Silva, Alberto Manuel Rodrigues da
Costa, Marco Bruno Correio, 1971-

ReacT-MDD : rastreabilidade reactiva de artefactos no desenvolvimento de sistema de informação

http://hdl.handle.net/11067/5218

Metadados

Data de Publicação 2010

Resumo A aproximação emergente MDD (Model Driven Development) preconiza o desenvolvimento de código a partir de modelos descritos pela linguagem UML ou seus dialectos. São necessários mecanismos que permitam a rastreabilidade entre os modelos criados, a vários níveis conceptuais, e o código gerado. Os modelos podem ser explicitados por diferentes diagramas, com possíveis relações entre si. Apesar de já existirem algumas ferramentas que realizam a sincronização entre código e modelos, em pequena escala,...

The MDD (Model Driven Development) approach, favors the development of code starting at models, eventually described by the UML language. Not only the produced code, but also the remaining artifacts are part of the solution, at the development, operation and maintenance stages. Models may be defined by diagrams, with different relations between them. There are already some tools that achieve some kind of synchronization between code and models, in small scale, but this is not the main focus of ...

Palavras Chave Bases de dados multidimensionais

Tipo article

Revisão de Pares Não

Coleções [ULL-FCEE] LEE, n. 10 (2010)

REACT-MDD – RASTREABILIDADE REACTIVA DE ARTEFACTOS NO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

Marco Costa
Doutor em Engenharia Informática e de Computadores
(Instituto Superior Técnico)

Alberto Rodrigues da Silva
Doutor pelo Instituto Superior Técnico
Investigador no INESC - ID e
Professor Associado no IST/UTL

Resumo: A aproximação emergente MDD (Model Driven Development) preconiza o desenvolvimento de código a partir de modelos descritos pela linguagem UML ou seus dialectos. São necessários mecanismos que permitam a rastreabilidade entre os modelos criados, a vários níveis conceptuais, e o código gerado. Os modelos podem ser explicitados por diferentes diagramas, com possíveis relações entre si. Apesar de já existirem algumas ferramentas que realizam a sincronização entre código e modelos, em pequena escala, são porém uma tecnologia proprietária e fechada ao utilizador. Propõe-se neste trabalho um modelo teórico para a rastreabilidade reactiva, consubstanciado pela *framework* ReacT-MDD.

***Palavras-chave*:** MDD, UML, Rastreabilidade.

Abstract: The MDD (Model Driven Development) approach, favors the development of code starting at models, eventually described by the UML language. Not only the produced code, but also the remaining artifacts are part of the solution, at the development, operation and maintenance stages. Models may be defined by diagrams, with different relations between them. There are already some tools that achieve some kind of synchronization between code and models, in small scale, but this is not the main focus of this kind of tools. In this research paper, we propose a mechanism that allows traceability between all artifacts during all project stages, not regarding its conceptual level. The ReacT-MDD framework is a contribution to link all kinds of artifacts in the application development context.

1 Introdução

O desenvolvimento de sistemas de informação é um campo suficientemente complexo para que exista uma grande diversidade de ferramentas usadas em todas as fases do ciclo de vida das aplicações que dele fazem parte e.g., compiladores, editores de texto, editores gráficos, controladores de versões, instaladores, produtores de instaladores, geradores de código, geradores de documentação de código, gestores de requisitos. Todas estas ferramentas produzem elas próprias novos dados que devem ser mantidos ao longo do tempo, com possíveis alterações. Infelizmente a sua integração está longe de ser total e, mesmo os produtos mais completos, não cobrem todas as necessidades. A manutenção dos artefactos gerados (i.e. os produtos resultantes da utilização de cada uma das ferramentas) torna-se assim em mais um aspecto a ter em conta no próprio processo de desenvolvimento. Considerando o processo de desenvolvimento de uma perspectiva sistémica, o aumento de artefactos pode levar ou não a um aumento da entropia do sistema. Se existirem ferramentas que os interligam de forma automática, relacionando-os inequivocamente, os conjuntos entretanto criados comportam-se como um único artefacto naquilo que diz respeito à sua coerência (mantendo-se constante a entropia do sistema, segundo a mesma analogia). Para o problema em causa, é relevante considerar uma aplicação desenvolvida através de um processo envolvendo o uso de modelos, ao longo de diversas fases. Os modelos criados podem referir-se a diversos níveis conceptuais, sendo o sistema modelado subjacente o mesmo.

