Uma Arquitetura para Reduzir a Complexidade e Aumentar a Produtividade do Ciclo de Vida do Desenvolvimento de Sistemas

André Luis Costa de Oliveira
José Palazzo Moreira de Oliveira

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Caixa Postal 15064
Porto Alegre - RS - Brasil

e-mail: andre@cs.inf.br, palazzo@inf.ufrgs.br

Resumo

O presente trabalho apresenta o resultado de uma cooperação entre o Instituto de Informática da Universidade Federal do Rio Grande do Sul e a Cumerlato & Schuster Informática S.A. Esta integração permitiu a elaboração de uma dissertação de Mestrado e logo a seguir, o desenvolvimento e a implementação do projeto proposto. Através da metodologia e ambiente apresentados é possível reduzir a complexidade encontrada no desenvolvimento de sistemas de Informação. As tarefas, ferramentas e modelos utilizados no desenvolvimento são decompostos em módulos e inseridos em uma estrutura que permite a integração das várias partes, a especificação abstrata e a automação de processos. Os Sistemas são especificados em Ferramentas CASE, orientadas por uma Metodologia de Desenvolvimento. As definições são armazenadas de maneira abstrata em um repositório (independente de tecnologia), o que permite a conversão em código fonte para a linguagem (PowerBuilder, Visual Basic, HTML, etc) e arquitetura desejada (Client/Server, 3-tiers, etc.). Desta forma, é possível manter a evolução tecnológica sem reescrever a aplicação. Atualmente a Estrutura de Módulos está sendo utilizada na Cumerlato & Schuster em 3 projetos e em breve se tornará padrão para os demais. A utilização da Estrutura está permitindo atingir níveis de 98% de geração de código de Interface e de Banco de Dados (as Regras de Negócio ainda são codificadas), com redução de aproximadamente 60% do tempo de especificação. Com estas medidas, os custos dos projetos são menores e o tempo de implementação é reduzido em mais de 50%.

1 Introdução

O presente trabalho tem como base uma Estrutura de módulos [6], que define um método para suporte ao ciclo de vida da aplicação de forma independente de tecnologia e de ferramentas. O objetivo principal é apresentar os problemas existentes no desenvolvimento, a especificação com algumas das ferramentas para produtividade construídas e os benefícios diretos e indiretos da estrutura de módulos. Para finalizar, são discutidas as principais diferenças encontradas entre o ambiente proposto e a forma de implementação anterior, utilizando para isto, as ferramentas de desenvolvimento da empresa e outras que foram construídas para oferecer maior produtividade. A idéia inicial da Estruutra de módulos, desde sua concepção até a comparação com outras ferramentas está descrita na Dissertação de Mestrado.

Entre os problemas encontrados durante o Desenvolvimento de Sistemas atualmente estão: a falta de recursos de muitas ferramentas, a integração entre produtos e a automatização de processos, ocasionando a execução manual de muitas tarefas existentes ao longo das etapas de especificação e implementação. Assim, as tarefas tornam-se muito lentas, sujeitas a erros, apresentam retrabalho e obrigam os profissionais a ter alto conhecimento técnico. Com isto é difícil manter padrões, pois muitas tarefas são subjetivas.

Dos projetos desenvolvidos atualmente apenas 16% são entregues no prazo, sendo que 31% são cancelados mesmo antes de terminar [11]. Isto geralmente ocorre porque o tempo entre o inicio e o fim do desenvolvimento é muito longo. Além disto, os dados levantados no início do projeto mudam, pois em um ambiente de concorrência acirrada as empresas devem ser flexíveis para conseguirem melhor resultado, exigindo mudanças freqüêntes em seu negócio.

Outro fato que ocorre quando o ciclo de desenvolvimento é longo e o resultado final demora a ser apresentado, é a mudança de paradigma. Isto acontece após o término do desenvolvimento da aplicação, no momento que é apresentada aos usuário ou implantada. Durante este período de utilização da aplicação os usuários podem ter uma visão mais abrangente ou novas idéias, solicitando características diferentes e melhores, que não estavam previstas no projeto [10].

Os fornecedores de *hardware* e *software* freqüentemente lançam versões evoluídas de seus produtos. Uma aplicação, mesmo no começo de sua implementação, já está sujeita a mudanças, pois o *software* ou tecnologia empregados também sofrem alterações de versões. Na maioria dos casos, a adequação da aplicação para a nova tecnologia exige alterações no projeto físico (por exemplo: objetos distribuídos, componentes, cliente/servidor, características para utilização na WEB, etc.). Para o usuário final, a aplicação (telas e funcionalidade) pode parecer a mesma, independente da tecnologia que foi utilizada. Mas o código interno é totalmente diferente, passando por todas as atividades do desenvolvimento novamente. Com a implantação da aplicação nas novas tecnologias, a empresa está sujeita a novos erros, custos com treinamento, custos com a compra de *software* e *hardware*, custo com desenvolvimento, pesquisas, bibliografia, etc.

