Construção de um Metamodelo para o ARIS Method: Escavação, Refatoração e Análise

Paulo S. Santos Júnior, João Paulo A. Almeida, Thiago Lavarezi Pianissolla

Departamento de Informática – Universidade Federal Espírito Santo (UFES)
Av. Fernando Ferrari, s/n, Vitória, ES, Brasil

paulossjunior@gmail.com, jpalmeida@ieee.org, thiagovni@gmail.com

Abstract. *This paper presents the construction of a metamodel for the ARIS Method using as a starting point the ARIS Toolset serialization format (the ARIS Markup Language). The result of the series of transformations presented here is a metamodel that allows one to store and manipulate models built in ARIS using Model-Driven Design techniques. Consequently, models can be accessed at a higher level of abstraction than previously made possible by using the ARIS Markup Language, facilitating the integration of the ARIS tools with external tools and applications. In addition, the proposed metamodel reveals the essential elements required to capture the semantics of models defined with the ARIS Method.*

Resumo. *Este artigo apresenta a construção de um metamodelo para o ARIS Method, partindo inicialmente do formato de serialização do ARIS Toolset (a ARIS Markup Language - AML). O resultado da série de transformações apresentadas é um metamodelo que permite o armazenamento e manipulação dos modelos construídos nas ferramentas ARIS com técnicas de desenvolvimento orientado a modelos. Desta forma, o acesso aos modelos pode ser feito em um nível de abstração mais alto do que o oferecido atualmente através da AML, facilitando a integração com aplicações externas. O metamodelo proposto revela ainda os elementos essenciais para capturar a semântica dos modelos construídos com o ARIS Method.*

1. Introdução

A abordagem ARIS (*ARchitecture for integrated Information Systems*) foi desenvolvida na universidade de Saarbrucken, Alemanha, em 1992 com o objetivo principal de permitir a descrição e desenvolvimento de sistemas de informação que estivessem integrados à estrutura da organização através de seus processos de negócio.

Esta abordagem propõe uma arquitetura de modelagem de organizações denonimada ARIS Method, estruturada em 5 diferentes visões (organizacional, dados, controle, função e saída) e 3 diferentes camadas de abstração (requisitos, projeto e implementação) [Scheer 2000]. Dado o escopo dos modelos suportados pelo ARIS Method, esta abordagem pode ser considerada uma técnica abrangente de modelagem de organizações e de arquiteturas de TI [Lankhorst 2005].

Com base nesta abordagem, foram desenvolvidas diversas ferramentas, como por exemplo, ARIS SOA Architect, ARIS Business Architect e ARIS Toolset. Estas ferramentas provêem um ambiente para modelagem, gerenciamento dos modelos produzidos e outros serviços de gerenciamento de processos [Kern e Kühne 2007]. Em particular, neste trabalho, estamos interessados no ARIS Method e seu suporte através

do ARIS Toolset. O interesse nesta ferramenta deve-se à grande importância industrial desta ferramenta na prática de modelagem de processos de negócio e arquiteturas de TI.

O ARIS Toolset permite: (i) a manipulação de modelos definidos no método ARIS (ARIS Method), (ii) a especialização e restrição do ARIS Method através do conceito de filtros e (iii) o acesso aos modelos através de uma linguagem (AML) que permite a comunicação entre o ARIS e o mundo exterior.

Para fazer a comunicação com aplicações externas o ARIS Toolset utiliza uma linguagem chamada de AML (ARIS *Markup Language*). Basicamente a AML é um formato XML (*Extensible Markup Language*) específico, que possibilita serializar os modelos e objetos definidos no ARIS method e configurados pelo filtro que estão armazenados na base de dados do ARIS Toolset, para que estes possam ser acessados por aplicações externas ao ARIS Toolset. De forma inversa, também é possível que aplicações externas construam modelos serializados em AML para adicioná-los no repositório do ARIS Toolset. O formato AML é definido em um documento DTD (*Document Type Definition*) que especifica a gramática de um documento AML.

Apesar do suporte à AML permitir a abertura do ARIS Toolset para integração e acesso aos modelos, esta forma de integração requer a manipulação direta e interpretação (*parsing*) de documentos XML. Desta forma, a manipulação de modelos produzidos no ARIS Toolset torna-se onerosa e com baixo nível de abstração, necessitando que o responsável pela interpretação dos documentos tenha que manipular diretamente arquivos XML e ter um grande conhecimento da AML.

Este artigo tem como objetivo apresentar a construção de um metamodelo para o ARIS Method, partindo inicialmente do formato de serialização do ARIS Toolset (a ARIS Markup Language) através de uma série de transformações no metanível. O resultado da série de transformações apresentadas é um metamodelo que permite o armazenamento e manipulação dos modelos construídos nas ferramentas ARIS em um repositório de modelos na plataforma Eclipse. Desta forma, o acesso aos modelos pode ser feito usando técnicas de desenvolvimento orientado a modelos (*Model-Driven Design*) em um nível de abstração mais alto do que o oferecido atualmente através da ARIS Markup Language, abrindo a ferramenta para integração com aplicações externas e ferramentas genéricas de transformação de modelos (como proposto em [Almeida et al. 2007] para a integração de ferramentas de desenvolvido orientado a serviços). O metamodelo proposto revela ainda os elementos essenciais para capturar a semântica dos modelos construídos com o ARIS Method. Com relação à AML, este metamodelo permite um melhor entendimento dos conceitos essenciais de um modelo produzido com o ARIS Method e pode servir como base para a definição da semântica dos conceitos do ARIS Method.

