O Diagnóstico de Requisitos como Tópico de Inovação Tecnológica – A Ferramenta iStar Diagnoses

Antonio de Padua Albuquerque Oliveira, Marinilza Bruno de Carvalho, Amanda Garcez Melgaço, Jonathan Dias Lima de Souza

Universidade do Estado do Rio de Janeiro – UERJ
Rua São Francisco Xavier, 524 - 6 andar - Maracanã - Rio de Janeiro, Brazil

{padua, mbruno}@ime.uerj.br amanda.garcez@yahoo.com.br
jonathanrj@outlook.com

Abstract: *In general way, innovation can be considered a propellant of productivity. Innovation works to achieve successful results (more effectiveness) with less effort or less waste of time (more efficiency). Innovation does not distinguish whether the object that receives innovation is an activity, human or not, or even if it involves the improvement of a tool. "Innovating is making, thinking and being different." Although requirements engineering is always growing, we observe that the activity of analysis (V & V - verification and validation of requirements) is not graced with innovative technologies with the same frequency as other activities of the process of software development. In this work we present the software tool "Istar Diagnoses" that supports a technique for analysis of requirements in models of the i* Framework, more precisely: in the article we summarize the method that has the technique of diagnoses, we explain the technique and exemplify the use of the tool with diagnostics in SD and SR models of the i* Framework.*

Resumo: *De um modo geral, a inovação pode ser considerada um propulsor da produtividade. A inovação trabalha para alcançar resultados de sucesso (mais eficácia) com menor esforço ou menor desperdício de tempo (mais eficiência). A inovação não distingue se o objeto que recebe a*

inovação é uma atividade, humana ou não, ou mesmo se envolve o aprimoramento de uma ferramenta. "Inovar é Fazer, Pensar e Ser DIFERENTE". Apesar da Engenharia de Requisitos estar sempre em evolução, observamos que a atividade de análise (V&V - verificação e validação de requisitos) não é agraciada com tecnologias inovadoras com a mesma frequência que outras atividades do processo de desenvolvimento software. Neste trabalho apresentamos a ferramenta de software "iStar Diagnoses" que apoia uma técnica para análise de requisitos em modelos do Framework i, mais precisamente: no artigo resumimos o método que possui a técnica de diagnósticos, explicamos a técnica e exemplificamos o uso da ferramenta com diagnósticos em modelos SD e SR do Framework iStar.*

1. Introdução

O Instituto de Propriedade Industrial – INPI, segundo a Lei de Inovação de 2004, define o Conceito de Inovação. Cita que: Inovação Incremental é o melhoramento do produto e/ou processo já existente, e Inovação Radical significa que o produto e/ou processo é completamente novo, tendo como referência a "instituição" [17]. Portanto, se é novo para a instituição é uma inovação. Seguindo nesta linha, as últimas estatísticas, com base na PINTEC 2011 (Pesquisa Industrial sobre Inovação Tecnológica, do IBGE no Brasil) [16], as empresas que mais cresceram foram as que mais inovaram, e muitas o fizeram na gestão, nas novas ações e o novo pensar. Usar o conhecido para gerar o novo, flexibilizar as fronteiras e dentro de todos os padrões éticos e morais, fazer diferente [12].

A inovação na Engenharia de Requisitos é estimulada pelo relato de fracassos no desenvolvimento de sistemas e, assim, é frequentemente contemplada com processos inovadores. Porém, historicamente, alguns levantamentos chegam a conclusão de que a atividade de análise é pouco executada. O que se traduz como uma prática, posto que as atividades de planejamento e avaliação ainda são rotineiramente insipientes. Os engenheiros optam, perigosamente, por deixar para a fase de testes o encargo de identificar inconsistências entre os requisitos e o software. Acreditamos que esse modo de agir

acontece por três motivos não isolados entre si: primeiro, a atividade é mesmo penosa; segundo, não é motivador para o ser humano encontrar erros no seu próprio trabalho e, por último, o engenheiro não dispõe de técnicas e ferramentas de software que facilitem a tarefa e motivem investir em prevenção. Estas questões ficam muito visíveis quando não existe planejamento, nem a possibilidade de simulação, testes de novos conceitos e abertura para mudanças, com ferramentas de gestão de risco, custo e prazo [18]. Neste ponto, o erro que deveria ser apenas uma informação de aprimoramento, é visto como um fracasso e desestimula a equipe [13].

