METACOMPILAÇÃO DE CLASSES PARA ACESSO A MODELOS IFC E SUGESTÕES PARA CRIAÇÃO DE CLASSES PARA ACESSO EM ALTO NÍVEL

COMPILATION OF CLASSES FOR ACCESS TO MODELS IFC AND SUGGESTIONS FOR CREATION OF CLASSES FOR HIGH-LEVEL ACCESS

doi>10.4237/gtp.v4i2.112

Cervantes AYRES Filho
Arquiteto e Urbanista e Mestre em Construção Civil pela UFPR
| ceayres@gmail.com |
| http://lattes.cnpq.br/0896114843042697 |

Sérgio SCHEER
Professor Associado, UFPR
| scheer@ufpr.br |
| http://lattes.cnpq.br/0695899382782312|

RESUMO

As *Industry Foundation Classes,* IFC, são um esquema criado em EXPRESS para descrição e transmissão inequívoca de informações na indústria da construção. Atualmente, são consideradas um dos mais importantes agentes na busca pela interoperabilidade entre os sistemas utilizados na indústria. Mais de dez anos após a sua primeira versão, entretanto, as IFC's continuam sendo utilizadas primordialmente em projetos-teste e experimentos isolados. Dentre as várias explicações para o baixo índice de implantação está o limitado número de ferramentas que permitam criar aplicações baseadas nele. A informação contida nos modelos de edifícios em formato IFC é extensamente particularizada, com as entidades que representam os elementos construtivos sendo reduzidas a tipos simples de dados e primitivos geométricos, separados em várias partes do arquivo. Essa característica precisa ser revertida durante a criação de aplicações que acessem modelos de edifícios, o que aumenta consideravelmente a tarefa de programação e dificulta a manutenção do código. Neste artigo são mostradas as principais formas de desenvolvimento de aplicações para acesso aos dados de modelos de edifícios em formato IFC. Também são apresentadas sugestões para a criação de uma ferramenta para metacompilação semiautomatizada de bibliotecas de classes Java, correspondentes às entidades IFC, que forneçam aos desenvolvedores métodos de acesso mais diretos e permitam a criação de aplicações de modo mais ágil.

Palavras-chave: BIM, IFC, ifcXML, metacompilação, binding.

ABSTRACT

The Industry Foundation Classes, IFC, are schemes created in EXPRESS for describing and transmitting unambiguous information on the construction industry. Currently, they are considered one of the most important agents in the quest for system interoperability in construction industry. More than ten years after its first version, however, the IFC's continued being used primarily on projects and test separated experiments. Among the various explanations for the low level of implementation is the limited number of tools (API) for developing new IFC-enabled applications. The information contained in the building model in IFC format is widely individualized, with the entities that represent building blocks being reduced to simple data types and geometric primitives, spread around different portions of the file. This feature needs to be reversed during the application creation that access building models, which greatly increases programming task and difficult to maintain code. This paper shows the main approaches for application development of building model data access in IFC format. In addition, it presents suggestions for the creation of an API that could allow semiautomatic metacompilation of Java Library classes corresponding to IFC entities. Such tool provides developers with more direct access methods. Thus, applications could be created easily and faster, helping to disseminate the IFC schema in the building industry.

Keywords: BIM, IFC, ifcXML, metacompilation, binding.

1. INTRODUÇÃO

A Modelagem de Informação da Construção, ou BIM (*Building Information Modeling*), é o processo de gestão de toda a informação produzida e utilizada nas diferentes fases do ciclo de vida de uma edificação, através da utilização de um modelo digital integrado. Este modelo representa tanto as características físicas como as funcionais dos elementos construtivos, reunindo dados provenientes de diferentes disciplinas envolvidas no processo construtivo, e permitindo análises e simulações diversas (EASTMAN, 1999; EASTMAN et al., 2004).

Por conta de certa desinformação a respeito desta tecnologia, não é raro encontrar usuários ou projetistas buscando a aplicação computacional definitiva, que auxilie desde as fases de concepção às análises de desempenho e simulações de operação do edifício. O escopo e as funcionalidades das aplicações computacionais destinadas a criar e processar modelos de edifícios são objeto de estudo e discussão desde o princípio do projeto auxiliado por computador (CAD), ainda na década de 1960. Desde muito cedo, ponderou-se que aplicações que pretendessem simular todas as atividades compreendidas na construção de edifícios seriam muito caras, complexas, de difícil implantação, operação e manutenção (AYRES, 2009).

Mesmo atualmente, a despeito dos enormes avanços na capacidade de processamento dos computadores, a proposta de criar grandes aplicações que contemplem todas as fases do desenvolvimento de edifícios continua sendo considerada ineficaz, como demonstram o trabalho de Magdy Ibrahim *et al*. (IBRAHIM *et al*., 2004) e o de Umit Isikdag (ISIKDAG *et al*., 2007). Além disso, o desenvolvimento de produtos, inclusive na indústria da construção, depende cada vez mais de consórcios de empresas trabalhando colaborativamente, com equipes isoladas geograficamente, atuando em segmentos específicos do projeto (GIELINGH, 2008). Aplicações de escopo muito abrangente não se adaptam bem a esta forma de trabalho, que exige adaptações rápidas e rotinas ajustadas para cada participante.

Ao invés de depender de aplicações computacionais isoladas e de escopo muito abrangente, a criação e a manutenção de modelos integrados de edifícios devem ocorrer em um sistema composto por vários tipos de aplicações. Cada uma destas possui diferentes objetivos e enfatiza diferentes porções da informação do modelo,

colaborando e compartilhando dados entre si (IBRAHIM et al., 2004). Eastman e outros autores chamam estas aplicações de BIM *Authoring Tools*, ou ferramentas BIM (EASTMAN et al., 2008).

1.1 Modelos de dados e interoperabilidade

Modelos de dados são estruturas compostas por constructos lógicos que definem a forma e o significado dos dados que representarão o ciclo de vida de um determinado produto. Durante a modelagem, as aplicações utilizam estes modelos para conformar a informação sobre o produto, gerando um modelo do produto, que é, naturalmente, uma instância do modelo de dados (LACROIX e PIROTTE, 1981; YANG *et al.*, 2008).

As diferentes aplicações utilizadas no processo de modelagem têm propósitos diversos e utilizam modelos de dados próprios – ou nativos – condicionados por especificidades dos campos e fases do processo de fabricação aos quais se destinam. Cada modelo de dados nativo é também determinado por questões relacionadas ao desempenho computacional, sistema operacional e a linguagem de programação utilizada no desenvolvimento da aplicação (PELS, 1996).

Para que possam atuar em conjunto, como um sistema, a troca de informações entre essas diferentes aplicações – ou seja – a transposição de dados entre diferentes modelos de dados – deve ocorrer sem sobressaltos, garantindo que a informação contida não seja prejudicada. Desse modo, o modelo do produto pode ser aprimorado e ampliado ao ser processado por cada uma das aplicações ao longo do seu ciclo de vida. O termo que define essa possibilidade é "interoperabilidade", que pode ser entendida como um mapeamento das entidades do modelo de dados nativo de uma determinada aplicação em relação a um modelo de dados universal, de formato neutro, que independa das especificidades e restrições impostas pelos diferentes ambientes para os quais as aplicações foram desenvolvidas.

Além de garantir que a informação possa transitar entre diferentes aplicações sem perda de significado, esse mapeamento torna desnecessário criar diferentes rotinas de transposição de informação entre o modelo de dados nativo de uma determinada aplicação e os de todas as outras com as quais a informação deve ser

compartilhada. Para cada nova aplicação desenvolvida, apenas um mapeamento é criado: entre o seu modelo de dados nativo e o modelo de dados neutro (NIBS, 2007).