Para que um sistema seja desenvolvido com sucesso, torna-se necessária uma metodologia que forneça a estrutura de trabalho para a arquitetura, infra-estrutura, escopo, modelagem, construção, aprovação, ajustes, testes, desenvolvimento e manutenção da aplicação. A mesma deve também prover uma completa solução para as informações, incluindo tarefas e papéis de cada pessoa, padrões para arquitetura e projeto, entrega do produto final com possibilidade de adaptações e estratégias para o sucesso (ex.: pontos de validação, negociação de cronograma, gerenciamento de riscos, etc.) [8]. Grande parte destas técnicas são implementadas por ferramentas diferentes, que não são integradas. A falta de integração exige muito retrabalho, resulta em perda de informação, está sempre sujeita a erros e demanda muito tempo. Desta forma, cada vez mais é necessária uma ferramenta que armazene, integre e gerencie as informações utilizadas e produzidas durante o ciclo de vida do desenvolvimento de sistemas, de maneira que não fique obsoleta ao longo do tempo e contemple todas as regras descritas pelas metodologias.

Os motivos citados acima são responsáveis, em grande parte, para a baixa produtividade do desenvolvimento de uma aplicação, provocando um aumento excessivo no tempo decorrido entre a etapa inicial e a final do ciclo de vida de um sistema. Além dos problemas específicos apresentados, outras questões foram analisadas para elaborar a Estrutura de Módulos, de forma que fosse possível oferecer maior produtividade durante todo o Ciclo de Vida do Desenvolvimento, ao invés de determinados processos isolados. Algumas destas questões são: Independência de ferramentas, diminuição do tempo de especificação, aumento da confiabilidade, redução de riscos, abrangência para vários tipos de Sistemas de Informação, utilização de tecnologias disponíveis, acompanhamento do avanço tecnológico, aproveitamento do conhecimento, reutilização de projeto e possibilidade de personalizar o produto final.

Para solucionar os problemas citados é necessário que as especificações sejam abstratas, que várias ferramentas possam ser integradas e que tarefas sejam automatizadas. A Estrutura de Módulos contempla estas características, oferecendo um retorno mais rápido, com mais confiabilidade e maior flexibilidade.

2 Desenvolvimento utilizando a Estrutura de Módulos

A estrutura de módulos, parte principal da Dissertação de Mestrado, é um conceito utilizado para gerenciar, organizar, integrar e armazenar o resultado das atividades (que são transformadas em módulos na estrutura) existentes ao longo do ciclo de vida (estrutura) da aplicação.

O objetivo principal da estrutura de módulos é oferecer uma arquitetura para o desenvolvimento de sistemas, visando obter maior produtividade e independência de tecnologia para o produto final, através de uma arquitetura modular. Utilizando-se a estrutura de módulos, é possível obter resultados mais rápidos no desenvolvimento, automatizar tarefas e manter o investimento aplicado na construção de sistemas.

As informações relativas à especificação dos sistemas, são armazenadas em um repositório de dados, representado pelo módulo central na estrutura. Este módulo é chamado de Metamodelo. O Metamodelo mantém as especificações em alto nível de abstração, sem detalhes físicos de implementação. Estas especificações podem ser convertidas para código fonte de qualquer linguagem ou ferramenta que esteja agregada na estrutura.

A estrutura é dividida em módulos, que representam as atividades ou ferramentas existentes no desenvolvimento de sistema. Estas são agrupadas em partes lógicas, de acordo com uma metodologia. A Figura 1 apresenta uma parte da Estrutura de Módulos, representando o processo de especificação dos modelos utilizados nas tarefas de Análise e Projeto. Os modelos são importados e convertidos (quanto mais interno for o módulo, mais abstrata é a informação) para armazenamento no Metamodelo.

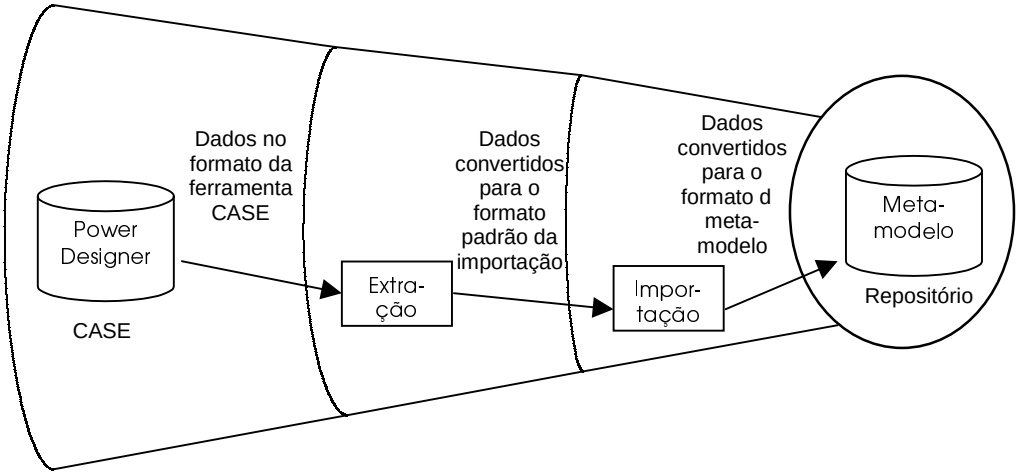

Figura 1 – Exemplo de tarefas que são tratadas como módulos

Alguns módulos como a codificação, a geração de diagramas, a validação e consistência de dados podem ser automatizados. Quanto mais módulos (atividades da metodologia de desenvolvimento) estiverem automatizados, produzindo o resultado esperado a partir das definições, mais rápido será desenvolvido o sistema, pois o trabalho manual e sujeito a erros é evitado [9].

