6
Conclusão

O ambiente HyperDE+DR foi desenvolvido com o objetivo de apoiar a captura e uso de *design rationale* durante a construção de aplicações hipermídia. O ambiente permite o registro das soluções utilizadas pelo usuário ao construir sua aplicação, possibilitando uma posterior visualização das mesmas. A recuperação do *design rationale* registrado facilita o entendimento da modelagem da aplicação, pois permite ver as questões sugeridas por cada idéia adotada, bem como os argumentos contra ou a favor desta idéia e a justificativa para cada escolha feita.

Um outro aspecto interessante do ambiente HyperDE+DR é a possibilidade de uso do *design rationale* de aplicações previamente construídas. O usuário pode selecionar os elementos de *design rationale* de uma aplicação existente e integrá-los a sua própria aplicação, enriquecendo seu modelo e reduzindo o esforço despendido.

A visualização e utilização do *design rationale* facilita a modelagem de diferentes aplicações baseadas no mesmo modelo formal e de um mesmo domínio. Espera-se com o ambiente HyperDE+DR prover uma maneira mais completa e fácil de construir aplicações baseadas nos métodos OOHDM e SHDM.

Nas seções seguintes são apresentados: (i) uma estimativa do esforço empregado para a extensão do ambiente HyperDE, (ii) as contribuições desta dissertação, (iii) alguns trabalhos relacionados existentes e (iv) algumas propostas de trabalhos futuros a serem desenvolvidos na mesma linha de pesquisa desta dissertação.

6.1.
Estimativa de esforço

Nesta seção é apresentada uma estimativa do esforço despendido para a realização das extensões que resultaram no ambiente HyperDE+DR. A estimativa é realizada através da comparação do tamanho dos dois ambientes,

HyperDE e HyperDE+DR. Algumas características dos dois ambientes são estimadas e comparadas, para ilustrar o esforço realizado.

Os arquivos de cada ambiente foram agrupados em pacotes, sendo os mais expressivos listados abaixo. Os pacotes correspondem às três camadas do framework MVC que constituem o ambiente.

- controllers: esse pacote contém os arquivos referentes às classes de controle do ambiente.
- models: contém os arquivos referentes às classes de modelo do ambiente.
- views: contém os arquivos referentes à camada de visão do ambiente, responsáveis pela apresentação das informações (interface).
- helpers: contém os arquivos que possuem funções auxiliares à camada de visão do ambiente.

Para cada pacote de cada ambiente foram estimados o número de arquivos e o número de linhas de código. Os resultados encontram-se na tabela a seguir.

Pacote	**Número de Arquivos**		**Linhas de Código**	
	HyperDE	HyperDE+DR	HyperDE	HyperDE+DR
controllers	12	17	1085	4014
models	24	31	1263	1764
views	32	65	688	1783
helpers	12	14	784	1264

Tabela 14 – Comparação entre estimativas do HyperDE e HyperDE+DR

Pode-se verificar que foram realizadas muitas alterações no código, resultando no acréscimo de cerca de três vezes o número original de linhas de código e no acréscimo de cerca de 50 arquivos.

Este esforço foi necessário para que o ambiente refletisse a alteração realizada em seu metamodelo, ao ser incorporado o metamodelo da abordagem Kuaba. Novas classes foram criadas, e várias alterações na interface foram

necessárias para permitir a captura e visualização do *design rationale* das aplicações.

As classes de modelo e de controle existentes precisaram ser modificadas para tratarem as questões relativas à captura e recuperação de *design rationale*. Um exemplo expressivo é o aumento do número de linhas de código das classes de controle, que praticamente quadruplicaram.

6.2. Contribuições

A principal contribuição do trabalho apresentado é o registro, recuperação e utilização das razões por trás das decisões tomadas por um projetista/desenvolvedor ao construir uma aplicação. No ambiente HyperDE+DR o registro do raciocínio é feito à medida que o usuário desenvolve a aplicação, ou seja, à medida que ele insere no ambiente os artefatos que compõem a aplicação. Desta maneira o projetista/desenvolvedor pode registrar as justificativas e argumentos que sustentam suas decisões de *design* e, conseqüentemente, as razões para a implementação dos artefatos que compõem a aplicação final.

