Uso de Metamodelos para apoiar o Registro semi-automático de *Design Rationale* em Projetos de Software

Thiago Ribeiro Nunes[1,3], Adriana Pereira de Medeiros[2]

[1] Universidade Candido Mendes – Campos dos Goytacazes, RJ – Brasil

[2] Departamento de Ciência e Tecnologia
Universidade Federal Fluminense (UFF) - Rio das Ostras, RJ - Brazil

{thiagorinu,adrianapm26}@gmail.com

Abstract. *This paper presents an extension of the ArgoUML software design tool, which implements the usage of metamodels proposed by the Kuaba approach to support the semi-automated design rationale recording in software design. Kuaba approach integrates its ontology with the design metamodels' formal semantics for representing rationale. This extension allows the designers record rationale while designing their artifacts and making their design decisions, automatically generating part of the design rationale representation from the meta-model. It reduces the extra effort required from them to record rationale during the software design and prevents the knowledge invested in the design from being lost.*

Resumo. *Este artigo apresenta uma extensão da ferramenta de design de software ArgoUML, que implementa o uso de metamodelos proposto pela abordagem Kuaba para apoiar o registro semi-automático de design rationale em projetos de software. A abordagem Kuaba integra sua ontologia com a semântica formal dos metamodelos para representar rationale. Esta extensão permite que os projetistas registrem seu conhecimento enquanto projetam seus artefatos e tomam suas decisões de design, gerando automaticamente parte da representação de design rationale a partir do metamodelo utilizado. Isto reduz o esforço requerido dos projetistas para registrar design rationale durante o design de software e evita que esse conhecimento seja perdido.*

1. Introdução

*Design Rationa*le (DR) é uma abordagem, proposta a partir do trabalho de Kunz e Rittel (1970) sobre o sistema IBIS (*Issue-Based Information System*), para descrever as razões por trás das decisões tomadas sobre o design de um artefato. Esta abordagem permite registrar as alternativas de solução avaliadas durante o design, o raciocínio usado para a escolha de uma dessas alternativas, as decisões tomadas e as dependências existentes entre elas [Lee 1997].

[3] Apoio parcial através de bolsa de iniciação científica da FAPERJ.

Os registros de DR têm grande importância em diversas tarefas de Engenharia de Software. Eles podem ser usados para melhorar a compreensão do desenvolvimento de um software, facilitando sua manutenção, sua evolução e o seu reuso em novos projetos. Por tornar os principais elementos de decisão explícitos, DR facilita a colaboração entre os engenheiros de software permitindo a análise das possíveis soluções para um determinado projeto, apontando as vantagens e desvantagens de cada solução considerada. Além disso, a captura de DR permite aos projetistas tornar seu conhecimento explícito e, mais tarde, examinar as justificativas de certas decisões de projeto como, por exemplo, na modificação de um sistema como conseqüência da evolução dos requisitos ou da disponibilidade de uma nova tecnologia. Isto é fundamental, uma vez que em muitos casos, após um longo período de tempo, os próprios engenheiros de software podem não se lembrar de todo o raciocínio que eles mesmos usaram em um determinado projeto.

Muitas abordagens já foram propostas para a representação de DR, como IBIS, QOC [MacLean et al. 1991], DRL [Lee 1991] e TEAM [Lacaze 2005]. No entanto, o conteúdo das representações de DR geradas a partir dessas abordagens é geralmente informal, o que impede o processamento computacional desse tipo de conhecimento. Além disso, quando essas abordagens são usadas no design de artefatos definidos formalmente (p. ex. artefatos de software), não é possível incorporar automaticamente as opções de design prescritas pelos métodos de design, assim como suas restrições. Ou seja, não é possível aproveitar a semântica formal do artefato fornecida pelo metamodelo que o descreve. Visando minimizar esse problema e permitir um uso mais efetivo de DR em designs baseados em modelos, foi proposta a abordagem de representação Kuaba [Medeiros 2006]. Nessa abordagem, a semântica formal do metamodelo usado no design dos artefatos é aproveitada na representação de DR. Isto permite a automatização de parte do processo de captura de DR e a realização de operações computáveis para apoiar seu uso no design de novos artefatos.

Apesar das várias ferramentas disponíveis para apoiar DR como, por exemplo, Compendium [Conklin et al. 2003], DREAM [Lacaze 2005], SEURAT [Burge e Brown 2004] e MVCASE [Paiva et al. 2006], DR não tem sido muito utilizado em projetos de software. As principais dificuldades encontradas para a captura e uso desse tipo de conhecimento são o consumo de tempo e o conseqüente aumento nos custos do projeto. As ferramentas existentes são, em geral, específicas para a captura de DR. Nelas, os usuários precisam registrar, manualmente, todas as informações de DR (problemas de design, as alternativas de solução para cada problema, os argumentos contra e a favor de cada alternativa, as decisões e suas justificativas). Isto gera um grande esforço adicional ao trabalho realizado pelos projetistas e reduz a aceitação deste tipo de ferramenta.

Este artigo apresenta uma extensão da ferramenta de design de software ArgoUML [Tolke 2007], que implementa o uso de metamodelos proposto pela abordagem Kuaba para apoiar o registro semi-automático de DR em projetos de software. A ferramenta estendida é parte da arquitetura conceitual do ambiente de design integrado proposto em [Medeiros 2006].

O uso do metamodelo de design proposto pela abordagem Kuaba permite que grande parte das estruturas de raciocínio e de decisão que compõem o DR de um artefato seja capturada e representada de forma automática, a partir da semântica descrita nesse metamodelo. Desta forma, o esforço requerido dos projetistas para

registrar este tipo de conhecimento é reduzido, uma vez que eles precisam informar apenas os argumentos para as soluções de design consideradas e as justificativas para as decisões tomadas. A grande vantagem é que os projetistas poderão fazer esse registro exatamente no momento em que as soluções estão sendo pensadas e as decisões tomadas, evitando, desta forma, que o conhecimento empregado no design do artefato seja perdido. Além disso, o trabalho extra para registrar este conhecimento usando uma ferramenta específica para DR é eliminado.

A seção 2 descreve brevemente os fundamentos da abordagem Kuaba. A seção 3 descreve a extensão da ferramenta de design de software ArgoUML para implementar essa abordagem, detalhando sua arquitetura e funcionamento. Os testes realizados e os resultados obtidos são apresentados na seção 4. A seção 5 apresenta alguns trabalhos relacionados. Por fim, as conclusões e alguns trabalhos futuros são apresentados.

2 A Abordagem de Representação Kuaba

Kuaba é uma abordagem baseada em argumentação para representação de DR. Seu principal objetivo é permitir o processamento computacional de DR para apoiar o reuso de designs baseados em modelo, particularmente, designs de software. A abordagem Kuaba difere das outras abordagens para DR propostas na literatura pelo fato de aproveitar a semântica formal fornecida pelos metamodelos de design para instanciar os elementos do seu modelo de representação, descrito na ontologia Kuaba [Medeiros et al. 2005]. O uso dessa semântica formal permite fornecer um suporte mais automatizado para o registro de DR, uma vez que grande parte da estrutura de DR pode ser obtida através da análise do metamodelo utilizado. Ele também define um padrão para a representação de DR usando a ontologia Kuaba, permitindo que os DRs gerados possam ser processados e combinados de forma automática para apoiar o design de novos artefatos.

