Avaliação de Modelos i* com o Processo AIRDoc-i*

Cleice Souza[1], Cláudia Souza[1], Fernanda Alencar[2], Jaelson Castro[1], Paulo Cavalcanti[1], Monique Soares[1], Gabriela Guedes[1], Eduardo Figueiredo[3]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
[2]Eletrônica e Sistemas – Universidade Federal de Pernambuco (UFPE)
[3]Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
{ctns, jbc, plc2, mcs4, ggs}@cin.ufpe.br; {ccnsouza, fernandalenc}@gmail.com; figueiredo@dcc.ufmg.br

Resumo. Engenharia de Requisitos é uma atividade chave no processo de desenvolvimento de software, pois ela é responsável pelo entendimento e definição do problema e das necessidades dos usuários. A abordagem GORE (*Goal-Oriented Requirements Enginering*) apresenta como foco capturar as reais intenções dos stakeholders através de técnicas como: V-Graph, NFR-framework, KAOS e a linguagem i*. A linguagem i* se caracteriza por modelar os objetivos e as intencionalidades dos atores de um contexto organizacional. Entretanto, os modelos produzidos em i* não estão livres de erros oriundos do mau uso dos construtores sintáticos de modelagem. Assim, faz-se necessário um processo de avaliação capaz de verificar a sintaxe da linguagem e propor correções em modelo i*. Neste sentido, este trabalho apresenta um processo proposto chamado de AIRDoc-i*. Neste processo, foram acrescidas atividades sequenciais para avaliar modelos i*.

Palavras-chave: Engenharia de Requisitos, Linguagem i*, processo AIRDoc.

1 INTRODUÇÃO

A Engenharia de Requisitos (ER) corresponde à atividade de entendimento das necessidades dos usuários no contexto do problema a ser resolvido [1]. Problemas da ER é amplamente pesquisados no Brasil [16] e no mundo. A ER pode representar as necessidades do usuário através de metas e intenções. Esta forma é usada na abordagem *Goal Oriented Requirements Enginering (GORE)* [2], que tem como foco capturar as reais intenções dos *stakeholders* por meio de metas que eles pretendem satisfazer.

Entre as várias técnicas GORE, podemos citar as linguagens V-Graph [3], NFR-framework [4], KAOS [5] e a linguagem i* [6]. Esta última é usada neste trabalho por ser uma linguagem bastante difundida na comunidade de ER [7, 13, 14]. A linguagem i* se caracteriza por modelar objetivos e as intencionalidades dos atores e suas dependências dentro de um contexto organizacional, permitindo descrever os relacionamentos estratégicos e intencionais em alto nível de detalhamento.

É importante ressaltar que modelos i* descreve detalhadamente os participantes do processo e o contexto que está sendo modelado o que facilita o seu entendimento. Santos [8], por exemplo, ressalta a importância dos modelos i* apresentarem descrições detalhadas do problema e interações modelados. Portanto, devem existir métodos avaliativos que verifiquem se essas descrições seguem o uso correto da linguagem i*. Não basta aos modelos i* estarem bem detalhados, eles também precisam estar corretos com relação a sintaxe da linguagem [6]. Em um trabalho anterior [17], nós apresentamos um processo chamado de ***A**pproach to **I**mprove **R**equirements **Doc**uments for **i*** models* (AIRDoc-i*) para avaliar a sintaxe de modelos i*. Este novo processo surgiu através da análise de atividades e de artefatos do processo AIRDoc [9] que avalia a sintaxe de modelos de casos de uso com objetivo de identificar os erros e quantificá-los.

Este artigo estende o processo AIRDoc-i* através da análise das atividades e dos artefatos do processo AIRDoc [9] foi montado um processo gráfico em BPMN [10] que detecta erros sintáticos da linguagem i* em seus modelos. Além disso, o novo processo avalia os modelos i*, identifica, armazena, e quantifica os erros e os corrige através de atividades sequenciadas do processo AIRDoc-i*.