A divisão em módulos também permite que as etapas iniciais do desenvolvimento de sistemas sejam reaproveitadas, pois as especificações são mantidas de forma independente de tecnologia e podem ser utilizadas em outros projetos. Da mesma forma, os níveis externos da estrutura, responsáveis pela geração do código, podem ser substituídos, sem a necessidade de alteração no Metamodelo. Assim a aplicação não precisa ser rescrita totalmente, bastando para isto a construção ou agregação de um módulo que substitua os existentes.

As entradas e saídas de cada módulo são definidas previamente, permitindo assim a fácil inclusão ou substituição de módulos. Desta forma, novas tarefas e recursos podem ser acrescentados na metodologia original e na estrutura de módulos existente (por exemplo: gerenciamento do desenvolvimento, validações dos modelos, geração de código, relatórios, etc.), através de módulos que são agregados no metamodelo. Como apresentado na Figura 2, o módulo “M” pode chamar tanto o “A”, “B” quanto o “C”, passando como parâmetro as informações estruturadas da mesma maneira.

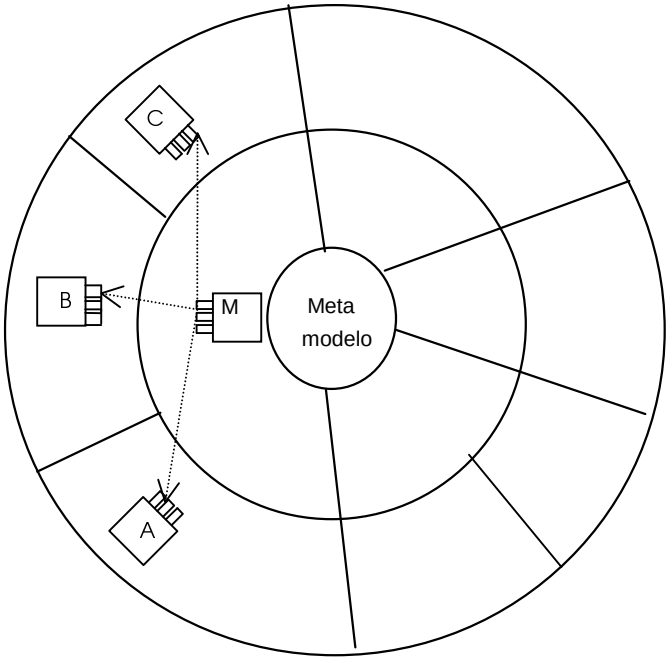

Figura 2 – Conversão e parâmetros entre os módulos

A comunicação entre os módulos é baseada em parâmetros, como apresentado na Figura 2, que são determinados na especificação da estrutura (a cada novo projeto a estrutura pode ser modificada, pois novos produtos ou ferramentas podem ser acrescentadas). Como os parâmetros são previamente estabelecidos, é possível montar uma tabela de conversão (tabela "De"-"Para", representando origem e destino). Esta tabela indica que cada elemento recebido ("De") do módulo anterior, será implementado de uma determinada maneira ("Para") no módulo posterior. No caso da substituição do módulo posterior (ex. nova linguagem de programação), os novos elementos ("Para") devem ser associados com os elementos do módulo anterior ("De") que não precisam ser modificados. A tabela de conversão permite que o sistema seja especificado de maneira abstrata no metamodelo e assim, traduzido (a partir das associações "De"-"Para") para os elementos desejados [12] [13]. Estes elementos podem ser componentes físicos de uma linguagem de programação (extração do metamodelo) ou metadados [1] recuperados de uma ferramenta CASE para o metamodelo (inclusão no metamodelo).

No Metamodelo (parte central da Estrutura de Módulos), o conhecimento do sistema está armazenado em um formato independente de linguagens e ferramentas, para que seja mantido de forma flexível, resistindo ao longo das mudanças tecnológicas.

De acordo com a estrutura, os demais módulos são construídos e agregados ao Metamodelo na forma de camadas. Desta forma, um produto existente pode ser utilizado e agregado à estrutura, inclusive ferramentas CASE e geradores de código, bastando para isto, criar módulos que fazem a interface entre o modelo e o novo produto [4] [7].

