2
Abordagens e Requisitos para a Representação de Design Rationale

A pesquisa em *design rationale* normalmente envolve três aspectos principais: a captura, a representação e o uso de *design rationale*. Geralmente, o esquema de representação determina os métodos usados para capturar e recuperar o *design rationale*, possibilitando sua utilização no design de novos artefatos. Esquemas de representação baseados em argumentação têm sido bastante usados para representar *design rationale*, por fornecerem uma estrutura para indicar quais decisões foram tomadas (ou não) e as razões por trás dessas decisões.

Neste trabalho decidimos representar *design rationale* usando uma abordagem baseada em argumentação. Consideramos que tal abordagem é adequada para expressar as razões para a escolha das diferentes opções de design usadas pelos projetistas durante o processo de design de software.

O trabalho de Toulmin (1958), particularmente o uso de uma representação gráfica semiformal para a visualização da estrutura de argumentos, pode ser visto como um precursor dos esquemas de representação baseados em argumentação para *design rationale* (Shum, 1991). De acordo com a análise da estrutura lógica de argumentos realizada por Toulmin (1958), um argumento é composto (de forma implícita ou explícita) de um fato ou observação (*Datum*), que via um passo lógico (*Warrant*) nos permite produzir uma conseqüente assertiva (*Claim*). A justificativa pode ser apoiada por um material de apoio (*Backing*) se necessário (o porquê da justificativa ser assumida como válida) e a afirmação (*Claim*) pode ser qualificada com uma refutação (*Rebuttal*), especificando exceções à regra.

Embora o trabalho de Toulmin (1958) possa ser visto como um precursor das notações existentes para argumentação, ele sofre de algumas deficiências expressivas no contexto de *design rationale*, como mostra a análise apresentada em (Lee & Lai, 1991b). Uma dessas deficiências é o fato de o modelo de Toulmin (1958) não ter a noção de alternativas ou medidas de avaliação (estado das

alternativas), como por exemplo “rejeitada” e “aceita”. Ou seja, seu escopo é limitado à representação de argumentos.

Neste capítulo apresentamos as principais abordagens que usam argumentação para representar *design rationale* (seção 2.1). Em seguida, apresentamos uma análise crítica sobre estas abordagens discutindo os principais problemas identificados (seção 2.2). Por fim, relacionamos alguns requisitos necessários para um modelo de representação mais expressivo para *design rationale*, que possa resolver alguns dos problemas verificados nas abordagens existentes (seção 2.3).

2.1. Diferentes Abordagens para Representar Design Rationale

Nesta seção apresentamos algumas das abordagens existentes na literatura que usam argumentação para representar *design rationale*.

2.1.1. A Abordagem IBIS

IBIS (*Issue Based Information System*) (Kunz & Rittel, 1970) foi desenvolvido, inicialmente, como um meio através do qual a deliberação aberta de temas pudesse ser realizada. Nesta abordagem, o design pode ser visto como um processo de deliberação, onde as possíveis soluções para um problema são apresentadas e submetidas à discussão sobre os seus prós e contras. IBIS foi a primeira representação explícita para raciocínio em um contexto de design – a primeira notação para *design rationale* (Shum, 1991).

Em IBIS, as questões de design são articuladas como *temas (issues)*, com cada tema seguido por uma ou mais *posições* que respondem ao tema. Cada posição pode potencialmente ser adotada como uma solução para o tema ou ser rejeitada. Os fatores favoráveis ou contrários às posições são descritos como *argumentos*. A Figura 1 ilustra as principais relações entre esses três elementos em IBIS.