O restante deste artigo esta organizado da seguinte forma. A seção 2 apresenta o objetivo da pesquisa. A seção 3 apresenta as principais contribuições do trabalho. A seção 4 as conclusões do trabalho. A seção 5 discute trabalhos futuros e em andamento.

2 OBJETIVOS DA PESQUISA

O principal objetivo da pesquisa foi responder a pergunta: *Como o processo AIRDoc-i* detecta e corrige erros sintáticos da linguagem i* ?*.

Para responder a pergunta da pesquisa foi realizada uma análise das atividades do processo AIRDoc-i* [17] aplicando-o a modelos i*. Para isso, foi necessário adequar as atividades do processo, pois a linguagem i* apresenta características próprias [6] que a torna diferente da linguagem de modelos de casos de uso UML [11]. O nosso objetivo foi alcançado através de melhorias do processo AIRDoc [9] que agora adaptado é um processo que avalia a sintaxe dos modelos i*. Este processo tem o intuito de avaliar e melhorar os construtores da linguagem i* em seus modelos.

3 CONTRIBUIÇÕES CIENTÍFICAS

Esta seção mostra a contribuição do trabalho, o processo AIRDoc-i* cujo objetivo principal é detectar erros sintáticos e corrigi-los. Temos o processo, mostrado na Fig. 1, modelado em BPMN (*Business Process Modeling Notation*) [10]. BPMN foi escolhida por utilizar ícones padrão que facilita a modelagem do processo AIRDoc-i*. Este processo é composto por fases, atividades e artefatos para avaliar os modelos i*, detectar os erros, armazenar os erros, quantificá-los e corrigi-los através de atividades sequenciadas. O processo AIRDoc-i* é explicado nas seguintes seções

3.1 AIRDoc-i*

O processo AIRDoc-i* está dividido em dois estágios que são Avaliação e Melhoria. O estágio Avaliação consiste em quatro Atividades: (E.1) Elaboração do plano de atividade; (E.2) Definir as Atividades do GQM; (E.3) Coletar os Valores da Métrica; e (E.4) Interpretar as Atividades do GQM. Além disso, o estágio de Melhoria consiste em duas Atividades: (I.1) Identificar as Correções e (I.2) Aplicar as Correções.

Primeiro Estágio: Avaliação

Este estágio apresenta como foco principal identificar, apontar e contar os erros sintáticos encontrados nos modelos i*.

E.1 Elaboração do plano de atividade. Nesta atividade, são definidas as pessoas para trabalhar no processo, as ferramentas, os modelos i*, os cargos, as datas, os prazos, as equipes, e o tempo para compor o projeto. As informações produzidas nesta atividade serão armazenadas nos artefatos 1, 3, 4, 5, 6 e 7.

E.2 Definir as atividades do GQM. Nesta atividade, são definidos o objetivo da avaliação, a pergunta da avaliação, a métrica, e a estrutura da avaliação. As informações produzidas nesta atividade são armazenadas nos artefatos 2, 8, 9, e 10.

E.3 Coletar os valores da métrica. Antes de realizar a coleta dos valores da métrica deve ser realizada a identificação dos erros sintáticos nos modelos i*, que acontece da seguinte forma:

1º passo: Realizar uma inspeção no modelo i* que será avaliado;

2º passo: Realizar uma inspeção no catálogo de erros do i*;

3º passo: Comparar a sintaxe do modelo i* com a sintaxe do catálogo.

Ao realizar estes passos, é possível identificar se o modelo apresenta erros sintáticos e apontá-los. Caso apresente erros sintáticos a métrica a ser aplicada nos modelos i* chama-se número de erros (E_1) retirada do trabalho de [8]. A métrica número de erros (E_1) está inserida no AIRDoc-i*. O valor da métrica (E_1) é variável de acordo com erros sintáticos encontrados. Após identificar os erros sintáticos o resultado pode ser coletado e armazenado no artefato 11.