1.2 STEP e EXPRESS

Um importante avanço da pesquisa de modelos de dados neutros foi a norma ISO 10303, também chamada STEP, acrônimo de *Standard for the Exchange of Product Model Data* (norma para transferência dos dados do modelo do produto). O seu objetivo é definir um formato neutro e interpretável para os dados do produto, durante todo o seu ciclo de vida, independente de sistemas específicos. A STEP é organizada em partes chamadas *Application Protocols* (protocolos de aplicação), ou APs, que definem padrões para estruturas de dados utilizadas por diferentes setores da indústria (SCRA-STEP, 2006).

Os modelos de dados STEP são definidos utilizando a linguagem de descrição de metadados chamada EXPRESS, apresentada em 1994, na parte 11 da norma STEP. A EXPRESS utiliza um esquema semântico de representação de dados, baseado em entidades, atributos e relacionamentos, e também possibilita criar generalizações e restrições para os dados (FOWLER, 1996).

Os constructos utilizados na definição de modelos de dados em EXPRESS são o *SCHEMA* – uma subdivisão funcional do modelo que permite a reutilização de informações entre modelos diferentes; o *TYPE*, que descreve tipos "primitivos" de dados, como inteiro, real, booleano, string, etc.; *ENTITY*, que representa as unidades básicas de informação, que compõem o *schema*; *SUBTYPE*, que estabelece relações hierárquicas entre entidades diferentes; *aggregations* (*SET*, *ARRAY*, *LIST*, *BAG*), que determinam coleções de tipos ou entidades; e também unidades algorítmicas: *FUNCTION*, *PROCEDURE* e *RULE*, que são utilizadas para adicionar restrições adicionais ao modelo de dados (SCHENCK e WILSON, 1994).

Modelos de dados STEP são consideravelmente mais eficientes na transmissão de informações de projetos do que os modelos de dados utilizados nos CADs baseados em primitivos geométricos. Björk, por exemplo, propôs utilizá-los para permitir a transferência de informações a respeito dos espaços, das fronteiras espaciais e das estruturas de fechamento dos edifícios, que são utilizadas em várias

aplicações de projeto e simulação (BJÖRK, 1992). Outro exemplo é a proposta de padronização de Fenves, que utiliza a EXPRESS para definir um núcleo comum de informações de projeto que viabilize a troca de dados entre vários setores da indústria. As entidades básicas do seu modelo semântico de dados são objetos genéricos capazes de descrever a forma e toda a informação complementar necessária para conduzir as atividades do projeto de produtos (FENVES, 2002).

1.3 Industry Foundation Classes - IFC

O principal objetivo da *Building Smart International* (BSI), criada em 1994, é o desenvolvimento de padrões independentes de fabricantes para interoperabilidade dos softwares utilizados na indústria da construção (EASTMAN, 1999; BSI, 2008a). O modelo de dados neutro da Building Smart International é chamado *Industry Foundation Classes*, ou IFC. A sua primeira versão foi lançada em 1997, e a versão atual é a 2X3, sendo que a 2X4 *alpha* já está disponível para testes na página da BSI na internet (BSI, 2008a). As IFC são atualmente o programa de padronização de modelos de edifícios mais proeminente da indústria da construção (HOWARD e BJÖRK, 2008).

A quantidade de dados que pode ser inserida em um modelo de edifício é enorme, pois cada elemento da construção (concreto ou abstrato) deve ser representado. O modelo IFC separa esses dados em quatro eixos de informação: ciclo de vida, disciplina, nível de detalhe e aplicações (software). A proposta da BSI não é criar uma representação detalhada e específica para cada elemento que pode ser encontrado na construção, já que isso daria origem a um modelo muito grande e de difícil implementação. Ao invés disso, os elementos construtivos são representados por classes genéricas, com informação suficiente para descrever as suas características principais. A isso se soma a possibilidade de estender uma descrição para que a representação se adapte melhor a um produto específico.

A arquitetura do modelo de dados IFC é composta por quatro níveis: *domain*, *interoperability*, *core* e *resource*. O nível *domain* é o mais alto, e permite a descrição de informações relacionadas às disciplinas envolvidas no desenvolvimento do edifício. No nível *interoperability* são descritos mapeamentos de dados para permitir a troca de informação entre diferentes *domains*. No nível *core* são descritas as

unidades de informação comuns a todos os domínios e mapeamentos, que podem ser especializadas por eles. O nível mais baixo é o *resource*, que contém a descrição de conceitos básicos e independentes, que são utilizados pelos níveis mais altos (LIEBICH e WIX, 2000).

Como o esquema IFC é desenvolvido em EXPRESS, as suas unidades básicas de informação são as *entidades*, que podem representar objetos, propriedades dos objetos, ou relações entre objetos (*ifcObject*, *ifcPropertyDefinition*, *ifcRelationship*). Objetos podem descrever elementos físicos, como paredes, espaços, equipamentos, e também elementos abstratos como tarefas, controles, processos, etc. As entidades são derivadas em sete tipos principais: produtos, processos, controles, recursos, atores, projetos e grupos (LIEBICH e WIX, 2000). As extensões das entidades básicas do modelo são criadas com a propriedade EXPRESS "*SUBTYPE OF*", e em teoria, desde que se enquadrem na arquitetura do modelo IFC, podem ser interpretadas inequivocamente em qualquer aplicação capaz de ler o modelo. O esquema IFC completo pode ser obtido no site da BSI (LIEBICH *et al.*, 2006).

1.4 SPF e ifcXML

Arquivos de modelos de edifícios criados a partir do esquema IFC podem apresentar-se em dois formatos (resultantes da mesma estrutura de meta-dados): podem ser descritos no formato SPF (STEP *Physical File*), ou então no formato ifcXML. Os arquivos de ambos os formatos são descritos em ASCII (*American Standard Code for Information Interchange*).

SPF é um formato de descrição de instâncias de modelos de dados criados em EXPRESS, e é definido na parte 21 da norma STEP, chamada "*Clear text encoding for the exchange structure*" (FOWLER, 1996; SCRA-STEP, 2006). Resumidamente, trata-se de um arquivo onde cada linha traz uma sequência de caracteres que representa o instanciamento de uma entidade do modelo EXPRESS. Em termos de linguagem de programação, pode-se imaginar o modelo EXPRESS (nesse caso as IFC) como o código no qual são descritos os tipos e as funções, enquanto o arquivo SPF seria uma sequência de chamadas para esses constructos, onde os parâmetros são as informações dos elementos que compõem o modelo do edifício.

Uma outra forma para os modelos IFC é o formato ifcXML. *Extensible Markup Language*, ou XML, é uma linguagem de metadados recomendada pelo World Wide Web Consortium (W3C) como padrão para troca de informações na Web (W3C, 2008b). Trata-se de uma linguagem textual para descrição de informações, que permite reunir em um só local tanto os dados como o significado semântico – os metadados. Ao contrário da EXPRESS, que foi criada com o propósito específico de representar modelos de produtos, a XML é essencialmente genérica, e pode ser utilizada para descrever qualquer tipo de informação. Este alto nível de abstração possibilitou a rápida disseminação da linguagem, mas é um obstáculo à garantia de consistência da informação.

Como não tem descritores e estruturas com significado fixo, a XML pode representar qualquer informação de infinitas maneiras, e qualquer uma delas será válida, desde que sigam umas poucas regras de sintaxe. Por exemplo, elementos construtivos podem ser chamados <segmento_de_parede> ou <parede>, e podem conter descritores diferentes para representar seus atributos. E ainda, a posição de uma parede pode ser descrita em relação a uma origem fixa, ou então em relação às outras paredes, e pode seguir um sistema cartesiano ou polar. Em resumo, há infinitas formas de descrever corretamente os elementos um modelo de edifício, o que para humanos torna-se uma questão de semântica rapidamente resolvida, mas para computadores pode demandar complexos mapeamentos de dados. Para auxiliar em situações como esta, foi desenvolvido XML *Schema Definition*, ou XSD. XSD é em uma linguagem de "metametadados". Ela permite descrever as estruturas que descrevem estruturas de dados (assim como a EXPRESS). Um esquema XSD define tanto os descritores que podem ser utilizados em um arquivo, como as relações possíveis e obrigatórias entre eles. Por exemplo, pode-se definir que um descritor <janela> tenha que ser obrigatoriamente associado a um descritor <parede>. Ao se criar arquivos XML a partir de esquemas XSD, é possível validar o conteúdo da informação utilizando o esquema como referência, e determinar assim a consistência da informação.