O desenvolvimento do ambiente HyperDE+DR foi realizado através da extensão do ambiente HyperDE (Nunes, 2005). A extensão envolveu o entendimento do metamodelo do ambiente HyperDE e a integração dos conceitos apresentados na abordagem Kuaba (Medeiros, 2006). Para isso ser possível, foi necessária a união do metamodelo referente à abordagem Kuaba com o metamodelo existente no ambiente HyperDE. Como resultado dessa união foi possível operacionalizar o trabalho apresentado em Medeiros (2006), pois o algoritmo proposto foi integrado a um ambiente de desenvolvimento baseado em modelos, o que permitiu o uso do método OOHDM para guiar a geração automática de idéias e questões sugeridas.

A abordagem adotada reduz o esforço requerido do usuário para o registro de seu raciocínio. Sua interferência é necessária apenas para informar questões de *design* que sejam relacionadas ao domínio da aplicação sendo desenvolvida. As questões de *design* relacionadas ao modelo formal adotado são geradas automaticamente pelo ambiente, interrompendo minimamente as atividades do projetista/desenvolvedor.

A posterior recuperação e visualização do *design rationale* da aplicação fornece ao usuário uma visão abrangente do raciocínio utilizado, o que pode

ajudá-lo a entender questões previamente solucionadas e a melhorar soluções anteriormente realizadas. A visualização e entendimento do *design rationale* de uma aplicação facilita sua manutenção e evolução, além de permitir o reuso de maneira eficiente das soluções desenvolvidas para a construção de novas aplicações.

Espera-se com este trabalho difundir os benefícios do registro e utilização do *design rationale* através da disponibilização de um ambiente de fácil entendimento e utilização. O HyperDE+DR oferece informações completas e precisas sobre as decisões tomadas nas aplicações desenvolvidas no ambiente, provendo meios para que estas informações sejam utilizadas de maneira a facilitar e enriquecer o desenvolvimento de novas aplicações.

6.3. Trabalhos Relacionados

Após a difusão do termo *design rationale* e dos possíveis benefícios de seu registro e recuperação, alguns pesquisadores desenvolveram ferramentas capazes de registrar e organizar essa informação e de torná-la disponível para acessos posteriores. Algumas ferramentas desenvolvidas nessa linha foram:

- SEURAT *(Software Engineering Using RATionale)*: é um sistema que oferece suporte à visualização e à inferência sobre o *rationale*, para indicar quaisquer problemas não resolvidos ou inconsistências (Burge & Brown, 2004). O sistema SEURAT é integrado a um ambiente de desenvolvimento de software, mas ainda assim a captura do *design rationale* é manual, ou seja, o usuário precisa informar explicitamente o raciocínio utilizado ao construir a aplicação. Outra diferença em relação ao ambiente HyperDE+DR é que o sistema SEURAT não oferece operações computáveis para apoiar o uso de *design rationale*.
- DREAM *(Design Rationale Environment for Argumentation and Modelling)*: é uma ferramenta case para a edição, análise e exploração de modelos baseados na notação TEAM. TEAM é uma extensão à notação QOC, introduzida por McLean et al. (1996) e que facilita o entendimento e visualização do *design rationale* capturado, mantendo a utilidade do mesmo em termos de estrutura (Lacaze et al., 2005). DREAM oferece suporte à verificação de

elementos não respondidos e inconsistências, argumentação e rastreamento dos modelos ao longo dos projetos e reuso dos modelos em projetos futuros (Lacaze et al., 2006). DREAM não é integrado a uma ferramenta de desenvolvimento de software, e as informações precisam ser inseridas manualmente pelo usuário.

- Compendium: é uma ferramenta hipermídia para a modelagem semi-formal e colaborativa. É baseado na abordagem Compendium, que surgiu como uma extensão às funcionalidades de QuestMap™ (Conklin, 1999) e gIBIS (Conklin & Begeman, 1988). As funcionalidades da ferramenta podem contribuir para qualquer atividade ou fase de engenharia de software onde seja necessária uma deliberação ou modelagem baseada em argumentos (Shum et al., 2005). Entretanto, Compendium está relacionado primariamente ao registro em tempo real de discussões em reuniões e a oferecer suporte ao trabalho colaborativo, não sendo integrado a nenhuma ferramenta de desenvolvimento de software, e exigindo a inserção manual dos elementos de raciocínio pelos usuários. Por ser genérico, Compendium não oferece suporte específico à manipulação de *design rationale*, sendo comumente utilizado para registrar idéias, soluções e questionamentos dos usuários.