A Figura 3 apresenta um exemplo da estrutura de módulos para suporte a todo o Ciclo de Desenvolvimento de Sistemas. A especificação dos modelos (DFD, DER, DE) é gerada a partir de uma Ferramenta CASE (ex. PowerDesigner ou System Architect). O módulo "Extrator de Dados" correspondente à Ferramenta CASE em questão é responsável por

recuperar das informações e enviar a um módulo genérico (“Importação”), que vai armazenar as informações de forma abstrata no Metamodelo. Depois que as informações relativas ao Sistema especificado na Ferramenta CASE estiverem no Metamodelo, as interfaces (módulo “DHF”) e as Regras de Negócio (módulo “Regras de Negócio”) são definidas. Depois de especificadas todas as informações relativas ao Sistema desejado, este pode ser implementado manualmente ou gerado automaticamente. Na Figura 3 existem módulos responsáveis por gerar a aplicação que está definida no Metamodelo. O módulo “Gerador” pode gerar a aplicação tanto na plataforma Cliente/Servidor (“Gerador Cliente” e “Gerador Servidor”) quanto em três camadas (“Gerador 3 Camadas”). O módulo “Gerador Cliente” recebe as informações do “Gerador” e pode enviar para o módulo “Gerador PowerBuilder” ou “Gerador WEB”, conforme o caso. Se for escolhido o “Gerador PowerBuilder”, este recebe as informações abstratas sobre as telas e converte para a linguagem do PowerBuilder, gerando automaticamente a interface da aplicação.

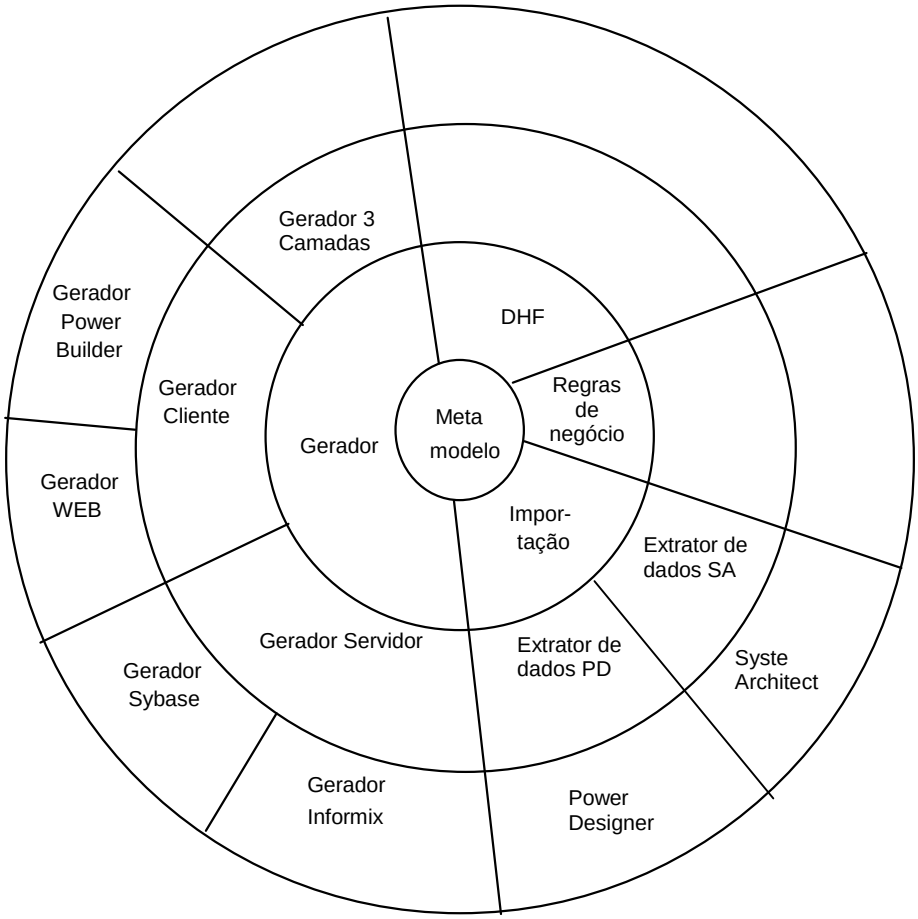

Figura 3 - Exemplo da estrutura de módulos

O desenvolvimento de sistemas utilizando a estrutura de módulos, permite que a especificação da aplicação seja realizada sem a preocupação constante da implementação nas ferramentas disponíveis e na situação tecnológica atual. As especificações são modeladas através de conceitos padronizados, utilizados nas ferramentas de modelagem. Os detalhes físicos não são incluídos nas definições, são convertidos pelos módulos respectivos que fazem a conexão entre o Metamodelo e o módulo onde a ferramenta está localizada.

Como a estrutura de módulos armazena as informações utilizadas no desenvolvimento de sistemas de maneira independente de formatos físicos, é possível a geração de código na linguagem desejada (desde que exista um módulo para isto). O metamodelo está preparado para armazenar as telas que serão apresentadas ao usuário final, as regras de negócio que controlam a aplicação (de maneira abstrata) e as especificações do Banco de Dados, desta forma, se existirem módulos para conversão do sistema para novas tecnologias, a especificação inicial da aplicação não é modificada. Isto pode ser comprovado utilizando como exemplo a utilização de vídeo e som na aplicação. A definição será a mesma, mas alguns tipos de dados como som e imagem serão incluídos nas telas (a partir das tabelas de conversão), permitindo a utilização destas características na aplicação. Estes detalhes de implementação (como deve ser apresentado um vídeo ao usuário ou como um som deve ser reproduzido) são implementados pelos módulos responsáveis pela geração da aplicação.