O ifcXML é justamente um esquema XSD para formatação de arquivos XML de acordo com as estruturas das IFC. No processo de criação do esquema XSD, que é conduzido pela IAI e segue a metodologia definida na STEP, as entidades e

relacionamentos do esquema EXPRESS IFC são mapeados e dão origem a descritores XML (NISBET e LIEBICH, 2007), que são então utilizados para representar os elementos dos modelos de edifícios. A primeira versão da ifcXML foi lançada em 2001, e a partir de então, para cada versão do esquema IFC, há uma versão ifcXML correspondente, mapeada automaticamente.

2. ACESSANDO ARQUIVOS DE MODELOS IFC

A despeito do seu desenvolvimento, a utilização do modelo IFC ainda é bastante limitada, e há vários desafios a serem vencidos, como a falta de recursos, ferramentas; o pequeno número de profissionais realmente envolvidos com o desenvolvimento do esquema; os processos de validação dos arquivos gerados; e as práticas de duplicação de informação que visam compensar partes do esquema que ainda não foram completamente implementadas. Entretanto, grande parte dos autores da área concorda que o modelo IFC ainda é a melhor alternativa para a viabilização dos ambientes de interoperabilidade entre as aplicações na indústria da construção (KIVINIEMI, 2006; KIM e SEO, 2008; NOUR e BEUCKE, 2008; PAZLAR e TURK, 2008).

Nesse sentido, desenvolver métodos para facilitar o desenvolvimento de aplicações baseadas no esquema IFC é essencial para fomentar a interoperabilidade. Assim como é observado repetidamente nas diversas áreas da informática – por exemplo, com a web – o envolvimento de um maior número de desenvolvedores pode aprimorar e alavancar as tecnologias.

2.1 Metacompilação e Binding

Há dois modos básicos para se extrair dados de um arquivo contendo um modelo de edifício em formato IFC, seja ele descrito em SPF ou ifcXML. Como os arquivos são codificados em ASCII, pode-se criar manualmente rotinas para leitura a partir da identificação das partes que compõem as suas linhas (*parsing* e *tokenização* de *strings*). Este modo de acesso é relativamente mais simples no caso dos arquivos ifcXML, pois ferramentas bem documentadas e facilmente encontradas, como a SAX e a DOM podem ser aplicadas para realizar o processo de *parsing*, que é a varredura dos nós do arquivo XML e a construção de uma estrutura hierárquica de

objetos que representam abstratamente as estruturas IFC (SAXPROJECT, 2008; W3C, 2008a).

Essa abordagem, embora aparentemente simples, torna-se muito trabalhosa devido à complexidade do modelo IFC, que tem mais de 600 entidades e milhares de parâmetros e relacionamentos entre elas. Nour, por exemplo, chegou a optar pela criação manual de classes para extrair dados de arquivos SPF, mas precisou selecionar apenas as entidades IFC mais comumente utilizadas, dada a enormidade do trabalho que seria necessário para cobrir todo o esquema IFC (NOUR, 2006). Mesmo que o enorme esforço de codificar classes para todos os componentes do esquema IFC fosse aceito, a aplicação criada seria pouco flexível e adaptável, pois sempre que o esquema fosse atualizado, seria necessário corrigir manualmente toda a estrutura de *parsing* criada.

Considerada apenas a extração de dados, a criação manual de rotinas de acesso aos arquivos IFC ou ifcXML já seria desaconselhável. Mas, obviamente, o processamento de modelos de edifícios exige também a modificação de elementos existentes e a inserção de novos elementos. Nessa situação, seria necessário criar rotinas para reorganizar ou recriar a sequência de caracteres do arquivo, após as transações de dados. Mais ainda, a sequência de caracteres gerada – ou seja, o conteúdo do arquivo SPF ou ifcXML gerado – precisa passar pelo processo de validação, para que se garanta a sua conformidade em relação ao esquema IFC ou o XSD da ifcXML (AYRES, 2009).

A forma recomendada para acessar dados de arquivos IFC é utilizar o processo de metacompilação e *binding*. Nesse caso, para os arquivos SPF, inicia-se com uma aplicação chamada *Standard Data Access Interface* (SDAI), que é utilizada para compilar a definição em EXPRESS das IFC, dando origem a estruturas de programação correspondentes às suas entidades. Essa operação segue especificações da própria norma ISO 10303, parte 22, que define um modelo para a criação de APIs (*Application Programming Interfaces*) para acesso a dados STEP (ISIKDAG *et al.*, 2007). Além das classes que correspondem às entidades do modelo, é gerado também um conjunto de classes responsáveis pelo gerenciamento das transações de dados realizadas nos arquivos SPF, ou seja, nos modelos de edifícios. Originalmente, SDAIs para C, C++ e FORTRAN foram definidas, mas

atualmente também há SDAIs para Java, como a JSDAI, por exemplo (LKSOFTWARE, 2008).

A figura 1 mostra um fragmento do esquema IFC 2X3, a entidade IfcBlock, que representa prismas de base retangular. Após a metacompilação na JSDAI, por exemplo, seria gerada uma classe correspondente a essa entidade, chamada IfcBlock(). Essa classe seria uma extensão da superclasse IfcCsgPrimitive3D(), e conteria três parâmetros: XLength, YLength e ZLength, todos do tipo IfcPositiveLengthMeasure. A metacompilação também providenciaria os métodos de acesso aos dados – neste caso, getXLenght(), setXLength(), getYLength() e assim por diante.

```
...
ENTITY IfcBlock
 SUBTYPE OF (IfcCsgPrimitive3D);
      XLength : IfcPositiveLengthMeasure;
      YLength : IfcPositiveLengthMeasure;
      ZLength : IfcPositiveLengthMeasure;
END_ENTITY;
...
```

Figura 1: entidade IfcBlock em EXPRESS

O processo de metacompilação também pode ser realizado para gerar classes que permitam acessar dados de arquivos ifcXML. Nesse caso, ao invés de compilar o esquema IFC original, utiliza-se o XSD da ifcXML. O fragmento do esquema ifcXML correspondente à entidade ifcBlock é mostrada na figura 2. Assim como a compilação do esquema EXPRESS da IFC, a classe equivalente, bem como os métodos de acesso aos seus dados seriam gerados automaticamente. Diferentemente da classe gerada a partir do esquema EXPRESS, porém, os métodos das classes geradas a partir da compilação do esquema XSD contém rotinas mais extensas, que percorrem recursivamente os nós do arquivo ifcXML, extraindo os dados no processo. Há várias ferramentas próprias para metacompilação de esquemas XSD disponíveis gratuitamente (W3C, 2008c). A XMLBeans, por exemplo, cria bibliotecas de classes automaticamente a partir da compilação do esquema XSD da ifcXML para Java (APACHE, 2008).

```
...
<xs:complexType name="IfcBlock">
  <xs:complexContent>
    <xs:extension base="ifc:IfcCsgPrimitive3D">
      <xs:sequence>
        <xs:element name="XLength" type="ifc:IfcPositiveLengthMeasure">
        </xs:element>
        <xs:element name="YLength" type="ifc:IfcPositiveLengthMeasure">
        </xs:element>
        <xs:element name="ZLength" type="ifc:IfcPositiveLengthMeasure">
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
...
```

Figura 2: entidade IfcBlock em ifcXML

O emprego do processo de metacompilação reduz drasticamente o trabalho de programação necessário para acessar os dados de modelos de edifícios, além de permitir uma rápida atualização das classes derivadas, sempre que o esquema IFC for atualizado pela BSI (NOUR e BEUCKE, 2008). A BSI sugere várias destas ferramentas de metacompilação – às quais chama *IFC Toolboxes* – em sua página da Internet (BSI, 2008b). As funcionalidades oferecidas pelas bibliotecas criadas na metacompilação dependem da *IFC Toolbox* utilizada. *Toolboxes* genéricas criam apenas as estruturas de classes equivalentes às entidades e um conjunto básico que ordena as transações de dados. Associar as diferentes entidades (por exemplo, todas as portas e janelas que fazem parte de uma parede) ou gerar representações e análises a partir delas fica a cargo do programador, como demonstrado no trabalho de Treeck e outros autores (2003). *Toolboxes* de programação em alto nível, por outro lado, além das estruturas de classes, geram rotinas para associar automaticamente as entidades distribuídas pelo arquivo SPF, podendo então fornecer informações mais complexas. Isso reduz o trabalho de programação necessário para se obter informações a partir dos modelos de edifícios. *Toolboxes* de programação em alto nível podem, por exemplo, conter rotinas que produzem a representação tridimensional das entidades do modelo, liberando o programador dessa tarefa.