Propõe-se a divisão dos artefactos do sistema aplicacional em dois tipos: artefactos conceptuais e operacionais. O primeiro tipo representa todos os artefactos que não estão directamente envolvidos na operação do sistema (e.g. classe de domínio, tabela relacional contida num diagrama de tabelas, descrição de um requisito). Inversamente o segundo tipo representa todos os elementos que podem eventualmente existir durante a operação da aplicação (e.g. classe C# da aplicação, tabela relacional numa base de dados usada pela aplicação). Desta classificação resulta que as alterações ao sistema, i.e. aos seus artefactos, podem ser sobre qualquer um dos tipos anteriores. Essas alterações sobre um dado artefacto podem implicar a alteração de outros artefactos que

estejam com ele relacionados (mesmo que estes pertençam a diferentes tipos de artefactos). Quando as alterações são feitas sobre artefactos conceptuais é muito provável que estas tenham de ser propagadas para os artefactos operacionais correspondentes. Os processos de geração automática favorecem este tipo de alterações permitindo em muitos casos uma actualização simples dos artefactos operacionais subjacentes [5, 10].

Das alterações, normalmente mais frequentes, sobre artefactos operacionais podem advir incoerências entre estes e os artefactos conceptuais, caso não sejam acauteladas as acções necessárias. Destas incoerências resulta um óbvio desfasamento entre a documentação do sistema e os seus artefactos operacionais. Uma alteração à estrutura de uma aplicação pode ter consequências não evidentes à partida. Como exemplo, tome-se uma alteração ao nome de uma classe Java. Não só todas as referências da aplicação a essa classe têm de ser alteradas (o que é conseguido usualmente pelo IDE) como pode ser necessário alterar todos os diagramas e demais artefactos documentais que referenciam essa classe. Alternativamente pode também ser necessário deixar inalterados os nomes correspondentes em modelos que lhe dizem repeito, estabelecendo-se uma relação entre eles. Essa relação entre artefactos com conteúdo diferente (na medida em que podem ser outros atributos para além do nome a serem alterados, e.g., o tipo de acesso) após ser estabelecida pode ser verificada e mantida.

Para resolver os problemas anteriores a tecnologia existente não é suficiente. Existem já geradores de código a partir de modelos [5, 6] mas essa abordagem não resolve o problema de ter de manter o código já realizado. É necessário algo mais, um modelo de coerência comum e uma nova classe de aplicações que permita, de forma aberta ao utilizador, definir as regras de rastreabilidade necessárias entre os diversos elementos do sistema em causa, mantendo-se assim a sua coerência. Deverá igualmente estar associado um mecanismo que verifique a todo o momento essa coerência, permitindo tomar as decisões necessárias. Neste trabalho é apresentado um modelo teórico para o tratamento da rastreabilidade reactiva. O modelo descrito foi aplicado na *framework* ReacT-MDD [1,2] cujos conceitos introdutórios são igualmente explicados.

2. Artefactos e Rastreabilidade

Os artefactos produzidos no âmbito do desenvolvimento de aplicações não devem ser considerados apenas meios para atingir um fim, neste caso a aplicação propriamente dita. Todos os diagramas, descrições textuais, ou outros artefactos necessários à compreensão do sistema, são também parte da solução do problema. Estes artefactos devem, por isso, permanecer actualizados apesar de todas as alterações produzidas pela utilização da aplicação ou por outros factores externos (e.g., novos requisitos resultantes de mudança no negócio). A justificação para esta afirmação é evidente se se considerar que a vida útil de uma

aplicação pode conter um número elevado de versões e o seu desenvolvimento deve ser constantemente dirigido por modelos. Se os modelos não reflectirem o estado actual do código implementado, o desenvolvimento pode ser feito sem que os modelos sejam actualizados, com as consequências óbvias para o futuro da aplicação. Nesse caso os modelos de nada servem, constituindo apenas um conjunto de esquemas e textos sem sentido. Porém, à medida que forem sendo produzidas novas alterações, a complexidade das aplicações for aumentando, as soluções encontradas tornarem-se cada vez mais particulares ao contexto específico, e as equipas forem enquadrando novos elementos, será cada vez mais questionável a opção tomada. Neste ponto pode já ser tarde demais para a equipa voltar a actualizar os modelos anteriormente criados, sendo eventualmente mais simples criar uma nova aplicação de raiz, eliminando os erros de concepção entretanto encontrados.

