UMA NOVA ABORDAGEM PARA ENGENHARIA DE SISTEMAS DE GERAÇÃO DE TEXTOS EM LINGUAGEM NATURAL: APLICAÇÃO NA DETERMINAÇÃO DO CONTEÚDO

ALEXANDRE MELO, MARCO FONSECA, LEONARDO JUNIOR, HENDRIK MACEDO

Departamento de Computação – Universidade Federal de Sergipe, 49100-000, São Cristóvão/SE – Brazil
E-mails: asmelo10@hotmail.com, marcos_ufs@yahoo.com.br, leonardobsjr@yahoo.com.br, hendrik@ufs.br

Abstract— Natural Language Generation (NLG) systems are usually built with the clearly intent of automating the process of linguistically correct texts from a data source. Traditionally, generators of such kind have been built using ad-hoc software engineering practices with no explicit development process, no standard software architecture and special-purpose languages. This work proposes a new development approach that leverages the most advanced languages and standards in modern software engineering to enhance the practical use of NLG applications. This innovative approach is applied to the NLG content determination layer and generation's prototype is presented to show the practicability of the proposal.

Resumo— Sistemas de geração linguagem natural (GLN) são usualmente construídos com o objetivo de automatizar o processo de geração de textos lingüisticamente corretos a partir de uma fonte de dados brutos, modelados de forma mais abstrata. Tradicionalmente, geradores desse tipo são construídos através de práticas ad-hoc de engenharia de software sem processo de desenvolvimento explícito, sem arquitetura de software padrão e com o uso de linguagens específicas ao domínio. Esse trabalho propõe uma nova abordagem de desenvolvimento que considera linguagens e padrões da engenharia de software moderna para impulsionar o uso prático de sistemas desse tipo. Essa abordagem inovativa é aplicada à etapa de determinação do conteúdo de GLN e um protótipo de geração é apresentado mostrando a plausibilidade da abordagem.

1 Introdução

A Geração de Linguagem Natural [Reiter and Dale 2000] (GLN) é uma tecnologia consolidada em termos conceituais. Ao longo da história, pesquisas na área esclareceram muitas questões fundamentais e produziram soluções que são suficientemente robustas e escaláveis para uso em aplicações práticas. O advento da Web como fonte quase inesgotável de conteúdo é claramente um grande impulsionador para o desenvolvimento de um número cada vez maior de sistemas que façam uso da tecnologia.

Infelizmente, essa não é a realidade. As técnicas de geração continuam virtualmente desconhecidas e/ou ignoradas no meio profissional. Diversas aplicações correntes que poderiam obter benefícios claros caso incorporassem a tecnologia, não fazem uso da mesma.

Uma das razões que levaram a essa situação é o fato de sistemas de GLN serem historicamente desenvolvidos utilizando práticas *ad-hoc* de engenharia de software e sem processo de desenvolvimento explícito, nem arquitetura de software padrão. O uso de linguagens e ferramentas de modelagem, implementação e representação de dados não usuais e de propósitos específicos é outro grande problema. O melhor exemplo disso é a atividade de realização lingüística, que é o aspecto de GLN que sempre recebeu mais atenção. Diversos componentes de realização lingüística foram construídos baseados em diferentes formalismos gramaticais e teorias usados para descrever GLN [Elhadad 1990].

Esse trabalho propõe uma abordagem inovadora para desenvolvimento de sistemas de GLN. A proposta considera o *pipeline* de atividades característico de um processo de geração de textos em linguagem natural como um conjunto de bases de regras de transformação de modelos. Essa metodologia de geração de aplicações baseadas em transformações sucessivas de modelos em níveis de abstrações diferentes foi recentemente popularizado com o advento do moderno paradigma de engenharia de software, MDA (Model Driven Architecture) [OMG 2001]. A abordagem proposta é aplicada à primeira atividade de um processo típico de geração, a determinação do conteúdo a ser gerado.

