isto inclui a *aquisição e desenvolvimento* de recursos reusáveis, o *gerenciamento* e o *uso* desses recursos. De acordo com Sodhi [SOD 98], basicamente três atividades ocorrem no processo de reuso: atividades de *engenharia do domínio*, atividades de *engenharia da aplicação* e atividades de *gerenciamento do reuso*. Estas atividades encontram-se representadas na Figura 1.

Domínio é definido por Sodhi [SOD 98] como o espaço do problema para uma família de aplicações com requisitos similares. *Engenharia do domínio* é um termo de reuso que descreve um caminho sistemático para a identificação de um modelo do domínio, semelhanças e diferenças, artefatos potencialmente reusáveis e uma arquitetura para habilitar o seu reuso. Engenharia do domínio pode ser interpretada como *engenharia-para-reuso* e representa, portanto, o lado do fornecimento. A engenharia do domínio compreende as atividades de *análise do domínio*, determinação da *arquitetura de software do domínio* e o desenvolvimento de *componentes reusáveis*, oriundos das aplicações. Estes componentes são trabalhados com a finalidade de torná-

los genéricos, reduzindo dependências e aumentando o seu potencial para ser reutilizado por outras aplicações.

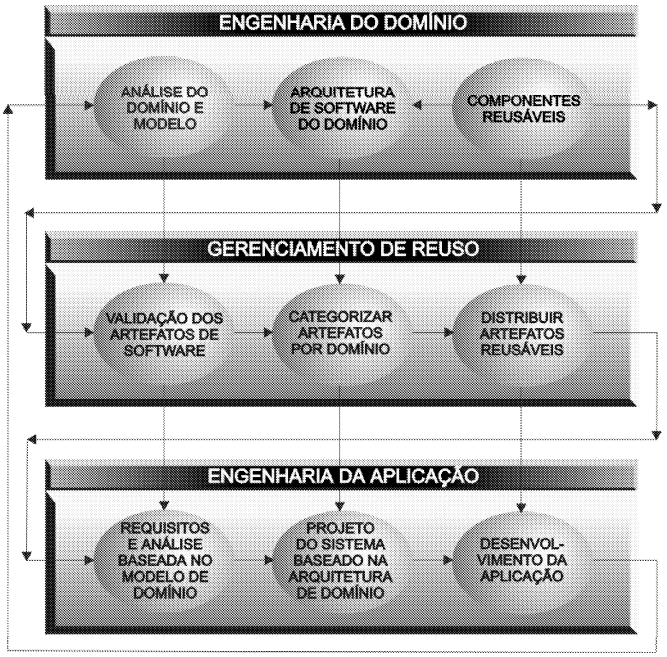

FIGURA 1 - Atividades para reuso de software [SOD 98]

O *gerenciamento do reuso* tem por função validar, classificar e distribuir os componentes para serem reusados. Entre o lado do fornecimento (engenharia do domínio) e o lado da demanda (engenharia da aplicação), o gerenciamento do reuso desempenha as funções de viabilizar as interações existentes, tornando o acesso aos componentes possível aos desenvolvedores. O gerenciamento de reuso abrange, além da validação dos componentes, um mecanismo para sua classificação e armazenamento.

A *engenharia da aplicação* utiliza os componentes disponibilizados pela engenharia do domínio quando novas aplicações são desenvolvidas. A engenharia da aplicação também identifica componentes depois de terem sido usados e modificados para novas aplicações. Estes componentes são então transferidos para o modo de engenharia do domínio para serem catalogados e classificados e minimizadas as suas dependências [SOD 98]. A engenharia da aplicação, portanto, é parte do processo que incorpora o ciclo de vida de um sistema; este processo organiza e direciona recursos para produção e suporte de sistemas através da aplicação do paradigma de reuso. A engenharia da aplicação refere-se à construção de sistemas novos ou sistemas reconstruídos através de um processo de reengenharia. A arquitetura da aplicação deverá ser derivada da arquitetura de *software* do domínio.