A ferramenta apresentada nesse trabalho apoia o processo de qualidade para análise (V&V - verificação e validação de requisitos) [11] de modelos do Framework i* (iStar). A ideia é investigar os modelos preparados para obter retorno positivo ou negativo (capacidades que devem ser implementadas ou características que devem ser evitadas) de maneira que a qualidade seja aprimorada para o atingimento do sucesso. A ideia aplicada pela técnica de diagnósticos e implementada pela ferramenta é semelhante a algumas técnicas de leitura que comportam métodos de inspeção.

O trabalho está organizado da seguinte forma: na seção 2 descrevemos brevemente os artefatos do Método ERi*c que são aplicados ao Framework i*. Na seção 3 apresentamos a ferramenta de diagnóstico de SDsituations e de SRconstructs. Na seção 4 apresentamos a aplicação da ferramenta, utilizando o exemplo "Gerenciamento de Exercícios para Grupos de Alunos" e, finalmente, na seção 5, concluímos e salientamos os trabalhos futuros.

2. Os Artefatos do Método ERi*c

A Ferramenta iStar Diagnoses [1] foi idealizada para aplicação no Método ERi*c – Engenharia de Requisitos Intencional [9].

ERi*c, ou "Intentional RE", é principalmente uma "expansão preparatória" do Framework i* [2]. Ele também aplica ideias do Framework NFR [3]. O método possui um conjunto de procedimentos e técnicas uteis para guiar a construção de modelos i*. A Figura 1 [10] mostra as seis etapas do método ERi*c.

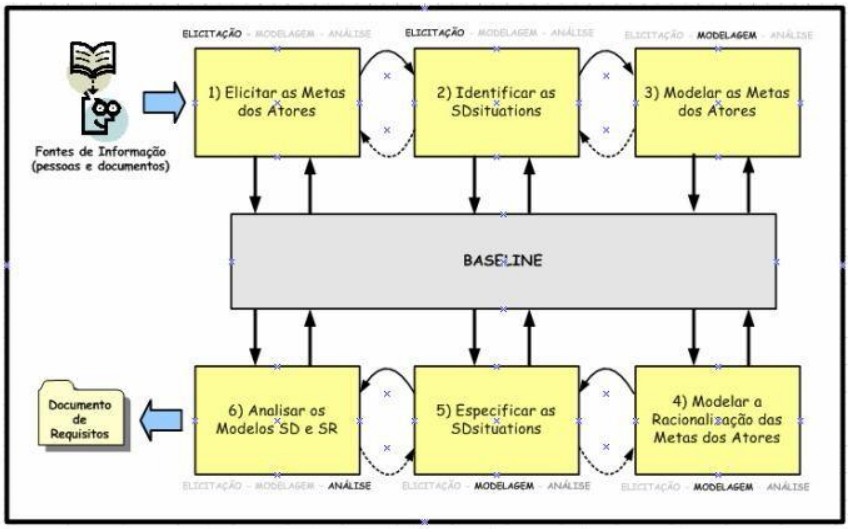

Figura 1 – Visão geral do método ERi*c ilustrando o encadeamento das etapas

O Método ERi*c aplica as atividades de ELICITAÇÃO, MODELAGEM e ANÁLISE como sendo as principais da Engenharia de Requisitos. A elicitação compreende o conhecimento do contexto que permeia o sistema desejado para a descoberta dos requisitos de software. A modelagem compreende a representação dos requisitos do software sob a forma de modelos ou diagramas. Análise compreende a verificação e validação de requisitos representados nos diagramas.

2.1 A ELICITAÇÃO: Metas e SDsituations

Apoiado na observação de Yu [2]: "A goal is a condition or state of affairs in the world that an actor would like to achieve", o método usou a ideia de que: "ações mudam estados e estados são metas" para capturar as metas a partir das ações executadas no contexto organizacional. Essa ideia para a elicitação das metas considera todos os tipos de ações identificadas pelo LAL – Léxico Ampliado da Linguagem [4]. Os detalhes do procedimento da elicitação estão descritos no trabalho de AGFL [6]. O LAL foi escolhido para a elicitação das ações porque ele tem a capacidade de identificar as ações que ocorrem no contexto organizacional sem viés de implementação, ou seja, sem considerar se existe ou não algum sistema de informação

já funcionando e sem distinguir, ou se preocupar, em que componente organizacional a ação foi executada.