A proposta é validada através da construção de gerador de relatórios de jogos de campeonatos de futebol. Os dados são obtidos através de um simulador implementado que simula resultados de partidas do campeonato brasileiro de futebol. A partir das instâncias do modelo de domínio definido geradas pelo simulador em formato apropriado, a atividade de determinação do conteúdo a ser incluído no texto final é executada.

O artigo está organizado está organizado da seguinte maneira: a seção 2 faz uma breve descrição das atividades gerais normalmente envolvidas no processo de GLN, a seção 3 apresenta a nova abordagem proposta com ênfase na atividade de determinação do conteúdo, o estudo de caso escolhido e resultados obtidos são apresentados na seção 4, e a seção 5 faz algumas considerações conclusivas a respeito do trabalho.

2 Geração de Textos em Linguagem Natural

Sistemas baseados em GLN produzem textos em linguagem natural a partir de dados brutos que representam informação sobre um determinado domínio, e que podem estar organizados em bases de dados convencionais ou bases de conhecimento, ou ainda produzidos como resultado do processamento de alguma aplicação.

O processo de geração de linguagem natural é tradicionalmente visto como um processo de comu-

nicação orientado à objetivo. Dessa forma, a produção do texto final, seja ele escrito ou falado, uma simples cláusula ou um documento mais elaborado composto de vários parágrafos com várias sentenças, será sempre uma tentativa de satisfazer um determinado *objetivo comunicativo*. Baseado nesse objetivo comunicativo, o gerador decide que informação da fonte de dados em questão deve ser comunicada no texto final gerado. Durante o processo de geração, esse objetivo geral é refinado em objetivos menores e mais específicos e algum tipo de planejamento é utilizado para convertê-los progressivamente juntamente com os dados em um texto final bem formado e linguisticamente correto.

Todo esse processo de geração é normalmente dividido em três tipos específicos de atividades (camadas): (1) *determinação do conteúdo*, (2) *organização do conteúdo*, e (3) *realização lingüística*.

Determinar conteúdo em GLN é o processo de decidir os pedaços de conteúdo, organizados a partir da fonte de dados de entrada, que irão compor o texto final. Cada pedaço de conteúdo representa uma unidade atômica de informação que não pode ser decomposta. Essas unidades de conteúdo, por sua vez, são usualmente agrupadas em uma unidade semântica mais complexa para um determinado domínio de aplicação. Essas unidades semânticas são chamadas de *mensagens.* Considerando um sistema responsável por gerar relatórios sobre rodadas de um torneio de futebol, as sentenças "O Tabajara FC ganhou do Barcelona FC no último domingo" e "Vitória do Tabajara FC sobre o Barcelona no último domingo" representam diferentes construções lingüísticas para um mesmo tipo de mensagem: "Vitória", por exemplo.

A organização do conteúdo trata de agrupar as mensagens selecionadas no processo anterior em unidades a cada nível da hierarquia lingüística: a unidade de comunicação, o parágrafo, a sentença, e a frase, por fim. Também é responsável por definir a ordem de apresentação de cada um dos agrupamentos formados dentro de um determinado nível e escolher dependências de coordenação ou subordinação entre esses agrupamentos.

Finalmente, a realização lingüística envolve a escolha do termo (palavra) e da construção sintática para expressar cada unidade de conteúdo. Essa escolha é restringida pelas regras léxicas e gramaticais da língua adotada. Símbolos de pontuação também são decididos nessa etapa.

3 Uma Abordagem baseada em Transformações de Modelos

Apenas recentemente, o campo de pesquisa em GLN parece ter começado a atentar para as potencialidades de uma eventual padronização. Um primeiro passo nessa direção foi dado com a tentativa de se fazer GLN através de transformações de árvores XML [Wilcock, 2001]. A abordagem mais sistemática e cuidadosa de se padronizar arquitetura de sistemas NLG é o projeto RAGS [Cahill et al, 2000]. O modelo RAGS propõe uma coleção de tarefas para o sistema geração baseada em módulos definidos classicamente pelo pipeline consensual proposto por [Reiter and Dale, 2000] apesar de não levar em consideração uma especificação dos problemas linguísticos que surgem quando se tenta criar algum texto.