3 Trabalhos Relacionados

Existem, basicamente, duas ferramentas no mercado com funcionalidades semelhantes às da presente proposta: uma é a *Structure for Information Systems Architecture* (SISA) da Framework Software, Inc., de Redondo Beach, Califórnia, e a outra MetaEdit+, da MetaCase Consulting, Finlândia. Ambas são ferramentas CASE que mantêm repositórios de componentes. Estas

ferramentas apresentam algumas características que tornam a sua utilização limitada. No caso da ferramenta SISA, esta trabalha somente ao nível do contexto definido por Zachman [INM 97]. A MetaEdit+, por sua vez, não possui interface para manipulação de objetos interoperáveis (OLE/DE/CORBA) e nem fornece suporte para controle de versões. A ferramenta descrita na presente proposta foi projetada para superar estas deficiências.

4 Descrição da Ferramenta

A ferramenta baseia-se nas atividades para reuso descritas por Sodhi [SOD 98] (Figura 1). Suas funcionalidades implementam o gerenciamento do reuso, que faz a comunicação entre as atividades de engenharia do domínio e a engenharia da aplicação. Através desta comunicação, o gerenciamento do reuso recebe solicitações de componentes da engenharia da aplicação, as quais ele tenta solucionar por meio da utilização dos componentes disponíveis no domínio. Uma interface permite a sua fácil localização e recuperação para reuso. Para popular e manter os repositórios da ferramenta, faz-se uso dos recursos de objetos interoperáveis, através das tecnologias COM (Component Object Model), da Microsoft ou CORBA (ORACLE/OMG). A utilização destas tecnologias possibilita o acesso a objetos criados e atualizados por outras ferramentas, permitindo que estes objetos sejam manipulados para compor uma aplicação específica.

A ferramenta proposta para gerência de reuso não é uma metodologia, e sim um esquema de classificação de produtos derivados de metodologias. O modelo apresentado provê uma arquitetura para o mapeamento de metodologias, utilizando os componentes necessários para o desenvolvimento de uma estratégia global de reuso dos produtos resultantes destas metodologias [HAE 00].

4.1 Classificação de Componentes

O processo de desenvolvimento de sistemas de informação com a finalidade de aplicar o paradigma de reuso, deve prover elementos para representar e classificar informações para melhor entendimento do problema e das funções nas quais cada um dos componentes pode atuar para atingir os objetivos propostos. Neste trabalho, utilizamos como principal elemento classificador uma matriz de dupla entrada, composta por linhas (*perspectivas*) e colunas (*dimensões*). Esta matriz fornece um contexto arquitetural para representar qualquer produto ou processo, ajudando a quebrar a arquitetura em pedaços administráveis. Nela, os vários componentes de um sistema em desenvolvimento podem ser analisados, planejados e visualizados através do mapeamento das varias etapas do ciclo de vida para as células correspondentes. As células da matriz representam o encontro das *perspectivas* com as *dimensões*.

Conforme a Figura 2, existe um mapeamento entre os níveis de detalhamento do domínio e da aplicação. Este mapeamento é iniciado através de uma consulta na tabela de dimensões e perspectivas do domínio. A partir desta consulta, são identificados os componentes do domínio passíveis de serem utilizados na aplicação. A seguir os mesmos são carregados para a aplicação, onde podem ser adaptados pelo desenvolvedor.

Uma vez definidas as perspectivas e dimensões para um contexto, os vários domínios pertencentes a este contexto vão se reportar ao esquema de classificação definido para o mesmo. No contexto de sistemas de informação, que é o nosso caso, existe uma tabela de

perspectivas/dimensões específica para cada domínio, com seus respectivos componentes. Cada célula da matriz, por sua vez, pode possuir diferentes tipos de componentes, sendo que cada tipo pode ter várias implementações. O mesmo se aplica para os níveis de detalhamento da *aplicação*, com a diferença de que esta apresenta níveis mais específicos do que os do *domínio*.

Uma vez definidas as perspectivas e dimensões para um contexto, os vários domínios pertencentes a este contexto vão se reportar ao esquema de classificação definido para o mesmo. No contexto de sistemas de informação, que é o nosso caso, existe uma tabela de perspectivas/dimensões específica para cada domínio, com seus respectivos componentes. Cada célula da matriz, por sua vez, pode possuir diferentes tipos de componentes, sendo que cada tipo pode ter várias implementações. O mesmo se aplica para os níveis de detalhamento da *aplicação*, com a diferença de que esta apresenta níveis mais específicos do que os do *domínio*.