A ontologia Kuaba descreve um modelo de representação de conhecimento para o domínio de DR. Ela provê um vocabulário para descrever DR e define um conjunto de regras que permitem a realização de inferências e operações computáveis para apoiar o uso do DR registrado. A ontologia Kuaba está descrita em duas linguagens formais de especificação diferentes, F-logic [Kifer e Lausen 1989] e OWL [W3C 2004].

Os elementos do vocabulário da ontologia Kuaba são apresentados de forma simplificada na Figura 1, usando uma notação gráfica no estilo da UML para auxiliar sua visualização. Algumas relações e restrições não são apresentadas para simplificar a apresentação. Neste vocabulário, os elementos de raciocínio representam as questões de design que surgem durante a elaboração do artefato, as possíveis idéias de solução para estas questões e os argumentos contra ou a favor das idéias apresentadas. O elemento decisão indica se uma determinada idéia foi aceita ou rejeitada como solução para uma questão e deve estar associada a uma justificativa final, sempre baseada nos argumentos associados à idéia. O elemento artefato representa o resultado de uma idéia de solução aceita para uma questão de design e pode ser um artefato atômico ou composto. O elemento método representa o método ou processo de design utilizado na modelagem dos artefatos. Por fim, o elemento modelo formal representa o metamodelo do método de design, ou linguagem de modelagem, utilizado para descrever os artefatos.

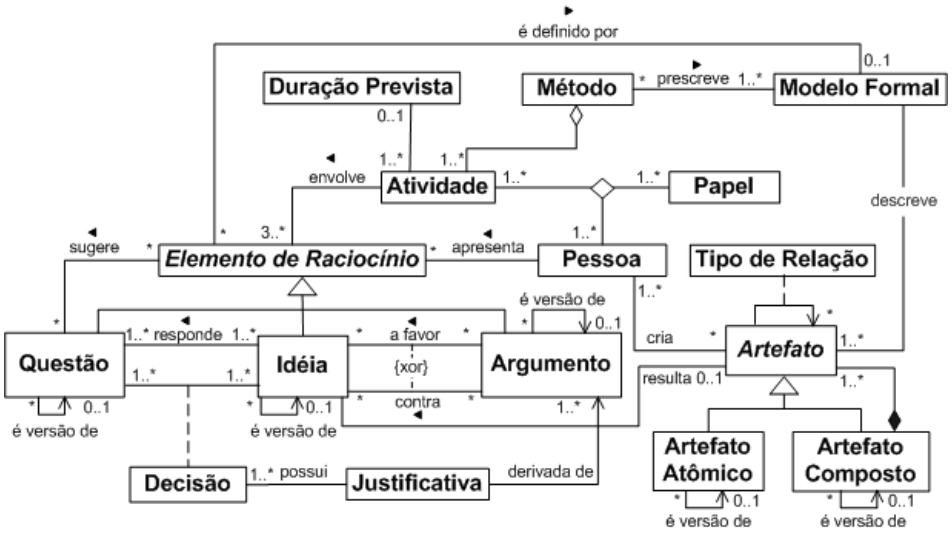

Figura 1. Elementos do vocabulário da ontologia Kuaba

A representação de DR para um artefato normalmente começa com uma questão genérica que estabelece o problema de design a ser resolvido. Esta questão genérica pode dar origem a novas questões que representam novos problemas de design relacionados ao problema principal. Para cada questão apresentada, os projetistas podem sugerir idéias, formulando possíveis soluções para o problema descrito na questão. Argumentos são apresentados contra ou a favor das idéias apresentadas e decisões são tomadas com base nesses argumentos. A Figura 2 mostra um exemplo de representação de DR utilizando a abordagem Kuaba para um modelo de domínio de um catálogo de CD, representado como um diagrama de classes da UML.

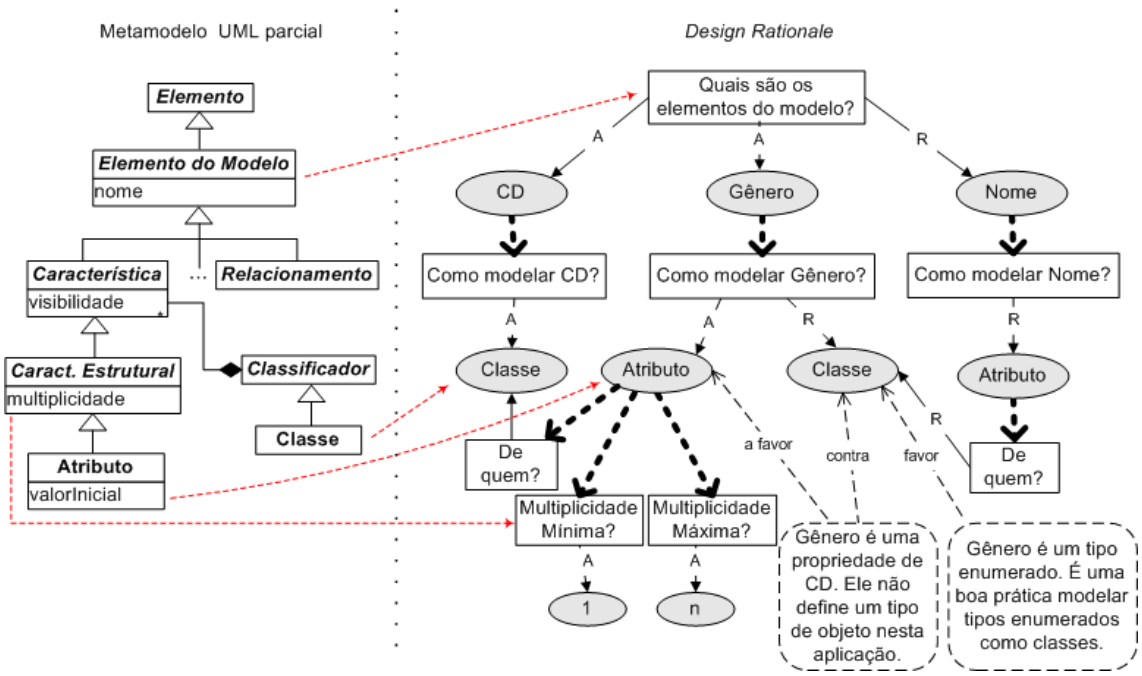

Figura 2. Exemplo de representação de DR usando a abordagem Kuaba e o metamodelo da UML para diagramas de classes

Este exemplo mostra como a semântica do metamodelo da UML é utilizada na representação de DR para instanciar os elementos da ontologia Kuaba. De acordo com esse metamodelo, o primeiro problema a ser resolvido no design de um diagrama de classes é a identificação dos elementos do modelo (metaclasse *Elemento do Modelo*). Aplicando o vocabulário da ontologia Kuaba, o resultado é a criação do elemento *"Questão"* (representado como um retângulo) com a instância *"Quais são os elementos do modelo?"*. Essa questão inicial é respondida pelas idéias *"CD"*, *"Gênero"* e *"Nome"*, representadas como elipses. Note que esses valores são determinados pelo conhecimento do projetista sobre o domínio, ou são extraídos do DR da fase de levantamento de requisitos anterior ao design. Essas idéias de domínio, por sua vez, sugerem as questões de design *"Como modelar CD?"*, *"Como modelar Gênero"* e *"Como modelar Nome?"*. As possíveis idéias de design que respondem essas questões também são determinadas pelo metamodelo da UML. Por questões de simplicidade, apenas as idéias de design *"Classe"* e *"Atributo"* foram consideradas. Da mesma forma, as questões *"Multiplicidade Mínima?"* e *"Multiplicidade Máxima?"* associadas à idéia *"Atributo"* também são instanciadas de acordo com o metamodelo da UML.

