5
Arquitetura da Cadeia de Modelagem de MDRM

Com MDRM, a tarefa de especificação da lógica de gerenciamento de recursos em ambientes distribuídos se torna um processo de modelagem bem definido. Cada etapa da *cadeia de modelagem de MDRM* se apóia em métodos e ferramentas que agregam facilidades para a visualização, construção, validação, transformação, implantação e manutenção dos modelos de gerenciamento de recursos. Este capítulo apresenta as etapas da cadeia de modelagem de MDRM, seus requisitos, protótipos propostos, assim como questões relacionadas às cadeias de desenvolvimento complementares, como os subdomínios de construção de estratégias de escalonamento e de admissão.

5.1.
Visão Geral

A cadeia de modelagem de MDRM não deve ser vista como um conjunto fechado de etapas de desenvolvimento, uma vez que a evolução da técnica vai trazer, certamente, novas funcionalidades à cadeia, novos subdomínios e novos requisitos. A Figura 11 apresenta uma visão geral da cadeia de modelagem atualmente em uso.

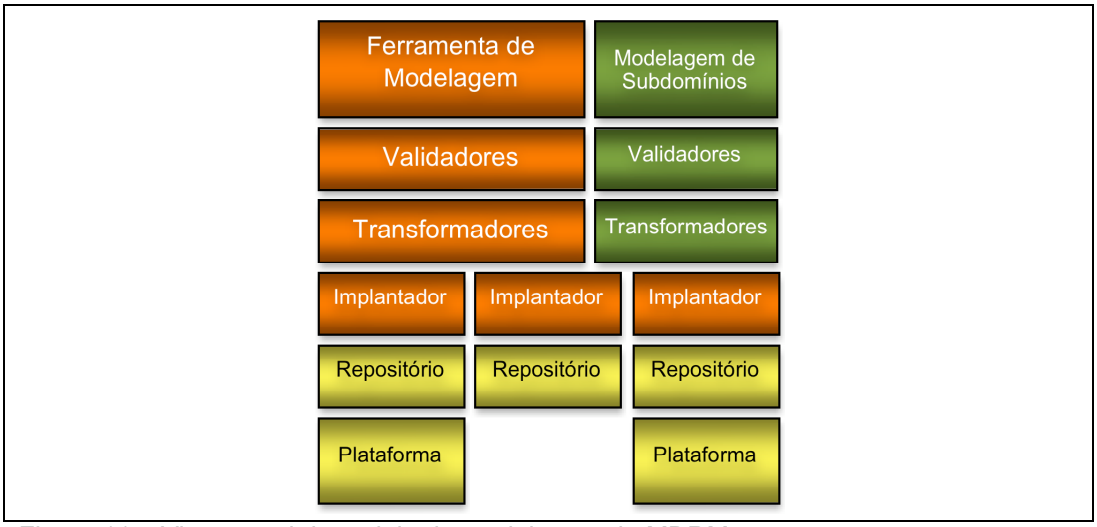

Figura 11 – Visão geral da cadeia de modelagem de MDRM

A cadeia de modelagem de MDRM é composta por: uma ferramenta para a construção de modelos baseada na linguagem Pan; validadores de especificações de modelos; transformadores de modelos; implantadores de modelos; repositórios de modelos e plataformas. As etapas da cadeia de modelagem encontram-se descritas individualmente nas próximas seções.

Dado que a linguagem Pan inclui os construtos necessários tanto para a modelagem quanto para a manutenção dos modelos, as mesmas ferramentas citadas podem também ser utilizadas na modificação e evolução dos modelos.

Paralelamente às ferramentas de modelagem em torno da linguagem Pan, a cadeia considera a existência de ferramentas pertencentes aos demais subdomínios de MDRM. A linha de produção em Pan se une à linha de produção dos subdomínios no momento da implantação do modelo, quando as referências aos componentes desenvolvidos em subdomínios são resolvidas pelos implantadores.