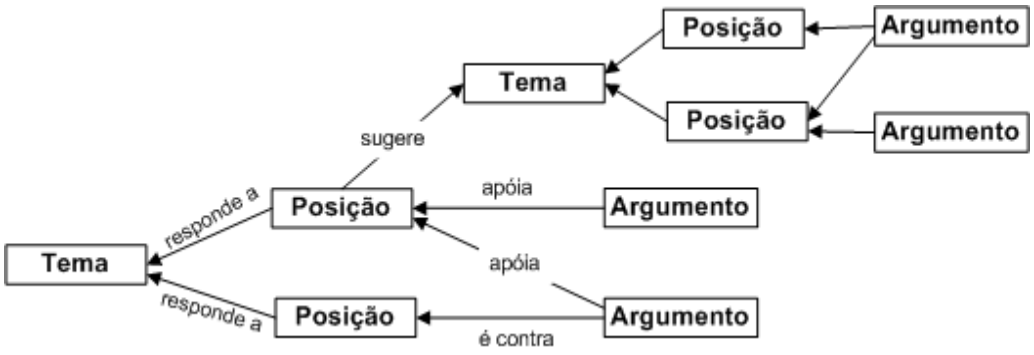

Figura 1 – A estrutura básica de IBIS

Muitos dos sistemas propostos para apoiar a captura de *design rationale* são baseadas em IBIS, como por exemplo, a ferramenta hipertexto gIBIS (Conklin & Begeman, 1988), QuestMap™ (Conklin, 1999) e Compendium (Conklin et al., 2003).

A ferramenta hipertexto gIBIS estende o vocabulário de IBIS e adiciona uma representação gráfica, na qual cada representação IBIS é apresentada como um grafo direcionado e o conteúdo dos nós e a navegação são apresentados em janelas distintas. Nela, os usuários podem navegar, consultar um índice hierárquico de todos os nós e acessar os atributos e conteúdos dos nós e elos. Menus sensíveis ao contexto garantem que apenas comandos legais podem ser realizados na criação dos nós. Assim, nenhum nó pode ser criado sem ser primeiro classificado e também relacionado a um nó apropriado, a menos que ele próprio seja estruturado como um tema. Cadeias de temas, suas posições e argumentos podem ser reunidos em uma sub-rede. No entanto, elos para nós localizados fora da sub-rede não são exibidos e a semântica e as inter-relações de temas são perdidas. Em resumo, gIBIS está configurada como uma ferramenta de navegação através da rede de temas mais do que para apoiar a avaliação do *rationale* ou a tomada de decisão.

As ferramentas QuestMap™ e Compendium são versões da ferramenta gIBIS. Estas ferramentas usam a notação IBIS com algumas modificações. Os nós *tema* e *posição* são chamados respectivamente de *questão* e *idéia* e o nó *Argumento* foi dividido em dois, um argumento positivo (a favor) e um argumento negativo (contra). Os nós são representados com ícones específicos e, para enfatizar os argumentos, os usuários podem anexar aos nós diferentes tipos de informação, como um vídeo, um texto, uma figura, etc. A representação inteiramente gráfica permite aos usuários organizar espacialmente esses nós.

Compendium é uma versão melhorada de QuestMap™. A melhoria está na capacidade de editar um nó. Na ferramenta Compendium existem janelas nas quais o usuário pode editar e visualizar informações adicionais relacionadas aos nós. Assim, o usuário pode adicionar informações textuais para descrever aspectos relevantes que complementam as informações registradas nos rótulos dos ícones utilizados na representação gráfica. Compendium foi usado em vários projetos industriais e acadêmicos. No entanto, os usuários acabaram não usando a ferramenta como os autores haviam planejado. Na verdade, eles usaram Compendium para manter um registro de suas idéias, soluções e questionamentos, ignorando alguns aspectos importantes da ferramenta, como por exemplo, a captura de *design rationale*.

2.1.2. O Modelo PHI

O modelo PHI (*Procedural Hierarchy of Issues*) (McCall, 1991) estende a notação IBIS expandindo a definição de *tema* e alterando a estrutura que relaciona temas. Em PHI, temas são relacionados por elos do tipo "*serve*", de forma que um tema A *serve* a um tema B se resolver A ajuda a resolver B. Além disso, PHI fornece dois métodos para tratar os temas de design: deliberação, no qual projetistas podem apresentar posições sobre os temas, e decomposição, onde o tema pode ser decomposto em uma variedade de sub-temas, que por sua vez podem ser deliberados ou decompostos. Assim, uma estrutura PHI é quase hierárquica (um tema pode ter mais de um pai) e, geralmente, tem sido representada como uma lista textual com sub-temas organizados em níveis.

O principal benefício de PHI é que ele foca na deliberação de temas que servem às afirmações estabelecidas no design (por exemplo, o tema principal), evitando tratar questões triviais e irrelevantes. Ou seja, se um problema não pode ser mostrado para servir parte da hierarquia, então ele é considerado irrelevante.