Dessa forma, é possível visualizar como a semântica formal do metamodelo é usada na abordagem Kuaba para automaticamente gerar parte da representação de DR (questões e idéias de design). Assim, os projetistas têm apenas o trabalho de registrar os argumentos (retângulos pontilhados na Figura 2) contra e a favor de cada idéia de solução e as razões para as decisões tomadas. Na Figura 2, as decisões do projetista são ilustradas com os rótulos *"A"* (para aceita) ou *"R"* (para rejeitada) nas setas entre questões e idéias.

3 Implementando a Abordagem Kuaba

Para apoiar o registro de DR usando a abordagem Kuaba, um ambiente de design integrado foi proposto em [Medeiros 2006] e discutido em [Medeiros e Schwabe 2008]. Esse ambiente tem como objetivo tornar a captura, a representação e o uso de DR parte do processo de design e, assim, reduzir o esforço requerido dos projetistas para registrar esse conhecimento. A Figura 3 mostra os principais componentes do ambiente proposto. Desses componentes, apenas o processador de *rationale* foi implementado em [Medeiros 2006]. Esse processador é responsável pelo processamento computacional de DR para apoiar o reuso de designs baseados em modelo.

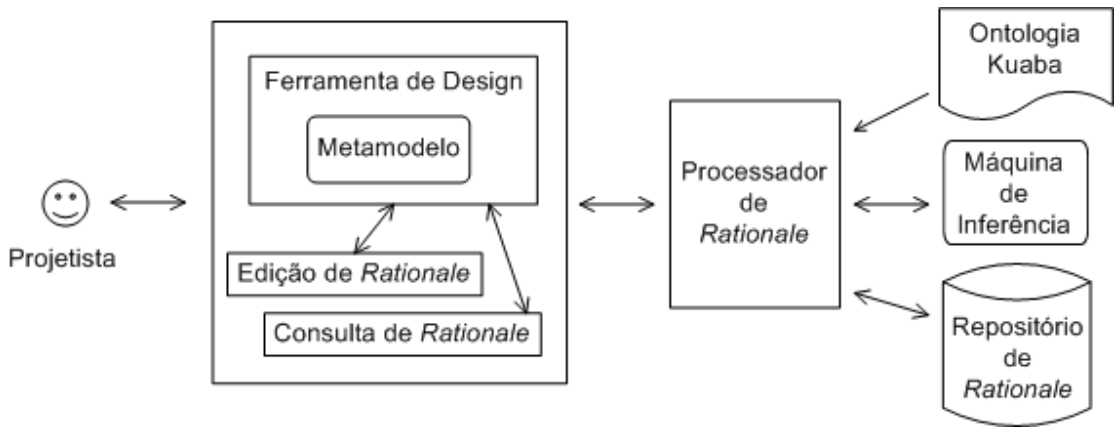

Figura 3. Componentes de um ambiente de design integrado para apoiar o uso da abordagem Kuaba na representação de DR

Como a maioria das ferramentas de design usa algum tipo de descrição formal (metamodelo) para os artefatos sendo projetados, a arquitetura conceitual mostrada na Figura 3 propõe a extensão de uma dessas ferramentas para permitir a captura, representação e processamento de DR usando a abordagem Kuaba. Esta extensão enriquece as ferramentas de design implementando a captura e representação semi-automática de DR a partir do metamodelo utilizado. Além disso, a extensão também inclui a implementação de duas camadas para apoiar a edição e a consulta de DR durante o design de um artefato. Na camada de edição, o projetista informa os argumentos contra ou a favor das idéias de solução consideradas e as justificativas para as decisões tomadas. Na camada de consulta, o projetista pode buscar designs existentes com suas respectivas representações de DR, formular questões sobre os designs encontrados e solicitar a integração dessas respresentações para iniciar um novo design. Nesta camada, o projetista também pode visualizar graficamente o DR do artefato que está sendo projetado, ou dos designs que estão sendo reusados em seu design.

A Figura 4 mostra a arquitetura definida neste trabalho para a completa implementação do ambiente de design integrado proposto em [Medeiros 2006].

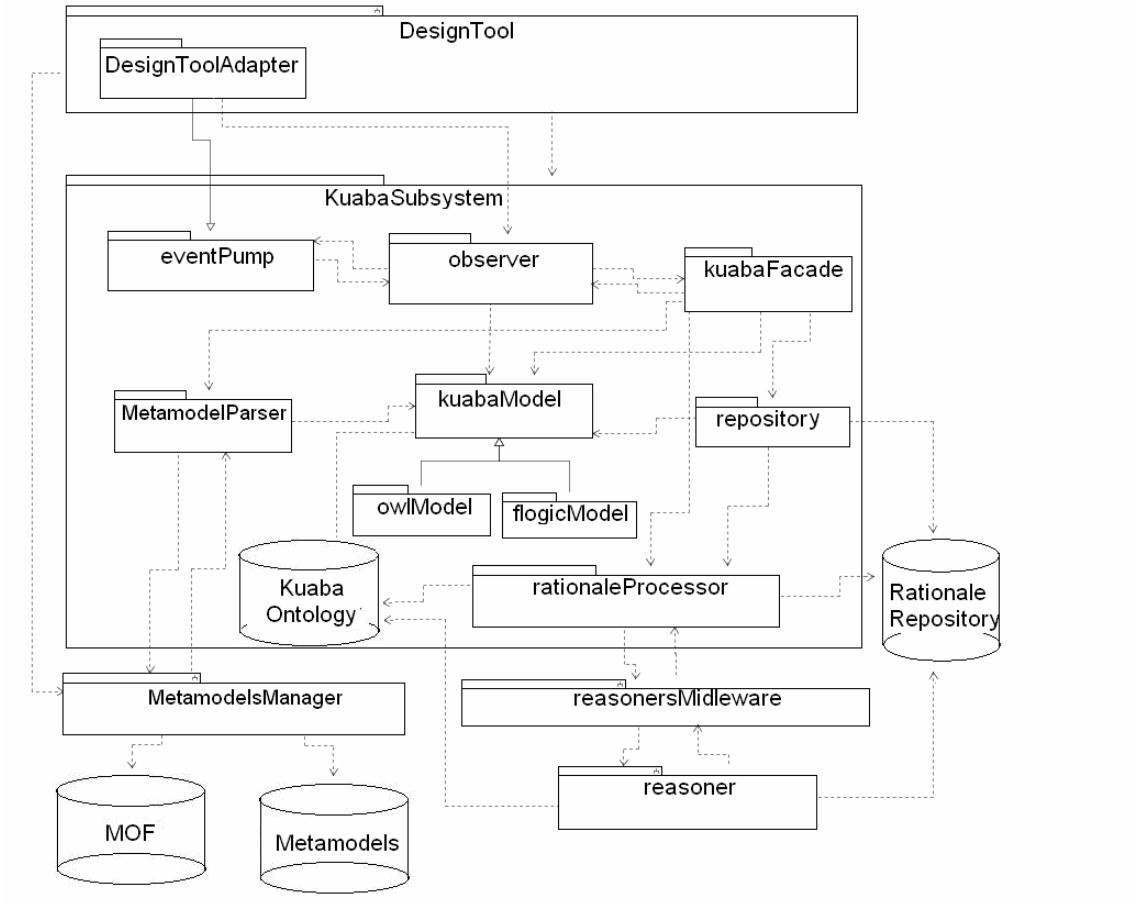

Figura 4. Arquitetura do Ambiente de Design Integrado

Essa arquitetura possui três componentes principais: *DesignTool*, *KuabaSubsystem* e o *MetamodelsManager*. *DesignTool* é responsável por permitir a edição de designs e pela notificação das modificações realizadas nos mesmos pelo projetista. Essas notificações são enviadas ao subsistema Kuaba por meio de eventos.