Apesar desta constatação, a experiência mostra que nem sempre existe uma consciência por parte dos profissionais do sector para a importância da totalidade dos artefactos [3, 4]. O problema não se resume a tornar a documentação de uma aplicação coerente com a aplicação propriamente dita. Pode acontecer igualmente que duas aplicações partilhem uma parte de um modelo conceptual que deve também ser mantido e ambas devem reflectir eventuais alterações ao modelo, ou pelo menos deve existir uma forma de entender em que é que uma aplicação não está de acordo com o modelo. De uma forma genérica, é importante ser assegurado que cada um dos artefactos existentes não entra em contradição com os restantes, ou seja que se mantém coerente.

3. Modelo Teórico da Rastreabilidade

No contexto do presente trabalho, designa-se por **projecto** não só um dado programa, ou conjunto de programas, como a entidade que agrega todos os artefactos que estão relacionados directamente com: a) a sua justificação, b) o tratamento de dados que realiza, c) os aspectos tecnológicos que lhe estão inerentes, d) o envolvimento humano necessário e e) a documentação necessária à produção, manutenção e operação da mesma. Designa-se por **elemento do projecto**, qualquer conceito que seja necessário à definição, ou operação, do projecto.

Designa-se por **artefacto** uma qualquer realização física de um elemento do projecto. Um elemento do projecto pode existir em mais do que um artefacto e um artefacto pode realizar mais do que um elemento do projecto. Alternativamente, pode considerar-se **artefacto** qualquer conjunto de símbolos, organizado segundo uma estrutura conhecida, que existe num dado período de tempo, num dado suporte, com o propósito de instanciar um conjunto de conceitos. Um artefacto pode incluir outros artefactos (e.g., a representação duma classe num diagrama de classes pode ser considerada ela própria um artefacto).

Um **suporte** é qualquer meio em que possa existir um certo tipo de artefactos (e.g., ficheiro de texto, tabelas de definição de uma base de dados num SGBD, repositório de uma ferramenta CASE).

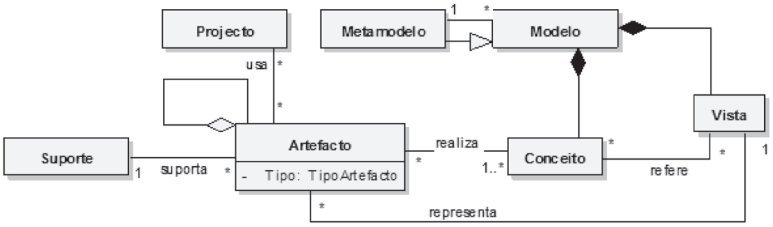

Fig. 1. Metamodelo dos conceitos básicos da rastreabilidade entre artefactos

Considera-se **modelo** um conjunto de elementos do projecto, que podem incluir relações entre eles (elas próprias sendo elementos). Um modelo é uma simplificação da realidade que, neste contexto, deve incluir todos os conceitos relevantes (do problema tratado no contexto do projecto) e apenas estes.

Considera-se **vista** uma forma de seleccionar, relacionar ou designar, um conjunto de conceitos da aplicação. A vista poderá ser indicada resumidamente como qualificador de um modelo (e.g., *modelo físico* que representaria abreviadamente uma *vista física de alguns elementos da aplicação*, ou ainda um *modelo dos elementos da aplicação com realização física*). Um diagrama é também um artefacto com a informação gráfica e lógica dos conceitos representados. Um **metamodelo** é um modelo que define um conjunto de conceitos presentes em modelos.

3.1 Relações de Dependência

Conforme foi considerado anteriormente, um artefacto pode conter outros artefactos. Importa agora considerar as diversas relações que se podem estabelecer entre esses artefactos. Considera-se que dois artefactos são equivalentes quando têm a mesma representação dos respectivos conceitos. Na Fig. 4, mesmo considerando que as classes Person e Human representam o mesmo conceito, pode-se afirmar que os artefactos 1 e 2 não são equivalentes.