O modelo PHI tem sido implementado em diversas ferramentas para captura de *design rationale*. Alguns exemplos são o sistema JANUS (Fischer & McCall, 1989) e o sistema PHIDIAS (McCall, 1990). JANUS é um sistema para design arquitetural que integra um editor CAD com uma crítica de design baseada em regra e com um ambiente para documentação hipertexto. O sistema JANUS monitora os desenhos em CAD enquanto eles são desenvolvidos, e uma crítica de

design baseada em regra avisa ao projetista se uma diretriz de design é violada. Neste caso, o projetista pode então requisitar a argumentação representada em PHI para explicar a regra. O sistema JANUS tem demonstrado que hipertexto pode ser usado em conjunto com um ambiente de design baseado em conhecimento para facilitar a captura do conhecimento de design.

PHIDIAS é um sistema hipermídia baseado em PHI que usa uma estrutura de nós e elos baseada em grafo para representar *rationale*. Este sistema apóia buscas estruturais da base de temas, esconde temas se decisões anteriores os tornaram irrelevantes, e permite que a argumentação relevante sobre um objeto seja exibida simplesmente pela seleção desse objeto. Embora PHIDIAS não use a crítica baseada em regra de JANUS, ele demonstra como os artefatos do design podem estar relacionados ao conhecimento de design que não pode ser facilmente formalizado como regras.

2.1.3. O Modelo Potts and Bruns

Em (Potts & Bruns, 1988) os autores apresentam um modelo genérico para representar deliberações de design e suas relações com os artefatos de design. O propósito deste modelo é representar o processo de deliberações que conduz a um dado design, e os designs intermediários que resultam neste processo. No modelo *Potts and Bruns*, a história de design é formada pela rede de designs intermediários representados pelos artefatos (especificações ou documentos de design), que são derivados uns dos outros através dos nós de deliberação baseados em IBIS, representados como temas, alternativas e justificativas. Os artefatos produzidos dependem do método de design de software que está sendo apoiado. A Figura 2 ilustra um exemplo da notação usada no modelo *Potts and Bruns*.

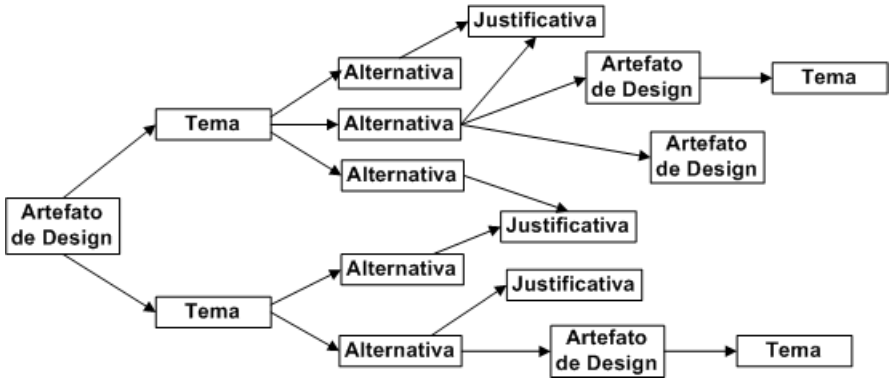

Figura 2 – A estrutura do modelo *Potts and Bruns*

A idéia principal do modelo *Potts and Bruns* é a integração de entidades dos métodos de design existentes com a deliberação de design baseada em IBIS. Esta integração é a diferença chave entre este modelo e os outros esquemas de representação para *design rationale*. Nela, as entidades do modelo genérico de Potts & Bruns são refinadas para acomodar o vocabulário de um método de design particular para derivar novos artefatos. Por exemplo, uma nova entidade, específica de um método de design, é incorporada ao modelo IBIS dando origem a um novo tema na deliberação do design. Assim, o modelo *Potts and Bruns* pode ser integrado a diferentes métodos de design.

Potts & Bruns apresentaram um exemplo do design de um formatador de texto usando o método de Liskov & Guttag (1986) para ilustrar essa integração. Neste exemplo, uma nova entidade *tarefa*, específica do método de Liskov & Guttag, é incorporada ao modelo IBIS dando origem a novos temas na deliberação do design do formatador de texto. As transformações dos artefatos através dos nós de deliberação foram representados usando o sistema hipertexto generalizado *PlaneText* (Gullichsen et al., 1986), cuja estrutura era verificada por regras Prolog que “conheciam” a sintaxe do método de Liskov & Guttag.