Dessa forma, o subsistema Kuaba identifica o tipo de evento ocorrido no design e instancia os elementos corretos (questões e idéias) da ontologia Kuaba para representar o DR, solicitando argumentos e justificativas aos projetistas quando necessário.

KuabaSubsystem é o componente responsável pela implementação da abordagem Kuaba. Ele contém diversos módulos. A manipulação da ontologia Kuaba é realizada pelo módulo *KuabaModel*, que fornece interfaces para manipulação de todos os elementos presentes no vocabulário dessa ontologia e para a especificação das instâncias desses elementos em uma linguagem formal, F-Logic ou OWL. O módulo *metamodelParser* é o responsável por analisar o metamodelo e realizar o registro semi-automático de DR. A consulta aos repositórios de *rationale*, incluindo operações de leitura e escrita, é implementada pelo módulo *repository*. O módulo *observer* contém classes que realizam o monitoramento do design e recebem as notificações das modificações ocorridas nos diagramas por meio de eventos gerados pela ferramenta de design. A partir desses eventos, alguns dos elementos da estrutura de DR (questões e idéias de design) são gerados automaticamente. O módulo *rationaleProcessor* disponibiliza as operações do processador de *rationale*, implementado em [Medeiros 2006], para o restante do subsistema. Ele conecta o subsistema Kuaba a esse processador e à máquina de inferência para a realização do processamento das representações de DR armazenadas no repositório de *rationale*.

O componente *MetamodelsManager* é responsável pela implementação dos metamodelos. Esse componente fornece toda a infra-estrutura necessária para gerenciar a criação, o armazenamento, o acesso, a descoberta e as consultas de metadados baseados na especificação MOF (*Meta Object Facility*) [OMG 2006] mantida pelo OMG (Object Management Group) para a definição de metamodelos.

A arquitetura definida neste trabalho para o ambiente de design integrado está sendo implementada, inicialmente, como uma extensão da ArgoUML, uma ferramenta de código aberto desenvolvida em Java para design de software com UML. Muitas outras ferramentas de design de código aberto foram pesquisadas e analisadas, entre elas JUDE[2], MVCASE e Poseidon for UML[3]. A ArgoUML foi selecionada pela sua robustez, escalabilidade, facilidade de extensão e qualidade de documentação.

3.1 Registro semi-automático de *Design Rationale*

Normalmente, o DR de um artefato envolve o registro de uma grande quantidade de informação. Isso torna o processo de captura e representação de DR custoso mediante a carga de trabalho extra exigida do projetista.O ambiente de design integrado proposto para a implementação da abordagem Kuaba difere das demais ferramentas existentes para DR, uma vez que ele não requer que o projetista registre manualmente toda a sua estrutura de raciocínio. Implementando a abordagem Kuaba, esse ambiente é capaz de analisar o metamodelo utilizado pela ferramenta de design para descrever os artefatos e extrair dele, automaticamente, as questões e opções de design disponíveis.

O registro de DR tratado neste artigo é dito semi-automático porque não é possível extrair automaticamente toda a estrutura de DR. O projetista precisa informar

[2] JUDE Design & Communication, http://jude.change-vision.com/jude-web/index.html

[3] GentleWare Models to Business, http://www.gentleware.com

os argumentos, contra e a favor das alternativas de solução, e as justificativas para as suas decisões. Essas informações caracterizam o conhecimento e a experiência do projetista investidos no design dos artefatos de software.

Como mencionado anteriormente, o componente responsável por extrair automaticamente as questões de design e as idéias de solução do metamodelo é o *metamodelParser*. Ele interage com o componente *metamodelsManager*, implementado pelo framework JMI (*Java Metadata Interface*)[4], que carrega o metamodelo de um arquivo XMI para a memória e analisa suas metaclasses. Considerando que o projetista pode não utilizar todo o metamodelo no design de seus artefatos, o componente *metamodelParser* pode ser configurado para selecionar apenas uma parte do metamodelo ao invés de trabalhar com todas as suas metaclasses. Isso é importante, por exemplo, quando o projetista utiliza apenas a notação de diagrama de classes no design de seus artefatos. Atualmente, o componente *metamodelParser* pode extrair questões e idéias de design de qualquer metamodelo descrito em MOF.

Primeiramente, o componente *metamodelParser* transforma todas as metaclasses concretas do metamodelo em idéias de design que respondem as questões de design relacionadas aos elementos do domínio. Na Figura 2, por exemplo, as opções *"Classe"* e *"Atributo"* extraídas do metamodelo UML representam idéias de design que respondem a questão "*Como modelar Gênero?*". Em seguida, todas as propriedades de cada metaclasse são transformadas em questões sugeridas pela idéia de design referente à cada metaclasse. Por exemplo, as questões de design *"De Quem?"*, *"Multiplicidade Mínima?"* e *"Multiplicidade Máxima?"*, mostradas no exemplo de DR da Figura 2, são sugeridas pela idéia de design *"Atributo"*. Finalmente, os possíveis valores que cada propriedade da metaclasse pode armazenar, são transformados em idéias de design que respondem as questões geradas a partir das propriedades das metaclasses analisadas. Na Figura 2, é possível observar que as questões de DR relacionadas à multiplicidade de atributo foram geradas a partir da propriedade *Multiplicidade* herdada pela metaclasse *Atributo* e a idéia de design *"1"* foi obtida da lista de possíveis valores para essa propriedade. Atualmente, todas as representações de DR geradas pela ferramenta de design estendida são especificadas internamente usando a linguagem OWL e armazenadas no repositório de *rationale* em formato de arquivo.

Esse processo implementado para apoiar o registro semi-automático de DR a partir da semântica definida no metamodelo reduz o esforço exigido do projetista para registrar DR. Além disso, ele define um padrão para gerar representações de DR com base na ontologia Kuaba, uma vez que o formalismo fornecido pelo metamodelo prescrito pelos métodos de design é usado na instanciação dos elementos da ontologia. Esta padronização permite realizar operações computacionais sobre o DR registrado para apoiar o reuso de design, que é um novo tipo de reuso. Esse reuso é alcançado através da integração de DRs existentes para iniciar o design de um novo artefato. Atualmente, a integração de DRs é realizada pela operação de união implementada pelo processador de rationale desenvolvido em [Medeiros 2006].