Nessa seção, é apresentada uma abordagem inovativa para a modelagem e desenvolvimento de sistemas de geração de textos em linguagem natural que utiliza linguagens e ferramentas usuais de desenvolvimento de software padrão. A abordagem utiliza o conceito de *transformação de modelos*, popularizado com o advento de um novo paradigma de engenharia de software e arquitetura de software ao mesmo tempo, denominado MDA.

Os princípios básicos dessa nova proposta são: (1) considerar as entradas e saídas de cada atividade do processo de geração como instanciações de modelos de dados previamente especificados em alguma linguagem de modelagem escolhida; (2) distinguir os tipos dos modelos de dados especificados para cada camada de transformação como modelos de dados do domínio, modelos de mensagens do domínio, modelos das estruturas de organização de dados, e modelos específicos de formatação de dados; (3) automatizar os passos do processo de engenharia de software aplicando regras de transformação genéricas entre esses diversos tipos de modelos.

As linguagens e/ou ferramentas de modelagem e implementação que constituem o núcleo da infra-estrutura da MDA, tais como UML, MOF, XMI, OCL e QVT [Eriksson], são de fato conceitualmente muito próximas das linguagens de propósito específicos tradicionalmente utilizadas em GLN, com a grande vantagem de serem padrões já bem estabelecidos para desenvolvimento de software e que são completamente independentes de plataforma e com amplo suporte do meio acadêmico e industrial.

OCL (Object Constraint Language) é uma linguagem textual e semi-formal utilizada para especificar restrições entre diversos elementos de um modelo UML. OCL possui uma sintaxe orientada a objetos intuitiva adequada para teoria dos conjuntos, lógica de predicados e construções algorítmicas. MOF (Meta-Object Facility) é uma pequena variante dos diagramas de classes UML utilizada para definir meta-modelos de dados no contexto da MDA. O XMI (XML Meta-data Interchange) provê um meio padronizado de se serializar um modelo UML ou meta-modelo MOF como um documento XML, definindo para isso uma *tag* XML específica para cada construção de um modelo UML ou MOF. QVT (Query-View-Transformations) é uma linguagem padronizada da OMG para especificação de transformações de modelos que permite se especificar uma transformação como um mapeamento entre elementos de um meta-modelo de entrada e um meta-modelo de saída, ambos definidos em MOF. QVT reutiliza OCL como uma sub-linguagem para identificar os elementos

desses metamodelos que são mapeados por essas transformações.

Dessa forma, a abordagem proposta nesse artigo considera o uso da linguagem UML/MOF para se especificar os modelos e meta-modelos de dados da arquitetura do gerador e uma linguagem padrão QVT para se especificar regras de transformação entre meta-modelos que definem modelos de dados entre camadas de atividade sucessivas do pipeline de geração (figura 1).

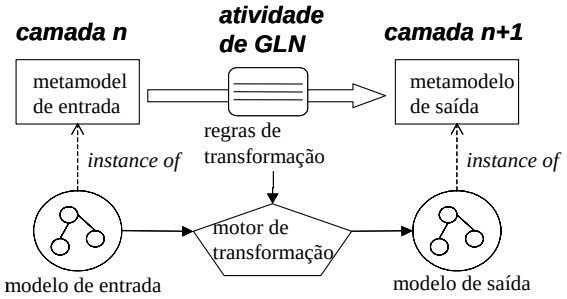

Fig. 1. Transformação de modelos entre camadas de um pipeline de GLN.

O propósito particular desse artigo é mostrar a plausibilidade dessa nova proposta, aplicando a abordagem na atividade de *determinação de conteúdo* de um processo de GLN.