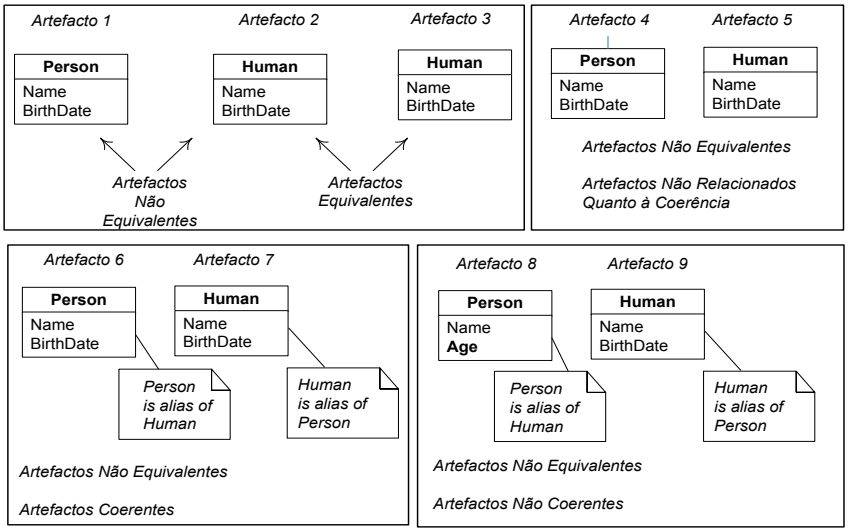

Fig. 2. Equivalência entre artefactos

A relação de equivalência é pouco interessante na medida em que é demasiado restritiva caso se queira modelar contextos de alguma complexidade. Neste tipo de contextos existe por vezes a necessidade de se criarem sinónimos para conceitos que são usados sob perspectivas diferentes. Nesse caso, o facto de os artefactos não serem equivalentes não significa que os conceitos não estão representados de forma coerente. Os artefactos 4 e 5 representam duas classes, em diferentes artefactos, que apesar de parecerem representar o mesmo conceito não têm informação associada que permita justificar essa afirmação. Não existe nenhuma relação de coerência sobre os dois artefactos que os permita relacionar quanto ao conceito que representam. Assim sendo não podem ser considerados coerentes.

Os artefactos 6 e 7 possuem informação associada que permite afirmar a coerência entre eles. Pelo contrário, os artefactos 8 e 9, sendo diferentes perspectivas da mesma classe, ao terem uma definição dos atributos diferente não são coerentes.

3.2 Conceitos

Para que a relação de coerência seja melhor definida, serão considerados em seguida alguns conceitos.

Seja T o conjunto dos artefactos presentes num dado domínio W, definido como o conjunto de aplicações tratadas. Considera-se a ausência de artefactos designada por $t_{Æ}$, sendo $t_{Æ} \hat{I} T$. Considera-se uma *expressão* E(t*) *sobre um conjunto de artefactos* t* qualquer expressão possível no contexto da definição

desses artefactos, sendo o seu valor calculável. Uma expressão E(t*) compreende usualmente os valores dos tipos básicos admissíveis (e.g., *int, string* ou *bool*), qualquer objecto cuja definição exista nesse âmbito, bem como operações aritméticas, lógicas ou outras que estejam definidas. Uma expressão E(t*) pode ser igualmente um valor constante (numérico ou outro qualquer). Considera-se por V o conjunto de todos os valores possíveis para as expressões produzidas sobre os artefactos duma aplicação, podendo afirmar-se que E: T ® V.

Considere-se a relação binária *identidade* entre valores designada por « e que resulta num valor booleano true caso ambas as expressões se refiram ao mesmo valor. Se, neste contexto, os elementos relacionados são objectos, o valor refere-se à identidade dos objectos, ou seja, se eles são o mesmo objecto. Apesar da relação binária identidade relacionar expressões pode dizer-se que estas são sempre convertidas em valores, após a sua avaliação, pelo que é possível dizer-se que « verifica as seguintes propriedades:

é reflexiva, ou seja, $a \in V: a \ll a$

« é simétrica, ou seja, $a, b \in V: a \ll b \Rightarrow b \ll a$

« é transitiva, ou seja, $a, b, c \in V: a \ll b \wedge b \ll c \Rightarrow a \ll c$

Como simplificação considere-se agora apenas as expressões sobre um artefacto E(t). Nesse caso, uma relação de coerência entre os artefactos t_x e t_y é definida por uma expressão:

$$E_m(t_x) \ll E_n(t_y) ; t_x, t_y \in T \quad (1)$$

Generalizando para mais de dois artefactos, é possível definir uma regra de coerência entre dois conjuntos de artefactos t^*_x e t^*_y como sendo:

$$E_m(t^*_x) \ll E_n(t^*_y) ; t^*_x , t^*_y \in T \quad (2)$$

Embora seja possível definir relações de coerência entre mais de dois artefactos, serão estas as relações consideradas doravante. Quando dois artefactos t_x e t_y possuem uma relação de coerência, esta pode ser usada para definir um rasto, r R:

$$r(t_x , t_y) \leftrightarrow E_m(t_x) \ll E_m(t_x) ; t_x , t_y \in T; m,n \in N \quad (3)$$

sendo $R: V \times V \rightarrow \{true, false\}$

e pode ler-se como: o rasto r entre os artefactos t_x e t_y está definido pela relação de coerência $E_m(t_x) \ll E_m(t_x)$ e o seu valor, num dado momento, é *true* se a avaliação das duas expressões $E_m(t_x)$ e $E_m(t_x)$ resulta em valores idênticos, ou *false* caso contrário. Designa-se como valor do rasto r, num dado momento, val(r) ao resultado da avaliação da relação de coerência respectiva. O rasto pode ser ou não válido, num dado momento. Por isso o seu valor pertence ao conjunto dos valores lógicos *{true, false}*. Note-se que quando um rasto não está definido não é possível dizer-se que o seu valor é *true*, nem sequer *false*. Pode considerar-se que dois artefactos t_i e t_j são coerentes quanto ao rasto r_x quando, num dado momento, a relação de coerência que o define é válida. Verifica-se que $r_x (t_i , t_j)$

pode ser eventualmente diferente de r_x (t_j , t_i). Como exemplo, considere-se o rasto:

$$r(t_i , t_j) \hat{U} NrClasses(t_i) \ll NrTabelas(t_j) ; t_i , t_j \hat{I} T \quad (4)$$

sendo: t_i uma colecção de classes Java, t_j uma base de dados, e as operações que calculam o número de classes e o número de tabelas presentes no artefacto respectivo.

No exemplo anterior é evidente que o rasto $r(t_j , t_i)$ com a mesma definição não seria idêntico ao anterior, caso pudesse sequer ser definido. Logo pode dizer-se que, apesar da definição das relações de coerência serem simétricas, os rastos não o são

$$r(t_i , t_j) \neq r(t_j , t_i) \quad (5)$$

Dois artefactos podem ter mais do que um rasto definido entre eles. Considera-se o subconjunto $R<t_i , t_j>$ de R definido como o conjunto de rastos definidos de t_i para t_j. Similarmente considera-se o subconjunto $R<t_i>$ de R definido como o conjunto de rastos no qual o artefacto t_i participa. Pode considerar-se que dois artefactos são *genericamente coerentes de* t_i *para* t_j, num dado momento, quando todos os rastos existentes de t_i para t_j são válidos:

$$r \hat{I} R< ti, tj >: val(r(ti, tj)) = true \quad (6)$$

Por extensão, pode considerar-se que dois artefactos são *genericamente coerentes*, num dado momento, quando todos os rastos existentes envolvendo os dois artefactos são válidos:

$$r \hat{I} R< t_i, t_j >: (val(r(t_i, t_j)) = true) \grave{U} (val(r(t_j, t_i)) = true) \quad (7)$$

Um rasto $r(t_i, t_j)$ pode ocorrer entre artefactos que realizam conceitos de um metamodelo (e.g., classe C# e tabela relacional). Quando tal acontece diz-se que $r(t_i, t_j)$ é um *meta-rasto entre* t_i *e* t_j. Nesse caso $val(r(t_i, t_j))$ é dado pela verificação simultânea de um subconjunto do produto interno das realizações de ambos os grupos de artefactos. A *sincronização*, como chamar-se-á à actividade de manter a coerência entre os diversos artefactos, deverá ser realizada de forma a que as alterações realizadas sobre um elemento sejam propagadas a todos os artefactos relacionados quanto à coerência.

3.3 Rastreabilidade

O significado usual do termo rastreabilidade é demasiado vago, quando aplicado ao contexto dos sistemas de informação. Definições como "a possibilidade de identificar a origem de um produto e de reconstituir o seu percurso desde a produção até à distribuição"[7], ou "a capacidade de relacionar cronologicamente entidades univocamente identificáveis e de uma forma relevante"[8], ou ainda "de que forma uma relação pode ser estabelecida entre dois ou mais produtos

do processo de desenvolvimento, especialmente produtos tendo relações dos tipos predecessor-sucessor ou grupo-detalhe"[18], têm de ser contextualizadas para que o seu significado possa ser compreendido e aplicado. Considerando os conceitos anteriormente definidos consideramos **rastreabilidade** como a acção de verificar os rastos existentes para um grupo de artefactos, tendo em vista um ou mais dos seguintes objectivos:

Análise de sensibilidade – Quando se realiza uma modificação ou eliminação (e em casos especiais uma criação) de um artefacto a rastreabilidade pode ser usada para se entender, quais as implicações que essa acção terá nos outros artefactos, i.e., quais os artefactos afectados pela mesma;

Validação de requisitos – A validação de um requisito pode ser feita seguindo os rastos necessários, do artefacto que contém o requisito até aos testes e respectivos resultados (eles próprios igualmente artefactos);

Verificação de conformidade – Os artefactos de código podem estar ou não de acordo com os modelos respectivos, sendo essa verificação feita a partir da validação dos rastos entre artefactos de código e artefactos dos modelos;

Pesquisa documental – A rastreabilidade pode ser usada para o conhecimento técnico da estrutura de uma aplicação, seguindo-se os rastos entre um dado artefacto e os demais.

Assim sendo, tomada como uma actividade, a rastreabilidade não tem como fim realizar alterações a uma aplicação, mas sim obter um determinado tipo de informação sobre a estrutura dessa aplicação. É também possível dizer-se que uma aplicação é rastreável ou um grupo de artefactos é rastreável, usando-se a rastreabilidade como uma propriedade. Na verdade, quando se usa a rastreabilidade como um qualificador, está-se a afirmar que sobre essa aplicação ou grupo de artefactos é possível realizar uma operação de rastreabilidade. Estabelecendo uma analogia com a linguagem SQL, a rastreabilidade seria comparável a uma operação *select*, enquanto a sincronização seria comparável às operações *insert, update* e *delete*. A rastreabilidade pressupõe uma sincronização prévia visto que para que uma aplicação seja rastreável é necessário que os seus artefactos estejam num estado coerente, i.e., que as relações de coerência definidas sejam válidas nesse momento. Caso contrário os resultados obtidos não serão exactos. A qualidade da rastreabilidade está por isso dependente da qualidade da sincronização. Por sua vez, a sincronização pressupõe que as relações de coerência entre os artefactos estejam criadas correctamente. A qualidade da sincronização depende da exactidão e completude da definição das relações de coerência entre os artefactos.

4 Componente Reactiva da Rastreabilidade

Como foi referido, tanto a rastreabilidade como a sincronização são operações que podem ser realizadas num determinado momento. Existem três

formas de agendar a execução destas operações: execução num intervalo de tempo, em momentos pré-determinados e através de eventos. Na execução num intervalo de tempo esta pode ser repetida a cada intervalo de tempo constante. Com a execução em momentos pré-determinados é possível ao utilizador desencadeá-la manualmente, nesse momento, ou num conjunto de momentos pré-determinados explicitamente. Este mecanismo resume-se a um mecanismo de agendamento de execuções. Na execução através de eventos, a execução é realizada quando se dá uma alteração ao sistema. Essa alteração, denominada de evento, é uma qualquer alteração ao estado de um artefacto abrangido por pelo menos uma regra de coerência.

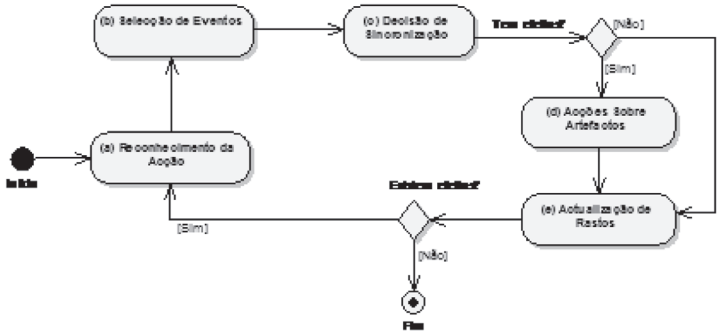

Fig. 3. Tratamento dos eventos

Define-se **rastreabilidade reactiva** como a actividade de rastrear e sincronizar artefactos através de um mecanismo de execução por eventos. As operações de sincronização são usadas a par das operações de rastreabilidade, pelo que se optou por limitar a designação desta actividade conjunta à mais conhecida e referida. Assim sendo, a rastreabilidade reactiva é um conjunto de operações que visam reagir a alterações provocadas sobre a estrutura da aplicação em causa.