[4] Sun Developers Network, http://java.sun.com/products/jmi/index.jsp

3.2 Camada de Edição de *Rationale*

A camada de edição de *rationale* é responsável pela instanciação da ontologia Kuaba para representar DR e pela captura dos argumentos e justificativas fornecidos pelo projetista. A instanciação da ontologia não é percebida pelo projetista. O principal módulo desta camada, *Observer*, monitora o design dos artefatos realizado na ferramenta de design e instancia os elementos (questões, idéias e decisões) da ontologia de acordo com as ações realizadas pelo projetista. Além disso, essa camada também identifica quando o projetista deve fornecer argumentos e justificativas e exibe as respectivas telas de edição. A Figura 5 mostra alguns dos observadores implementados nesse módulo.

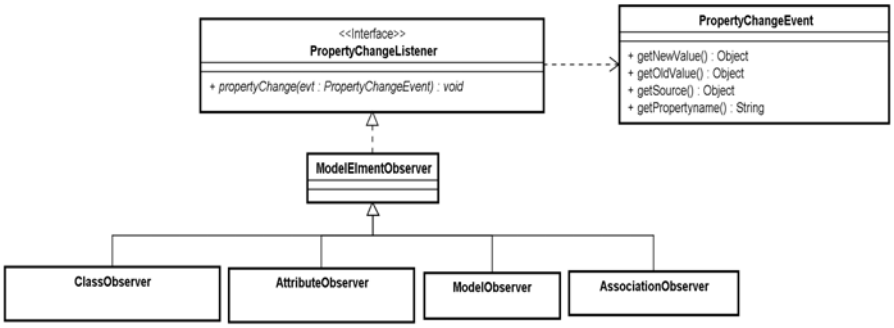

Figura 5. Exemplos de Observadores

Cada metaclasse do metamodelo utilizado tem um observador associado, responsável por monitorar as mudanças nas instâncias de sua respectiva metaclasse. Quando o projetista atribui um nome válido a algum elemento de design como, por exemplo, uma classe, o respectivo observador apresenta uma tela de cadastro de argumentos a favor para a idéia de design que corresponde a esse elemento. A Figura 6 mostra um exemplo da tela de entrada de argumentos, exibida no momento em que o projetista atribui o nome "*Pedido*" a uma classe na ferramenta de design ArgoUML.

Quando o projetista altera o nome de um elemento de design, o sistema registra uma nova idéia de domínio referente ao novo nome e pede argumentos contra a idéia de design anterior. Os argumentos digitados são, simultaneamente, contrários à idéia design anterior e a favor da idéia que corresponde à nova idéia de domínio.

A inclusão de elementos de design relacionados a outros elementos sempre requer a entrada de argumentos a favor das idéias de design relacionadas. Por exemplo, quando um atributo *número* é incluído na classe *Pedido*, a ferramenta solicita argumentos a favor da idéia de solução de fazer *número* um atributo da classe *Pedido*. Essa relação de pertinência é extraída da propriedade *owner* da metaclasse *Atributo* definida no metamodelo UML. A exclusão de qualquer elemento de design requer a entrada de argumentos contra a idéia de design associada ao elemento removido.

3.3 Camada de Consulta de *Rationale*

A camada de consulta de *rationale* permite ler e usar as instâncias da ontologia Kuaba (representações de DR) armazenadas no repositório de *rationale*. O módulo *repository* contém as classes e interfaces usadas para acessar essas instâncias, entre elas, as classes

FlogicRepository e *OwlRepository*, que implementam operações de consulta para instâncias da ontologia Kuaba representadas em Flogic e OWL, respectivamente.

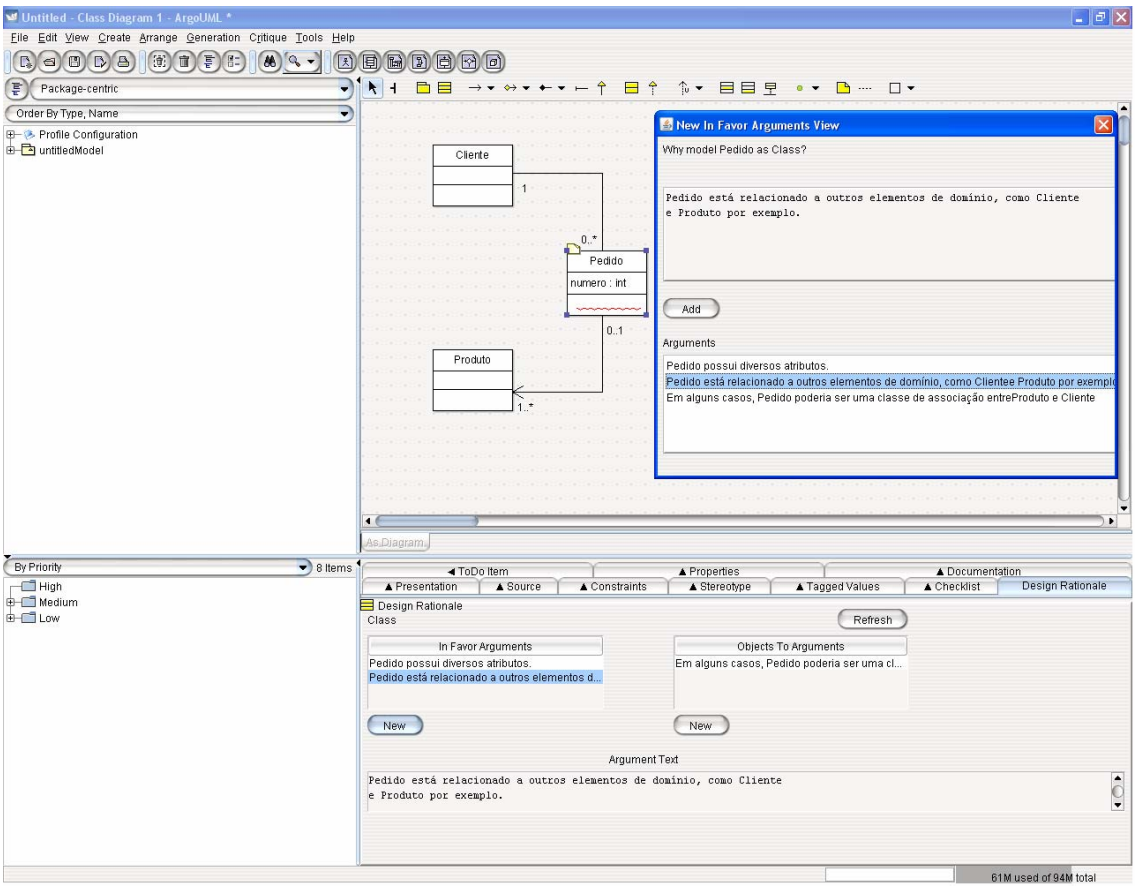

Figura 6. Exemplo de tela para apoiar a edição e consulta de DR

Para apoiar a visualização dos argumentos registrados para as idéias de design consideradas pelo projetista durante o design dos artefatos, foi adicionada uma aba chamada *Design Rationale* ao painel de propriedades da ArgoUML, como mostra a Figura 6. Quando o projetista seleciona a classe *Pedido*, os argumentos registrados anteriormente contra e a favor da idéia de design *Class* são carregados e exibidos na aba *Design Rationale*. Nessa aba, o projetista pode, também, cadastrar novos argumentos para a idéia de design relacionada ao elemento de design selecionado, basta que o projetista clique em um dos botões com rótulo *new*.