Com a cadeia proposta, pode ser observado que um modelo de gerenciamento de recursos possui um ciclo de vida bem definido, desde sua especificação até sua implantação e manutenção. Esse ciclo de vida encontra-se descrito a seguir e ilustrado pela Figura 12:

1. *Em edição*: O modelo está sendo criado, em edição por meio de alguma ferramenta de modelagem, período em que ele possui especificação ainda inconsistente em relação ao meta modelo e aos modelos implantados que ele referencia (importa);
2. *Validado*: O modelo possui especificação em concordância com todas as restrições sintáticas e semânticas definidas pela linguagem Pan, estado este garantido por ferramentas de validação. O modelo pode voltar a ser modificado antes do início do processo de implantação, deixando de ser válido e voltando ao estado de edição;
3. *Transformado*: O modelo foi submetido a ferramentas de transformação, que iniciam o processo de implantação. O modelo validado encontra-se transformado em outros modelos, também validados, voltados a armazenamento em repositórios. Esse tipo de transformação somente é necessária se forem utilizados repositórios distribuídos.
4. *Implantado*: Os modelos transformados foram submetidos por implantadores aos respectivos repositórios que os abrigarão. Os modelos ficam disponíveis para referências futuras visando reuso de suas

especificações, integração com outros modelos, ou para manutenções em suas estruturas.

5. *Evoluído*: O modelo implantado em um repositório foi modificado por quaisquer operações de manutenção submetidas por um implantador. Manutenções sucessivas fazem o modelo permanecer neste estado indefinidamente.

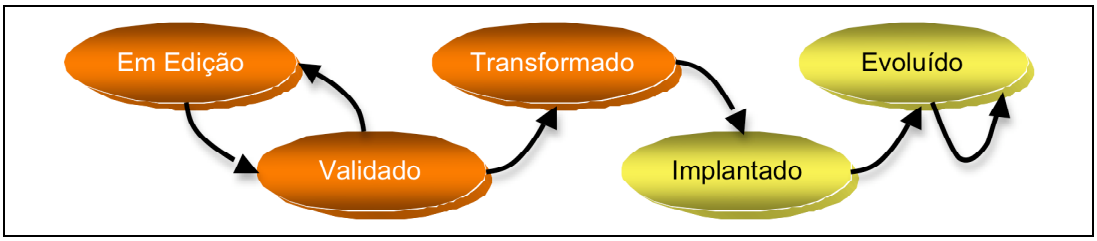

Figura 12 – Ciclo de vida de um documento de modelo Pan

Nota-se que, uma vez validado, o modelo poderia ser submetido a testes de simulação, para verificações de desempenho e corretude da lógica de gerenciamento especificada. Ferramentas de simulação, apesar de não estarem incluídas na cadeia de modelagem neste momento, podem ser adicionadas ao processo para agregar maior confiabilidade aos modelos.

Observa-se, também, que um ciclo de vida semelhante é atribuído a documentos de manutenção de modelos, no que diz respeito à edição, validação e transformação, conforme ilustrado pela Figura 13. A diferença está no fato de que documentos de manutenção não são armazenados em repositórios, mas sim, os resultados das operações por eles especificadas, quando bem sucedidas, se tornam parte dos respectivos modelos modificados.

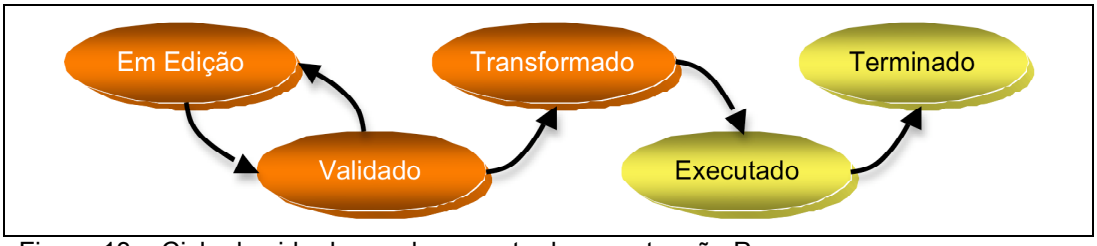

Figura 13 – Ciclo de vida de um documento de manutenção Pan

5.2. Subdomínios

Os principais subdomínios de MDRM compreendem o desenvolvimento de estratégias de escalonamento de recursos e de estratégias de admissão. Apesar de

não representarem exatamente o foco do presente trabalho, alguns comentários acerca de seus processos de desenvolvimento podem ser traçados.