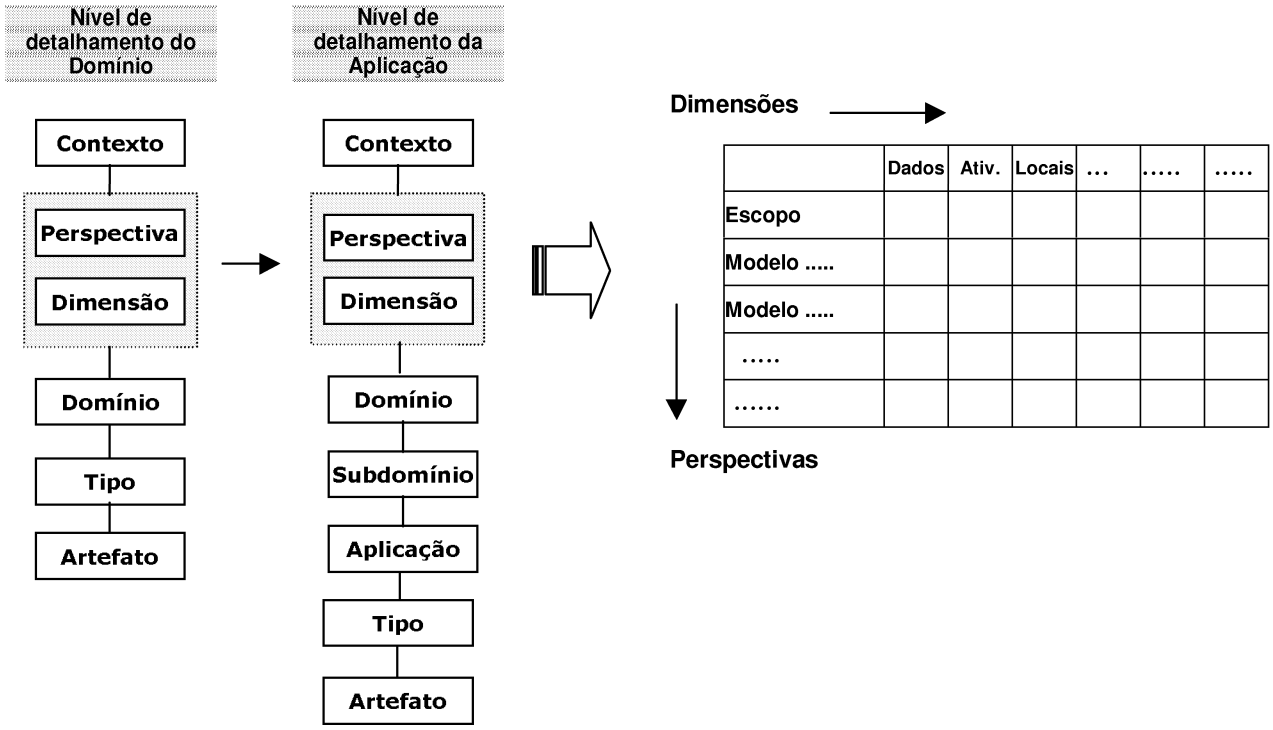

FIGURA 2 – Mapeamento de componentes de domínio em aplicações

A tabela que aparece ao lado direito da Figura 2 é um exemplo de aplicação que pode construída utilizando a ferramenta, a partir do cadastramento de *perspectivas* e *dimensões*. Dentro de um determinado *contexto*, poderíamos mapear para as linhas (*perspectivas*) da matriz os diferentes níveis de detalhamento de um sistema de informação (escopo, modelo da organização, modelo do sistema, modelo técnico e modelo de componentes) e para as colunas (*dimensões*) as diferentes visões (dados, atividades, locais, pessoas, tempo e motivação), como proposto por Zachman [ZAC 89].

A principal função do nível *contexto* é permitir a criação de vários esquemas diferentes para classificar representações descritivas de objetos complexos. A estrutura proposta por Zachman, é derivada de estruturas análogas que são encontradas nas mais antigas disciplinas de arquitetura e engenharia, que classificam e organizam os componentes de um processo complexo, como por exemplo, a construção de um edifício ou a fabricação de um avião [INM 97]. Sendo a lógica da

ferramenta genérica, ela pode ser utilizada para classificar a descrição de empresas, computadores, sistemas, *data warehouses* [HAE 00a], ou qualquer outra coisa.