Como resultado das atividades de elicitação (1) e (2), da Figura 1, foi elaborada a Tabela A, aplicando a ideia para o contexto "Gerenciamento de Exercícios para Grupos de Alunos". Este exemplo apresenta as relações de dependência entre os atores do contexto (teacher, student e group). O "grupo", considerado como um ator, depende estrategicamente de cada "estudante" e consequentemente o "professor" depende do "grupo" de estudantes. Essas dependências aparecem modeladas e organizadas em grupamentos de encadeamento de tempo chamadas SDsituations. As atividades de elicitação usaram como fonte de informação exclusivamente as ações do exemplo "e-learning" [8]. Na tabela estão apresentadas as metas dos atores: professor, aluno e grupo (teacher, student e group) grupadas por SDsituations.

Pelo procedimento de captura das metas, fica determinado o "papel" do ator "depender" e "dependee", respectivamente: quem depende (depender) e de quem se depende (dependee) para o atingimento da meta. O procedimento também identifica os agrupamentos das metas para a solução de uma SDsituation. Uma SDsituation é um construto de dependências estratégicas com uma intencionalidade (uma meta) situacional temporalmente compartilhada por alguns atores [5]. Para o contexto do exemplo [8], cinco SDsituations foram mapeadas. São elas: 1 - Organização dos Grupos, 2 - Definição do Schedule, 3 - Proposição do Exercício, 4 - Solução do Exercício e 5 - Avaliação do Exercício. O "ID" da SDsituation está identificado na segunda coluna da tabela A. O número indica para qual SDsituation cada meta trabalha.

Um benefício de se dividir o sistema segundo o conceito de SDsituations é a administração da complexidade, trabalhando o Engenheiro de Requisitos com diferentes situações em diagramas ou modelos distintos.

Tabela A – Metas grupadas por ator "depender" em ordem cronológica

DEPENDER	(SDsituation)					DEPENDEE
teacher						
accurate [solution]	(5)	exercise	BE	learned	by	**student**
	(5)	results	BE	published		
easy [delivery]	(5)	exercise	BE	evaluated		
	(4)	exercise	BE	done	by	**group**
	(4)	schedule	BE	obeyed	by	**group**
	(3)	exercise	BE	understood	by	**group**
	(3)	exercise	BE	understood	by	**student**
	(3)	exercise	BE	proposed		
fare [schedule]	(2)	schedule	BE	defined		
proper [delivery]	(2)	preferences	BE	considered		
reliable [group]	(1)	group	BE	created		
	(1)	group	BE	organized	by	**student**
student						
	(5)	results	BE	published	by	**teacher**
	(5)	solution	BE	discussed		
	(5)	exercise	BE	evaluated	by	**teacher**
quality [solution]	(4)	exercise	BE	done		
	(4)	schedule	BE	obeyed		
reliable [exercise]	(3)	exercise	BE	understood		
	(2)	schedule	BE	defined	by	**teacher**
	(2)	preferences	BE	considered		
	(1)	group	BE	organized		
group						
	(5)	exercise	BE	evaluated	by	**teacher**
quality [solution]	(5)	solution	BE	discussed		
	(4)	exercise	BE	done		**student**
quality [solution]	(4)	exercise	BE	shared		
	(4)	schedule	BE	obeyed		

2.2 A MODELAGEM: Diagramas IP, Modelos SD, Modelos SR e Especificação das SDsituations

Como resultado das atividades (3), (4) e (5), (da Figura 1), foi preparado originalmente para todas as SDsituation três produtos: um Diagrama IP, um Modelo SD e um Modelo SR. Os Diagramas IP, os modelos SD e os Modelos SR por motivos de limitação de espaço são os exemplos para apenas uma SDsituation. A Figura 2 mostra o Diagrama IP e a Figura 3 mostra o Modelo SR, ambos da SDsituation – Solução do Exercício. O modelo SD não foi apresentado por limitação de espaço e, mesmo porque, ele pode ser interpretado do modelo SR, o qual é o detalhamento do modelo SD correspondente.