Entretanto, apesar de vários esforços terem sido realizados no sentido de oferecer registro, organização, verificação e recuperação de *design rationale*, não há ferramentas disponíveis que realmente sejam integradas a ambientes de desenvolvimento, permitindo a captura semi-automática das informações e possibilitando o seu reuso.

6.4. Trabalhos Futuros

Nesta seção são apresentadas algumas sugestões de trabalhos a serem desenvolvidos como decorrência do trabalho proposto nesta dissertação.

6.4.1. Importação de repositórios externos

Na versão atual do ambiente HyperDE+DR, os repositórios exportados por ferramentas externas não podem ser importados por aplicações desenvolvidas no ambiente. Por exemplo, as aplicações geradas pelo ambiente HyperDE não podem ser importadas pelo HyperDE+DR, pois não contêm as informações referentes ao *design rationale* necessárias para a sua correta interpretação.

Uma primeira sugestão de trabalho seria tornar os repositórios gerados pelo ambiente HyperDE compatíveis com o ambiente HyperDE+DR. Para que isso ocorra será necessário um procedimento de geração automática do *design rationale* das aplicações previamente desenvolvidas no HyperDE. As idéias, questões sugeridas e decisões devem ser criadas automaticamente durante a conversão, para que os repositórios exportados pelo HyperDE possam ser interpretados pelo ambiente HyperDE+DR.

Este procedimento pode ser feito usando os artefatos da aplicação sendo convertida para criar as idéias consideradas e, com base no método formal utilizado, sugerir questões e respostas para essas questões, criando decisões entre elas.

6.4.2. Análise automática do *design rationale* capturado

Uma funcionalidade a ser integrada ao ambiente é a análise automática do *design rationale* registrado de uma aplicação. A análise seria muito útil para detectar possíveis inconsistências no raciocínio capturado, auxiliando o projetista/desenvolvedor a avaliar seu *design rationale*.

O desenvolvimento desta funcionalidade permitiria ao ambiente verificar automaticamente o raciocínio registrado, alertando o projetista/desenvolvedor sobre a existência de elementos em não conformidade com o método de *design* utilizado. Um possível aviso de inconsistência seria uma questão do tipo XOR com mais de uma idéia aceita como resposta. Um possível aviso de alerta seria uma questão que não possua nenhuma resposta, como por exemplo, uma classe navegacional que não possua atributos.

6.4.3. Geração de relatórios

Com o objetivo de apoiar a funcionalidade descrita na seção 6.3.2, seria de grande utilidade que o ambiente produzisse relatórios das análises dos *design rationales* registrados. Desta maneira o usuário poderia receber uma lista dos erros e inconsistências em seu raciocínio, facilitando a identificação e resolução dos mesmos. Exemplos de itens que poderiam constar nos relatórios incluem: questões sem resposta, decisões sem justificativa, idéias e decisões sem argumentos, entre outros.

Outra alternativa extremamente útil seria a representação gráfica dos resultados da análise do design. A visualização dos avisos de alerta e erro seriam muito facilitadas através de ícones indicativos, como por exemplo um X vermelho para erros e um sinal de exclamação amarelo para os alertas.

6.4.4. Registro de soluções não aceitas

Uma limitação do ambiente é a impossibilidade de registro das soluções não aceitas pelo projetista/desenvolvedor. O HyperDE+DR permite apenas o registro das idéias de solução aceitas pelo usuário. Um trabalho futuro proposto seria a extensão do ambiente HyperDE+DR para que este permita a captura de todo o *design rationale* utilizado, e não somente as idéias finais aceitas pelo projetista/desenvolvedor. Tal extensão proveria uma maior flexibilidade e abrangência à ferramenta.

6.4.5. Sugestão de argumentos e justificativas

Os elementos do *design rationale* que são externos ao método de *design* adotado devem ser inseridos manualmente pelo projetista/desenvolvedor. Isto significa que elementos diretamente relacionados com o domínio da aplicação, como argumentos e justificativas, necessitam da intervenção do usuário para serem registradas.