3.1 Determinação do Conteúdo

A atividade de determinação de conteúdo consiste de uma coleção de regras que devem ser aplicadas aos dados de entrada desse sistema para produzir um conjunto de mensagens apropriadas. Essas regras devem modelar conhecimento especialista do domínio do sistema em questão, especificando condições e circunstâncias sob as quais determinados grupos de mensagens devem ser considerados ou não. O objetivo comunicativo é uma dessas circunstâncias. Considerando mais uma vez o exemplo de um sistema que gera relatórios sobre torneios de futebol, os tipos de mensagens produzidas sob o objetivo comunicativo de relatar os vitoriosos da rodada anterior são diferentes dos das mensagens que deveriam ser produzidas caso o objetivo comunicativo fosse o de relatar quem subiu e quem desceu na tabela de classificação do torneio, por exemplo.

Essas regras podem ser especificadas com QVT, consistindo de fato num mapeamento entre o metamodelo MOF de entrada da atividade e o metamodelo MOF de saída. O meta-modelo de entrada é o *modelo de dados do domínio* da aplicação enquanto que o meta-modelo de saída da atividade é o *modelo de mensagens* do domínio da aplicação.

3.2 Modelagem do Domínio

O modelo de dados do domínio representa o "mundo" do sistema em voga. Em outras palavras, ele descreve as entidades existentes, suas propriedades, relações entre essas entidades e conceitos que servem para definir classes de entidades que compartilham características comuns (ontologia).

A abordagem aqui proposta sugere o uso da linguagem UML/MOF para especificar a ontologia do sistema de geração. Trabalhos recentes consideram a utilização de UML como uma linguagem para modelagem de ontologia [Cranefield and Purvis 1999]. De fato, uma análise mais profunda mostra que modelos UML possuem características tradicionalmente associadas ao paradigma de representação declarativa do conhecimento tais como a natureza abstrata da linguagem de modelagem que não se amarra a nenhum tipo de aplicação em particular, e a facilidade com que um modelo UML pode ser modificado sem afetar outras características no modelo.

3.3 Definição de Mensagens

Quando textos sobre um determinado domínio são gerados, é desejável que se expresse a informação sob a forma de organizações particulares de elementos presentes na ontologia. Mensagem é o nome dado a cada organização desse tipo. A mensagem "Vitória" exemplificada anteriormente poderia ser modelada como uma relação entre duas entidades da classe "time" e uma entidade da classe "jogo" que por sua vez possui como propriedades o número de gols de cada uma das entidades time integrantes daquele jogo. Dessa forma seria possível expressar em forma de texto natural uma sentença indicando o vencedor da partida em questão.

A definição dos tipos de mensagens que devem compor esse segundo meta-modelo não é um processo trivial. A abordagem proposta sugere que os tipos de mensagens sejam definidos tomando-se como base um corpus de textos naturais finais. A metodologia para definição de mensagens consiste então em: (1) identificar sentenças individuais em cada um dos textos do corpus considerado, (2) identificar frases constituintes dessas sentenças que correspondem a mensagens individuais, tomando o cuidado de distinguir variações de cunho meramente lingüístico de variações de caráter semântico, (3) agrupar as mensagens identificadas em classes de mensagens e, finalmente, (4) modelar as classes de mensagens como meta-modelos MOF.

Uma vez especificados os meta-modelos MOF para a ontologia (representação da entrada da atividade de determinação de conteúdo) e para as mensagens (representação da saída da atividade de determinação do conteúdo), as regras de transformação em QVT que realizam o mapeamento entre os dois meta-modelos, e os dados provenientes da fonte de entrada como instanciação do meta-modelo MOF da ontologia, o motor de transformação fica encarregado de aplicar as regras especificadas e criar como resultado uma instanciação coerente do meta-modelo MOF das mensagens.

4. Estudo de Caso: Campeonatos de Futebol

O estudo de caso escolhido para mostrar a aplicação da abordagem de determinação de conteúdo apresentada anteriormente, foi um sistema para geração de relatórios sobre o jogos de campeonatos de futebol.