Se num dado momento t_0 ocorre uma acção (criação, modificação e eliminação) sobre um artefacto t_i, então todos os rastos $R< t_i >$ têm de ser verificados para que possam ser propagadas as alterações necessárias. Como simplificação, considera-se que essa verificação e as eventuais alterações são atómicas em t_0. Considera-se que uma acção a sobre um artefacto t_i dada por $a(t_i)$ pertence ao conjunto A e ainda que A: T ® T. Quanto ao tipo de cada acção é possível indicar dois casos particulares:

a) Uma acção de criação implica uma relação Æ ® T sendo indicada por $a(t_Æ)$.

b) Uma acção de eliminação implica uma relação T ® Æ sendo indicada por $a^Æ(t_i)$.

De a) torna-se evidente que uma acção, genericamente, não é uma aplicação no sentido algébrico do termo [9]. Um *evento* é materializado após existir

conhecimento de uma acção sobre um artefacto. Esse conhecimento pode ser *a priori* ou *a posteriori*. Como muitos dos artefactos são produzidos por ferramentas específicas e exteriores à própria aplicação de rastreabilidade, esse conhecimento variará consoante o nível de controlo sobre os próprios artefactos.

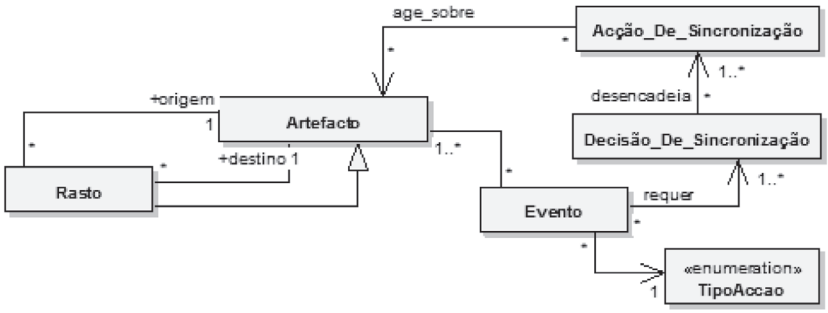

Fig. 4. Metamodelo do mecanismo de eventos sobre artefactos

Caso exista um mecanismo que permita conhecer a *intenção* de realizar uma acção sobre um artefacto, bem como de interromper essa acção, esse conhecimento pode ser *a priori*. Porém, na generalidade dos casos é possível assumir que o evento acontecerá *a posteriori*. Como um rasto é ele próprio um artefacto é possível considerar rastos sobre rastos, bem como eventos sobre rastos. O algoritmo de tratamento do evento envolve as seguintes actividades: (a) reconhecimento da acção, (b) selecção de eventos, (c) decisão de sincronização, (d) acções sobre artefactos, (e) actualização de rastos. Em (a) existe a tomada de conhecimento que foi produzida uma alteração ao estado do sistema, materializada por uma acção sobre um artefacto conhecido. O artefacto é então acrescentado a uma lista de artefactos alterados (LAA). Em sequência, para cada artefacto da lista de artefactos alterados (b) verifica-se se existe um evento associado a esse artefacto, se não existir a acção é ignorada e o artefacto sai da LAA. Quando existe um evento (e este cumpre as condições de activação, i.e., a acção que o desencadeia é válida) em (c) são invocadas as decisões de sincronização. Neste momento pode ser necessário o utilizador tomar uma decisão, caso a decisão de sincronização implique uma escolha.

Em alguns casos a decisão é determinística e (d) actua imediatamente. Em resultado de eventuais acções sobre o artefacto podem existir efeitos sobre outros artefactos que têm de ser propagadas. Esta actividade pode levar a que o processo seja repetido para os artefactos entretanto alterados. Em (e) é verificada a lista de rastos existentes, bem como os artefactos correspondentes. Quando, entretanto, houve uma acção sobre esse artefacto este é acrescentado à LAA e o processamento continua em (a). A decisão de sincronização (c), não tendo efeitos, pode levar a um estado de incoerência que tem de ser resolvido em (e). Quando a

LAA está vazia o sistema voltou ao estado de coerência. Pode então dizer-se que todos os artefactos tratados são genericamente coerentes, conforme a definição dada.