2.2 Late Binding e Early Binding

Há duas formas de associar as classes geradas pelo processo de metacompilação às aplicações em desenvolvimento: *late binding* e *early binding*. No *late binding*, apenas as classes genéricas que aceitam parâmetros abstratos e as classes que controlam as transações de dados são utilizadas. Como exemplo, tem-se após a compilação do esquema IFC a classe IfcEntity(), que pode ser utilizada para acessar qualquer entidade do arquivo, já que todas são subclasses da primeira. Nesse caso, os atributos tornam-se também abstratos, e a sua natureza precisa ser identificada através rotinas formadas por sequências de testes lógicos com constantes literais.

Resumidamente, usando *late binding*, não haveria um método getId() para extrair o valor do parâmetro Id de uma determinada entidade IFC. O que seria disponibilizado para o programador seria o método genérico getAttribute(), que pode se referir a qualquer parâmetro de qualquer entidade. Seria então passado para este método, via variável *string* ou constante literal, o nome do parâmetro desejado (nesse caso, "Id"). As classes genéricas incluídas na biblioteca se encarregam de gerar uma exceção caso identifiquem que a classe correspondente à entidade que está sendo acessada não possua um parâmetro chamado "Id", porém antes que isso seja possível, cabe ao programador identificar a natureza da própria classe – ou seja a *qual* entidade IFC essa classe se refere, pois o acesso via *late binding* não provê essa funcionalidade.

Abstração nesse nível é um problema para modelos de dados bem definidos como as IFC. É preciso garantir que toda a informação processada e produzida pela aplicação seja validável – ou seja, que esteja de acordo com o esquema EXPRESS ou ifcXML. Utilizando classes genéricas e passando para elas o nome dos atributos através de strings, não haveria empecilho para a criação de um arquivo contendo uma entidade chamada "minha_parede", que evidentemente não existe no esquema original. Mesmo que o nome da entidade entre aspas se referisse a uma entidade IFC válida, sempre há o risco de serem cometidos erros de digitação. Ambientes de programação, como o Eclipse, contém funcionalidades que auxiliam o programador na identificação desses erros, quando ocorrem na porção interpretável do código. Porém se o erro ocorrer no interior de uma constante literal – por exemplo, em getEntity("IfcWaal") ao invés de getEntity("IfcWall") –

não haverá qualquer alerta durante a compilação e, ainda pior, talvez sequer se identifique o problema durante a execução do programa, dependendo das rotinas de exceção programadas.

Classes e métodos abstratos são úteis para acessar pequenos arquivos e para manter o código do programa flexível e adaptável, capaz de interpretar vários esquemas EXPRESS ou XSD diferentes. Entidades simples, compostas de parâmetros atômicos, são facilmente acessadas por rotinas básicas. Porém, o trabalho com estruturas mais complexas exige muita programação adicional (AYRES, 2009). No caso do esquema IFC, as principais entidades são constructos que relacionam várias outras entidades aninhadas, em uma estrutura hierárquica. Por isso, embora a metacompilação auxilie com a geração das classes genéricas e as rotinas de transação de dados, a utilização do *late binding* acaba por demandar a criação de rotinas extensas para discernir cada uma das classes localizadas acaba sendo um trabalho quase tão exaustivo como programar manualmente as classes para acessar os arquivos, sem utilizar qualquer tipo de *binding*.

Outro ponto desfavorável ao uso do *late binding* é a relativa estabilidade do esquema IFC. As suas atualizações, por parte da BSI, ocorrem em intervalos razoavelmente longos, e a extensão das suas entidades segue regras que permitem que novos elementos e propriedades sejam representados sem a criação de novos elementos atômicos. Por isso, não há grandes vantagens em criar aplicações com classes muito genéricas ou abstratas.

No *early binding*, por outro lado, classes e métodos concretos são utilizadas no desenvolvimento de aplicações. Isso significa, basicamente, substituir o uso de constructos abstratos e dependentes de constantes literais como getEntity("IfcWall") por getIfcWall(), ou getAttribute("Id") por getId(). Essa abordagem limita a ocorrência de erros de digitação – que são informados em alertas durante a compilação – e também de definição incorreta de entidades – que produzem exceções durante a execução e permitem identificar facilmente a porção do código a ser corrigida. Uma estrutura bem definida de classes é também uma ótima maneira de facilitar o entendimento da estrutura de dados para os programadores envolvidos no desenvolvimento da aplicação.

Por exemplo, a classe metacompilada IfcWallStandardCase(), que representa a entidade IFC de mesmo nome, contém métodos concretos que retornam o valor dos atributos, como getOwnerHistory() e getObjectPlacement(). Quando os atributos a serem extraídos são diretos e simples, como no caso de tipos atômicos (números, strings, booleanos), métodos *get* e *set* fornecem meios rápidos para manusear os dados. Nas figuras 3 e 4 são mostrados fragmentos de arquivos SPF e ifcXML, respectivamente, contando instâncias da entidade IfcWallStandardCase, que é utilizada para representar paredes simples.

```
...
#129= IFCWALLSTANDARDCASE('3JEUWb65nCGPaW0ls_s4xI',#13,' SW -
002',$,$,#126,#207,'3D68AD52-C2B9-4884-B3-BC-6FB7E720C6F6');
...
```

Figura 3: fragmento de arquivo SPF mostrando instância de IfcWallStandardCase

```
...
<IfcWallStandardCase id="i1714">
  <GlobalId>1WfU$nJCrCZO_HrHunLtl5</GlobalId>
    <OwnerHistory>
      <IfcOwnerHistory xsi:nil="true" ref="i1568"/>
    </OwnerHistory>
    <Name>SW - 002</Name>
    <ObjectPlacement>
      <IfcLocalPlacement xsi:nil="true" ref="i1711"/>
    </ObjectPlacement>
    <Representation>
      <IfcProductDefinitionShape id="i1793">
        <Representations ex:cType="list">
          <IfcShapeRepresentation pos="0" xsi:nil="true" ref="i1753"/>
          <IfcShapeRepresentation pos="1" xsi:nil="true" ref="i1786"/>
        </Representations>
       </IfcProductDefinitionShape>
    </Representation>
    <Tag>3B09AD2E-868F-48F9-A4-80-F4F953F6B0E1</Tag>
</IfcWallStandardCase>
...
```

Figura 4: fragmento de arquivo ifcXML mostrando instância de IfcWallStandardCase

Nestas duas situações, para obter o valor do parâmetro "Name" (que designa o nome do segmento de parede), basta utilizar o método getName(), que neste caso retornaria "SW – 002".