O modelo SR detalha o diagrama IP, este último pode ser considerado como uma prévia (ou um rascunho) para o modelo SD. As dependências, as correlações e as contribuições determinadas no diagrama IP são mapeadas no modelo SR. Observe que quando as dependências são escolhidas algumas metas podem receber uma

alteração pequena de semântica no nome (ex. “colaborative exercise BE done”).

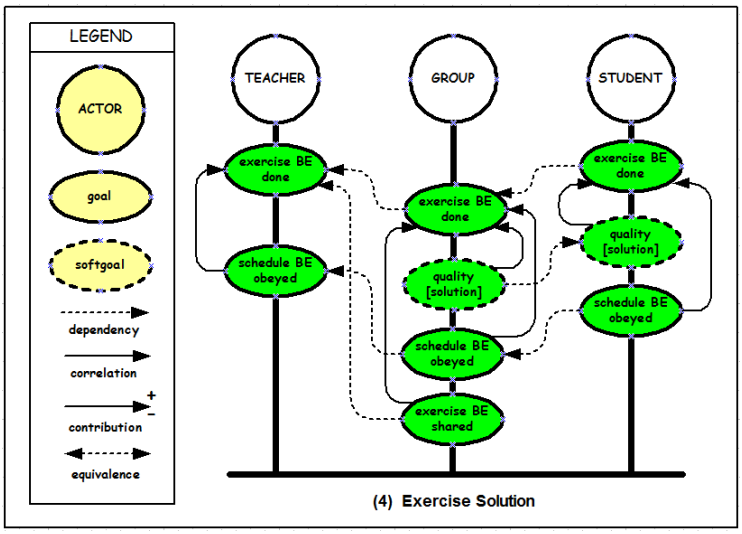

Figura 2 – Diagrama IP – SDsituation: 4 - Solução do Exercício

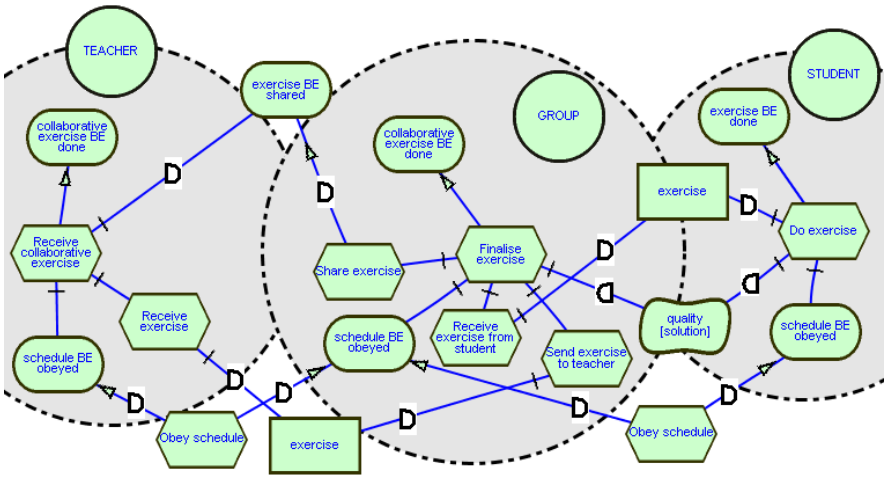

Figura 3 – Modelo SR – SDsituation: 4 - Solução do Exercício

2.3 A ANÁLISE: Formulários (Templates) do Diagnóstico

Interpretando os modelos i* preparados anteriormente, podemos considerar que existem duas estruturas básicas (estruturas canônicas [7]) ou construtores que são frequentes em modelos i*.

A primeira estrutura básica, a SDsituation, aparece no modelo SD. Cada SDsituation é formada por uma ou mais dependências e qualquer SDsituation pode ser visualizada separadamente das demais SDsituations. Uma SDsituation deve ser identificada como uma unidade de negócio. Como a modelagem se refere a um único contexto, a afirmação deve ser interpretada como: "É uma boa ideia trabalhar com SDsituations separadamente, porém cada uma depende criticamente das demais".

A segunda estrutura canônica, o SRconstruct, aparece no modelo SR. Um SRconstruct, é formado por uma meta (o nome da meta é o nome do SRconstruct porque existe somente uma meta fim "end" no SRconstruct) e por pelo menos uma tarefa (como o meio de atingir o "end"). Além da meta e das tarefas ("means"), fazem parte da estrutura todos os componentes e subcomponentes necessitados pelas tarefas que são: subtarefas, recursos, metas flexíveis (softgoals) e metas.