3 Metamodelo

O metamodelo permite a criação de uma grande base de conhecimento sobre as informações existentes na empresa [5]. O metamodelo armazena as informações de alguns modelos utilizados na especificação de sistemas, como: DFD, DER, DTE, DE, DHF, etc. A descrição dos processos do sistema é realizada através da descrição de Regras de Negócio, representadas por um diagrama. Este diagrama permite que as Regras sejam ativadas através de eventos (manuais ou automáticos).

As Regras de Negócio são especificadas em um formato estruturado [2] [3], tendo como base alguns tipos de regras existentes. A estrutura da definição é dividida em três partes: evento "QUANDO" (E), condição "SE" (C) e ação "ENTÃO-SENÃO" (A), como apresentado na Figura 4. A estrutura ECA permite melhor modularidade e maior possibilidade de reaproveitamento.

REGRA: Contato
QUANDO (*Recebe telefonema de uma pessoa*) OU (*Recebe uma carta*)
SE (*a pessoa for um cliente*)
ENTÃO(*atendimento personalizado*)
SENÃO(*cadastra atendimento*)

Figura 4 – Exemplo de especificação de Regras de Negócio

O repositório é implementado em um SGBD relacional, permitindo que o acesso seja livre para que qualquer ferramenta possa operar com os dados armazenados. O SGBD permite a criação de uma base de dados com acesso via ODBC, pois existe uma grande variedade de *drivers* já construídos. Com a utilização deste padrão, o acesso aos dados pode ser realizado a partir de vários tipos de aplicação, até mesmo de processadores de texto para emitir relatórios. O Diagrama ER do Metamodelo apresentado na Dissertação de Mestrado, é também demonstrado na Figura 5. O Diagrama Entidade-Relacionamento da Figura 5 possui informações relativas aos modelos (especificação) DFD, DTE, Regras de Negócio, DER e das Telas da aplicação.