3. DEFICIÊNCIAS DA METACOMPILAÇÃO DE CLASSES

As deficiências das bibliotecas geradas pela metacompilação começam a surgir quando os atributos a serem extraídos são complexos – ou seja – são outras entidades IFC, e não tipos simples de dados. Quando certo atributo de uma entidade é uma outra entidade, surge no arquivo SPF uma referência, indicada pelo caractere # seguido de um número. Este código refere-se à sequência da entidade referenciada, dentro do bloco DATA do arquivo SPF, onde estão enfileiradas todas as entidades que compõem o modelo. No exemplo da figura 3, o atributo OwnerHistory é a décima terceira entidade do bloco DATA. No caso do formato ifcXML, há duas formas possíveis para referenciar atributos complexos: as entidades referenciadas podem ser descritas logo abaixo do nó XML que fez a referência (entidades aninhadas), como se pode ver no atributo Representation da figura 4; ou então através do atributo "ref", que aparece no atributo OwnerHistory da mesma figura. O atributo "ref" indica um código seqüencial que identifica cada instância de entidade IFC descrita no arquivo ifcXML.

Essa estratégia de referenciamento tem dois objetivos: permitir que porções de informação sejam reutilizadas (quando são referenciadas por várias entidades simultaneamente); e reduzir a complexidade do arquivo. Sem referenciamentos, a leitura das linhas do arquivo SPF ou dos nós do arquivo ifcXML seria difícil, se é que seria praticável, dada a complexidade de algumas entidades IFC. Disso se conclui que identificar entidades referenciadas que são descritas em diferentes partes do arquivo SPF ou ifcXML é situação das mais recorrentes durante o processamento de modelos IFC.

Porém, embora as classes geradas pela metacompilação possam identificar o valor do atributo complexo como uma entidade, não provém rotinas para localizar a sua descrição dentro do arquivo. Utilizando a JSDAI, por exemplo, seria retornado pelo método um objeto condizente com o tipo do atributo, porém nulo, com exceção do parâmetro Id (ou outro similar, utilizado para identificar as entidades contidas no arquivo). Cabe então ao programador desenvolver rotinas para primeiramente identificar a ocorrência do atributo complexo e do retorno de um atributo nulo (contendo apenas o código que indica a entidade referenciada) e em seguida

percorrer todas as entidades do arquivo em busca da que contenha o atributo de identificação com o mesmo valor.

Essa é a principal dificuldade no trabalho com as classes metacompiladas – a falta de métodos mais diretos para o acesso de atributos. Informações descritas em arquivos IFC são decompostas até os primitivos de dados, uma qualidade necessária para manter o esquema flexível e robusto. Em entidades mais simples, sem aninhamentos ou atributos complexos, esta característica não chega a ser um problema. Porém acessar atributos de entidades mais elaboradas é um desafio, porque a informação necessária pode encontrar-se distribuída por dezenas de outras entidades em locais diferentes do arquivo. Na figura 5 é apresentado um esquema da sequência de atividades necessárias para se obter, a partir de um modelo de edifício em ifcXML, o volume de uma parede e o seu material.

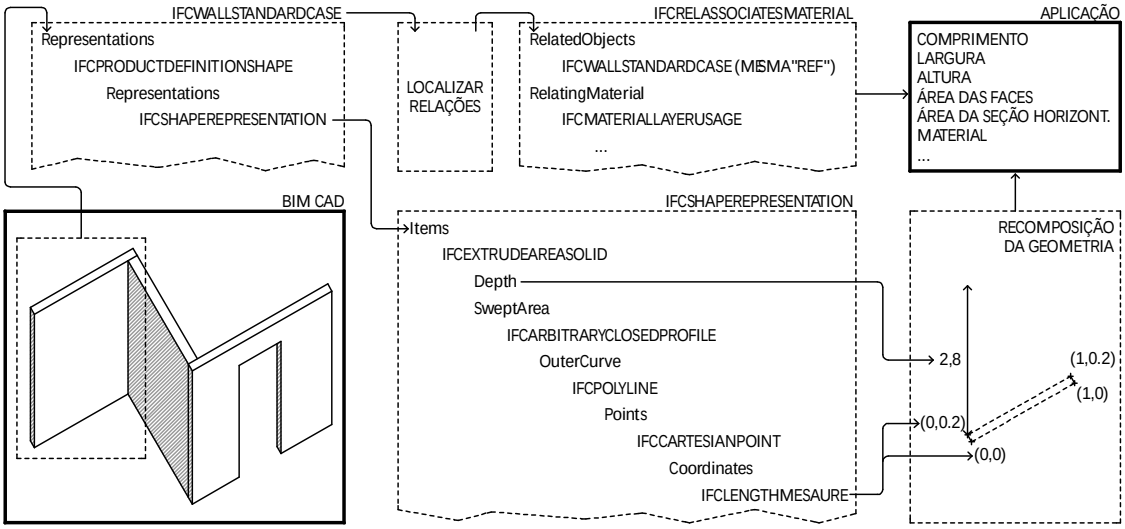

Figura 5: Etapas necessárias para reconstrução da informação a partir de um modelo

A exportação do modelo de uma aplicação CAD para o formato SPF ou ifcXML é um processo de mapeamento do modelo de dados proprietário (interno ao sistema) para o padrão neutro IFC. Os diferentes objetos paramétricos que representam os elementos construtivos são convertidos em entidades IFC. Objetos paramétricos representando paredes simples, de seção regular, como as mostradas na figura 5, podem ser convertidos em entidades IfcWallStandardCase. Para identificar o seu volume, é necessário acessar o atributo Representation, o que retorna duas

referências a entidades IfcShapeRepresentation. No esquema IFC, diferentes representações podem ser atribuídas a uma entidade, e no caso da IfcWallStandardCase há no mínimo duas: uma para o eixo da parede, representado no plano horizontal, e outra para o prisma que a representa tridimensionalmente. A IfcShapeRepresentation pode ser formada por diferentes entidades de construção geométrica, reunidas no atributo Items. Neste exemplo, há apenas uma entidade de representação, a IfcExtrudedAreaSolid, que constrói sólidos a partir da extrusão de um polígono. O atributo Depth da IfcExtrudedAreaSolid é o valor geométrico mais facilmente acessado desta entidade, e representa, neste caso, a altura da parede. Para identificar as outras dimensões da parede, é necessário localizar os pontos cartesianos que formam o polígono que dá origem à operação de extrusão, e calcular as distâncias entre eles. Cada ponto geométrico é desmembrado em dois números reais que formam a abscissa e a ordenada, e é armazenado em uma entidade IfcLengthMeasure separada. As expressões da figura 6, que acessam a representação e em seguida a componente X de um dos pontos da base da parede dão uma idéia da tarefa necessária.

```
...
IfcShapeRepresentation shape = (IfcShapeRepresentation)
    wall.getRepresentation().getIfcProductRepresentation()
    .getRepresentations().getIfcRepresentationArray(0);

[...]

IfcExtrudedAreaSolid extruded = (IfcExtrudedAreaSolid)
    shape.getItems().getIfcRepresentationItemArray(0);

[...]

IfcPolyline poly = (IfcPolyline)
    extruded.getSweptArea().getIfcArbitraryClosedProfileDef().
    getOuterCurve().get(0);

Real x1 = poly.getPoints().getIfcCartesianPointArray(0).getCoordinates().
    getIfcLengthMeasureArray(0).getDoubleValue();
...
```

Figura 6: obtendo a coordenada x de um dos pontos geométricos da base de uma parede

Vale lembrar que essas operações são intercaladas por rotinas (representadas na figura por [...]) de identificação dos tipos contidos nos objetos retornados, porque conjuntos que comportam mais de um tipo de entidade são abstratos, e não fornecem os métodos necessários para acessar os dados. Também é preciso criar

rotinas para fazer buscas no arquivo pelas entidades complementares, porque os resultados das duas primeiras expressões são referências, e não entidades.