2.1.4. A Linguagem DRL

DRL (*Decision Representation Language*) foi originalmente projetada para representar o *rationale* de decisões. Em Lee (1991), o autor apresenta uma pequena extensão da DRL original, visando torná-la uma linguagem de *design rationale*. A linguagem DRL apresentada no trabalho de Lee (1991) pode ser vista também como uma extensão do modelo *Potts and Bruns* descrito acima, pois preserva a estrutura básica deste modelo acrescentando a ele mais poder expressivo e computacional. A Figura 3 ilustra os elementos e relações que formam o vocabulário da linguagem DRL.

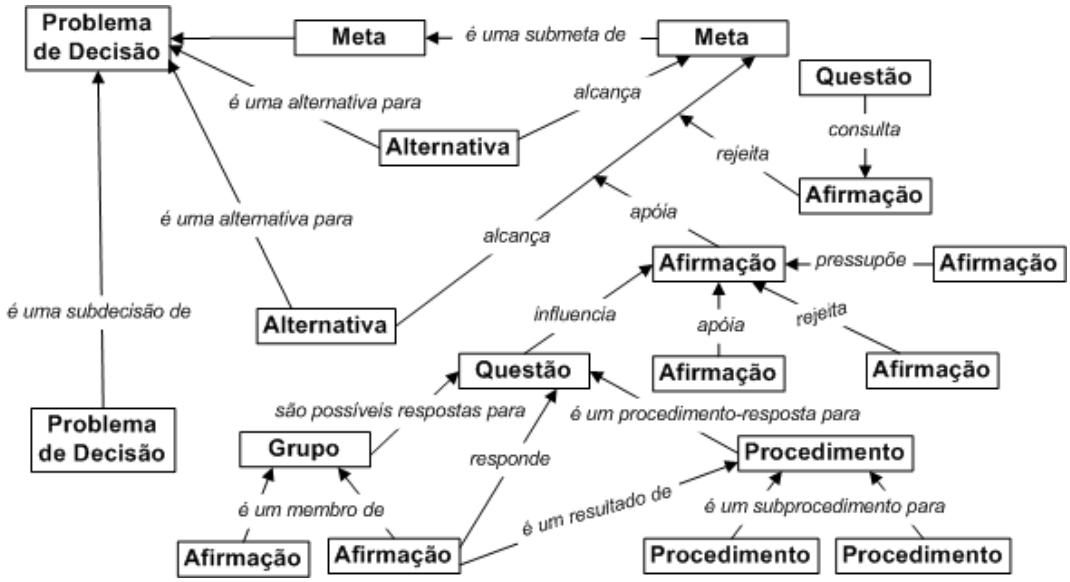

Figura 3 – O vocabulário da linguagem DRL

DRL foi desenvolvida para ser usada no sistema SIBYL (Lee, 1990) com o objetivo de fornecer suporte a serviços computacionais, como o gerenciamento de dependência (monitorar decisões que dependem umas das outras), o gerenciamento de precedência (decisões compartilhando as mesmas metas) e o gerenciamento de plausibilidade (a força da argumentação em favor de uma alternativa). DRL enfatiza a gerência dos elementos qualitativos da tomada de decisão e o gerenciamento de dependência. Ela expressa a avaliação das alternativas por referência a metas explícitas que capturam os objetivos do processo de design. O foco dessa linguagem de representação está em gerenciar o peso para uma decisão de design particular de uma maneira consistente e confiável.

SIBYL é um sistema de gerenciamento de decisão qualitativo, que auxilia os usuários no gerenciamento e na representação dos aspectos qualitativos do processo de tomada de decisão. Este sistema apóia as tarefas de decisão usando a linguagem DRL para representar a argumentação por trás das decisões tomadas, e para criar “gráficos de decisão” mostrando as diferentes alternativas e suas avaliações.

SIBYL consiste de duas partes: a interface do usuário que torna mais fácil para as pessoas usar DRL para representar suas tomadas de decisão, e o conjunto de serviços (gerenciamento de dependência, de precedência e avaliações), que exploram a estrutura de DRL para fornecer apoio à decisão qualitativa, auxiliando o usuário a tomar melhores decisões baseadas nesta estrutura.