4.2 Meta-modelo e repositório de componentes

O diagrama da Figura 3 mostra o meta-modelo da ferramenta, evidenciando um mapeamento entre *Artefato do Domínio* e *Artefato da Aplicação*. Este mapeamento é materializado por uma relação biunívoca entre as células do nível do domínio e da aplicação, identificadas pelo mesmo par perspectiva/dimensão. Em outras palavras, através das células do *framework*, é possível relacionar artefatos entre os repositórios do domínio e da aplicação. Neste meta-modelo estamos usando o termo *artefato* porque a ferramenta armazena elementos de diferentes contextos, onde os mesmos podem ou não ser reusáveis.

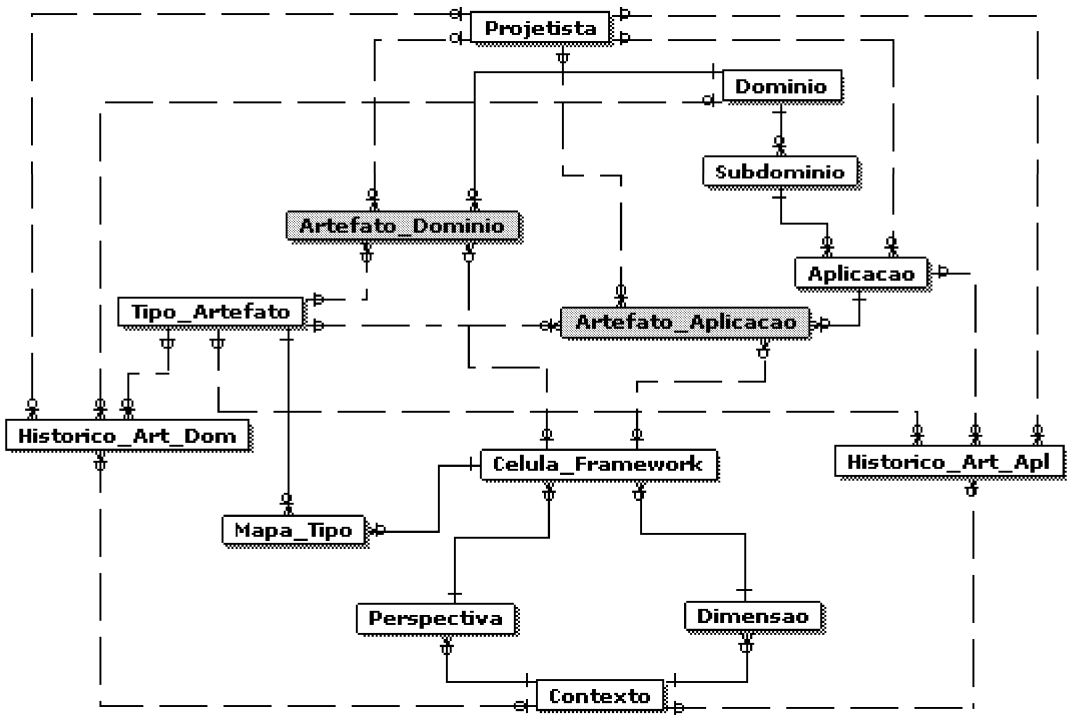

FIGURA 3 – Meta-modelo da ferramenta

Como pode ser visto na Figura 3, as entidades *Artefato_Domínio* e *Artefato_Aplicação* se relacionam com a entidade *Célula_Framework*, que representa o encontro das linhas e colunas da matriz e que será um dos pontos fundamentais para o endereçamento lógico do componente do domínio ou da aplicação. Os artefatos podem se tornar componentes a partir do esforço de um desenvolvedor do domínio no sentido de torná-los reusáveis.

Uma importante funcionalidade provida pela ferramenta é o controle de versões dos componentes. Podem ser recuperadas, tanto para os componentes do domínio quanto para os componentes da aplicação, diferentes versões do componente ou o histórico das modificações efetuadas nos componentes ao longo do tempo. Conforme Figura 3, estas informações são armazenadas nas entidades *Historico_Art_Dom* e *Historico_Art_Apl*.