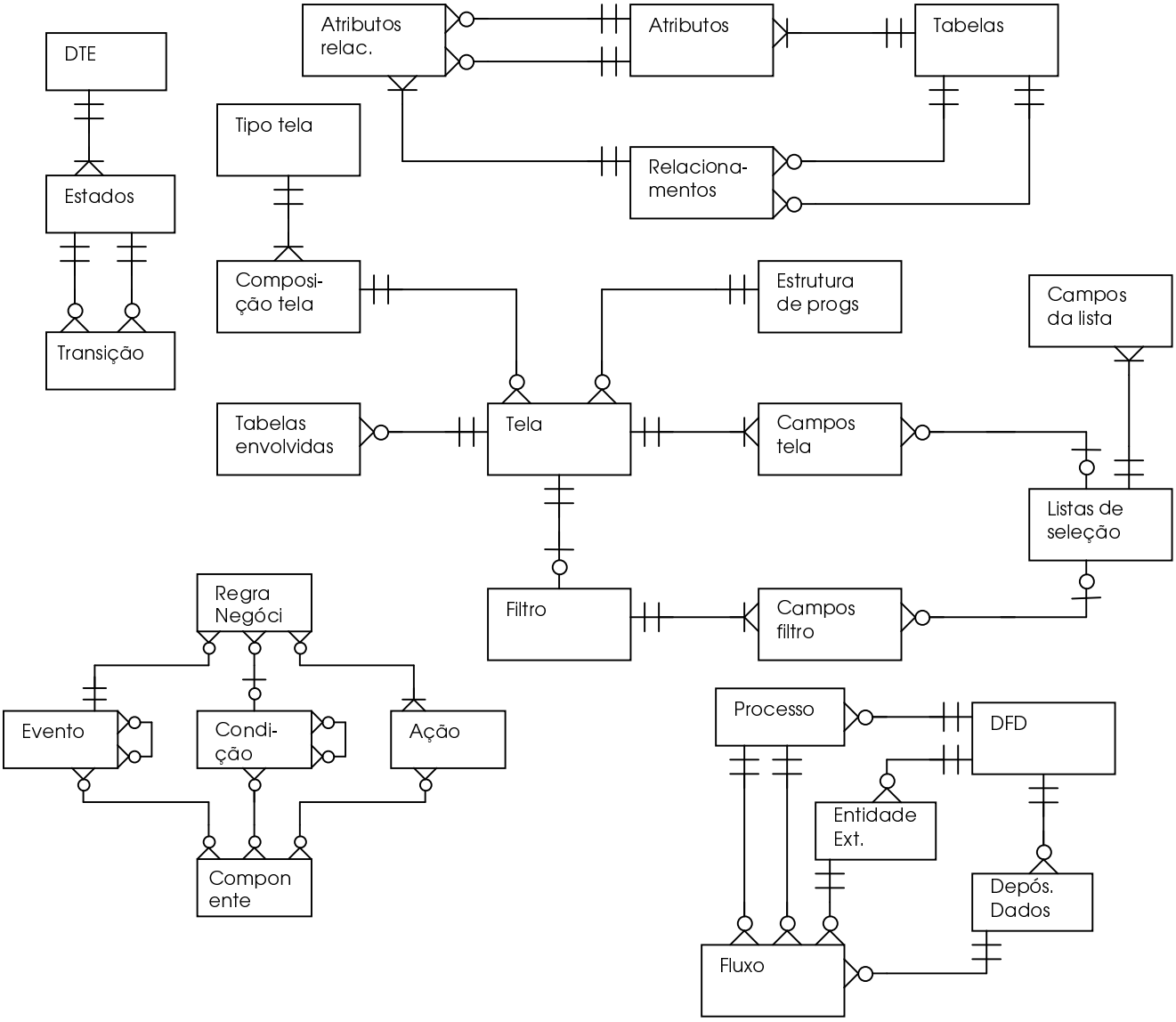

Figura 5 – Diagrama ER do Metamodelo

O Metamodelo apresentado na Figura 5 foi estruturado com base nos modelos utilizados pela Cumerlato & Schuster ao longo do desenvolvimento de um sistema. Entre os modelos de especificação armazenados no Metamodelo, três destes são peças chave para

armazenar o conhecimento do sistema. Os três modelos são os que armazenam informações relativas ao repositório de dados da aplicação (Banco de Dados), das funcionalidades (Regras de Negócio) e da interface (Telas). Com base nestes três modelos é possível gerar a aplicação para as tecnologias existentes atualmente.

Se for necessário utilizar uma nova tecnologia (ex. Banco de Dados Orientado a Objetos, N-Camadas, Bancos de Dados heterogêneos simultaneamente, etc.) e o Metamodelo não estiver preparado para isto, existem várias alternativas a serem seguidas, entre elas:

- Podem existir conversões pré-definidas, especificando que sempre que houver um determinado tipo de elemento, seja realizada uma conversão para outro
- O metamodelo pode ser estendido e os novos elementos necessários acrescentados (novas tabelas), associados às especificações existentes
- Novos atributos podem ser acrescentados ao Metamodelo, completando as informações necessárias
- Outros repositórios encontrados em ferramentas que integram a Estrutura de Módulos podem ser utilizados.

4 Estudo de caso

A Figura 5 representa o Diagrama ER utilizado na Dissertação de Mestrado. Para possibilitar a implementação em um Banco de Dados (com concorrência e transação) e a utilização efetiva da Estrutura de Módulos na Cumerlato & Schuster, foram necessários alguns ajustes, entre eles:

- Permitir o controle de vários projetos simultâneos

As tabelas do Metamodelo apresentadas na Dissertação aceitavam especificações de apenas um projeto de cada vez no Banco de Dados. Foi incluída uma tabela para armazenar os projetos existentes e o atributo correspondente ao código do projeto propagado para as tabelas necessárias.

- Divisão do projeto em módulos

Mesmo depois do Metamodelo aceitar a especificação de vários projetos simultâneos, foi necessária a divisão em módulos, permitindo armazenar grandes Sistemas de Informação (mais de 100 entidades). Desta forma, cada projeto do Metamodelo, é classificado internamente por módulo.

- Utilização opcional de *Folder* nas telas geradas

Algumas telas possuem muitos campos ou informações de várias origens, que precisam ser agrupadas e separadas. Assim, foi necessário que o Metamodelo possibilitasse o armazenamento de informações relativas ao relacionamento dos campos da tela com os *folders* existentes.

- Associação dos eventos de Interface com as Regras de Negócio

Alguns eventos de interface como inclusão, alteração, exclusão, saída/entrada da tela, saída de campo, execução de uma validação depois da inclusão, etc., ativam Regras de Negócio. Estas Regras podem estar implementadas em linguagens de programação diferentes ou ficarem localizadas em outros equipamentos. Desta forma, foi necessário que os eventos de interface fossem associados com as Regras de Negócio disponíveis.

- Associação dos campos de tela com os campos de tabela

No Metamodelo original os campos da tela especificada não estavam relacionados com o campo da tabela correspondente. Para manter a integridade do modelo e evitar redundância de informações, foram criados os relacionamentos.

- O Diagrama de Transição de Estados não está sendo utilizado

Os estados e eventos são definidos através de Regras de Negócio, desta forma as tabelas relativas ao DTE foram removidas do Metamodelo.

- Tabela de Domínios

Os atributos que possuem domínio discreto são especificados na Ferramenta CASE. Os valores possíveis de serem utilizados no atributo devem estar cadastrados no Metamodelo, pois quando for criada a tela pode-se optar por um *Radio Button* ou um *Check Box*, que precisa apresentar todos os valores disponíveis.

- Tabelas de Conversão (“De” – “Para”)

Algumas tabelas de conversão foram acrescentadas no Metamodelo. Uma das tabelas é utilizada para montar uma associação entre a Ferramenta CASE e a Linguagem de Programação, por exemplo: Um atributo especificado com o tipo " *integer*" na Ferramenta CASE, deve ser declarado com o tipo " *int*" no PowerBuilder. Esta tabela é utilizada somente quando o programa fonte está sendo criado (Módulo Gerador), para permitir a conversão de tipos de dados.

- Associação de parâmetros na chamada entre duas telas

A partir de uma tela, é possível chamar outra, passando as informações existentes como parâmetro. Para possibilitar a geração de código, foi necessário incluir uma tabela para armazenar os campos da tela chamadora que serão passados como parâmetro para a tela chamada.