Uma maneira de tentar reduzir o esforço do usuário é permitir que o ambiente sugira alguns dos argumentos e justificativas dos artefatos criados, utilizando como base o método OOHDM. Por exemplo, pode ser criada uma base de argumentos pré-definidos para determinados elementos específicos do método de *design*, como índices, classes e contextos. Alguns elementos, tais

como os tipos dos atributos de índices e de classes navegacionais, também poderiam ter seus argumentos e justificativas sugeridos ou preenchidos automaticamente pelo ambiente.

Estes conjuntos de argumentos pré-definidos foram chamados de *rationale patterns* por Burge & Brown (2000), ou seja, os padrões de *rationale* consistem em conjuntos de argumentos que freqüentemente aparecem juntos ao avaliar determinada alternativa.

Uma proposta semelhante foi realizada no sistema SEURAT, através da representação RATSpeak (Burge & Brown, 2004). Nessa representação é definida uma ontologia de argumentos que consiste em uma hierarquia de razões contra e favor das alternativas de *design*. Este tipo de abordagem hierárquica para a definição e classificação de argumentos poderia ser utilizada para auxiliar a construção de uma especificação de argumentos pré-definidos para o ambiente HyperDE+DR.

6.4.6. Disponibilização de padrões de *design*

Durante a modelagem de aplicações segundo os métodos OOHDM e SHDM existem soluções de *design* que são recorrentes, ou seja, comumente utilizadas. Com base na avaliação dessas soluções é possível criar um conjunto de soluções pré-definidas para problemas de *design* conhecidos. Tais 'pacotes' de soluções constituiriam padrões de *design*, a serem disponibilizados no ambiente para auxiliar a construção de novas aplicações.

Um exemplo de solução de *design* comumente utilizada é o acesso de elementos de um conjunto conceitual da aplicação através de índice. Esta solução consiste na definição de um contexto ordenando os itens do conjunto conceitual alfabeticamente. A partir deste contexto é definido um índice, que serve para que o usuário acesse as informações desejadas. A solução está ilustrada no esquema de contexto de navegação apresentado na figura a seguir.

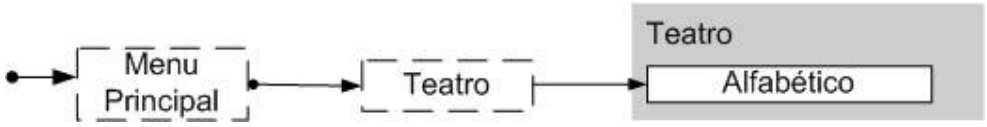

Figura 58 – Exemplo de padrão de *design*

No exemplo ilustrado, o padrão de solução foi adotado para prover acesso aos elementos da classe Teatro, ordenados alfabeticamente, através de índice baseado em contexto.

6.4.7. Criação das questões sugeridas por análise automática do metamodelo

Na versão atual do HyperDE+DR as questão são sugeridas com base no método OOHDM, mas não são geradas automaticamente por meio de análise do metamodelo do ambiente. Em outras palavras, atualmente as questões sugeridas estão diretamente associadas ao método OOHDM. Caso essas questões fossem geradas automaticamente por meio de uma varredura do metamodelo, poder-se-ia trocar o metamodelo que o mecanismo continuaria funcionamento perfeitamente.

Exemplo: temos que o metamodelo do ambiente HyperDE+DR, descrito sucintamente na figura 9, mostra algumas relações entre seus elementos. Por meio do conhecimento dessas relações pode-se gerar automaticamente questões sugeridas para artefatos criados no ambiente. Sabe-se que um artefato do tipo índice pode ter duas formas: baseado em contexto e baseado em query (consulta). Sendo assim, por varredura automática do metamodelo o ambiente pode sugerir para uma idéia correspondente a um índice as perguntas “Qual a query do índice?” ou “Qual o contexto do índice?”. O mesmo ocorreria para um outro metamodelo, como por exemplo o metamodelo do UML. Nesse caso, ao ser criado um atributo, poderia ser derivado de seu metamodelo as suas respectivas questões sugeridas, tais como: “A que classe pertence?”, “Qual a multiplicidade mínima?” ou “Qual a multiplicidade máxima?”.

Esta independência de metamodelos permitiria uma flexibilidade do ambiente em um alto nível de abstração.