Para popular e manter o repositório *Artefato_Domínio*, cada potencial componente deve conter: (a) os identificadores de *Contexto*; (b) as coordenadas das células (*Perspectiva* e *Dimensão*, presentes na entidade C*élula_Framework)*; e (c) o *Tipo_Artefato*, considerando os tipos válidos para aquela célula, presentes na entidade *Mapa_Tipo*. A maioria destes atributos são capturados no momento em que as ações vão sendo selecionadas na interface pelo usuário.

4.3 A Interface do Usuário

O mecanismo principal no processo de gerenciamento de reuso é suportado por duas interfaces que trabalham em sincronismo: a interface do *domínio* e a interface da *aplicação*. Na verdade, a interface começa a ser montada pelo usuário utilizando um configurador disponibilizado pela ferramenta. Neste configurador, é possível ao usuário especificar *projetistas*, *contextos*, *perspectivas* e *dimensões* e *tipos* de artefato. Para cada *contexto*, podem ser criadas as *perspectivas* e *dimensões* correspondentes, específicas daquele contexto. O usuário não percebe que a interface vai sendo construída na medida que as linhas (*perspectivas*) e colunas (*dimensões*) do *framework* vão sendo cadastradas no configurador.

Uma opção de projeto de interface adaptável possibilita ao usuário tornar a interface mais apropriada para realizar suas tarefas. Desta forma, mais rapidamente ele poderá construir projetos e protótipos e menos tempo será gasto na seleção e localização dos componentes do *domínio* ou da *aplicação*. Considerando que a mente humana opera através de associações, são previstos recursos para que o usuário possa utilizar descrições, ícones e metáforas que traduzam o *contexto* onde a ferramenta será aplicada.

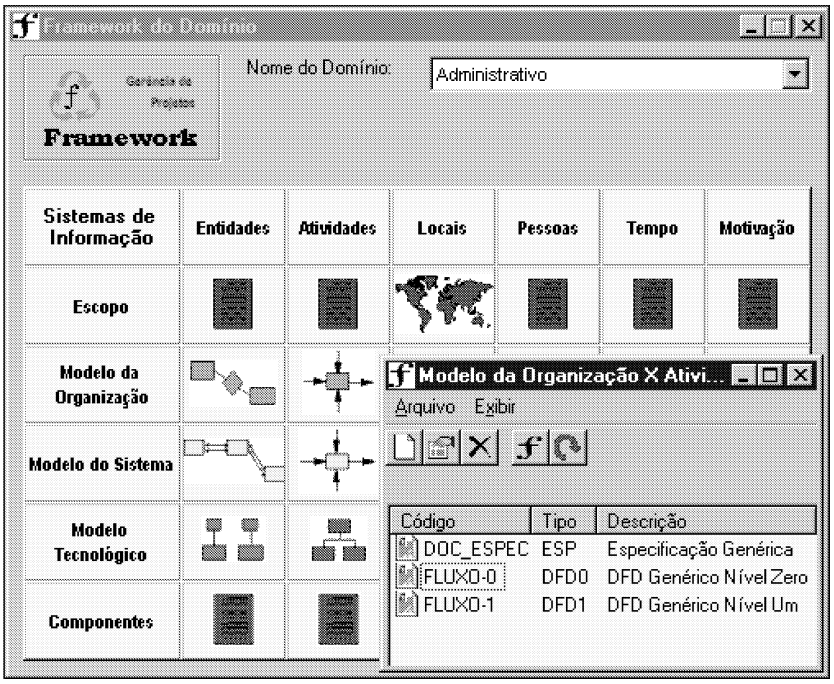

FIGURA 4 – Componentes contidos em uma célula (nível do domínio)

A Figura 4 exibe o conteúdo de uma célula (Modelo da Organização X Atividades) do repositório do domínio, tomando como exemplo o modelo proposto por Zachman [ZAC 89] para

classificar componentes de sistemas de informação. O contexto de Zachman foi tomado apenas como exemplo, sendo que qualquer outro, como o contexto da Engenharia da Informação [FEL 88], poderia ter sido utilizado. Esta característica visa superar a limitação apresentada pela ferramenta SISA.

A Figura 5 mostra o conteúdo de uma célula (OrganizaçãoXAtividades) do repositório da aplicação. Comparando as taxonomias dos dois níveis – domínio e aplicação – da Figura 2, observa-se que no nível da aplicação está prevista a classificação dos componentes por subdomínio e aplicação, em adição às demais classificações existentes em ambos os níveis. Isto deve-se ao fato de que os componentes de uma mesma aplicação devem ficar reunidos para facilitar a sua manipulação e a compreensão das suas funcionalidades.