Ambos subdomínios podem ser regidos por técnicas de desenvolvimento de software dirigido por modelos, para trazer portabilidade, melhores abstrações por meio de DSLs e, notavelmente, segurança. Por isso, a Figura 11 ilustra a linha de produção desses subdomínios por meio de termos comuns a MDSD, tais como modelagem, validação e transformação.

O caso do subdomínio de estratégias de escalonamento é o mais crítico deles, uma vez que, normalmente, os algoritmos de escalonamento precisam ser implantados em níveis muito baixos nas plataformas, como em núcleos de sistemas operacionais, micro-código de *network processors*, módulos de roteadores extensíveis, etc (Gottlieb, 2002). Nesses níveis, requisitos como desempenho e segurança são fundamentais. No caso do subdomínio de estratégias de admissão, por outro lado, pode ser facultada sua implantação em tais níveis, a não ser que os critérios de avaliação incluam informações sobre recursos e fluxos acessíveis apenas dessa forma em certas plataformas-alvo.

A definição de DSLs para a modelagem de estratégias de escalonamento e de admissão traz diversos benefícios (Barreto, 2002), entre os quais a facilidade de especificação de estratégias e a avaliação de questões de segurança são os principais. Apesar de ter sido desenvolvido com foco específico sobre o escalonamento de processos, certos aspectos do trabalho citado podem ser tomados como guia para a definição da cadeia de desenvolvimento de estratégias de escalonamento em geral.

É importante mencionar que Barreto (2002) define uma DSL e uma infra-estrutura de suporte em plataforma completamente orientadas a eventos, o que facilita muito o processo de modelagem e validação. Por outro lado, isso levou a profundas modificações no núcleo do sistema operacional alvo (Linux), cuja API de escalonamento de processos não é orientada a eventos. Sua decisão levou à necessidade de modificação de todo o escalonador de processos do sistema, e de todos os pontos de chamada ao escalonador ao longo do núcleo, o que se mostrou tarefa árdua. Esses aspectos devem ser contrabalançados no projeto da DSL e sua respectiva complexidade de suporte em plataformas.

Particularmente, o escalonamento de processos é, sem dúvida, o caso mais complexo de escalonamento a ser tratado, dado que diversas partes do sistema

operacional influenciam ou são influenciadas por ele. Normalmente, o código de escalonamento de processos em núcleos de propósito geral possui pouca modularidade, pois é criado pensando no ganho de desempenho que soluções fortemente acopladas provêem. No entanto, recentemente, o sistema operacional Linux teve seu escalonador de processos redesenhado para obter, internamente, uma estrutura modular de especificação de estratégias de escalonamento (políticas de escalonamento) (Kumar, 2008). Elas continuam não sendo carregáveis em tempo de execução, nem implementáveis via módulos de núcleo, mas apresentam uma interface de programação bem definida, o que pode colaborar na identificação das necessidades de especificação da DSL dedicada a estratégias de escalonamento.

Nota-se que, para que uma mesma DSL possa ser capaz de especificar estratégias de escalonamento para recursos como CPU, filas de rede e outras filas de E/S, ela terá que uniformizar conceitos comuns no escalonamento desses recursos. Ao mesmo tempo, para que não perca seu poder de expressão, a DSL deve oferecer também as abstrações específicas a cada tipo de recursos, organizando-as em áreas funcionais, por exemplo.

No caso das estratégias de admissão, a tarefa é muito mais simples, uma vez que tal estratégia não representa dependência para as demais áreas de um sistema operacional. A DSL para estratégias de admissão deve se concentrar exclusivamente na especificação de parâmetros e comparações que levem à avaliação da demanda submetida.

Por isso, transformações de modelos para código específico de plataforma para estratégias de admissão tendem a ser simples. Como pôde ser visto, o mesmo não pode ser dito de estratégias de escalonamento. O presente trabalho considera as linhas de produção dos subdomínios como soluções existentes, e, portanto, faz uso de código específico de plataforma para as estratégias de escalonamento e de admissão, simulando a etapa final dessas linhas.