A Seção 2 introduz a abordagem utilizada para construção do metamodelo do ARIS Method, a Seção 3 apresenta o passo de escavação inicial do metamodelo que consiste no mapeamento do DTD do AML para ECORE (a técnica da metametamodelagem do *Eclipse Modeling Framework*), a Seção 4 apresenta os conceitos de conjunto dos elementos presentes na AML, a Seção 5 apresenta a análise e refatoração do metamodelo escavado na Seção 3, e a Seção 6 apresenta o metamodelo básico do ARIS Method e uma discussão sobre as limitações do mesmo. Finalmente, a Seção 7 apresenta as conclusões e trabalhos futuros.

2. Abordagem

Para explicitar o metamodelo do ARIS Method a partir do DTD da AML, foram necessárias uma série de transformações, apresentadas na Figura 1. Através destas transformações, será possível: (i) construir um metamodelo para o ARIS Method em ECORE, a linguagem de definição de metamodelos do Eclipse Modeling Framework (EMF) [Eclipse Foundation 2008]; (ii) analisar e re-estruturar o metamodelo do ARIS Method e; (iii) identificar os elementos básicos do ARIS Method.

A primeira transformação realizada tem o objetivo de mapear todos os elementos do DTD AML para EMF (*Eclipse Modeling Framework*) Esta transformação foi chamada de **escavação** e é uma transformação isomórfica que pode ser considerada simplesmente uma tradução entre *technological spaces* [Kurtev et al. 2002].A partir desta transformação, o acesso aos modelos pode ser feito usando técnicas de desenvolvimento orientado a modelos, ao invés da manipulação de documentos XML.

A segunda transformação denominada de **refatoração** consiste na análise e na re-estruturação do metamodelo AML. Esta transformação explicita os relacionamentos que originalmente eram codificados através de identificadores nos elementos da AML. Nesta transformação a informação dos modelos é preservada completamente, porém há uma melhoria na representação dos relacionamentos no metamodelo e nas instâncias deste metamodelo.

Por fim, a última transformação (**seleção**) possui o objetivo de revelar os elementos considerados essenciais para a captura da semântica dos modelos construídos com o ARIS Method, excluindo elementos considerados de infra-estrutura ou estritamente relacionados à sintaxe concreta da linguagem [Harel e Rumpe 2000].

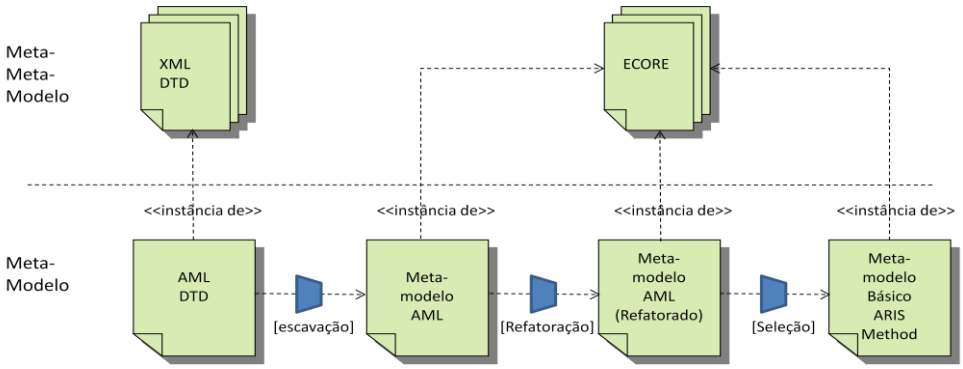

Figura 1 - Transformações

3. Transformação do DTD da AML para o Metamodelo AML em ECORE

Os passos utilizados na **escavação** têm o objetivo o desenvolver o metamodelo AML em ECORE a partir do DTD da AML.

As metaclasses no ECORE são representas pelo elemento EClass, o relacionamento entre duas metaclasses é expresso pelo elemento EReference e os atributos pertencentes as metaclasses (EClass) é representado pelo elemento EAttribute. Para o desenvolvimento do metamodelo AML em ECORE foi utilizado o EMF (*Eclipse Modeling Framework*) como framework de metamodelagem.

A construção do metamodelo do ARIS Method em um framework de meta-modelagem (EMF) promove a abertura da ferramenta para integração com outras

ferramentas de modelagem de processos de negócios e arquiteturas organizacionais, além de permitir a definição de transformações de modelos do ARIS para modelos de outras linguagens.

O primeiro passo da escavação consiste na identificação dos elementos do DTD. Em seguida, cada elemento identificado deve ser mapeado como uma EClass com o mesmo nome do elemento do DTD, e cada um dos seus atributos será mapeado como um EAttribute da metaclasse correspondente.