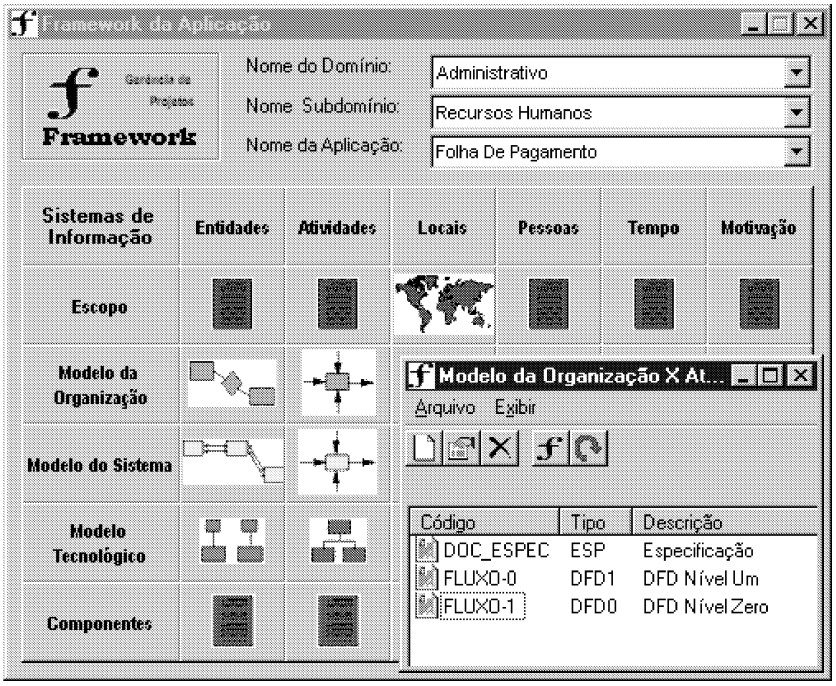

FIGURA 5 – Componentes contidos em uma célula (nível da aplicação)

A partir da janela da Figura 5, pode-se incluir, excluir ou editar um componente dos níveis do domínio ou da aplicação. Uma inclusão no nível da aplicação é ilustrada pela Figura 6. Consta desta figura um diagrama de entidade-relacionamento sendo editado: no lado esquerdo são especificados os atributos do componente e no lado direito o próprio componente.

Um detalhe importante é que o componente pode ser manipulado dentro da própria janela do *framework*. Os botões *importar* e *exportar* permitem a troca de componentes entre os lados da oferta e da demanda. Quando um componente é incluído ou editado em uma aplicação ou no domínio, a ferramenta, através do recurso de automação OLE, fornece uma interface. Esta interface permite que objetos não relacionados troquem informações sem terem sido explicitamente programados para suportar uns aos outros. A implementação possibilita trabalhar com as duas formas possíveis de interoperabilidade: vinculada (*linked*) e incorporada (*embedded*).

A vinculação possibilita, por exemplo, que um diagrama E-R construído usando a ferramenta CASE ERWin possa ser editado na aplicação mantendo o vínculo com a ferramenta.

Quando ele é carregado, o recurso OLE permite que a edição do objeto seja feita na própria célula do *framework*, muito embora este objeto permaneça vinculado à aplicação de origem. Uma alteração em um objeto **vinculado** será propagada a todas as aplicações que utilizam este objeto. Por outro lado, na **incorporação**, o objeto criado a partir de uma aplicação passa a ser um componente estático da aplicação, ou seja, passa a ser incorporado como se fosse um anexo à aplicação. Qualquer objeto criado em aplicações do sistema operacional hospedeiro pode ser acessado através do recurso de automação OLE.