Cada SRconstruct (formado por uma meta "end", por tarefas, recursos, metas flexíveis (softgoals) e metas), existe para atingir um fim ("end") bem determinado e, os componentes do SRconstruct estão essencialmente juntos para apoiar os meios ("means") para atingir o fim ("end") na estrutura.

3. Diagnóstico de SDsituations e de SRconstructs

Dada cada estrutura básica das SDsituations e das SRconstructs, as questões formuladas devem ser respondidas (veja tabelas B e C). As palavras em negrito são "lacunas" ou "place-holders" para serem substituídas pelos nomes reais dos elementos que aparecem nos modelos SD e SR, a partir das duas estruturas canônicas.

Uma SDsituation é formada por elementos de dependência, entretanto a finalidade de diagnosticar uma SDsituation é descobrir quais dependências são "problemáticas" (ou não contribuem para a meta) e assim questionar também se existe algum outro ator que poderia ser melhor colaborador para aquela SDsituation. A Tabela B [7] contém as questões preparadas pelo método para serem aplicadas às SDsituations.

A ideia de diagnosticar SRconstructs é descobrir quais componentes são "problemáticos" (ou seja, não contribuem para a

meta), como também identificar se há alguma outra maneira que poderia ser usada para que a meta do SRconstruct seja alcançada. Na Tabela C [7] mostramos as perguntas preparadas pelo método para serem aplicadas pelo Engenheiro de Requisitos.

Tabela B – Questões relativas ao diagnóstico de SDsituations

SDSITUATION: "nome da SDsituation"

I. QUESTÕES EXTERNAS
1. *Quem mais poderia colaborar com o* "***depender***" para atingir "***nome da meta da SDsituation***"? Quanto ele pode colaborar*? (completamente ou parcialmente)*
2. *Porque o* "***dependee***" colabora com o "***depender***" para atingir "***nome da meta da SDsituation***"?
3. *Quais* SDsituations acontecem antes de "***nome da SDsituation***"?
4. *Que* problemas com as SDsituations anteriores podem ser identificados para atingir "***nome da meta da SDsituation***"?
5. *E se o* "***dependee***" não puder colaborar na "***nome da SDsituation***"?

II. QUESTÕES INTERNAS
6. *Quais são os problemas dentro da* "***nome da SDsituation***"? Que tipos de problemas (precisão, deficiências, ambiguidades ou omissões) são identificados na "***nome da meta da SDsituation***"?
7. *Que detalhes o* "***depender***" necessita?
   a) Caso: dependência de recurso – Quais são os problemas de disponibilidade (tempo e precisão) de "***nome do recurso***"? Quando*?* Como*? Quanto?*
   b) Caso: dependência de meta-concreta – Quais são os problemas que "***nome da meta-concreta***" tem para ser alcançada pelo "***dependee***"? (tempo, habilidade) Quando*?* Como*? Quanto?*
   c) Caso: dependência de meta-flexível – Quais são os problemas que "***nome da meta-flexível***" tem para ser razoavelmente satisfeita pelo "***dependee***"? (capacidade) Existe "***nome da meta-flexível***" na conclusão de "***nome da SDsituation***"? Porque? Quem está demandando pela meta-flexível?
   d) Caso: dependência de tarefa – O "***dependee***" recebeu as orientações de como fazer "***nome da tarefa***"? Pode o "***dependee***" fazer a tarefa? (tempo, habilidade)
8. *Qual dependência possui a maior responsabilidade para atingir* "***nome da meta da SDsituation***"? Porque?

Tabela C – Questões relativas ao diagnóstico de SRconstructs

SRCONSTRUCT: "*nome da meta final do SRconstruct*"

I. QUESTÕES EXTERNAS
1. *Quem mais tem como meta "nome da meta final"?*
2. *Quais são as alternativas possíveis para que "nome da meta final" seja atingida? Por que?*
3. *Quais são os elementos de dependência dos dependees?*
4. *Que tipos de problemas (precisão, deficiências, ambiguidades ou omissões) podem ser vislumbrados? Quantos? E se os recursos ficarem indisponíveis? Quem tem a culpa? Como evitar tais problemas?*
5. *E se a* "***nome da meta final***" for partilhada com outro ator?
6. *Que outro construto depende dessa meta? Por que? Quanto?*