E.4 Interpretar as Atividades do GQM. Nesta atividade será interpretado o resultado da métrica utilizando o objetivo e as perguntas da atividade E.2. As informações produzidas nesta atividade serão armazenadas nos artefatos 12 e 13.

Segundo Estágio: Melhoria

O segundo estágio é composto por duas atividades que são: Identificar as Correções e Aplicar as Correções. Este estágio apresenta como foco corrigir os erros sintáticos encontrados nos modelos i*. As correções serão realizadas através da consulta do catálogo de erros frequentes em i*.

I.1 Identificar as Correções. Nesta atividade ocorre a identificação da correção. A identificação acontece através da inspeção no catálogo de erros frequentes em i*, artefato 14, que exibe a forma certa de utilizar os construtores sintáticos linguagem. O artefato 13 armazena os erros sintáticos encontrados no modelo i*.

I.2 Aplicar as Correções. Nesta atividade são aplicadas as correções dos erros sintáticos identificados nos modelos i* localizados no Catálogo de Erros Frequentes

em i*. O artefato 15 inclui todos os modelos i* corrigidos. As correções são aplicadas de acordo com erro. Por exemplo, o modelador identifica o erro, consulta a localização do erro no catálogo e, posteriormente, observa no catálogo a forma correta de realizar a modelagem e aplica a correção no erro encontrado.

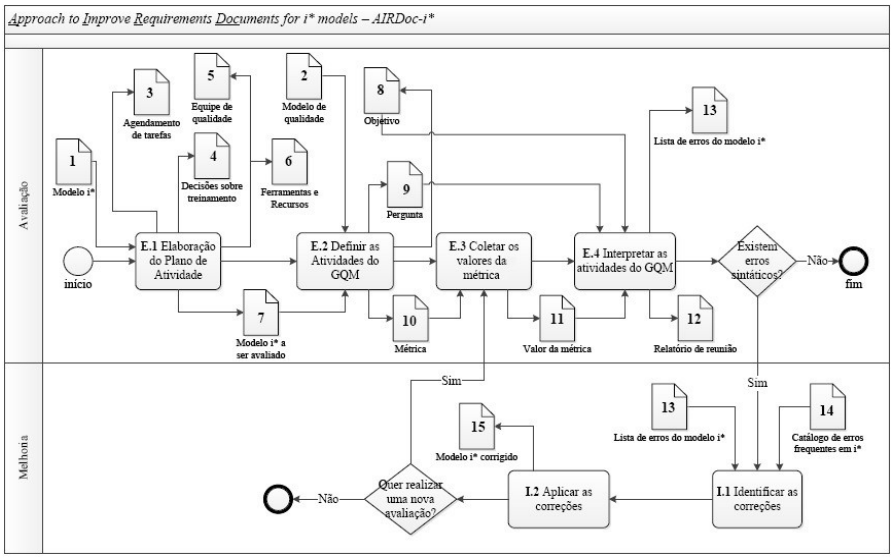


Fig. 1 O processo AIRDoc-i*

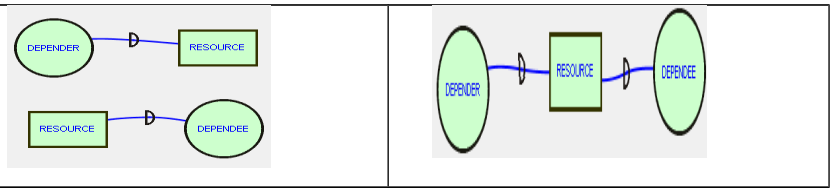
3.2 Catálogo de erros frenquentes da linguagem i*

Modelos i* não estão livres de descrições sintáticas erradas que prejudicam o entendimento por parte dos interessados. Para facilitar a identificação dos erros que podem aparecer nos modelos i* o catálogo proposto anteriormente [8] foi incrementado com novos erros sintáticos. Este novo catálogo é chamado de *Catálogo de Erros Frequentes da Linguagem i**. Os objetivos deste novo catálogo são:

- Ser inserido dentro do processo avaliativo de modelos i* chamado de AIRDoc-i*, para corrigir os erros sintáticos encontrados nos modelos i*; e
- Ser usado pela pessoa interessada em aumentar seus conhecimentos sobre erros sintáticos que os modelos i* podem apresentar.