Para definir as características do atributo presente em cada EClass é preciso verificar as propriedades attribute-type e default-value no DTD. O attribute-type como apresentado em [W3Schools 2008], define os tipos primitivos que um atributo pode assumir como, por exemplo, CDATA, ID, IDREF entre outros. A propriedade default-value define a obrigatoriedade do atributo e em alguns casos o conjunto de valores que um atributo pode assumir ou ainda, a possibilidade ou não de modificar o valor do atributo [W3Schools 2008]. A Tabela 1 apresenta os possíveis valores e significados da propriedade default-value.

Tabela 1- Interpretação da propriedade default value de um atributo

Propriedade *default-value*	Significado
#REQUIRED	O atributo é obrigatório.
#IMPLIED	O atributo não é obrigatório.
valor1 | *valor2* | *valor3* […]	O atributo é obrigatório e assume um valor dentro de uma enumeração.
#FIXED *valor*	O valor do atributo é uma constante determinada por *valor* e não pode ser alterado.

No caso de um atributo possuir um conjunto de valores pré-determinados optou-se por representar tal conjunto como uma enumeração em EMF (ENum).

O último passo da escavação envolve representar os relacionamentos entre as metaclasses (com suas cardinalidades). No DTD, o relacionamento entre elementos é representado como demonstrado na Figura 2.

```
<!ELEMENT elemento (elemento1<tiporelacionamento>,elemento2<tiporelacionamento>,....)>
```

Figura 2 -Template de relacionamento no DTD

A Tabela 2 abaixo apresenta os possíveis tipos de relacionamentos no DTD .

Tabela 2 - Relacionamentos do DTD

Cardinalidade	Descrição	Símbolo
1	Uma única ocorrência do elemento (opção padrão, não possuindo símbolo próprio para representação no DTD)	-
1..*	Uma ou mais ocorrências do elemento, representado pelo símbolo	+
0..*	Zero ou mais ocorrências do elemento, representado pelo símbolo	*
0..1	Zero ou uma ocorrência do elemento, representado pelo	?

Todos os relacionamentos entre os elementos do DTD foram mapeados como relacionamentos de agregação no EMF, porque se considerou que todos esses relacionamentos representavam composições entre elementos.

Dois casos particulares de relacionamento entre elementos no DTD mereceram atenção especial para sua representação em EMF. O primeiro caso envolve o

relacionamento entre um elemento de um tipo x com um outro elemento que pode ser uma instância de um tipo y ou uma instância de um tipo z. Nesse caso, no metamodelo AML a metaclasse correspondente a x possui dois relacionamentos de agregação sendo o primeiro relacionamento com uma metaclasse correspondente a y e um segundo relacionamento com uma metaclasse correspondente a z. Porém, a restrição imposta no DTD não pode ser representada diretamente no metamodelo, apesar de poder ser representada como uma restrição OCL [OMG 2008] no metamodelo.

O segundo caso ocorre quando um elemento possui um relacionamento com outro elemento do tipo #PCDATA, ou seja, um tipo composto. Nesse caso, esse tipo de relacionamento foi mapeado como um atributo com o nome **mixed** do tipo String na representação do elemento no modelo do EMF.

A Figura 3 apresenta um fragmento do DTD da AML que foi mapeado para ECORE como demonstrado na Figura 4. Na Figura 3 o elemento Language possui três atributos, sendo dois obrigatórios (LocaleId do tipo NMTOKEN e Codepage do tipo CDATA) e um não obrigatório (Language.ID do tipo ID). Além disso, o elemento Language possui um relacionamento como outros dois elementos: LanguageName e LogFont.

```
<!ELEMENT Language (LanguageName?, LogFont?)>
<!ATTLIST Language
        Language.ID ID #IMPLIED
        LocaleId NMTOKEN #REQUIRED
        Codepage CDATA #REQUIRED
>
```

Figura 3 - Exemplo de um elemento do DTD

A Figura 4 revela a metaclasse Language, com os três atributos correspondentes aos atributos do elemento Language do DTD da AML apresentado na Figura 3. A metaclasse Language está relacionada às metaclasses LanguageName e LogFont. Ambos os relacionamentos possuem cardinalidade 0..1, correspondente ao símbolo "?"na Figura 3.

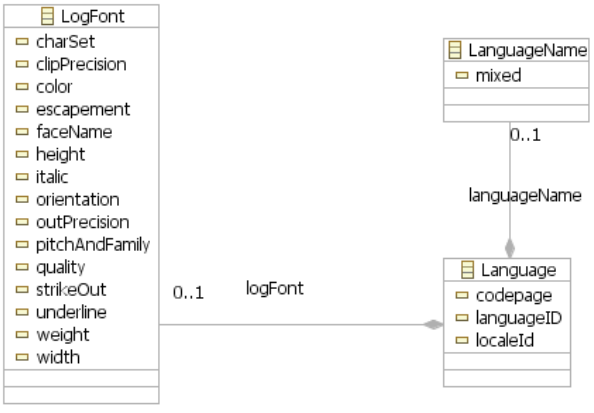

Figura 4 - Exemplo de mapeamento entre DTD e o EMF

3.2. Automação da escavação

A primeira etapa da automatização da escavação foi a transformação do DTD do AML em um documento XSD (XML *Schema*). Para aumentar a agilidade dessa primeira etapa foi utilizado o serviço presente em [HITSoftware 2008], que de forma automática,

transforma um documento DTD em um documento XSD. (Foi necessário realizar pequenos ajustes no documento XSD gerado. Estes ajustes foram omitidos por restrições de espaço.)