Em uma cadeia MDRM ideal, no entanto, somente no momento da implantação de modelos Pan deveria ser solicitada a transformação dos modelos de escalonamento e de admissão referenciados em código específico das plataformas envolvidas. Outro requisito desejável é a geração automática de especificações de parâmetros e de categorias de serviço compatíveis com as estratégias modeladas, que possam ser disponibilizadas prontas para uso na

construção de modelos Pan, tal como descrito na Seção 4.2.5. Da mesma forma, relatórios de compatibilidade iniciais referentes às estratégias modeladas podem ser gerados, em conformidade com a descrição da Seção 4.2.4.

5.3. Ferramenta de modelagem Pan

Uma ferramenta de modelagem Pan é aquela usada para a criação de modelos e de documentos de manutenção, para uso pelos diversos atores possíveis em cenários distribuídos. Por isso, deve incluir diferentes visões e várias funcionalidades que facilitem a tarefa de modelagem.

A visão textual de um modelo de gerenciamento de recursos representa a listagem do código Pan que descreve o modelo. Uma edição textual rica pode ser oferecida por meio de recursos tais como: marcação sintática por diferenciação de cores (*syntax highlighting*); indentação automática para fácil acompanhamento da hierarquia de elementos; ocultação de partes da hierarquia de elementos (*code folding*), entre outros. Esses recursos são normalmente encontrados em simples editores de texto com suporte a código XML. Um recurso extremamente desejável é o complemento automático de código (*autocomplete*) que, baseado no *schema* XML da linguagem, lista quais são os elementos e atributos possíveis de serem especificados em um dado contexto do código. O complemento automático acelera tanto o desenvolvimento de modelos quanto a curva de aprendizado da linguagem.

A visão gráfica de um modelo representa por meio de imagens e outras estruturas gráficas os elementos de um modelo, como florestas, árvores de recursos virtuais e recursos virtuais. O objetivo da visão gráfica é dispensar, tanto quanto possível, a especificação textual de modelos, de forma a incluir atores não-especialistas, agilizar o desenvolvimento de modelos e diminuir a possibilidade de erros. Uma DSL gráfica é necessária para uma representação fiel de todos os elementos e relacionamentos do meta modelo por meio de notação pictórica. Todas as tarefas de modelagem são mapeadas sobre interações com botões e áreas de desenho em uma interface gráfica de usuário.

Por fim, a visão estrutural de um documento Pan é uma árvore de elementos XML e seus atributos, de fácil navegação e modificação. Seu grau de facilidade de uso é mediano, entre a visão textual e a visão gráfica.

De forma ideal, as visões devem estar integradas e sincronizadas, para que o projetista rapidamente passe de uma visão a outra percebendo que as especificações descritas estão consistentes. A integração da ferramenta de modelagem com validadores, descritos na próxima seção, é mais uma facilidade bem-vinda, pois, assim, possíveis erros retornados na validação podem ser indicados claramente nas visões.

Outra funcionalidade desejável é a disponibilização de interfaces simples, em forma de formulário, dedicadas a tornar transparente ao projetista a complexidade de especificação de categorias de serviços, regras de controle de acesso e de classificação, conforme apresentadas nas seções 4.2.5, 4.2.6 e 4.2.7, respectivamente.

Conforme mencionado anteriormente, a ferramenta de modelagem deve ser capaz de manipular modelos e manutenções de modelos. Para a criação de um modelo, a ferramenta deve gerar um "documento Pan de especificação de modelo de gerenciamento de recursos", em conformidade com a descrição apresentada na Seção 4.2.1. Para a descrição de uma manutenção, a ferramenta deve criar um "documento Pan de especificação de manutenção de modelos de gerenciamento de recursos", em conformidade com a descrição apresentada na Seção 4.2.2. Em ambos os casos, o documento deve ser salvo localmente, para aguardar o acionamento das demais etapas da cadeia de modelagem.