2.1.5. A Análise do Espaço de Design - QOC

A análise do espaço de design posiciona um artefato em um espaço de possibilidades e busca explicar porque esse artefato foi escolhido dentre estas possibilidades. Trata-se de um processo que envolve a descoberta das dimensões importantes em um espaço (quais são as Questões principais), a exploração do espaço de alternativas (Opções) que definem os espaços locais ao redor dessas dimensões, e a justificativa do porque um ponto em um espaço local é melhor que um outro ponto (baseada em Critérios e Argumentos).

QOC (*Questions, Options and Criteria*) (MacLean et al., 1991) é uma notação semiformal, baseada em argumentação, de análise do espaço de design para sistematicamente representar e reformular visões do "espaço de design" ao redor de um design. Uma representação QOC difere daquelas baseadas em IBIS e PHI, cujo propósito principal é registrar a deliberação do design. QOC foca nos três conceitos básicos indicados em seu nome: *questões* identificam os principais temas para estruturar o espaço de alternativas; *opções* fornecem as possíveis respostas a estas questões, e *critérios* formam as bases para avaliação e escolha entre as opções. Estes conceitos são ilustrados na figura abaixo.

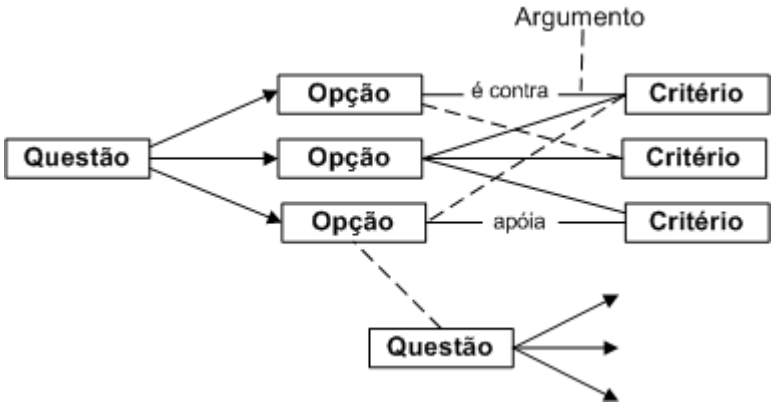

Figura 4 – O vocabulário da notação QOC

De acordo com Shum (1991), tornar o espaço de design explícito expressando-o como estruturas QOC e usando alguma forma de hipertexto, permite ao projetista reestruturar o espaço (reformulando questões) à medida que surgem novas idéias a respeito de como o design pode ser melhor visualizado. Essa ênfase em desenvolver uma representação logicamente coerente é talvez a característica marcante de QOC para *design rationale*. Algumas notações de *design rationale* propostas na literatura são baseadas em QOC, como as notações TEAM (Lacaze,

2005), DQN (Bramwell, 1995; Bramwell et al., 1995) e QUIDS (Hordijk et al., 2004).

A notação TEAM (*Traceability, Exploration and Analysis Mode*) estende QOC para tratar sistemas interativos e também abordar algumas das questões de usabilidade de QOC. Uma das extensões propostas, a inclusão das entidades "modelos de tarefas" e "cenários", tratam explicitamente dos modelos de tarefas e dos cenários que correspondem aos caminhos de execução nestes modelos. A análise e modelagem de tarefa são elementos críticos em métodos de design centrados no usuário, comumente usados na engenharia de sistemas interativos. Outras extensões, visando a usabilidade da notação, foram implementadas. Por exemplo, a representação dos requisitos expressos pelos usuários como "fatores" e a definição de métricas nos elos entre opções e critérios.

TEAM é apoiada por uma ferramenta chamada DREAM (*Design Rationale Environment for Argumentation and Modelling*) para a edição, análise e exploração de modelos. DREAM também apóia o gerenciamento das equipes de trabalho e seções, tratando as decisões de acordo com as pessoas responsáveis por elas e também de acordo com o momento no qual essas decisões foram tomadas.

DQN (*Design Question Notation*) é uma outra notação de *design rationale* para sistemas interativos baseada em QOC. Nesta notação cada elemento é descrito por uma linguagem formal, que é uma linguagem híbrida entre CSP (*Communicating Sequential Processes*) (Hoare, 1985) e *Action System* (Butler, 1992). A principal diferença entre as notações DQN e QOC é que DQN foi projetada para apoiar desenvolvimento formal (usando métodos formais) e QOC foi projetada para uso mais geral. A notação DQN mostra que *design rationale* também pode ser formal.