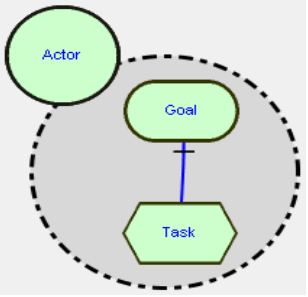
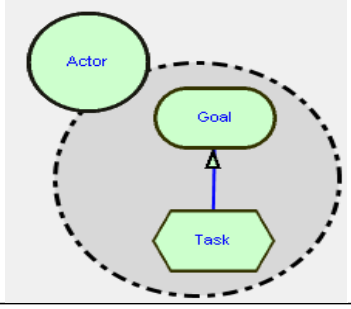
Tabela 1. Ligação de dependência.

Ligação de Dependência 001	Ligação de dependência só deve acontecer se houver *depender*, *dependum* e *dependee*.
Ligação de dependência sem *dependum*.	O *dependum* só deve ter uma única ligação vinda do *depender* e uma ou mais ligações indo para *dependee diferentes*.
Errado	Certo



A seguir estão alguns dos novos erros inseridos dentro do catálogo de erros frequentes em i*. A Tabela 1 apresenta o erro de Ligação de Dependência, no qual a ligação de dependência só deve acontecer se houver *depender*, *dependum* e *dependee*. Além disso, o *dependum* só deve ter uma única ligação vinda do *depender* e uma ou mais ligações indo para *dependee diferentes*. Ademais, a Tabela 2 apresenta o erro de *Goal Refinement*, no qual todo *goal* só deve ser o fim (*end*) num relacionamento do tipo *means-end link*.

Tabela 2. Refinamento: Goal.

Goal Refinement 001	*Goal element* deve ser o fim (*end*) em uma ligação do tipo *means-end*
Goal sendo refinado através de uma *task-decomposition*.	O elemento interno *goal* deve ser satisfeito através de uma ligação de *means-end link*.
Errado	Certo
	

4 CONCLUSÕES

Este artigo apresenta como contribuição um processo chamado AIRDoc-i* [17] que identifica e corrige os erros sintáticos encontrados em modelos i*. Este processo possui quatro atividades que são Elaboração do Plano de Atividade, Definir as atividades do GQM, Coletar os valores das métricas e Interpretar as atividades do GQM. Estas atividades têm como objetivo definir a equipe de avaliação, definir a pergunta, a métrica, apontar os erros sintáticos nos modelos i* e coletar o resultado da métrica. A fase de Melhoria possui duas atividades que são Identificar as Correções e Aplicar as Correções. A identificação das correções acontece quando o *stakeholder* recorre ao catálogo de erros frequentes em i* para encontrar a forma correta de realizar aquela modelagem na qual o erro identificado. A aplicação da correção do modelo i* acontece na segunda atividade desta fase produzindo um modelo i* melhorado.

5 TRABALHOS FUTUROS EM ANDAMENTO

Como trabalhos futuros, nós pretendemos fazer: (i) criar extensões do AIRDoc-i* para avaliar a completude, a ambiguidade e o detalhamento dos modelos i*, além de detectar erros semânticos; (ii) aumentar o catálogo de erros frequentes da linguagem i*, com o propósito de incluir novos erros sintáticos e adicionar erros semânticos que ocorram com frequência em modelos i*; (iii) especificar em OCL (*Object Constraint Language*) [12] os erros do Catálogo de Erros Frequentes da Linguagem i*, cuja importância é definir condições e/ou restrições para os construtores da linguagem i*, e (iv) aplicar o processo em aplicações modeladas em i* como linha de produtos de software [18] e sistemas multi-agentes [15].