Um protótipo de ferramenta de modelagem Pan foi desenvolvido para ilustrar as funcionalidades descritas nesta seção. Plat (*Pan language authoring tool*), como foi denominado, foi implementado como um conjunto de *plug-ins* para o ambiente Eclipse (Eclipse.org, 2007), agregados a outros, como OxygenXML Editor (SyncRO Soft, 2007). A Figura 14 apresenta uma tela capturada do ambiente Plat em execução. Dentre as funcionalidades desejáveis descritas, Plat somente não implementa i) a sincronização entre as visões, devendo o projetista salvar a visão para ver replicadas suas alterações e ii) a visão gráfica completa, dado que ainda não foi definida uma DSL gráfica adequada ao meta modelo VRT.

Na Figura 14, a janela ① lista os documentos Pan disponíveis localmente em um dado diretório. Quando um documento é selecionado para abertura (clique duplo), abrem-se uma visão gráfica e uma visão textual referentes ao documento.

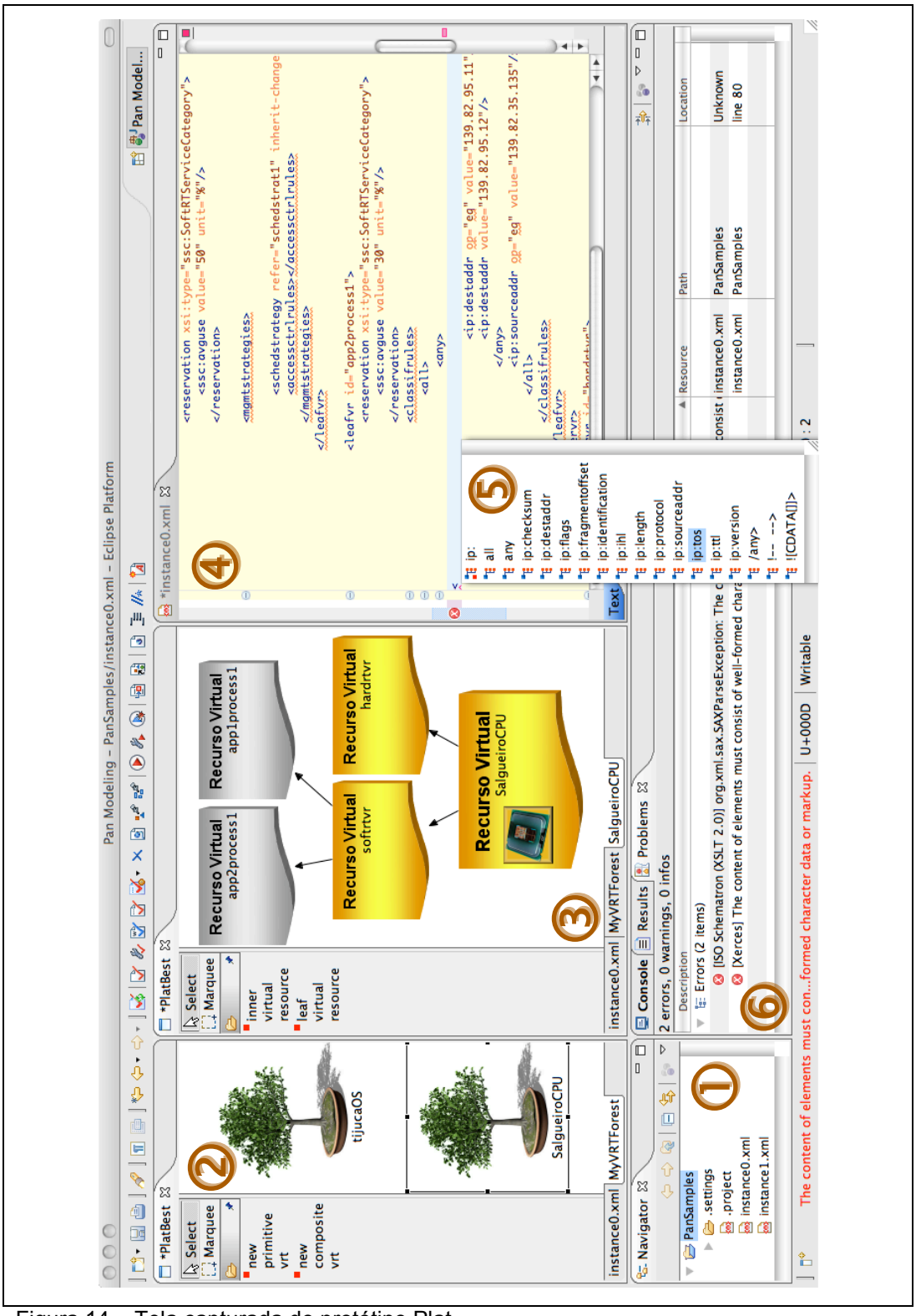

Figura 14 – Tela capturada do protótipo Plat

A visão gráfica é composta pelas janelas ② (visualização de um floresta) e ③ (visualização de uma árvore de recursos virtuais). A janela ④ é a visão textual do documento aberto, provida pelo plug-in OxygenXML, que inclui marcação de sintaxe, ocultação de blocos de código, indentação automática, entre outras funcionalidades. A lista *drop-down* ⑤ foi aberta no momento em que um novo elemento Pan começou a ser definido, e é resultado da capacidade de OxygenXML para o complemento automático de código baseado em *schemas* XML (no caso, o *schema* de Pan). A lista no exemplo exibe um conjunto de regras de classificação, no escopo dos *namespaces* conhecidos, que poderiam ser declaradas naquela linha. A integração com os validadores XML embutidos na plataforma Eclipse e no plug-in OxygenXML possibilita que os erros encontrados na validação sejam listados na janela ⑥.

5.4. Validadores

Validadores de documentos Pan se destinam à verificação estrutural, sintática e semântica do documento, tendo como base o conjunto de restrições definidos na especificação da linguagem. Essas restrições descrevem condições para a boa formação de um documento Pan e para a construção de modelos em conformidade com o meta modelo VRT.