O DR registrado também pode ser visualizado pelo projetista como um mapa. Esse tipo de visualização oferece uma visão mais completa do DR. No entanto, este tipo de visualização requer algum conhecimento sobre o vocabulário utilizado na representação de DR, uma vez que é exibida toda a estrutura de raciocínio capturada de acordo com a ontologia Kuaba, incluindo questões, idéias, argumentos e decisões. A Figura 7 mostra um exemplo de consulta ao DR registrado, usando a visualização em forma de mapa.

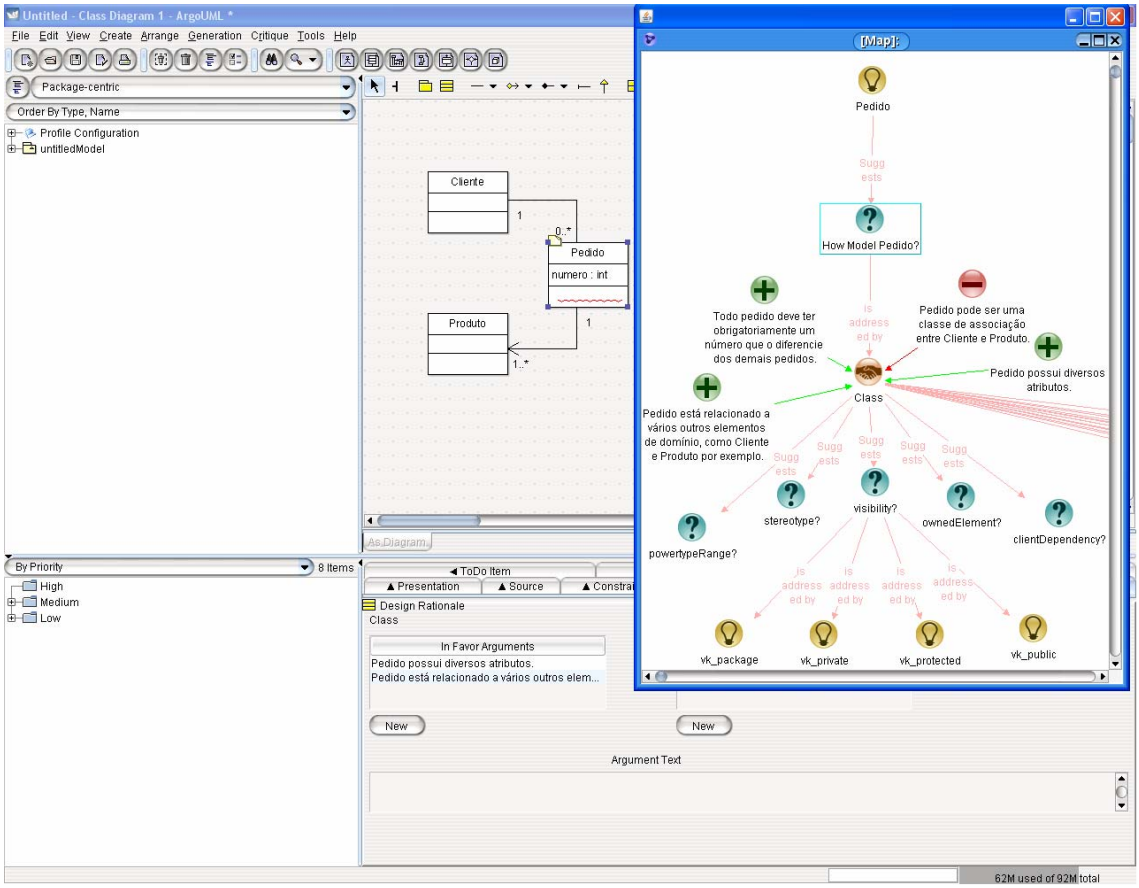

Figura 7. Exemplo de mapa para exibição de DR

Na figura acima, temos a idéia de domínio *"Pedido"* que sugere a questão *"How to model Pedido?"*. A idéia de design *"Class"*, extraída do metamodelo UML, endereça essa questão. Essa idéia, por sua vez, sugere outras questões e idéias de design com base no metamodelo. A decisão de modelar o elemento de domínio "*Pedido*" como classe também está representada no mapa. Por fim, temos a representação gráfica dos argumentos contra e a favor da idéia de design "*Class*". Apenas esses últimos elementos são cadastrados pelo usuário. O restante dos elementos da ontologia Kuaba foram extraídos do metamodelo UML de forma automática enquanto o projetista desenhava seus artefatos. Devido a complexidade do mapa extraído do metamodelo, foram exibidas na Figura 7 apenas as idéias de design que possuem decisões aceitas.

3.4 Interface com a Ferramenta de Design

Para conectar os componentes *KuabaSubsystem* e *DesignTool* ilustrados na Figura 4, é necessário que os observadores do módulo *Observer* sejam adicionados à ferramenta de design, e que esta emita os eventos gerados à medida que o projetisa altera o seu design. Visando alcançar a independência entre esses componentes, foi definida a interface *EventPump*. Essa interface pertence ao módulo *eventPump* do componente *KuabaSubsystem* e contém operações para cadastrar e remover observadores de acordo com os elementos de design editados na ferramenta. A interface *EventPump* é implementada pelo componente *DesignTool*, no módulo *designToolAdapter*, fazendo com que o subsistema Kuaba possa cadastrar e remover observadores, independente da ferramenta de design utilizada.

A ferramenta ArgoUML já possui uma interface para cadastro e remoção de observadores e emissão de eventos. Assim, a implementação da interface *EventPump*, no módulo *designToolAdapter*, é realizada por um adaptador para a interface fornecida pela ArgoUML, como mostra a Figura 8.