5 Conclusões

Existe actualmente um esforço crescente no campo das linguagens de transformação de modelos. Iniciativas como o QVT (*Queries, Views, Transformations*) [11,12], o MISTRAL [13], o ModelMorf [14] dão visibilidade a esta tendência de utilizar a geração automática, quer de código através de modelos, como de modelos a partir de outros. Naquilo que diz respeito à rastreabilidade e à sincronização de artefactos (sejam eles respeitantes a modelos ou a código), é possível identificar duas perspectivas: a perspectiva *generativa* e a perspectiva *relacional* [1]. Na primeira, considera-se que os artefactos são gerados ou produzidos (não necessariamente de forma automática) a partir de outros. Quando uma alteração é feita sobre um artefacto desencadeia-se um processo de geração [6] que propaga um conjunto de alterações a outros artefactos. Na segunda perspectiva considera-se que os artefactos já existem, estando relacionados entre si (neste contexto através de rastos). As alterações sobre um artefacto podem produzir efeitos noutros artefactos através dos rastos existentes. Esta segunda perspectiva adequa-se à manutenção de sistemas já existentes enquanto a primeira pode ser usada com maior facilidade num desenvolvimento de raíz. A rastreabilidade, enquanto tema de investigação, está a suscitar um interesse crescente relacionando artefactos tão diferentes como os requisitos [17], os modelos [15, 19] ou o próprio código [16]. Com níveis conceptuais tão diversos é necessário encontrar uma plataforma comum que permita estabelecer os *rastos.* O modelo apresentado é suficientemente genérico para permitir a instanciação em diferentes plataformas tecnológicas e aplicações. O recurso à metamodelação e à transformação de modelos através do QVT [12,13] foi um ponto de partida para esta abordagem que o complementa. Embora os aspectos tecnológicos estejam fora do âmbito deste artigo, é relevante notar que os conceitos apresentados estão a ser aplicados em protótipos já produzidos [1,2].

Bibliografia

1. Costa, M., da Silva, A.R.: RT-MDD Framework – A Practical Approach. 3rd ECMDA Traceability Workshop, Haifa, Israel (2007)
2. Costa, M., da Silva, A.R.: Synchronization Issues in UML Models. 9th International Conference on Enterprise Information Systems, Funchal, Portugal (2007)
3. Iivari, J.: Communications of the ACM. *Why Are CASE Tools Not Used*, Oct.1996,

Vol.39, Nr.10, Pgs. 94-103, Association for Computing Machinery (1996)
4. Welsh, T.: *How Software Modelling Tools Are Being Used.* In Enterprise Architecture Advisory Service Executive Update Vol. 6 , N. 9, 03-12, Cutter Consortium (2003)
5. Herrington, J. 2003. *Code Generation In Action*. Manning Pub. Co (2003)
6. da Silva, A. R., Lemos, G., Matias, T., Costa, M.: *XIS Generative Programming Techniques,* Generative Programming and Component Engineering (GPCE'03), Erfurt, Alemanha (2003)
7. Porto, Ed.: *Dicionário da Língua Portuguesa*, Porto Editora, Portugal (2008)
8. Wikipedia: *Traceability,* Wikipedia, http://en.wikipedia.org/wiki/Traceability
9. Monteiro, A. A.: *Álgebra Linear e Geometria Analítica*, Ed. Assoc. dos Estudantes da Fac. de Ciências de Lisboa (1989)
10. Dollard, K.: Code Generation In Microsoft .NET. Apress (2004)
11. OMG: Object Management Group, 2005. *MOF QVT Final Adopted Specification* http://www.omg.org/docs/ptc/05-11-01.pdf
12. QVTP, 2003. *Revised submission for MOF 2.0 Query / Views / Transformations RFP, Vers. 1.1*, QVT-Partners, http://qvtp.org
13. Kurtev, I., Berg, K. v. d., *MISTRAL: A Language for Model Transformations in the MOF Meta-modeling Architecture*, Lecture Notes in Computer Science, Springer (2005)
14. Tata (ed.): *ModelMorf – A Model Transformer*. www.tcs-trddc.com/ModelMorf/index.htm (2007)
15. Oldevik, J., Neple, T.: *Traceability in Model to Text Transformations*. 3rd ECMDA Traceability Workshop (2006)
16. Perini, A., Susi, A.: *Automating Model Transformations in Agent-Oriented Modelling*. Proceedings of 6th International Workshop AOSE 2005, Utrecht (2005)
17. Palo, M.: *Requirements Traceability*. Seminar Report, Department of Computer Science. University of Helsinki (2003)
18. IEEE (ed.). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. Institute of Electrical and Electronics Engineers (1990)
19. Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., Shaham-Gafni, Y.: *Model Traceability*. IBM Systems Journal, Vol. 45, Nr. 3 (2006)