A unidade básica de DQN é uma questão de design, que tem um escopo e um conjunto finito e não vazio de opções de design. DQN inclui ainda o conceito de prioridade entre as propriedades definidas para as opções de design. As propriedades são comparáveis aos critérios definidos na notação QOC. Este conceito de prioridade permite ao usuário expressar suas necessidades com relação ao problema que está sendo tratado. De acordo com as prioridades definidas sobre as propriedades, é possível deduzir a opção de design selecionada.

Diferente de outras notações, a notação DQN define um conjunto de operadores (produto, união e interseção) que permitem ao programador estruturar,

remover e adicionar refinamentos para questões de design que ocorrem durante o desenvolvimento formal de um programa ou sistema interativo. Estes operadores tornam possível automaticamente identificar inconsistências no design, manipulando matematicamente as questões de design, seus escopos e opções de design. A notação define, ainda, um operador de avaliação que remove opções de uma questão de design de acordo com as preferências estabelecidas pelo usuário.

QUIDS (*QUality Indicators in Design Spaces*) é uma abordagem recente apresentada na forma de um espaço de design para arquiteturas de software. Esta notação é parecida com QOC, mas seu foco está no impacto e nas conseqüências das escolhas realizadas durante o design. QUIDS está dividida em três elementos: problemas de design; opções de solução para estes problemas; e indicadores de qualidade que influenciam as várias opções de solução. Uma vez que a solução é escolhida, a notação torna possível modelar os problemas inerentes a esta escolha (contexto do problema). QUIDS foi concebida visando responder problemas específicos do design de arquiteturas e os indicadores de qualidade definidos são específicos para este tipo de problema.

2.1.6. A Abordagem ADD

A abordagem ADD (*Active Design Documents*) (Garcia, 1992) se baseia na idéia de capturar o *rationale* através da transformação dos documentos de design de repositórios estáticos de dados em modelos computacionais do próprio design. Assim, o projetista interage com o modelo computacional do processo de design utilizado pelo sistema ADD para registrar diretamente os dados relevantes do *rationale*. ADD representa *design rationale* como uma combinação de *rationale* baseado em argumentação e em modelo. A recuperação desse *rationale* se dá através das explicações fornecidas pelo documento ativo a partir do modelo computacional e das informações registradas.

No sistema ADD o *rationale* é capturado usando o computador como um "aprendiz do projetista". ADD permite que o projetista defina os valores para os parâmetros de design que especificam o artefato a ser projetado e, baseado em seu modelo computacional, gera expectativas para as decisões de design à medida que estes valores são definidos. Se o projetista propõe um valor diferente do esperado,

ADD solicita que ele reconsidere o valor proposto ou ajuste o seu modelo para refletir o novo processo de design.

Em ADD o usuário pode explorar o *design rationale* de diversas maneiras: através da árvore de história, da árvore de dependência, de anotações e também formulando questões diretas. No entanto, as informações fornecidas pelo sistema ADD não são explicações bem escritas sobre as condições de design e muitas vezes não fazem referência a dados relevantes na base de conhecimento. Além disso, observou-se que os usuários da documentação não exploram completamente as perspectivas existentes e o conhecimento embutido no *rationale* recuperado. Isto justificou as extensões incluídas no sistema ADD, que deu origem ao ADD+ (Garcia & Souza, 1997).

ADD+ usa o mesmo modelo básico de ADD, mas melhora as interações do sistema com o usuário incluindo estruturas retóricas nos documentos ativos. Em ADD+, a riqueza do conhecimento armazenado na base de conhecimento do sistema ADD é organizada em um esquema de alto nível, baseado na Teoria de Estrutura Retórica (Mann & Thompson, 1987), e mapeada nas configurações de entrada e saída de tela que permitem a interação entre sistemas e usuários. Comparado a ADD, ADD+ tem um modelo de comunicação explícito com o usuário para transmitir mensagens que reforçam a usabilidade do modelo.

Em (Varejão et al., 1996a) os autores propõem uma outra extensão do sistema ADD chamada de ADD-GHS, que captura anotações estruturadas feitas pelo usuário durante o processo de design. Esta extensão incorpora ao sistema ADD um mecanismo que permite capturar *design rationale* através da combinação de um modelo formal do processo de design, com o histórico do processo e com argumentos informais obtidos através de anotações estruturadas. As anotações estruturadas são anotações informais em linguagem natural organizadas por uma estrutura de indexação que permite captar o contexto no qual essas anotações foram feitas.