A validação de um documento Pan é, na realidade, um processo definido em dois passos. O primeiro, de validação estrutural e sintática, consiste na verificação do documento contra o *schema* XML da linguagem Pan (listado no Anexo I). Existem vários validadores de documentos XML baseados em *schemas* disponíveis de forma livre, tais como *Apache Xerces 2* (Apache.org, 2007).

O segundo passo consiste na validação sintática e semântica, por meio da verificação do documento contra o *schema* complementar da linguagem Pan, descrito em *Schematron* (listado no Anexo II). Os validadores de código aberto para *Schematron* são, na realidade, documentos de transformações XSLT que devem ser usados por um processador, tal como Saxon-B 9.0 (Kay, 2007b), tendo como entrada o documento Pan a ser validado. O resultado da transformação é uma lista de erros, se existirem, que descrevem os pontos de violação das regras.

Se o documento Pan passa pelos dois passos de verificação, ele é dito ser um documento válido e pode, assim, ser submetido para implantação. Por outro lado, basta que um dos passos retorne erros para que o documento seja dito inválido, pelo menos até que os erros reportados sejam corrigidos e novo processo de validação seja iniciado.

Se a validação for integrada à ferramenta de modelagem, a região separada para o reporte de erros de validação será preenchida com os erros retornados pelos validadores. Adicionalmente, a integração da ferramenta de modelagem com validadores pode ser ainda maior, por meio de mecanismos como a validação ativa, no qual o documento ou a parte dele sendo modificada é submetida à validação durante a digitação das declarações de elementos Pan na visão textual. Essa funcionalidade encontra-se ilustrada na Figura 15. O erro reportado na figura foi capturado no segundo passo de validação, indicando que o identificador colocado no atributo `refer` não é uma estratégia de escalonamento.