O domínio é composto por vinte equipes que se enfrentam em turno e returno. A equipe vitoriosa de um confronto conquista três pontos e a equipe derrotada não marca ponto algum; em caso de empate, ambas equipes conquistam um único ponto. O campeão é definido como aquela equipe que ao final de 38 rodadas conquistar o maior número de pontos dentre todas participantes. Todas as equipes jogam em todas as rodas e apenas uma vez por rodada.

4.1 Gerador de Relatórios

O gerador tem como objetivo gerar relatórios em linguagem natural sobre o andamento de edições do campeonato brasileiro. Uma aplicação real bastante interessante para um gerador desse tipo seria a geração automática de variações de textos para sítios Web de notícias sobre futebol. Um exemplo possível de texto gerado é mostrado na figura 2.

```
"Hoje ocorreram 4 jogos da 15ª Rodada do 2º
Turno do Campeonato Brasileiro. Às 16:00h
aconteceram os confrontos entre São Paulo(3)
e (2)Ponte Preta, Vasco(0) e (0)Figueirense,
São Caetano(2) e (1)Paraná, e às 18:00h o
confronto entre Grêmio(1) e (3)Fortaleza.
Com esses resultados o Santos continua líder
com 56 pontos. Subiram na tabela: Vasco(+1-
>10º), São Paulo(+2->3º), São Caetano(+1-
>11º) e Fortaleza(+1->8º). Desceram na
tabela: Paraná(-1->13º), Grêmio(-2->12º).
Permanecem na mesma situação:
Figueirense(7º) e Ponte Preta(5º)."
```

Fig. 2. Exemplo de relatório de jogos de campeonato de futebol.

4.2 Modelos e Meta-modelos

O domínio consiste de dez conceitos distintos para representar entidades respectivas. Esses conceitos são especificados em forma de diagrama de classes UML/MOF ilustrado na figura 3.

Para a definição do conjunto de mensagens a ser gerado, um conjunto de corpus de textos de relatórios sobre o domínio em questão foi construído. Cada corpus representa um objetivo comunicativo diferente. O texto ilustrado na figura 2 mais acima, por exemplo, é resultado do processo de geração que respeita dois objetivos comunicativos distintos: (1) relatar o resultado das partidas pertencentes à rodada mais recente do campeonato (parágrafo 1), e (2) relatar as mudanças de posição das equipes na tabela de classificação após a realização dessa rodada em questão (parágrafo 2).

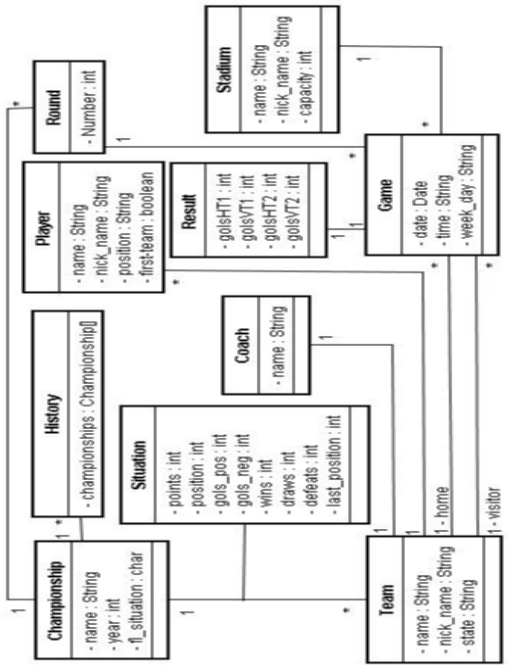

Fig. 3. Diagrama de classes UML/MOF do modelo de domínio.

A partir da análise dos textos desse corpus, de suas sentenças e, finalmente, das frases constituintes dessas sentenças, foram definidas as classes de mensagens a serem consideradas como meta-modelo de saída da atividade de determinação de conteúdo do gerador. Para satisfazer esses dois objetivos comunicativos em particular, foram definidas as seguintes classes de mensagens: (1) MsgGameResult: mensagens que informam o resultado de um jogo; (2) MsgGameTime: mensagens que informam o horário de um jogo; (3) MsgNumberGames: mensagem que informa o número de jogos em um determinado dia; (4) MsgPoint: mensagens que informam os pontos de um time; (5) MsgPositionChange: mensagem que informa quantas posições um time ganhou ou perdeu em relação a sua última posição; (6) MsgRound: informa a rodada do campeonato; (7) MsgTeamPosition: mensagem que informa a posição de um time em um campeonato.