- Inclusão de maior controle

Foi necessário incluir algumas regras de consistência no Metamodelo para garantir a integridade das informações, por exemplo: permitir a especificação de telas somente com tabelas que possuem relacionamentos diretos (entre si), exigir a que os campos da chave primária das tabelas estejam presentes nas telas e permitir a chamada de *Pick-List* somente se o campo tiver um relacionamento com a tabela correspondente.

- Regras de Negócio associadas com componentes

Várias Regras de Negócio podem ser agrupadas para formarem outra com maior funcionalidade. O Metamodelo foi alterado para permitir o agrupamento de Regras de Negócio.

- Controle de programação das Regras de Negócio

Como as Regras de Negócio ainda não estão sendo geradas automaticamente, foi mais produtivo criar tabelas para controlar o fluxo de implementação. Nestas tabelas são indicadas as Regras enviadas para a Programação, as datas de envio para implementação, de retorno e de testes.

A partir da inclusão dos itens citados acima no Metamodelo, foi possível construir alguns módulos para oferecer novas funcionalidades a serem disponibilizadas na Estrutura. O primeiro módulo construído foi o Importador, que recupera as informações da Ferramenta CASE e insere no Metamodelo. Logo depois foi construída uma ferramenta no estilo "Assistente" para a especificação das telas. Depois das telas especificadas, foi construído o gerador, que a partir das informações do Metamodelo converte para código fonte de uma linguagem de programação. A Ferramenta CASE utilizada foi o PowerDesigner. Os módulos para importar as informações, especificar e gerar programas foram construídos em Microsoft Visual Basic. O gerador foi construído inicialmente para PowerBuilder.

O Diagrama ER do Metamodelo que está sendo utilizado atualmente é composto por 48 tabelas e 347 atributos, enquanto no DER apresentado na Dissertação haviam 27 tabelas.

Os próximos passos serão a construção de três módulos para serem inseridos na Estrutura: uma ferramenta para especificar Regras de Negócio, um gerador para Regras de Negócio e um gerador para HTML.

5 Conclusão

O objetivo principal da estrutura de módulos, é permitir a especificação de sistemas de informação de maneira independente de tecnologia, com base em definições abstratas. Várias ferramentas, linguagens de programação e métodos podem ser integrados, produzindo o resultado final em menor tempo e com maior confiabilidade.

A estrutura de módulos permite que várias ferramentas (CASE, Linguagens de Programação, Ambientes de Desenvolvimento, Geradores de Código, Bancos de Dados, *Middleware*, etc.) utilizadas atualmente no desenvolvimento sejam integradas, para formarem um conjunto que seja aplicado durante todo o ciclo de vida do Sistema. Desta forma, as melhores características de cada ferramenta podem ser utilizadas, permitindo a integração de cada uma delas através do metamodelo. Isto é possível através da construção de um novo módulo, responsável pela integração da ferramenta com o metamodelo. A integração das ferramentas existentes permite que seja criado um ambiente para o desenvolvimento do sistema, podendo substituir cada um dos produtos utilizados conforme a necessidade da aplicação a ser construída.

Os módulos agregados ao metamodelo são independentes e autônomos. Isto significa que os módulos podem ser desenvolvidos por empresas diferentes e agregados ao mesmo metamodelo. A agregação de diferentes módulos é possível pela padronização dos parâmetros enviados e recebidos. Desta forma, antes da construção de um novo módulo, os parâmetros que devem ser recebidos e enviados já são conhecidos, facilitando o desenvolvimento e a agregação ao metamodelo.

As atividades de gerência de projeto e acompanhamento de tarefas podem ser incluídas na estrutura de módulos. Para isto, é necessário agregar módulos de gerenciamento ao metamodelo, possibilitando controlar e consistir as informações utilizadas durante o ciclo de vida do desenvolvimento de sistemas. Os módulos gerenciais podem controlar atividades como: auxílio aos procedimentos de documentação, padronização de especificação, controle das equipes de desenvolvimento e das tarefas realizadas, controle do cronograma, auxílio na documentação e dos ajustes no produto final e acompanhamento dos testes finais. Estes módulos podem ser integrados com processadores de texto, planilha de cálculo ou algum *software* utilizado para controle de cronograma (ex. Microsoft Project).

Com base em estudos realizados na Cumerlato & Schuster a utilização da Estrutura de Módulos, como apresentado na Figura 3, com módulos para especificação de programas (DHF) e para geração de código, otimiza o desenvolvimento de sistemas e aumenta a confiabilidade da aplicação. A Tabela 1 apresenta um estudo com tempos aproximados de especificação e implementação de sistemas. É apresentado também um comparativo de tempos utilizando e sem a utilização da Estrutura de Módulos.

As informações da Tabela 1 relativas às telas consideram um Programador com pelo menos 1 ano de experiência na ferramenta PowerBuilder e a existência de padrões e bibliotecas de objetos de interface. Para cálculo do tempo médio, cada programa é considerado com pelo menos 2 *Pick-Lists* e 1 Filtro de consulta. As informações da Tabela 1 relativas às Regras de Negócio, são divididas em três níveis de complexidade: simples (3 a 5 validações), médias (manutenção em tabela e mais de três validações) e complexas (várias manutenções em tabelas, cerca de 5 validações e processamentos).