A segunda e última etapa da automatização da escavação foi realizada utilizando um *plugin* do Eclipse que transforma de forma automática um documento XSD (*XML Schema*) para um modelo EMF [Eclipse Foundation 2008b].

4. Metamodelo AML

Nesta seção são apresentados alguns dos elementos do metamodelo AML. A semântica destes elemento foi inferida a partir de [IDS Scheer 2006b], de conhecimentos adquiridos através da utilização do ARIS Toolset e da análise de documentos AML extraídos da ferramenta.

A Figura 5 apresenta um fragmento do metamodelo de alguns elementos referentes à infra-estrutura necessária para a captura de modelos produzidos pelo ARIS Toolset. Neste metamodelo, a raiz de um documento AML é uma metaclasse denominada AML responsável por agrupar os demais elementos do modelo de serialização. Esta metaclasse é composta por: HeaderInfo, Language, Prefix, Database, User, UserGroup, FontStyleSheet, FFTextDef, OLEDef, Group, Delete.

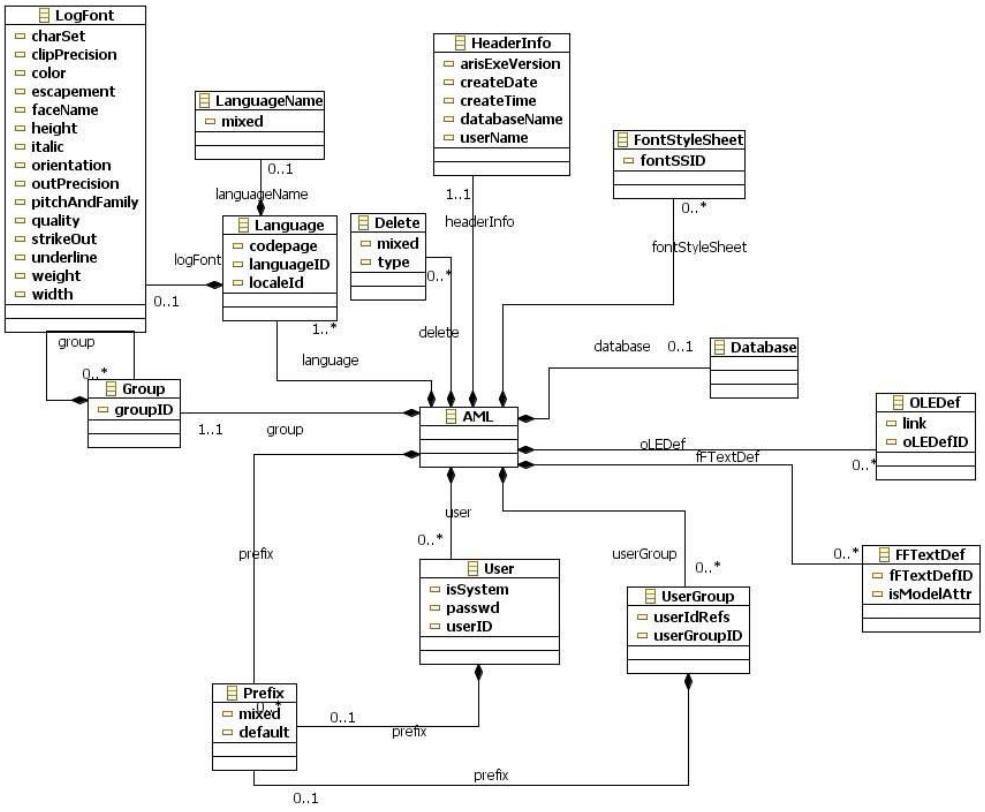

Figura 5 - Fragmento do metamodelo a partir do elemento raiz AML

A metaclasse HeaderInfo representa propriedades gerais do modelo: hora e data da criação do modelo, nome do banco de dados, nome do usuário e a versão do ARIS na qual o modelo foi criado.

A metaclasse Language representa o idioma selecionado para a criação dos objetos. Esta metaclasse é necessária pois o ARIS Toolset tem suporte à internacionalização, e portanto, a maioria dos campos textuais depende do idioma selecionado.

A metaclasse Group de um documento AML representa um agrupamento de diagramas (instâncias de Model) e elementos de modelagem como objetos e conexões entre objetos. Objetos são instâncias de ObjDef que representam os elementos de modelagem básicos do ARIS Method como unidades organizacionais, funções e eventos. A Figura 6 apresenta um fragmento do metamodelo que revela os elementos de um Group.