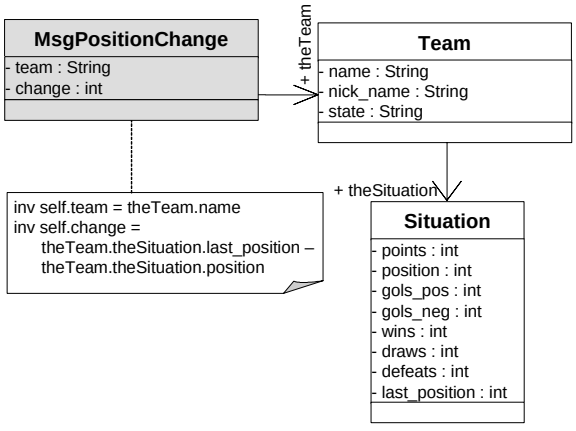

Fig. 4. Modelagem da classe MsgPositionChange.

A figura 4 ilustra a modelagem UML/MOF das mensagem MsgPositionChange. Nessa representação as classes de cor mais clara representam conceitos que

modelam dados do domínio. As restrições OCL especificadas nos diagrama definem formalmente a relação entre os conceitos das duas camadas.

4.3 Transformações de Modelos

Para a transformação do modelo de domínio para o de mensagens foi utilizada a linguagem ATL (ATLAS Transformation Language) [Bézivin et al. 2003]. ATL pode ser usada tanto para descrição de transformações sob o prisma de QVT quanto para implementação das mesmas. Uma regra de transformação ATL é declarativa e consiste de um conjunto de variáveis sintaticamente tipadas e restrições OCL agindo como filtros que acessam o modelo de entrada e o transformam de acordo com a especificação do meta-modelo de saída.

As regras ATL definidas mapeiam atributos das classes de domínio em atributos das classes de mensagens. Para cada classe de mensagem um conjunto de regras de transformação foi especificado. A figura 5 mostra o código ATL da regra de transformação para gerar mensagens do tipo MsgNumberGames.

A regra de transformação para gerar mensagens do tipo MsgNumberGames faz uso de um método declarativo auxiliar definido como *helper* em ATL. A função desse *helper* é retornar o número de jogos que ocorreram em uma determinada data passada como parâmetro. As mensagens MsgNumberGames possuem um único atributo "nGames" que armazenam o número de jogos naquela data.

```
helper def:countGames(dia:Integer,mes:Integer,year:Integer):
Integer=Domain!Game.allInstances()->asSequence()->iterate(game;
sum:Integer=0| if (game.day=dia and game.month=mes and
game.year=year) then sum+1 else sum endif );

rule MsgNumberGames {
  from a : Domain!Game
  to q : Message!MsgNumberGames
    (nGames<-thisModule.countGames(6,2,2007))
}
```

Fig. 5: Regra de transformação ATL para gerar a mensagem MsgNumberGames.

4.4 Implementação

4.4.1 Simulador de Campeonatos

O padrão de entrada para o gerador de relatórios é um conjunto de instâncias das classes que modelam o domínio da aplicação em formato XMI. Para testar a acuidade do gerador, foi implementado um simulador na linguagem Java responsável por gerar instâncias já serializadas nesse formato.