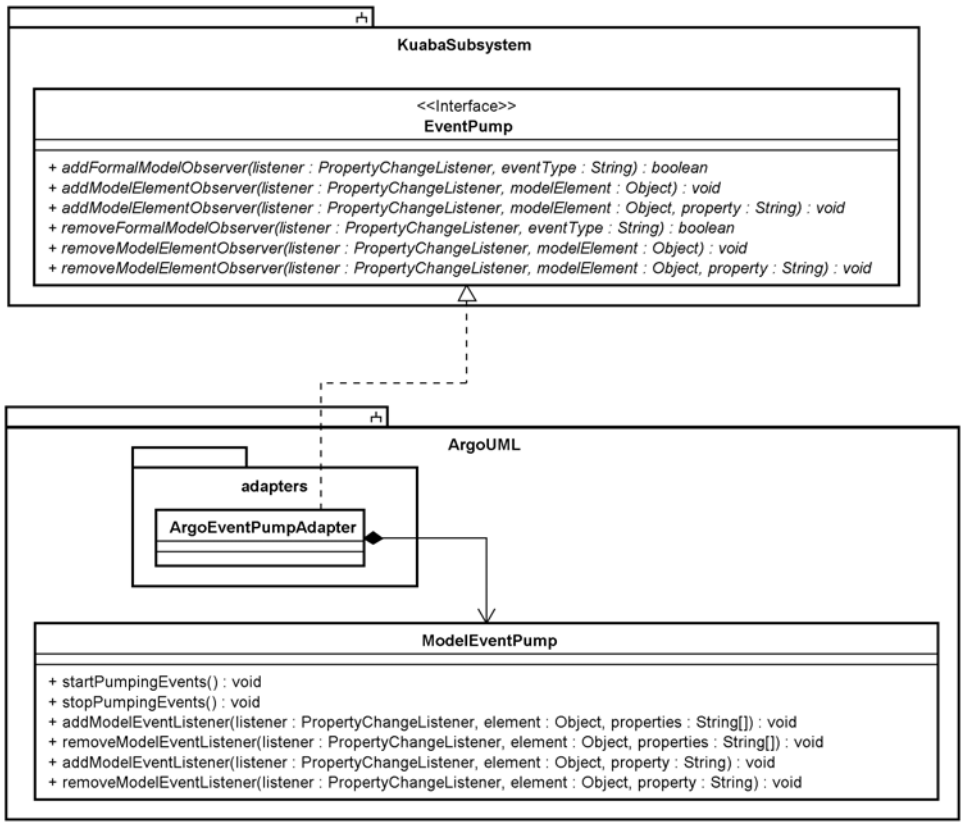

Figura 8. Adaptação da interface *EventPump*

4 Avaliação da Ferramenta estendida

Para avaliar a extensão realizada na ferramenta de design ArgoUML para implementar a abordagem Kuaba, foi realizado um experimento com cinco alunos da Universidade Candido Mendes - Campos, quatro deles cursando o nono período do curso de Ciência da Computação e um o curso de mestrado. O experimento foi realizado com os seguintes objetivos: avaliar se o registro semi-automático de DR viabilizado pelo uso de metamodelos pode reduzir, de fato, o esforço requerido do usuário para registrar DR; avaliar a importância dada pelos usuários ao DR registrado e identificar possibilidades de melhorias na interface da ferramenta ArgoUML estendida.

Nesse experimento os participantes foram solicitados a resolver um problema de design registrando as informações solicitadas pela ferramenta de design e, posteriormente, trocar os designs elaborados entre eles. Após a troca dos designs, foi solicitado a cada participante que tentasse entender a solução de design desenvolvida pelo seu colega através da consulta do DR registrado. Ao final, os participantes responderam um questionário fornecendo *feedback* sobre sua experiência em usar a ferramenta. Este questionário incluiu questões sobre a usabilidade da ferramenta, o esforço requerido para registro de DR, o suporte fornecido à captura e consulta de DR

durante o design, o desempenho da ferramenta e possibilidades de melhorias na interface com o usuário.

O problema utilizado nesse experimento envolveu a modificação de uma solução de design inicial envolvendo três classes (*CaixaRegistradora*, *Cozinha* e *Comida*) para registrar a venda de comestíveis em uma aplicação de bar-café, visando acomodar um novo requisito: "*a aplicação deve apoiar, também, a venda de coisas não comestíveis como Jornais e Revistas, por exemplo*". Essa solução foi elaborada usando técnicas de modelagem orientada a objetos. Cada participante desenvolveu sua própria solução de design, partindo do design original, cadastrando argumentos contra e favor de suas idéias de solução. Esses argumentos foram, posteriormente, consultados por um outro participante para tentar entender as razões por trás das diferentes soluções de design para o mesmo problema.

Após a realização do experimento, constatou-se que todos os participantes reconheceram a importância das informações de DR capturadas. Todos alegaram que elas são fundamentais para agilizar o entendimento de um design realizado por outra pessoa. Além disso, afirmaram que a solicitação dos argumentos, no momento em que o design está sendo realizado, faz com que o projetista reflita melhor sobre a solução que ele está modelando, podendo melhorar, assim, a qualidade da solução desenvolvida. Os participantes também descartaram a possibilidade de utilizar duas ferramentas distintas, uma para realização do design e outra para o registro de DR.Constatou-se, também, que o fato da ferramenta utilizada realizar a captura e representação de DR de forma semi-automática, restando ao projetista apenas registrar os argumentos para as idéias de solução consideradas e as justificativas para as decisões tomadas, motivou os participantes a registrar DR para seus artefatos.

Os principais problemas identificados durante o experimento estão relacionados à interface com o usuário, ao desempenho da ferramenta e à freqüência com que solicitação de argumentos é feita durante o design. Houve uma certa dificuldade por parte dos participantes em realizar a consulta ao DR registrado. Eles alegaram que a interface gráfica é pouco explicativa no caso das consultas, uma vez que eles não possuíam nenhum conhecimento prévio sobre DR. Com relação ao desempenho, 10% dos participantes tiveram problemas com performance devido à extensa manipulação do design e, conseqüentemente, da grande quantidade de objetos gerados em memória. Outra dificuldade identificada pelos alunos é a quantidade de argumentos solicitados. Eles consideraram que boa parte dos argumentos poderia ser extraída da fase de levantamento de requisitos ou até mesmo de boas práticas de desenvolvimento orientado a objetos.

5 Trabalhos Relacionados

Muitos sistemas têm sido propostos na literatura para integrar o registro de DR ao processo de design, como SEURAT, Sysiphus [Dutoit et al. 2005], *TechSolution* [Figueiredo et al. 2005] e ABCDE-DR [Souza et al. 1998]. Contudo, nenhum deles oferece um mecanismo similar ao apresentado neste trabalho para automatizar a geração de parte das informações de DR, aproveitando a semântica do metamodelo usado pelos projetistas para descrever seus artefatos. Isso faz como que o usuários tenham que registrar, manualmente, não apenas os argumentos e justificativas para suas decisões, mas toda a estrutura de DR, que inclui todas as questões tratadas e as possíveis opções

de design (muitas delas pré-definidas pelo metamodelo), tornando o registro de DR uma atividade extremamente onerosa e ineficaz.

O sistema SEURAT (*Software Engineering Using RATionale*) possui uma arquitetura conceitual similar à arquitetura conceitual do ambiente de design integrado apresentada neste trabalho (Figura 3), e também propõe a integração das ferramentas de design com ferramentas de suporte à captura e representação de DR. No entanto, o sistema SEURAT não usa a semântica do metamodelo utilizado e apóia apenas o uso das representações de DR para identificar inconsistências durante o processo de manutenção de software.

Sysiphus é um conjunto de ferramentas voltado para o meio acadêmico que permite que os estudantes desenvolvam e compartilhem seus modelos de sistema e discutam, justifiquem e colaborem uns com os outros utilizando modelos de DR similares ao QOC [MacLean 1991]. O ponto chave do *Sysiphus* é que ele atribui igual importância para os modelos de sistema e para os modelos de *rationale*. De forma similar à ferramenta estendida nesse trabalho, o sistema *Sysiphus* permite que o DR seja registrado no momento em que o design está sendo realizado, mas não possui nenhum tipo de automatização da captura e representação de DR. Ou seja, toda a estrutura de raciocínio é registrada manualmente, gerando uma carga de trabalho excessiva para seus usuários.

A ferramenta *TechSolution* está presente nos Ambientes de Desenvolvimento de Software (ADS) e nos Ambientes de Manutenção de Software (AMS) instanciados a partir da Estação TABA [Rocha et al. 1990]. Essa ferramenta permite que os projetistas façam o registro e a consulta de DR no mesmo ambiente em que desenvolvem seus artefatos. Isso elimina a necessidade de utilizar duas ferramentas distintas, uma para o design e outra para o registro de DR. Por exemplo, a ferramenta *TechSolution* apresenta uma base de conhecimento para determinados tipos de decisões, como a escolha do estilo arquitetural a ser utilizado. Neste caso, é apresentada ao projetista uma lista de alternativas de solução, critérios para avaliar as soluções e a avaliação destes critérios em relação às alternativas de solução. Com essas informações previamente cadastradas, há uma redução no esforço extra requerido do projetista para registro de DR. No entanto, o registro de DR é feito de forma manual e informal.