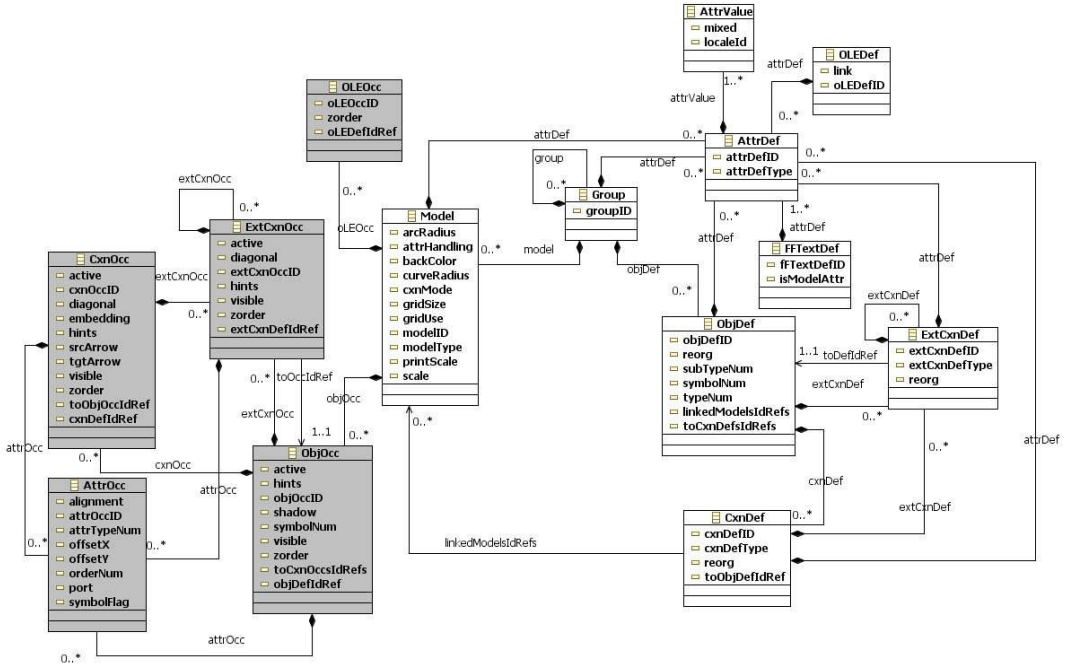

Figura 6 - Fragmento do metamodelo a partir do elemento Group

Este fragmento revela uma distinção conceitual importante no ARIS Toolset: a separação entre o conteúdo de modelo (a definição de objetos como unidades organizacionais, funções, eventos, regras, etc. e suas conexões [Scheer 2000]) e os diagramas (instâncias de Model) que incluem a ocorrência de objetos e conexões, além de informações diagramáticas Usando a separação entre o conteúdo de modelos e diagramas, um mesmo elemento de definição (instâncias de ObjDef, CxnDef, AttrDef, etc.) pode ocorrer mais de uma vez em diagramas diferentes ou ainda no mesmo diagrama com notações diferentes através de ocorrências de objeto (instâncias de ObjOcc, CxnOcc, AttrOcc, etc.).. As metaclasses que correspondem à ocorrência de objetos estão representadas em cinza na Figura 6. Com exceção das metaclasses Model e Group, as demais metaclasses do modelo correspondem a definição de objetos (representado em branco na figura). Este conceito assemelha-se ao apresentado em [Lankhorst 2005] sob o nome de *Model*/*View*/*Visualisation*.

Os objetos de definição possuem as seguintes informações: nome, tipo (typeNum), símbolo padrão (symbolNum), conexões com outros objetos, assim como modelos associados ao objeto e que revelam refinamentos da descrição de um objeto.

O elemento que une dois objetos em um modelo é chamado de conexão. Este elemento representa o elo entre um objeto origem e um objeto destino. Uma conexão possui como atributos um tipo e um nome. Os conceitos de elemento de definição e elemento de ocorrência também se aplicam a conexões. Estes conceitos são representados por CxnDef e CxnOcc, respectivamente.

Todo elemento (conexão, modelo e objeto de modelagem) no ARIS possui uma lista de atributos e cada atributo é representado pelo elemento AttrDef. Este elemento possui como propriedade tipo, nome e conteúdo (elemento AttrValue). Além disso, o elemento AttrOcc define como as informações armazenadas no AttrDef serão visualizadas nos elementos do ARIS.

5. Refatoração do metamodelo AML

Esta seção apresenta a refatoração do metamodelo escavado. A abordagem empregada consiste na análise dos atributos de cada elemento buscando explicitar relações que não aparecem diretamente no metamodelo escavado.

Ao analisar os elementos do AML no metamodelo de serialização e no DTD da AML, que define as regras do AML, observou-se que alguns elementos possuem atributos do tipo IDREF ou do tipo IDREFS.