O simulador implementa as dez classes de domínio definidas no gerador e uma classe adicional, chamada Simulator, responsável por gerar instâncias das classes de domínio segundo as regras do campeonato brasileiro de futebol. A figura 6 ilustra o conjunto de instâncias XMI como resultado de um processo de simulação. A figura 7 mostra a representação visual dessas instâncias em um diagrama de objetos UML.

```
<Championship xmi:id="Campeonato2007" name="Campeonato
Brasileiro 2007" year="2005" fl_situation="r" rounds="Rodada10"
teams=" Vasco Flamengo"/>
<Round xmi:id="Rodada10" number="01" games =
"Vasco_X_Flamengo"/>
<Game xmi:id="Vasco_X_Flamengo" day="17" month="02" year="2007"
time="16:00" week_day="Domingo" teamH="Vasco" teamV="Flamengo"
stadium="Maracana" round="Rodada10" result="ResultadoVasFla"/>
<Result xmi:id="ResultadoVasFla" golsHT1="1" golsVT1="1"
golsHT2="2" golsVT2="0"/>
<Stadium xmi:id="Maracana" name="Jornalista Mario Filho"
nick_name="Maracana" capacity="90000"/>
<Team xmi:id="Flamengo" name="Flamengo" nick_name="Mengo"
state="Rio de Janeiro" championship="Campeonato2007"
situation="Situacao_Mengo"/>
<Team xmi:id="Vasco" name="Vasco" nick_name="Vascao" state="Rio
de Janeiro" championship="Campeonato2007" coach="RenatoGaucho"
players="Romario" />
<Coach xmi:id="RenatoGaucho" name="Renato Portaluppi"/>
<Situation xmi:id="SituacaoVasco" points="6" position="1" gols_pos="7"
gols_neg="2" wins="2" draws="0" defeats="0" last_position="2"/>
<Player xmi:id="Romario" name="Romario de Souza Faria"
nick_name="Romario" position="Atacante" first_team="true"/>
```

Fig. 6. Representação XMI dos objetos gerados pelo simulador.

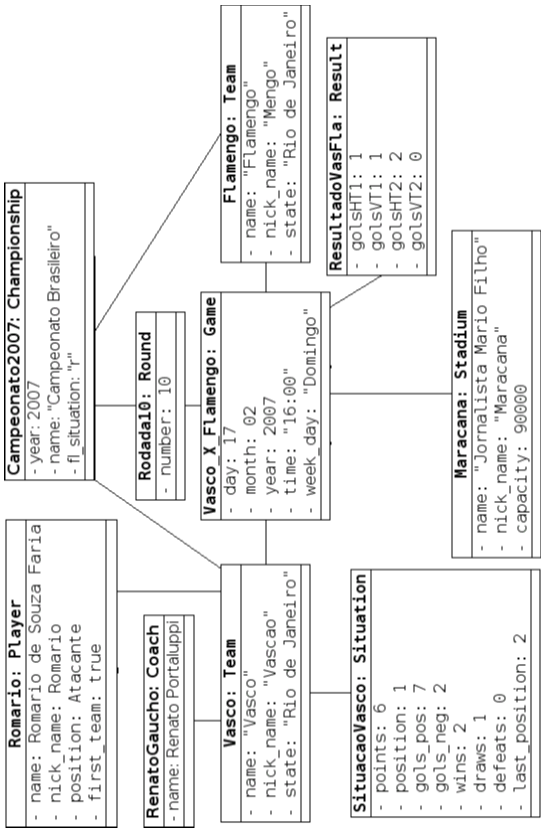

Fig. 7. Diagrama de objetos UML representativo das instâncias geradas.

4.4.2 Ambiente de simulação

As transformações ATL funcionam através de um plug-in (ADT) para o ambiente Eclipse de desenvolvimento. Um outro plug-in Eclipse (EMF) é responsável pela codificação dos modelos e meta-modelos no formato XMI 2.0. Os meta-modelos correspondem à modelagem do domínio e das mensagens, já os

modelos correspondem às instâncias do domínio e das mensagens. O ADT fornece uma notação textual mais simples (KM3) para especificação dos meta-modelos no EMF. A vantagem de se utilizar o KM3 é se evitar a sintaxe verborrágica inerente do formato XMI.. O plug-in ADT encarrega-se de gerar automaticamente as versões XMI correspondentes dos meta-modelos KM3 definidos.