II. QUESTÕES INTERNAS (PARA CADA TAREFA MEIO)
7. *Quais são os problemas* com a tarefa "***nome da tarefa meio***"? *Por que*?
8. Para cada tarefa "***nome da tarefa meio***" que componentes *são* necessários para atingir "***nome da meta final***"?
   a) Caso: recurso – Quais *são* os problemas de disponibilidade do recurso "***nome do recurso***"? (tempo, precisão). Quando*?* Como*?*
   b) Caso: subMeta-concreta – Quais *são* os problemas para que "***nome da submeta-concreta***" seja atingida pelo "dependee/ator"? (tempo, habilidade). Quando*?* Como*?*
   c) Caso: meta-flexível – Quais *são os problemas para que* "***nome da meta-flexível***" seja razoavelmente satisfeita pelo "***dependee/ator***"? (capacidade) Existe "***nome da meta-flexível***" no final do "***SRconstruct goal's name***"? *Por que*? Quais *são* as contribuições, recebidas ou fornecidas por ***"nome da meta-flexível"***?
   d) Caso: subTarefa – Pode o ator "***dependee/ator***" fazer "***nome da tarefa***"? (tempo, habilidade)
9. Existe algum detalhe omitido na operacionalização? De que tipo? Por que? Como? Quanto?
10. Existe algum recurso faltando? De que tipo? E se o recurso não estiver disponível?

Os diagnósticos de SDsituations e os diagnósticos de SRconstructs são intrinsecamente interligados pelas perguntas e, consequentemente, pelos problemas. Um problema que seja diagnosticado por algum dos dois templates certamente será

identificado pelo outro template de diagnóstico. Por isso, Oliveira et al., [7] afirma que o processo conduz a duas oportunidades de identificar um mesmo problema: uma pelo Diagnóstico da SDsituation e outra pelo Diagnóstico do SRconstruct.

Depois de terminados os diagnósticos, o engenheiro, com o uso da ferramenta, pode pedir a elaboração do Quadro - Metas X Problemas [7], através do UC 7 – Exibir Matrix. O quadro "Metas X Problemas" é útil para apoiar um plano de priorização da solução dos problemas detectados na fase de diagnósticos. No quadro, veja ilustração na tabela D, a ferramenta iStar Diagnoses evidencia, de modo grupado, um resumo que indica todas as metas que são afetadas por cada problema individualmente.

Tabela D – Ilustração do Quadro - Metas X Problemas

Metas → Problemas	M1	M2	M3	M4	M5	Mi	Mj
P1		X	X				X
P2			X			X	
P3					X	X	
Pi				X		X	
Pm	X	X	X		X	X	X
Pn						X	
Pz			X			X	

4. A Implementação da Ferramenta

4.1. Descrição sucinta da iStar Diagnoses

A ferramenta construída para fazer os diagnósticos auxilia o engenheiro de requisitos a analisar os modelos SD e os modelos SR do Framework i*. A análise feita pelo engenheiro de requisitos é baseada no questionário gerado para cada SDsituation e para cada SRconstruct. Os questionários criados são preparados dinamicamente a partir dos componentes dos modelos.

Os componentes dos modelos, para uso da ferramenta iStar Diagnoses, necessitam que o engenheiro de requisitos faça a "especificação textual" do modelo, ou seja: o cadastramento dos atores do sistema e, depois disso, o cadastramento de cada SDsituation, preenchendo na interface da ferramenta os seus componentes para salvá-la no banco de dados. A partir da SDsituation cadastrada, pode-se gerar o questionário baseado nos componentes do modelo SD, respondê-lo, e salvar no banco de dados. Mais detalhes são mencionados no capítulo 4.4, o qual faz a descrição das interfaces.

Os SRconstructs só podem ser cadastrados se forem referenciados a alguma SDsituation, da mesma forma que seus componentes são preenchidos na interface e a "especificação textual" do modelo SR pode ser salvo. O engenheiro de software gera o questionário baseado no modelo preenchido e este é salvo no banco de dados. Os questionários das SDsituations e das SRconstructs podem ser abertos para posterior análise.