REFÊNCIAS

1. Asghar, S., Umar, M.: Requirement Engineering Challenges in Development of Software Applications and Selection of Customer-off-the-Shelf (COTS) Components. International Journal of Software Engineering (IJSE), Volume (1): Issue (1), 2010.
2. Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In: 5th IEEE International Symposium on Requirements Engineering (p. p. 249). Washington, DC, 2001.
3. Yu, Y., Leite, J. C., Mylopoulos, J.: From goals to aspects: Discovering aspects from requirements goal models. 12th Intl. Conf. on Requirements Engineering. 2004.
4. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. 1. ed. [S.l.]: Springer, 1999.
5. Lamsweerde, A. V.: Requirements Engineering in the Year 00: A Research Perspective. In: Keynote paper, Proc. ICSE'2000 - 22nd Intl. Conference on Software Engineering. IEEE Computer Society Press, 2000.
6. Yu, E.: Modelling Strategic Relationships for Process Reengineering. Tese (Doutorado). Canadá: University of Toronto, Department of Computer Science, 1995.
7. Horkoff, J. M. : Using i* Models for Improvement. Dissertação (Mestrado). Toronto, Canadá: Department of Computer Sciences, 2006.
8. Santos, E. B. D.: Uma Proposta de Métricas para Avaliar Modelos i*. Dissertação (Mestrado). Recife: UFPE, 2008.
9. Ramos, R. A.: AIRDoc – An Approach to Improve the Quality of Requirements Documents: Dealing with Use Case Models. Tese (Doutorado). Recife: UFPE, 2009.
10. BPMI. Business Process Modeling Notation, OMG Avaible Specification. Object Management Group, 1.1 edition. 2008.
11. UML. Unified Modeling Language. 2009. Disponível em: http://www.uml.org/. Acesso em: 20 Setembro de 2011.
12. OMG. Object Management Group. 2001. OCL 2.0. Acesso em 23 de dezembro de 2012, disponível em Object Constraint Language: OMG Available Specification: http://www.omg.org/spec/OCL/2.0/
13. J. M. Conejero, E. Figueiredo, A. Garcia, J. Hernandez, and E. Jurado. On the Relationship of Concern Metrics and Requirements Maintainability. In Information and Software Technology (IST), 54(2), pp. 212-238, 2012.
14. D. Demerval; M. Soares; F. Alencar; E. Santos; C. Souza. Towards an i*-based Architecture Derivation Approach. In: Fifth Internatiol i* Workshop, Trento, 2011.
15. A. Oliveira, L. Cysneiros, J. Leite, E. Figueiredo, and C. Lucena. Integrating Scenarios, i*, and AspectT in the Context of Multi-Agent Systems. In proceedings of

the 16th Conference of the Center for Advanced Studies on Collaborative research (CASCON), Article No. 16. Ontario, Canada, 2006.
16. K. Oliveira; J. Pimentel; E. Santos; D. Demerval; G. Souza; C. Souza; M. Soares; J. Castro; F. Alencar; C. Silva. 25 years of Requirements Engineering in Brazil: a systematic mapping to WER 2013. Workshop Engineering Requirements, 2013.
17. C. Souza,; F. Alencar ; G. Souza ; M. Soares ; C. Souza; R. Ramos ; J. Castro. AIRDoc-i*: um Processo para Avaliação de Modelos i*. In: 15th Workshop on Requirements Engineering, Buenos Aires, 2012.
18. G. Souza; C. Silva; J. Castro ; M. Soares; D. Demerval ; C. Souza. GS2SPL: Goals and Scenarios to Software Product Lines. In: 24th International Conference on Software Engineering & Knowledge Engineering (SEKE), 2012.