Como descrito anteriormente, os modelos de dados (instâncias) são fornecidas pelo simulador já em formato XMI e servem de entrada para o gerador. O motor de transformação ATL (motor de inferência) embutido no plug-in ADT encarrega-se de aplicar as regras ATL codificadas entre os meta-modelos de entrada e saída, gerando como resultado as instâncias do meta-modelo de mensagens em XMI (figura 9).

```
<MsgPoint team="Flamengo" points="6"/>
<MsgTeamPosition team="Flamengo"
position="1"/>
<MsgGameResult teamH="Flamengo"
teamV="Vasco" scoreH="1" scoreV="3"/>
<MsgGameTime time="10:30" teamH="Flamengo"
teamV="Vasco"/>
<MsgPositionChange team="Flamengo"
change="1"/>
```

Figura 8. Instâncias XMI das classes de mensagens geradas pelo gerador.

5. Conclusão

Nesse artigo foi apresentada uma abordagem inovativa para o desenvolvimento de sistemas de Geração de Linguagem Natural. Seguindo a filosofia MDA de construção de software, é proposto que o processo de geração de textos em linguagem natural passe a ser baseado em transformações de modelos. As representações de dados de entrada e saída de atividades específicas do pipeline de geração são definidos como meta-modelos MOF e regras de transformação entre esses dois meta-modelos distintos são especificadas usando uma sintaxe declarativa. Um motor de aplicação de regras é responsável por transformar instâncias do meta-modelo de entrada em instâncias condizentes com a meta-modelagem de saída. Essa nova abordagem que reutiliza linguagens e ferramentas padrões para engenharia e implementação de softwares usuais aparece como uma alternativa real para solução de um problema que limita o uso da tecnologia de geração de linguagem natural nas soluções do dia a dia: a falta de metodologia e arquitetura padrão para construção de sistemas reais. Um gerador de relatórios sobre campeonatos de futebol foi apresentado como estudo de caso. Um simulador de resultados de jogos de campeonatos de futebol foi desenvolvido para criar a fonte de dados de entrada para o gerador. Os resultados das simulações realizadas mostraram que o protótipo desenvolvido em ambiente preparado realiza satisfatoriamente a atividade de determinação de conteúdo de sistemas GLN, deixando claro que a abordagem proposta é totalmente plausível. O protótipo está sendo estendido para cobrir as demais etapas do processo de geração de linguagem natural e um sistema de extração inteligente de informação da Web está sendo desenvolvido para que possamos realizar testes utilizando conteúdo de sites de notícias de futebol como fonte de dados para o gerador.

Referências Bibliográficas

Bézivin, J., Dupé, G., Jouault, F. and Rougui, J. (2003). First experiments with the ATL model transformation language: Transforming XSLT into XQuery, *Proceedings of the OOPSLA'03 Workshop on Generative Techniques in the Context of the MDA*.

Cahill, L., Doran, C., Evans, R., Kibble, R., Mellish, C., Paiva, D., Reape, M., Scott, D. and Tipper, N. (2000). Enabling Resource Sharing in Language Generation: an Abstract Reference Architecture, *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Cranefield, S. and Purvis, M. (1999). UML as an Ontology Modelling Language, *Proceedings of the Workshop on Intelligent Information Integration*, 16th International Joint Conference on Artificial Intelligence (IJCAI-99).

Elhadad, M. (1990). Types in Functional Unification Grammars, *Proceedings of the 28th. Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp: 157-164.

Eriksson, H.E., Penker, M. Lyons, B. and Fado, D. (2003). UML 2 Toolkit, John Wiley & Sons.

Object Management Group. (2001). Model Driven Architecture (MDA), OMG Document ormsc/2001-07-01 edition.

Reiter, E. and Dale, R. (2000). Building Applied Natural Language Generation Systems, Cambridge University Press.

Wilcock, G. (2001). Pipelines, Templates and Transformations: XML for Natural Language Generation, *Proceedings of the first NLP and XML Workshop*; Workshop session of the 6th Natural Language Processing Pacific Rim Symposium, Tokyo.