Segundo os autores, as principais vantagens da integração do mecanismo de anotações estruturadas ao sistema ADD são permitir a documentação do conhecimento informal e proporcionar a formação de uma rede de informações que possibilita capturar outros tipos de relações, além das relações causais já recuperadas por ADD, por exemplo, relações temporais, estruturais e funcionais, dentre outras.

2.1.7. O Modelo DRIM

DRIM (*Design Recommendation and Intent Model*) (Pena-Mora, 1994) fornece primitivas para representar conhecimento de design em termos do processo de raciocínio usado pelos projetistas para gerar um artefato que satisfaça suas intenções de design. As principais primitivas do modelo são as intenções de design, as propostas de projeto baseadas nestas intenções e os artefatos que representam o produto em um processo de design. Uma intenção de design refere-se ao objetivo do projeto de software, às restrições envolvidas, à função considerada e à meta de projeto. Para cada intenção de design, o projetista pode apresentar diferentes propostas de projeto. Uma proposta de projeto inclui a recomendação do projetista e a justificativa do porquê a proposta é recomendada. Esta justificativa explica porque a recomendação satisfaz a intenção de design proposta. O artefato possui propriedades estruturais e comportamentais e inclui o sistema de software e seus componentes.

O modelo DRIM foi estendido em (Pena-Mora & Vadhavkar, 1996) para ser usado em um *framework* que combina padrões de projeto com *design rationale* para apoiar o design de sistemas de software reusáveis. O modelo estendido chamado DRIMER (*Design Recommendation and Intent Model Extended to Reusability*), integra os conceitos de design e reuso de código. Esta integração dá origem à abordagem "padrões por intenção", que se refere ao processo de selecionar padrões de projeto baseado nas suas intenções de design iniciais e então refinar a escolha do padrão usando restrições específicas. O *framework* proposto atua como uma ferramenta de design que facilita o reuso de software por usar uma biblioteca de componentes de software testados, registrar e permitir uma fácil recuperação das decisões tomadas durante o processo de design.

2.1.8. O Sistema SEURAT

O sistema SEURAT (*Software Engineering Using RATionale*) (Burge & Brown, 2003, 2004) foi desenvolvido para apoiar o uso de *design rationale* em manutenção de software. Seu principal objetivo é apoiar a visualização de *rationale* e a realização de inferências para apontar questões não resolvidas ou inconsistências geradas pelas modificações realizadas no software. SEURAT usa uma representação de *rationale* baseada em DRL chamada

RATSpeak. Nesta representação as alternativas para cada problema de decisão podem ser discutidas por suas relações com seus requisitos, com outras alternativas e com suposições e restrições que apóiam ou rejeitam as alternativas. Um dos elementos chave de RATSpeak é a ontologia de argumentos, uma hierarquia de tipos de argumentos comuns que servem como tipos de restrições que podem ser usadas em um software. Um dos tipos de argumentos descreve se uma alternativa satisfaz ou viola um requisito. Outros argumentos referem-se a suposições feitas ou dependências entre alternativas. Por fim, um outro tipo especifica se uma alternativa apóia ou rejeita um requisito não funcional. Estes argumentos compõem uma lista de razões para as escolhas de design feitas pelos desenvolvedores e mantenedores do software.

O sistema SEURAT está integrado com o ambiente de desenvolvimento Eclipse[3], como um *plug-in* Java, para tornar a captura e o uso de *rationale* integrado com o processo de desenvolvimento de software. Desta forma, os usuários podem visualizar o *design rationale* existente e informar o novo *rationale* para as modificações realizadas no código usando uma mesma ferramenta.

2.2. Principais Problemas com as Abordagens Existentes

Como mencionado anteriormente, muitas das abordagens existentes para *design rationale* geram representações incompletas ou informais. As representações de *design rationale* geradas pelas abordagens IBIS, PHI, QOC e suas extensões são consideradas incompletas porque elas não representam explicitamente as decisões tomadas pelos projetistas durante o design. Não existe em seus vocabulários um elemento específico para descrever decisões. As decisões tomadas muitas vezes se confundem com os problemas de design tratados ou temas apresentados para o design. A aceitação ou não de uma alternativa como uma solução para o design é representada como uma propriedade da alternativa. Isto dificulta a identificação de quais alternativas foram aceitas e quais foram rejeitadas e o porquê. Além disso, os elementos usados para descrever os argumentos ou justificativas contra ou a favor das alternativas propostas não deixam claro se o que está representado é a justificativa final para ter escolhido ou rejeitado uma dada

[3] http://www.eclipse.org

alternativa, ou se os argumentos apresentados para a alternativa compõem uma justificativa geral para a decisão tomada.