De acordo com [W3Schools 2008b], atributos do tipo IDREF representam uma referência a um identificador de um elemento contido no documento XML e atributos do tipo IDREFS representam um seqüência de IdRefs separados por espaços em branco. A Figura 7 apresenta um exemplo de elemento do DTD do AML com atributos do tipo IdRef e idRefs (em negrito).

```
<!ELEMENT CxnDef (GUID?, AttrDef*, ExtCxnDef*)>
<!--Format for CxnDef.Type: CxnBaseType or FromObjType.CxnBaseType.ToObjType-->
<!ATTLIST CxnDef
        CxnDef.ID ID #REQUIRED
        CxnDef.Type NMTOKEN #REQUIRED
        ToObjDef.IdRef IDREF #REQUIRED
        LinkedModels.IdRefs IDREFS #IMPLIED
        Reorg (DELETE|NODELETE) "DELETE"
>
```

Figura 7 – Fragmento do DTD da AML referente a CxnDef com IDREF e IDREFS

Após localizar os atributos do elemento que serão transformados em associações, a segunda etapa consiste em identificar qual o elemento referenciado por um atributo. Apesar dos identificadores usados em referências no arquivo XML não serem tipados foi possível inferir as metaclasses relacionadas através da análise do nome de cada atributo selecionado (ToObjDef.IdRef e LinkedModels.IdRefs) e através da análise de documentos AML exportados pela ferramenta. No fragmento apresentado na Figura 7 é possível identificar a associação de CxnDef com ObjDef e a associação de CxnDef com Model.

A última etapa consiste em determinar a cardinalidade de cada associação identificada anteriormente. Isso é feito analisando duas propriedades do atributo: se ele é obrigatoriamente requerido (com #REQUIRED) ou não (com #IMPLIED) e; se o atributo é do tipo IDREF ou do tipo IDREFS. Conjuntamente, estas propriedades definem a cardinalidade das associações de acordo com a Tabela 3.

Tabela 3 - Obrigatoriedade x Tipo de atributo

	#REQUIRED	#IMPLIED
IDREF	1..1	0..1
IDREFS	1..*	0..*

No fragmento apresentado na Figura 7, foi criada uma relação de associação de CxnDef com ObjDef com multiplicidade 1..1, ou seja, uma instância de CxnDef está associado a um único elemento ObjDef.

A Figura 8 apresenta um fragmento do metamodelo da AML refatorado com as novas relações do tipo associativa entre as metaclasses (construídas a partir da abordagem apresenta nesta seção) e ainda as relações do tipo agregação construídas pelo passo de escavação como apresentado na seção 3.

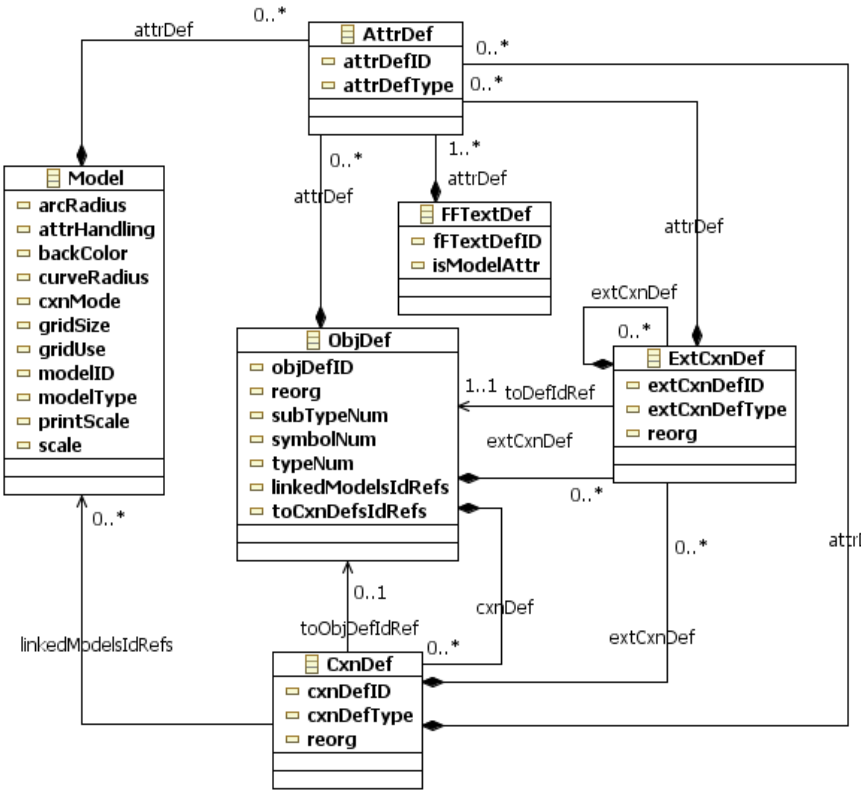

Figura 8 - Fragmento do metamodelo de serialização refatorado

6. Metamodelo Básico do ARIS Method

Esta seção apresenta o modelo básico do ARIS Method extraído do metamodelo de serialização refatorado. Esse metamodelo revela os elementos que foram considerados essenciais para capturar a semântica dos modelos construídos com o ARIS Method. Foram excluídos os elementos considerados de infra-estrutura ou estritamente relacionados à sintaxe concreta da linguagem [Harel e Rumpe 2000] e que não possuem relevância semântica no ARIS Method.

A Figura 9 apresenta o metamodelo básico do ARIS Method contendo os elementos Model, Group, ObjDef, ObjOcc, CxnDef, CxnOcc, AttrDef, AttrValue, AttrOcc, ExtCxnOcc, ExtCxnDef seus principais atributos e relações.