Depois de identificar toda a informação e elaborar uma expressão para calcular o volume resultante a partir da área formada pelos quatro pontos e a altura da extrusão, o programador precisa criar uma rotina para realizar outra série de buscas por entidades relacionadas, agora para identificar o material associado à parede. Esse segundo grupo de operações traz uma desvantagem: a entidade IfcWallStandardCase não tem um atributo para descrever o material. Esta informação é relacionada indiretamente, por uma instância da entidade IfcRelAssociatesMaterial, localizada em outra parte do arquivo. Depois de uma sequência de localização e identificação de entidades referenciadas, obtém-se o nome e os demais atributos do material associado.

Estas limitações das classes geradas automaticamente demonstram que o processo de metacompilação padrão, que utiliza apenas o esquema EXPRESS ou o XSD, não atende adequadamente as necessidades dos programadores durante o desenvolvimento de aplicações para a BIM. Acessar a informação dos modelos IFC no seu nível mais baixo, com a maior decomposição possível das entidades é útil em algumas situações, mas na maioria das ocasiões o programador precisará de acesso em níveis mais altos.

As bibliotecas de classes disponibilizadas pelos fabricantes de CADs para acesso aos seus modelos de dados proprietários, como os formatos de arquivo do Revit ou do ArchiCAD, possuem métodos mais diretos, facilitando o trabalho de programação. A mesma informação obtida pelos processos acima exigiria poucas expressões – por exemplo, wall.getVolume() e wall.getMaterial().getName(). Por outro lado, modelos de dados proprietários são rígidos demais para acomodar a diversidade de visões sobre os dados que é inerente ao ambiente de interoperabilidade BIM (AYRES, 2009).

Uma possível solução é a associação das vantagens das ambas as abordagens: a liberdade de extensão de dados e a robustez do modelo neutro IFC e as classes com métodos mais diretos oferecidas pelas APIs (*Application Programming Interfaces*) dos fabricantes de *softwares*.

4. SUGESTÕES PARA UMA METACOMPILAÇÃO EM NÍVEL MAIS ALTO

Durante o desenvolvimento de aplicações típicas para a BIM, o programador necessita de métodos diretos, que facilitem o acesso aos dados do modelo em um nível razoavelmente alto. Expressões desejáveis seriam semelhantes às apresentadas na figura 7.

```
real x = wall.getLength();
real vol = wall.getVolume();
string material = wall.getMaterial().getName();
```

Figura 7: expressões mais simples para acessar modelos IFC

Como exposto na seção 2, a criação manual de classes com funcionalidades finamente sintonizadas com as informações que são mais importantes para o desenvolvimento de aplicações para a indústria da construção é uma tarefa árdua. Há mais de 600 entidades no esquema IFC, com milhares de atributos a serem revistos, e a dificuldade de atualização da biblioteca resultante quando o esquema IFC fosse atualizado surge como séria desvantagem nesta abordagem. Uma outra solução, mais robusta e flexível, seria criar essas classes automaticamente, através de um processo de metacompilação. Porém, como foi demonstrado, as classes resultantes carecem de métodos de acesso mais diretos, para facilitar a tarefa de desenvolvimento de aplicações para a BIM.

A abordagem proposta por este trabalho combina as vantagens destas duas. A solução sugerida é associar mais de um esquema durante a metacompilação. O esquema original EXPRESS ou XSD, que contém as unidades de informação (entidades, atributos e relacionamentos) e esquemas complementares, que forneceriam relacionamentos adicionais entre as entidades e atributos. A partir disso, classes com métodos de acesso em nível mais alto seriam geradas automaticamente.

Semântica é a questão chave para a criação automática dessas classes. O esquema original fornece a natureza dos elementos construtivos e seus componentes, e os esquemas complementares fornecem o conhecimento sobre como combinar os elementos para produzir as informações de nível mais alto. Por exemplo, para

originar o método de acesso getVolume(), para uma classe que represente a entidade IfcWallStandardCase, a metacompilação combinaria as definições do esquema original com outro contendo "instruções" sobre como combinar e calcular os atributos do esquema original para que se obtenha o resultado: onde se localizam os dados necessários, que tipo de equações são utilizadas, que formato tem o dado de saída, etc.

Idealmente, o processo de geração dos esquemas complementares deveria ser igualmente automatizado: uma compilação a partir de um conjunto de ontologias que definissem claramente, por exemplo, o que é uma parede, que elementos a compõem, e como o conhecimento sobre a sua construção é agregado ao esquema IFC. Os recentes desenvolvimentos das bibliotecas IFD (*International Framework for Dictionaries*) estendem o significado das associações de entidades IFC e criam um esquema sobreposto que descreve de maneira mais fiel os objetos dos modelos BIM, além de relacionar grupos de entidades que representam conceitos que variam de país para país, como por exemplo, que entidades IFC compõem uma porta (IAI, 2008b; a).

Esquemas IFD poderiam ser utilizados como complementação semântica ao esquema IFC, porém apenas dados, por mais claramente definidos que sejam, ainda não são suficientes para gerar classes mais funcionais. É necessário também incluir os procedimentos a serem realizados com esses dados pelas classes geradas, o que exigiria um esquema ainda mais ampliado. Ontologias poderiam tornar-se uma resposta para esta necessidade de complementação do esquema IFC com procedimentos, mas desenvolvimentos nesse sentido ainda são muito preliminares e antes que venham a permitir automatização completa da geração do esquema complementar, essa operação poderia ser realizada de forma parcialmente automatizada, em duas etapas, descritas a seguir.

4.1 Etapa automatizada

Grande parte das dificuldades apresentadas pelo uso das classes geradas a partir de processos de metacompilação vem da estrutura hierárquica que é produzida. Embora a sua complexidade reflita a complexidade do esquema original, não há muita inteligência na associação das classes resultantes. Pode-se mesmo assumir

que a estrutura hierárquica das bibliotecas de classes metacompiladas não chega a ser complexa, apenas extensa.

O escopo limitado das classes não lhes permite reconhecer relacionamentos transcendentes – ou seja, os métodos das classes que são referenciadas por outras classes. O escopo da entidade IfcWallStandardCase, por exemplo, não inclui referência alguma à entidade IfcPolyline, por isso não é possível obter diretamente as polilinhas que representam a base de uma parede. A questão fundamental para criação de métodos mais diretos, portanto, é "informar" às classes quais são as entidades que podem ser referenciadas a ela, dentro de um contexto pré-determinado de níveis hierárquicos.

Uma possibilidade seria converter o esquema IFC ou o ifcXML para a linguagem OWL, e extrair da ontologia resultante os relacionamentos transcendentes entre as entidades, por inferência. Porém esse tipo de conversão já foi testado e gerou discrepâncias (SCHEVERS e DROGEMULLER, 2006), o que poderia comprometer a validade da informação contida nos modelos gerados pelas aplicações.

Outra abordagem, mais direta, seria incluir estes relacionamentos adicionais diretamente em um esquema EXPRESS ou XSD. Uma rotina recursiva leria os descritores do esquema original, reconheceria os relacionamentos hierarquicamente inferiores, e geraria descritores adicionais nos níveis superiores, dando origem a um esquema aumentado. Estes descritores simulariam a presença das estruturas de dados que ocorrem somente nos níveis inferiores, e quando a metacompilação fosse realizada, o escopo das classes dos níveis superiores passaria a incluir métodos correspondentes aos descritores adicionais. Por exemplo, a entidade IfcWallStandardCase teria um descritor para acomodar a entidade ifcShapeRepresentation, que no esquema original encontra-se 4 níveis hierárquicos abaixo.

Evidentemente, os descritores adicionais não têm por objetivo redefinir a estrutura de dados da IFC. Eles seriam apenas atalhos para forçar a metacompilação a criar métodos capazes de acessar a informação de modo mais direto. Por isso, devem ser desenvolvidas estratégias para separar os descritores adicionais dos originais, possivelmente inserindo um prefixo antes do nome do descritor adicional. Outra

questão a ser avaliada é o nível de recursividade desta operação, ou seja, até que ponto compensa fazer o "escalamento" dos métodos dos níveis inferiores para os superiores.