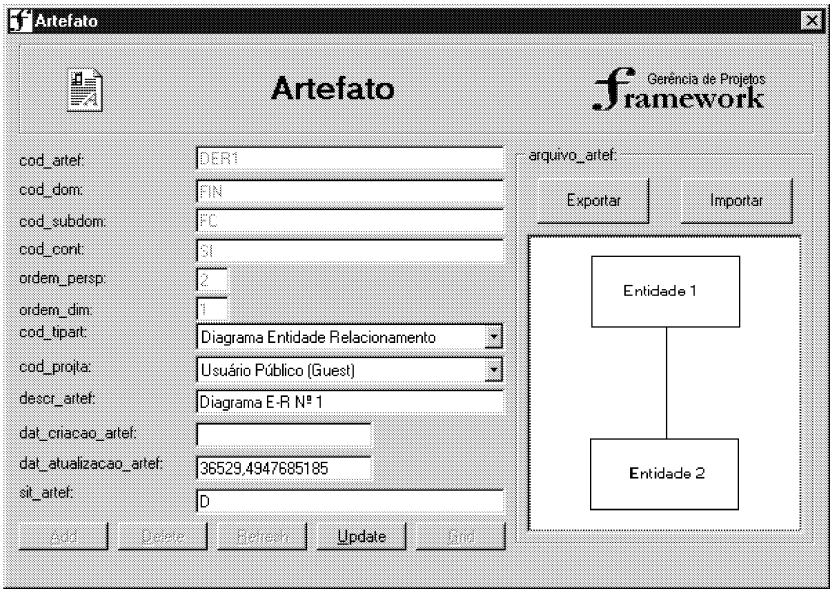

FIGURA 6 – Componente importando do repositório do domínio

5 Contribuições e trabalhos futuros

Apresentamos neste trabalho uma ferramenta para suportar o processo de reuso sistemático de *software* com diferenciais importantes com relação às ferramentas similares existentes no mercado. Nomeadamente, são providas funcionalidades para o controle de versões de componentes, flexibilização do contexto e suporte a interoperabilidade. Nesta primeira versão, a ferramenta suporta componentes gerados a partir da plataforma Windows, permitindo a manipulação de componentes interoperáveis dentro do padrão ActiveX. Sua extensão para o padrão CORBA encontra-se em desenvolvimento.

Referências:

[BIG 89] BIGGERSTAFF, T.J. **Software Reusability - Concepts and Models**. New York: ACM Press, 1989. 425 p.

[FEL 88] FELICIANO NETO, A. e FURLAN, J. D. **Engenharia da Informação: Metodologias, Técnicas e Ferramentas.** Sâo Paulo: McGraw-Hill, 1988. 263 p.

[HAE 00] HAENDCHEN FILHO, A. **Uma Estrutura para Desenvolvimento e Gerenciamento de Software Reusável para Sistemas de Informação**. Porto Alegre: Universidade Federal do Rio Grande do Sul, CPGCC, 2000. Dissertação de Mestrado.

[HAE 00a] HAENDCHEN FILHO, A., PRADO, H.A.; TOSCANI, S.S. **Evolving a Legacy Data Warehouse System to an Object-Oriented Architecture** em: XX INTERNATIONAL CONFERENCE SCCC, 2000. Proceedings... Los Alamitos,CA,USA: IEEE Computer Society Press, 2000.

[INM 97] INMON, W.H. et al. **Data Stores, Data Warehousing and the Zachman Framework**. New York: McGraw-Hill, 1997. 358 p.

[JAC 98] JACOBSON, I. **Software Reuse - Architecture, Process and Organization for Business Sucess**. New York: Addison-Wesley, 1998. 497 p.

[KAN 87] KANG, K.C. A Reuse-Based Software Development Methodology. In: BOOCH, G.; WILLIAMS, L. **Workshop on Software Reusability and Maintanability**. [S.l.]: Rocky Mountain Inst. of Software Engineering, 1987.

[McI 68] McILROY, D. **Mass Produced Software**. *1968 NATO Conference on Software Engineering*, pp.138-55.

[MOO 91] MOORE, J.M. Domain Analysis: Framework for Reuse, In: PRIETO-DIAZ, R.; ARANGO, G. **Domain Analysis and Software Systems Modeling**. Los Alamitos, CA: IEEE Computer Society Press Tutorial, 1991.

[SOD 98] SODHI, J. et al. **Software Reuse - Domain Analysis and Design Process**. New York: McGraw Hill, 1998. 344 p.

[ZAC 89] ZACHMAN, J. Enterprise Architecture: The Issue of the Century. **Database Programming and Design Magazine**, New York, Mar. 97. Disponível por www em: http://www.zifa.com/zifajz01.htm.