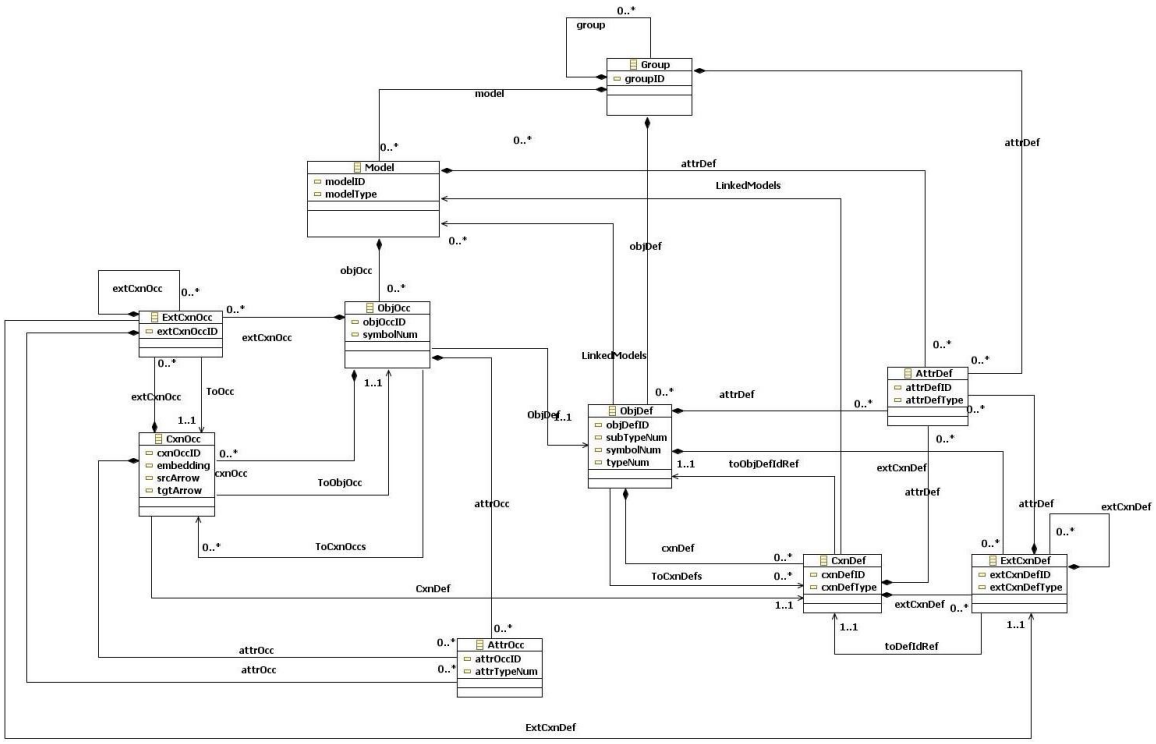

Figura 9 - Metamodelo básico do ARIS Method

6.1. Discussão

Como pode ser observado no metamodelo básico do ARIS Method apresentado na Figura 6, os diferentes tipos de objetos do ARIS Method, como, por exemplo, unidades organizacionais (*Organizational Units*), atividades (*Functions*) e eventos (*Event*) [Scheer 2000], não são representados explicitamente através de metaclasses. Estes tipos são codificados em identificadores dentro dos elementos como, por exemplo, *modelType*, *typeNum* e *cxnDefType* nos respectivos elementos *Model*, *ObjDef* e *CxnDef*. Como conseqüência não é possível através de uma inspeção rápida do modelo diferenciar tipos que representam categorias conceituais completamente diferentes, como, por exemplo, unidades organizacionais e atividades.

Apesar disso, esta representação evita fixar os tipos dos objetos e conexões que podem ser suportados pela representação, e, portanto o metamodelo como apresentado permaneceria estável frente a mudanças nestes tipos.

Levando em consideração estas observações é possível concluir que o metamodelo construído é praticamente neutro com relação às distinções conceituais do ARIS Method, e desta forma se aproxima de um metametamodelo. Porém, este metamodelo introduz ainda a relação entre objetos de ocorrência e objetos de definição, o que é uma característica particular do ARIS Method não presente em vários metametamodelos como EMF e MOF.

7. Conclusão e Trabalhos Futuros

Neste trabalho, foi proposto um conjunto de metamodelos do ARIS Method construído em uma série de transformações no metanível a partir do DTD da AML. Tomando como base os metamodelos propostos, o artigo apresentou também a semântica dos elementos essenciais de modelos construídos com o ARIS Method, assim como uma discussão sobre os metamodelos propostos.

A partir dos metamodelos propostos é possível construir interfaces que realizem a integração entre as ferramentas ARIS e ferramentas de simulação, análise, geração automática de relatórios, etc., em um nível mais alto de abstração usando técnicas de desenvolvimento orientado a modelos (*Model-Driven Design*). Desta forma, não será necessário construir interpretadores para um documento AML (XML) como é feito atualmente em [IBM 2008] e nas interfaces do próprio ARIS Toolset como apresentado em [IDS Scheer 2006] e; implementar transformações de modelos diretamente através de transformações de documentos XML como é o caso da transformação XSLT apresentada em [Mendling e Nüttgen 2004] para a conversão de eEPCs do ARIS [Scheer 2000] para EPML (EPC Markup Language) [Mendling e Nüttgen 2004].