Os métodos gerados pelos descritores adicionais passariam parâmetros para rotinas genéricas de localização e identificação das entidades referenciadas, retornando-as diretamente, de modo transparente para o desenvolvedor. Essas rotinas seriam o núcleo da biblioteca de classes e eliminariam as enormes cadeias de métodos "get" que são necessárias quando se utiliza classes produzidas por um processo de metacompilação padrão, baseado apenas no esquema IFC EXPRESS ou XSD original. Além disso, os métodos adicionais poderiam retornar coleções de entidades relacionadas, agrupadas por atributos de relacionamento. Por exemplo, a classe correspondente à entidade IfcWallStandardCase poderia ter um método chamado getReferencedIfcCartesianPoints(), que retornasse uma coleção contendo toda e qualquer entidade IfcCartesianPoint relacionada, em qualquer nível hierárquico dentro da entidade principal. Esses pontos poderiam conter parâmetros simulando os relacionamentos inversos, que não são providos pelo esquema ifcXML, indicando ao programador a sua procedência.

4.2 Etapa manual

Por melhor que sejam descritos os dados, algumas situações mais especializadas requerem também procedimentos. Como não há esquemas disponíveis para automatizar métodos capazes de atender a esta necessidade, seria necessário fornecer aos desenvolvedores a possibilidade de refinar as classes manualmente. Por exemplo, a criação automática de um método chamado getVolume() para a ifcWallStandardCase dependeria de um esquema semântico que associasse implicitamente as entidades que contém a informação necessária e a operação matemática a ser realizada a partir delas. Na falta deste esquema, um ambiente de aprimoramento das metaclasses poderia ser criado para associar manualmente os dados necessários, e para permitir a definição dos procedimentos diretamente em linguagem de programação.

Este ambiente poderia se valer de uma interface gráfica, semelhante às utilizadas em ferramentas de definição de esquemas RDF ou EXPRESS-G, onde o

desenvolvedor selecionaria entidades e atributos que pretendesse associar, seja no esquema IFC original ou em esquemas complementares, que se sobrepõem ao esquema original, e reorganizam as suas entidades para se adaptar às visões específicas sobre os dados (incluem-se aí diferenças entre disciplinas da indústria da construção ou entre países, por exemplo). A figura 8 mostra um esquema básico da ferramenta sugerida.

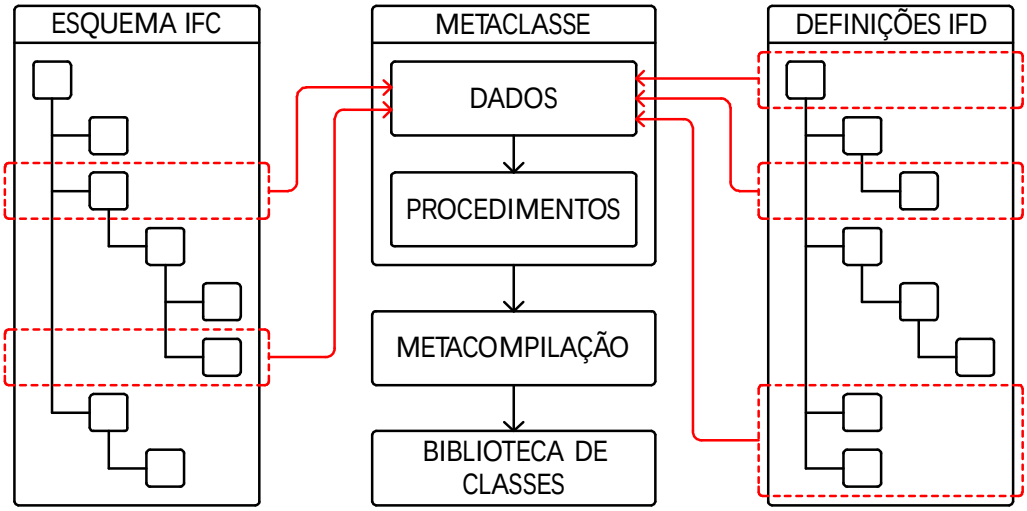

Figura 8: Esquema básico da ferramenta sugerida

Depois da definição dos dados, é criado o código para processá-los. A melhor linguagem para isso seria a própria linguagem na qual as classes seriam geradas. Uma eficiente abordagem seria utilizar os recursos já disponíveis no ambiente de programação Eclipse, e criar um plug-in ou mesmo uma interface complementar (*benchwork*) ao invés de uma aplicação completamente nova. Neste ambiente, as classes geradas automaticamente na primeira fase seriam utilizadas ou refinadas, dando origem a bibliotecas mais refinadas. Na figura 8, por exemplo, é mostrada a associação hipotética entre entidades do esquema IFC e de definições de um esquema IFD.

Desse modo, garante-se a flexibilidade dos esquemas, e ao mesmo tempo automatiza-se consideravelmente a geração de classes. A biblioteca de classes obtida pode ser utilizada em várias aplicações, reduzindo o trabalho de programação. De fato, não é necessário que todos os desenvolvedores realizem

estas etapas. Pode-se imaginar vários níveis de acesso à metacompilação, e o desenvolvimento de bibliotecas modulares, que atendam as diferentes necessidades dos desenvolvedores de aplicações para as diferentes disciplinas da construção. Por exemplo, pode ser desenvolvida uma biblioteca genérica para acesso a geometria em alto nível, que pode ser utilizada por aplicações orientadas para qualquer disciplina da indústria. Ou então bibliotecas especializadas, para atuar em conjunto com as primeiras e fornecer funcionalidades próprias de cada disciplina. Pode-se imaginar uma biblioteca de classes que atuassem como pré-processador para aplicações de simulação de desempenho físico, e assim por diante.

Desenvolvedores poderiam também acessar essas bibliotecas em nível intermediário, fazendo ajuste nas associações ou refinando os processamentos, dando origem a uma biblioteca mais ajustada às suas necessidades. Finalmente, um número maior de programadores sequer precisaria realizar qualquer tarefa de metacompilação, pois um grande número de bibliotecas de classes de acesso em alto nível poderia ser disponibilizado, e a sua combinação seria suficiente para a maioria dos desenvolvimentos de aplicativos de acesso ao modelo.

5. CONSIDERAÇÕES FINAIS

Neste trabalho foram relatadas brevemente algumas observações sobre as dificuldades encontradas no desenvolvimento de aplicações para acessar modelos IFC. A complexidade das estruturas de dados do modelo é um resultado direto da complexidade dos objetos a serem representados e das características das informações produzidas na indústria da construção. A decomposição da informação ao nível dos primitivos é uma estratégia de flexibilização que garante a robustez do padrão de dados. Porém, para as situações cotidianas de desenvolvimento de aplicações – principalmente em desenvolvimentos rápidos – é necessário prover formas de acessar as informações em um nível mais alto, agilizando a geração do código.

O processo de metacompilação de classes é uma solução interessante para estas situações, entretanto resulta em classes com funcionalidades muito limitadas, que não se ajustam bem às especificidades do acesso de dados em modelos IFC. Foram

propostas sugestões para o desenvolvimento de uma biblioteca de classes com métodos de acesso mais diretos, em um processo semi-automatizado. A partir da varredura do esquema IFC EXPRESS ou XSD seriam incluídos descritores adicionais, resultando em um esquema aumentado, que seria então metacompilado. Em seguida, as classes geradas poderiam ser refinadas em um ambiente com interface gráfica que permitisse a seleção de entidades do esquema original e esquemas complementares, como os IFD. As classes geradas teriam uma funcionalidade mais adaptada às necessidades diferenciadas do desenvolvimento de aplicações para a indústria da construção.

A maior facilidade no desenvolvimento de aplicações que acessem modelos em formato IFC pode colaborar para a disseminação do padrão, que atualmente ainda é tido como a melhor alternativa para permitir a interoperabilidade entre aplicações utilizadas na BIM.