Tabela 1

	Sem a Estrutura		Com a Estrutura	
Programa	Especificação	Programação	Especificação	Programação
Telas: 1 a 5 campos	30 min	36 min	1 min	10 seg
Telas: 6 a 10 campos	1 hora	51 min	2 min	10 seg
Telas: 11 a 15 campos	2 horas	72 min	3 min	10 seg
Telas: 16 a 20 campos	3 horas	90 min	4 min	10 seg
Regras de Negócio simples	1 a 2 horas	2 a 4 horas	1 horas	1 a 3 horas
Regras de Negócio médias	2 a 3 horas	5 a 12 horas	1,5 a 2 horas	4 a 10 horas
Regras de Negócio complexas	3 a 5 horas	12 a 30 horas	2 a 4 horas	8 a 20 horas

É importante observar que as diferenças entre as horas de programação também ocorrem porque a estrutura, além de agilizar o trabalho de especificação, reduz o volume de código e diminui os erros de implementação.

Com base na Tabela 1, uma aplicação com 60 Tabelas, 65 Telas (35 com 1 a 5 campos, 20 com 6 a 10 campos e 10 com 11 a 15 campos) e 40 Regras de Negócio (5 complexas, 5 médias e 30 simples), sem contabilizar as etapas de Planejamento e Análise, seria desenvolvida de acordo com o número de horas apresentado na Tabela 2.

Tabela 2

	Sem a Estrutura		Com a Estrutura	
Programa	Especificação	Programação	Especificação	Programação
Telas de 1 a 5 campos: 35	17,50	21,00	0,58	0,10
Telas de 6 a 10 campos: 20	20,00	17,00	0,67	0,06
Telas de 11 a 15 campos: 10	20,00	12,00	0,50	0,03
RN simples: 30	45,00	90,00	30,00	60,00
RN médias: 5	12,50	42,50	8,75	35,00
RN complexas: 5	20,00	105,00	15,00	70,00
Total	135,00	287,50	55,50	165,18
Total Geral		422,50		220,68

Como apresentado na Tabela 2, a aplicação seria construída em cerca de 220,68 horas (55,50 horas de especificação e 165,18 de programação), ao invés de 422,50 horas (135 horas de especificação e 287,5 de programação) da forma tradicional. De acordo com o número de programas e regras da aplicação, houve um ganho geral de 47,7%, 58,8% de especificação e 30% nas Regras de Negócio.

Com base nos resultados apresentados na Tabela 2 é possível concluir que a Estrutura de Módulos realmente oferece maior produtividade para o desenvolvimento de sistemas. A partir do momento que as Regras de Negócio também forem armazenadas de forma mais abstrata e houverem módulos para gerá-las, o tempo de desenvolvimento será reduzido ao mínimo necessário.

9 Referências

[1] APT Data Group. **Briefing Paper:** What is Metadata. [S.l.]: APT Data Group, 1996. Disponível por HTTP em http://www.computerwire. com/bulletinsuk/212e_1a6.htm (10/03/1997).

[2] HERBST, H. et al. **The Specification of Business Rules:** A Comparision of Selected Methodologies. Berne, Switzerland: Institute for Information Systems - University of Berne, 1994.

[3] HERBST, H. **Business Rules in System Analysis:** A Meta-Model and Repository System. Berne, Switzerland: Institute for Information Systems - University of Berne, 1996.

[4] INFORMIX INC. **Informix's Web DataBlade Module.** [S.l.]: Informix Inc., 1997. Disponível por HTTP em http://www.informix.com/informix /corpinfo/zines/tms0197/tms1h97.htm (10/05/1997).

[5] INFORMIX INC. **The Informix Datablate Technology.** [S.l.]: Informix Inc., 1997. Disponível por HTTP em http://www.informix.com/ informix/bussol/iusdb/databld/dbtech/techbro/db_intro.htm (11/05/1997).

[6] Oliveira, André L. C. **Metodologia para desenvolvimento de sistemas de informação através da utilização de módulos autônomos**. Porto Alegre. CPGCC da UFRGS, 1998. Dissertação de Mestrado.

[7] ORACLE INC. **Oracle Designer - User Guide.** [S.l.]: Oracle Inc., 1997.

[8] SHIMBERG, David. Following a Client/Server Database Tecnhology. **DBMS Magazine**, [S.l.], v. 8, p. 48, May 1995.
[9] SPERTUS, Michael. Programming Automation. **Object Currents Magazine**, [S.l.], Jan. 1996.
[10] SPENCER, Kenneth L.; MILLER, Ken. **Client/Server Programming with Microsoft Visual Basic.** Redmond, Washington: Microsoft Press, 1996.
[11] STANDISH GROUP. **Chaos.** [S.l.:s.n.], 1995. Disponível por HTTP em http://www.standishgroup.com/chaos.html.
[12] TUCKER, Michael. Lords of the jungle: DB middleware helps you find your data. **Datamation**, [S.l], Sept. 1997.
[13] KNOWLES, Anne. A database with a head for business **Datamation**, [S.l], Sept. 1997.