Diferentemente do trabalho apresentado em [Kern e Kühne 2007], este artigo concentra-se na apresentação do metamodelo do ARIS Method em uma derivação sistemática através de transformações da AML, enquanto o primeiro concentra-se na ponte entre os repositórios de ARIS e EMF. Além disso, o metamodelo do ARIS Method apresentado neste artigo possui um maior nível de detalhes (revelando um número maior de metaclasses e muitas associações implícitas) buscando esclarecer a semântica dos elementos do ARIS Method.

Com relação à AML, o metamodelo proposto neste artigo permite um melhor entendimento dos conceitos essenciais de um modelo produzido com o ARIS Method. Não apenas este metamodelo exclui uma série de elementos considerados de infra-estrutura ou estritamente relacionados à sintaxe concreta da linguagem, como também facilita a visualização de relacionamentos entre metaclasses que na AML são codificados através de atributos de elementos XML.

É possível concluir que o metamodelo derivado a partir da AML é neutro com relação às distinções conceituais do ARIS Method. Portanto, os próximos passos deste trabalho se concentrarão na construção de um metamodelo que represente diretamente as diferentes categorias conceituais presentes no ARIS Method em seus diferentes domínios de modelagem organizacional e de arquiteturas de TI, como, por exemplo, os domínios de estrutura organizacional, processos de negócio, objetivos organizacionais, entre outros. Este metamodelo revelará metaclasses como unidades organizacionais, atividades, eventos, regras, etc., explicitando relações conceituais entre estes elementos de modelagem como relações todo-parte, relações de refinamento, especialização, etc. Este metamodelo "semântico" do ARIS Method servirá como base para avaliação ontológica do ARIS Method frente a ontologias de fundamentação como UFO [Guizzardi 2005] [Guizzardi 2008] e para a definição rigorosa da semântica de fragmentos do ARIS Method.

Agradecimentos

Este trabalho foi desenvolvido com recursos da Fundação de Apoio à Ciência e Tecnologia do Espírito Santo (FAPES) no escopo do projeto INFRA-MODELA e com

recursos do Fundo de Apoio à Ciência e Tecnologia do Município de Vitória (FACITEC) no escopo do projeto MODELA.

Referências

Almeida, J. P A.; Iacob, M.-E.; Jonkers, H.; Lankhorst, M. e van Leeuwen, D. (2007) "An Integrated Model-Driven Service Engineering Environment", Enterprise Interoperability: New Challenges and Approaches, Springer.

Eclipse Foundation (2008) "Eclipse Modeling - EMF - Home", http://www.eclipse.org/modeling/emf/

Eclipse Foundation (2008b) "Generating an EMF Model using XML Schema (XSD)", http://eclipse.org/modeling/emf/docs/2.x/tutorials/xlibmod/xlibmod_emf2.0.html.

Guizzardi, G. (2005) Ontological Foundations for Structural Conceptual Models, Telematica Instituut Fundamental Research Series, vol. 015, Enschede, Países Baixos, Telematica Instituut.

Guizzardi, G., Vascocelos, J., Segrini, B., Falbo, R., e Guizzardi, R. (2008) "Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology", IDEAS, XI workshop Iberoamaricano, Pernambuco, Brasil.

Harel, D. e Rumpe, B. (2000) Modeling Languages: Syntax, Semantics and All That Stuff, Part1: The Basic Stuff, The Weizmann Institute of Science, Israel.

HITSoftware (2008) "XML Tools : DTD, XML schema and XML document conversion software tool : XML Utilities", http://www.hitsw.com/xml_utilites/.

IBM (2008) "XML Import tool from ARIS to IBM WebSphere Business Modeler", http://www.alphaworks.ibm.com/tech/arisimport/requirements.

IDS Scheer (2006) "Aris Platform: Overview of Intefaces to ARIS 6.1x / 6.2x / 7.0x", White Paper.

IDS Scheer (2006b) "ARIS Platform: XML Export and Import", White Paper.

Kern, H. e Kühne, S. (2007) "Model Interchange between ARIS and Eclipse EMF". The 7th OOPSLA Workshop Domain-Specific Modeling, Montreal, Canada.

Kurtev, I., Bézivin, J. e Aksit, M. (2002) "Technological spaces: An initial appraisal", CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, USA.

Lankhorst, M. (2005) Enterprise Architecture at Work - Modeling, Communication, and Analysis, Springer.

Mendling, J. e Nüttgen, M. (2004) "Transformation of ARIS Markup Language to EPML", Proc. 3rd GI Workshop on Event-Driven Process Chains, Luxemburgo.

OMG (2006) Object Constraint Language, http://www.omg.org/docs/ptc/05-06-06.pdf.

Scheer, A. W. (2000) ARIS-Business Process Modeling, Springer, 3a edição.

W3Schools (2008) "DTD Tutorial", http://www.w3schools.com/dtd/default.asp.

W3Schools (2008b) "Introduction to XML Schema", http://www.w3schools.com/Schema/schema_intro.asp.