Geralmente, as representações geradas por estas abordagens não são descritas por uma linguagem de representação formal com uma semântica expressiva. Todo o *rationale* é registrado e recuperado como pedaços de texto não-interpretado fornecidos pelos usuários, o que resulta em uma representação de *design rationale* genérica e informal. Isto impede a realização de operações computáveis sobre o *rationale* registrado, dificultando seu reuso em novos contextos de design. Os tipos de operações que podem ser realizadas sobre uma representação de *design rationale* por um sistema determinam fortemente a facilidade de gerenciar o *rationale* registrado e como este *rationale* poderá ser recuperado e reusado em outros designs. Normalmente, as operações implementadas nos sistemas de apoio a *design rationale* permitem apenas a consulta e indexação do *rationale* registrado e não consideram, por exemplo, a integração de soluções registradas nos *rationales* de artefatos similares.

Muitos dos sistemas de *design rationale* que utilizam notações baseadas em argumentação, como Compendium, SIBYL, DREAM e SEURAT, não se beneficiam da semântica formal fornecida pelos métodos de design para representar *design rationale*. Além disso, com exceção do sistema SEURAT, esses sistemas também não consideram a integração da representação de *design rationale* com as ferramentas de design existentes. Assim, grande parte do *rationale* precisa ser informada interativamente pelos usuários usando uma ferramenta específica para *design rationale*, o que gera um esforço adicional ao trabalho realizado pelos usuários e reduz a aceitação desses sistemas.

2.3. Requisitos para a Representação de Design Rationale

Com base nas características e nos principais problemas identificados nas abordagens apresentadas, geramos a seguinte lista de requisitos para a representação de *design rationale*:

1. *Representação explícita de decisões.* Uma mesma alternativa de design pode ser considerada, ou não, uma solução para questões de design diferentes. Portanto, as decisões tomadas sobre a aceitação ou não desta

alternativa como uma solução para cada questão de design devem ser registradas separadamente.

2. *Distinção entre argumentos e justificativa final*. Algumas vezes o projetista decide aceitar uma idéia de solução que possui apenas argumentos contrários. Podem existir também idéias que, pelo número de argumentos a favor, parecem representar uma melhor solução de design do que a idéia escolhida pelo projetista para uma dada questão. Isto pode indicar que existe um *rationale* que não está sendo registrado ou que a escolha desta idéia de design deve ser reconsiderada pelo projetista. Geralmente, o *rationale* que está faltando pode ser registrado como um novo argumento. Porém, em alguns casos não se trata de um novo argumento para a idéia de solução proposta, mas sim de uma justificativa final para a decisão tomada pelo projetista após analisar os argumentos apresentados. Por exemplo, esta justificativa final poderia registrar os motivos pelos quais o projetista decidiu usar uma idéia de solução, apesar de todos os argumentos apresentados para ela serem contrários à sua aceitação.

3. *Integração da argumentação com as descrições dos artefatos gerados*. A complexidade e o volume de informações presentes nas representações de *design rationale* torna difícil para o projetista compreender as diversas idéias de solução e o resultado das decisões tomadas. A relação entre a argumentação registrada e o artefato produzido pode facilitar esta compreensão e ajudar o projetista a decidir se vale a pena reusar um artefato ou não.

4. *Registro de informações sobre o contexto de design*. Registrar apenas a argumentação usada pelos projetistas não é suficiente para compreender o conhecimento usado por eles durante o design. É necessário também ter informações sobre o contexto de design, como por exemplo, o método de design utilizado, as atividades realizadas e os responsáveis pelas decisões tomadas.

5. *Integração do design rationale com o método de design*. Representações genéricas e informais de *design rationale* dificultam o uso do *rationale* em novos designs. A integração do *rationale* com a semântica formal

definida pelos métodos de design torna o conteúdo registrado mais rico e expressivo, possibilitando a realização de operações computáveis capazes de apoiar novos usos de *design rationale*, como por exemplo, a combinação de *rationales* de artefatos similares.