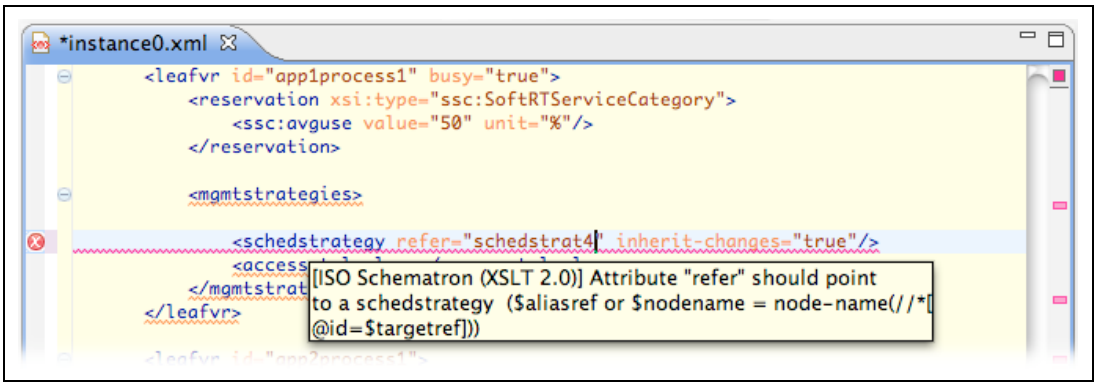

Figura 15 – Validação ativa na visão textual

5.5. Transformadores

Transformações de modelos descritos em Pan podem originar outros modelos também em Pan, por exemplo para desmembrá-los em descrições mais concisas, cada qual dedicada a ser implantada em um certo repositório. Essa política de distribuição de modelos somente faz sentido se o ambiente implementa repositórios distribuídos. Nota-se, portanto, que é tarefa dos transformadores arbitrarem sobre a distribuição de modelos entre repositórios, por meio da lógica de transformação especificada.

Essas transformações modelo-para-modelo na cadeia de modelagem de MDRM são, normalmente, muito simples, consistindo apenas de substituições de cadeias de caracteres nos documentos dos modelos e geração de outros documentos com as cadeias substituídas. A especificação de uma transformação ganha forma por meio de um documento XSLT, ideal para esse tipo de tarefa. Um processador XSLT, baseado nas regras do documento de transformação, recebe como entrada um documento Pan e gera outros documentos Pan, dependendo das substituições especificadas. Processadores XSLT de código aberto são amplamente disponíveis, tais como Saxon-B 9.0 (Kay, 2007b).

Há também um outro tipo de transformação, modelo-para-código, no qual um modelo Pan que está por ser submetido a um repositório deve também ser traduzido para código específico de uma ou mais plataformas envolvidas, provavelmente por possuir VRTs primitivas que especificam a lógica de escalonamento de um recurso real. Juntamente com a solicitação de implantação desse modelo, o implantador do repositório encaminha os códigos específicos de plataforma, se for o caso, para implantação nas plataformas envolvidas.

5.6. Implantadores, repositórios e plataformas

Um implantador de modelos Pan é responsável por encaminhar um modelo Pan derivado de uma transformação para o seu repositório de modelos. Se todas as árvores de recursos virtuais ali especificadas são compostas, provavelmente não haverá necessidade de implantação em plataformas, mas apenas no repositório.

Por outro lado, se houver VRTs primitivas no modelo submetido à implantação, e se essas VRTs contiverem lógica de escalonamento, o implantador acionará novas transformações para a geração de código específico a cada plataforma definida como alvo dessas árvores.

Plataformas representam os frameworks para suporte ao meta modelo VRT em algum nível de abstração. Um caso específico de plataforma é um sistema operacional e sua infra-estrutura de suporte a VRT, tal como a especificação VRT-FS Linux descrita no Capítulo 6, a seguir. Nesse caso, todas as funcionalidades previstas no meta modelo devem ser suportadas, uma vez que esse tipo de plataforma lida tanto com recursos reais, quanto com recursos

compostos. Fica evidente que a simples representação de um documento Pan nessas plataformas não é o bastante para configurar a lógica de compartilhamento dos recursos, e, por isso, é necessária a geração de código mencionada no parágrafo anterior.

Um outro caso específico de plataforma é o repositório, que armazena os modelos Pan, mesmo aqueles instanciados em sistemas operacionais. Para os modelos que são instanciados em sistemas operacionais, repositórios representam somente um ponto de armazenamento para referência futura em manutenções, por exemplo.

Por outro lado, para os outros modelos, repositórios são mais que uma base para a disponibilização de documentos Pan, uma vez que as árvores e florestas ali armazenadas possuem alguma lógica de gerenciamento de recursos, como estratégias de admissão e regras de controle de acesso. Essa lógica deve ser acionada quando um repositório recebe uma operação de manutenção a ser executada sobre um de seus modelos armazenados. Por isso, repositórios oferecem implementações de controladores de admissão e controladores de acesso.