ABCDE-DR é um protótipo de sistema para o domínio de diagramas de objetos em engenharia de software. Ele apóia a representação semi-formal de DR usando um modelo de anotações. A captura e representação de DR ocorrem no momento em que o design é realizado, a partir das anotações anexadas aos objetos do diagrama. O próprio sistema é responsável por estruturar as informações de DR e por armazená-las para uso futuro, atribuindo certo grau de automatização do processo. No entanto, todas as informações de DR precisam ser registradas manualmente na forma de anotações, o que agrega um alto custo ao processo de design.

6 Conclusões

Neste trabalho foi apresentada uma extensão da ferramenta de design de software ArgoUML realizada com o objetivo de implementar o uso de metamodelos proposto pela abordagem Kuaba para apoiar o registro semi-automático de DR, validando, assim, a arquitetura conceitual do ambiente de design integrado definida em [Medeiros 2006].

A captura e representação semi-automáticas de DR, foco deste trabalho, foram implementadas com o objetivo de reduzir o esforço requerido do projetista para o registro de DR. Para isso, a ferramenta de design estendida analisa o metamodelo utilizado e extrai automaticamente parte da estrutura de DR (questões e idéias de design) à medida que o projetista desenha seu artefato. Em outras palavras, o *rationale* é capturado no momento em que o design é realizado de acordo com as opções de design pré-definidas pelo metamodelo utilizadas pelo projetista. Assim, o projetista não precisa conhecer os elementos do modelo de representação de DR, ele deve informar apenas seu raciocínio sobre as idéias de design consideradas e o porquê de suas decisões. Além disso, a ferramenta estendida também gera as especificações formais das representações de DR em OWL, e futuramente em F-logic, permitindo seu processamento.

Nossa pesquisa atual inclui: a implementação da captura e representação semi-automática de DR para outros diagramas da UML, como diagramas de casos de uso e de sequência; a investigação do uso de ontologias de argumentos para reduzir a quantidade de requisições feitas ao usuário pela ferramenta; a implementação da entrada das justificativas para as decisões tomadas durante o design; e a integração da ferramenta ArgoUML estendida com o processador de *rationale* desenvolvido em [Medeiros 2006], permitindo o processamento computacional das representações de DR para apoiar reuso de design. Essa integração visa permitir ao projetista consultar DRs de diferentes artefatos e combiná-las para iniciar o design de um novo artefato, a partir de idéias de solução já utilizadas em outros artefatos.

Como trabalho futuro, será estudada a possibilidade de realizar a carga das instâncias da ontologia Kuaba sob demanda para manter em memória apenas os objetos necessários às ações do projetista a cada momento, visando melhorar o desempenho da ferramenta. Um outro trabalho futuro é a construção de uma perspectiva de DR, na qual o projetista poderá visualizar e editar não só os argumentos, mas toda a estrutura de DR registrada. Esse trabalho permitirá avaliar a possibilidade de gerar os diagramas UML a partir das modificações realizadas nos elementos de DR. Essas modificações seriam automaticamente refletidas no design e, como o DR é baseado no metamodelo que descreve o artefato, seria possível gerar novos diagramas.

Referências

Burge, J., & Brown, D. C. (2004). “An Integrated Approach for Software Design Checking Using Rationale”, Design Computing and Cognition, Kluwer Academic Publishers, 557-576.

Conklin, J., Selvin, A., Buckingham, S. S., Sierhuis, M. (2003) “Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS”, In: LAP'03: 8th International Working Conference on the Language - Action Perspective on Communication Modeling, Tilburg, The Netherlands.

Dutoit, A. H., Wolf, T., Paech, B., Borner, L., Rückert, J. (2005). “Using Rationale for Software Engineering Education”, Proceedings of the 18th Conference on Software Engineering Education & Training (CSEE&T 2005), IEEE Computer Society, p. 129-136.

Figueiredo, S., Rocha, A. R., Santos, G., Montoni, M. e Natali, A. C. (2005) “Apoio à Manutenção de Software através de Design Rationale em Ambientes de Manutenção de Software Taba”. Anais do II Workshop de Manutenção de Software Moderna.

Kifer, M. and Lausen, G. (1989) F-Logic: “A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme”. ACM SIGMOD May,134-146.

Kunz, W. and Rittel, H. W. J. (1970) “Issues as Elements of Information Systems”. Institute of Urban and Regional Development, Working Paper 131, Univ of California, Berkeley, CA.

Lacaze, X. (2005) “Conception rationalisée pour les systèmes interactifs – Une notation semi formelle et un environnement d’édition pour une modélisation des alternatives de conception”. These de Doctorat de l’Université Toulouse I.

Lee, J. (1991) “Extending the Potts and Bruns Model for Recording Design Rationale”. Proc. of the 13th International Conference on Software Engineering, Austin, 114-125.

Lee, J. (1997) “Design Rationale Systems: Understanding the Issues”. IEEE Expert Volume 12, No. 13, p. 78-85.

MacLean, A., Young, R., Bellotti, V. and Moran, T. (1991) “Questions, Options, and Criteria: Elements of Design Space Analysis”. Human-Comput. Interaction, No. 6 (3-4) 201-250.

Medeiros, A. P., Schwabe, D., Feijó, B. (2005) “Kuaba Ontology: Design Rationale Representation and Reuse in Model-Based Designs”. *Proc. ER 2005*, LNCS 3716, 241-255.

Medeiros, A. P. (2006) “Kuaba: Uma Abordagem para Representação de Design Rationale para o Reuso de Designs baseados em Modelo. Tese de Doutorado, Departamento de Informática, PUC-Rio.

Medeiros, A., Schwabe, D. (2008) “Kuaba Approach: Integrating Formal Semantics and Design Rationale Representation to Support Design Reuse”, In: Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2008-22, p. 399-419. Cambridge University Press.

OMG (2006) “Meta Object Facility (MOF) Core Specification”.

Paiva, D. M. B., Lucrédio, D., Fortes, R. P. M. (2006) “MVCASE – incluindo *design rationale* para auxílio à modelagem em projetos de pesquisa”, XX SBES - Simpósio Brasileiro de Engenharia de Software (SBES 2006), Florianópolis, SC, Brasil

Rocha, A. R. C., Aguiar, T. C., Souza, J. M. (1990) "TABA: A Heuristic Workstation for Software Development". In: *COMPEURO 90*, Tel Aviv, Israel, May.

Souza, C. R. B., Wainer, J., Santos, D. B., Dias, K. L. (1998) “A model and tool for semi-automatic recording of design rationale in software diagrams.” *Proceedings of the 6th String Processing and Information Retrieval Symposium & 5th International Workshop on Groupware*, Cancun, Mexico, p. 306–313.

Tolke, L. Klink, M., van der Wulp, M. (2007) “Cookbook for developers of ArgoUML”, http://argouml-stats.tigris.org/documentation/defaulthtml/cookbook/

W3C (2004) Web Ontology Language Reference, February.