6. REFERÊNCIAS

APACHE. **XMLBeans** (página da internet). 2008. http://xmlbeans.apache.org/, acessado em 12.2008.

AYRES, C. **Acesso ao Modelo Integrado do Edifício**. Curitiba, 2009, 149 p. Dissertação (Mestrado). Programa de Pós Graduação em Construção Civil, Universidade Federal do Paraná. Disponível em http://hdl.handle.net/1884/20219. Acessado em: 10.2009.

BJÖRK, B.-C. A conceptual model of spaces, space boundaries and enclosing structures. **Automation in Construction**, v. 1, n. 3, 1992, p.193-214. Disponível em http://www.sciencedirect.com/science/journal/09265805. Acessado em: 12.2008.

BSI. **Building Smart International** (página da internet). 2008a. http://www.buildingsmart.com/, acessado em 12.2008.

BSI. **IFC Tools for Developers** (página da internet). 2008b. http://www.iai-international.org/software/Tools%20for%20IFC%20developers.html, acessado em 12.2008.

EASTMAN, C. M. **Building Product Models: Computer Environments Supporting Design and Construction**. Boca Raton: CRC Press, 1999, 424 p.

EASTMAN, C. M.; SACKS, R. e LEE, G. **Functional modeling in parametric CAD systems**. In: ACADIA Conference 2004, 2004, Toronto. Disponível em http://bim.arch.gatech.edu/data/reference/Functional%20modeling%20in%20parametric%20CAD%20systems_GCAD2004.pdf. Acessado em: 12.2008.

EASTMAN, C. M.; TEICHOLZ, P.; SACKS, R. e LISTON, K. **BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors**. Hoboken: Wiley, 2008, 490 p.

FENVES, S. J. **A Core Product Model for Representing Design Information - NISTIR 6736**. Boca Raton: NIST - National Institute of Standards and Technology, 2002, 424 p.

FOWLER, J. **STEP for Data Management Exchange and Sharing**. Twickenham: Technology Appraisals, 1996, 222 p.

GIELINGH, W. An assessment of the current state of product data technologies. **Computer-Aided Design**, v. 40, n. 7, 2008, p.750-759. Disponível em http://www.sciencedirect.com/science/journal/00104485. Acessado em: 12.2008.

HOWARD, R. e BJÖRK, B.-C. Building information modelling - Experts' views on standardisation and industry deployment. **Advanced Engineering Informatics**, v. 22, n. 2, 2008, p.271-280. Disponível em http://dx.doi.org/10.1016/j.aei.2007.03.001. Acessado em: 11.2008.

IAI. **IFD Library** (página da internet). 2008a. http://dev.ifd-library.org/index.php/Main_Page, acessado em 12.2008.

IAI. **IFD Library White Paper**. IAI, 2008b, 9 p.

IBRAHIM, M.; KRAWCZYK, R. e SCHIPPOREIT, G. **Two Approaches to BIM: A Comparative Study**. In: eCAADe Conference, 22, 2004, Copenhagen. 610-616. Disponível em http://cumincad.scix.net/cgi-bin/works/Show?2004_610. Acessado em: 12.2008.

ISIKDAG, U.; AOUAD, G.; UNDERWOOD, J. e WU, S. **Building information models: a review on storage and exchange mechanisms**. In: CIB W78 International Conference on Information Technology on Information Technology, 24, 2007, Maribor. 135-144. Disponível em http://itc.scix.net/data/works/att/w78-2007-020-068b-Isikdag.pdf. Acessado em: 12.2008.

KIM, I. e SEO, J. Development of IFC modeling extension for supporting drawing information exchange in the model-based construction environment. **Journal of Computing in Civil Engineering**, v. 22, n. 3, 2008, p.159-169. Disponível em http://cedb.asce.org/cgi/WWWdisplay.cgi?0803518. Acessado em: 12.2008.

KIVINIEMI, A. **Ten years of IFC development - why are we not yet there?** Espoo: VTT, 2006, 43 p.

LACROIX, M. e PIROTTE, A. **Data structures for CAD object description**. In: Annual ACM IEEE Design Automation Conference, 18, 1981, Nashville. New York: 1981 653-659. Disponível em http://portal.acm.org/citation.cfm?id=802371. Acessado em: 12.2008.

LIEBICH, T.; ADACHI, Y.; FORESTER, J.; HYVARINEN, J.; KARSTILA, K. e WIX, J. **Industry Foundation Classes IFC2x Edition 3**. BSI, 2006.

LIEBICH, T. e WIX, J. **IFC Technical Guide**. BSI, 2000, 46 p.

LKSOFTWARE. **JSDAI Express API** (página da internet). 2008. http://www.jsdai.net/, acessado em 12.2008.

NIBS. **National Building Information Modeling Standard**. National Institute of Building Sciences, 2007, 183 p.

NISBET, N. e LIEBICH, T. **ifcXML Implementation Guide**. 2. ed. Building Smart International - Modeling Support Group, 2007, 50 p.

NOUR, M. M. **A Flexible Model for Incorporating Construction Product Data into Building Information Models**. Weimar, 2006, 186 p. Tese (Doutorado). Construction Informatics, Bauhaus. Disponível em http://e-pub.uni-weimar.de/volltexte/2006/778/. Acessado em: 12.2008.

NOUR, M. M. e BEUCKE, K. An Open Platform for Processing IFC Model Versions. **Tsinghua Science and Technology**, v. 13, n. S1, 2008, p.126-131. Disponível em http://qhxb.lib.tsinghua.edu.cn/myweb/english/2008/2008es1/126-131.pdf. Acessado em: 12.2008.

PAZLAR, T. e TURK, Z. Interoperability in practice: geometric data exchange using the IFC standard. **International Journal of Production Research**, v. 13, 2008, p.362-380. Disponível em http://www.itcon.org/data/works/att/2008_24.content.00881.pdf. Acessado em: 12.2008.

PELS, H. J. Product and process data modelling. **Computers in Industry**, v. 31, n. 3, 1996, p.191-194. Disponível em http://www.sciencedirect.com/science/journal/01663615. Acessado em: 12.2008.

SAXPROJECT. **SAX - Simple API for XML** (página da internet). 2008. http://www.saxproject.org/, acessado em 12.2008.

SCHENCK, D. A. e WILSON, P. R. **Information Modeling: The EXPRESS Way**. New York: Oxford University Press, 1994, 388 p.

SCHEVERS, H. e DROGEMULLER, R. **Converting the Industry Foundation Classes to the Web Ontology Language**. In: International Conference on Semantics, Knowledge and Grid, 1, 2006, Beijing. 2005 73-75. Disponível em http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F11136%2F35651%2F01691494.pdf&authDecision=-203. Acessado em: 12.2008.

SCRA-STEP. **STEP Application Handbook - ISO 10303 - Version 3**. North Charleston: SCRA, 2006, 175 p.

TREECK, C. V.; ROMBERG, R. e RANK, E. **Simulation based on the product model standard IFC**. In: International IBPSA Conference, 2003, Eindhoven. 1293-1300. Disponível em http://www.inive.org/members_area/medias/pdf/Inive%5CIBPSA%5CUFSC75.pdf. Acessado em: 12.2008.

W3C. **DOM - Document Object Model** (página da internet). 2008a. http://www.w3.org/DOM/, acessado em 12.2008.

W3C. **Extensible Markup Language - XML - 1.0, Fifth Edition** (página da internet). 2008b. http://www.w3.org/TR/REC-xml/, acessado em 12.2008.

W3C. **XML Schema** (página da internet). 2008c. http://www.w3.org/XML/Schema, acessado em 12.2008.

YANG, W. Z.; XIE, S. Q.; AI, Q. S. e ZHOU, Z. D. Recent development on product modelling: a review. **International Journal of Production Research**, v. 46, n. 1, 2008, p.6055-6085. Disponível em http://www.ingentaconnect.com/content/tandf/tprs/2008/00000046/00000021/art00008. Acessado em: 12.2008.