4.2. Diagrama de Classes

Na Figura 4 as classes de entidades na cor amarela são provenientes de um "meta modelo" [19] do modelo SD do Framework i* e as classes de entidades na cor azul são provenientes da estratégia de diagnósticos [7]. Uma rápida observação, pelo diagrama de classes (UML) da Figura 4, nos leva a conclusão que as classes das entidades na cor azul: Diagnostico, Questão, SDsituation e SRconstruct são muito importantes no modelo. Essa conclusão é devido ao número de ligações que essas classes de entidade possuem, devendo se iniciar por essas classes a descrição do modelo que é aplicado a cada projeto.

Em uma SDsituation participam "2 ou mais" (2..*) atores e em cada ligação de dependência entre atores (com papeis de "depender" e "dependee") participa apenas "1" elemento. O elemento da dependência desempenha o papel de "dependum". E, cada dependum pode ser de um dos quatro tipos (Tarefa, Recurso, Meta flexível ou Meta). Uma SDsituation envolve "2 ou mais" (2..*) SRconstructs, isso porque cada ator, e eles são no mínimo 2 para cada SDsituation, deverá possuir pelo menos uma meta. E, um Diagnóstico de perguntas de SDsituation examina (verifica através de questões) "uma ou mais" (1..*) SDsituations.

Um SRconstruct é deliberado por uma única (1) Meta (meta fim) e essa meta pode ser atingida por "1 ou mais" (1..*) Tarefas (tarefas meio). Cada Tarefa é formada por "1 ou mais" (1..*) elementos, de um dos quatro tipos (Tarefa, Recurso, Meta flexível ou Meta), podendo algum desses elementos possuir o papel de "dependum" em alguma dependência. Um Diagnóstico do SRconstruct analisa (verifica através de questões) "1 ou mais" (1..*) SRconstructs.

A última relação ainda não descrita, e não menos importante que as demais, indica que um Ator se associa a outros Atores com papéis de "depender" e "dependee" e cada associação determina um

elemento da associação, o "dependum" que é o elemento da dependência estratégica.

O esquema do BD derivado do modelo de classes, da Figura 4, tem a seguinte formatação:

Diagnoses = {diagnosisID (pk)}
Question = {questionID (pk), diagnosisID (fk), questionText, answerText problemFlag, elementID (fk)}
SDsituation = {sdSituationID (pk), sdSituationName, diagnosisID (fk)}
SRconstruct = {srGoalID (pk, fk), sdSituationID (fk), diagnosisID (fk)}
Participation = {numSDsituation (pk), dependerID (fk), dependeeID (fk), dependumID (fk)}
Dependency = {dependerID (pk, fk), dependeeID (pk, fk), dependumID (pk, fk)}
Decomposition = {taskID (pk, fk), elementID (pk, fk)}
Actor = {actorID (pk), actorName, actorType} actorType = role | position | agent
Element = {elementID (pk), elementType, dependumFlag}
Task = {taskID (pk), taskName, endGoal (fk)}
Resource = {resourceID (pk), resourceName}
Softgoal = {softgoalID (pk), softgoalType, softgoalTopic}
Goal = {goalID (pk), goalSymbol, goalVerb}

A Ferramenta iStar Diagnoses implementou o modelo de arquitetura de Software em 3 camadas “Model-View-Controller (MVC)” atualmente considerado um “Design Patterns” (arquitetura padrão) utilizado na Engenharia de Software. A arquitetura em 3 camadas está relacionada com a divisão de papéis ou responsabilidades em: camada de apresentação, de negócio e de acesso aos dados. A ideia desse modelo de desenvolvimento é separar, para dar independência, as três camadas do software: "entity”, “boundary” e “control”.

4.3. Diagrama de Casos de Uso

A ferramenta iStar Diagnoses é composta por sete casos de uso (UCs), veja Figura 5, entre esses UCs três são considerados preliminares ao diagnóstico, porque eles estabelecem a “especificação textual” das estruturas canônicas, pois a ferramenta não implementa automaticamente uma interface gráfica. O Engenheiro de Requisitos depois de criar o projeto que é objeto de análise (com mnemônico e nome do Projeto) deve: identificar os atores do contexto organizacional, através do UC 1 – Editar Atores; criar cada SDsituation modelada nos diagramas SD, através do UC 2 – Editar SDsituations; e criar cada SRconstruct modelada nos diagramas SR e ligada a cada SDsituation anteriormente representada, através do UC 3 – Editar SRconstructs.

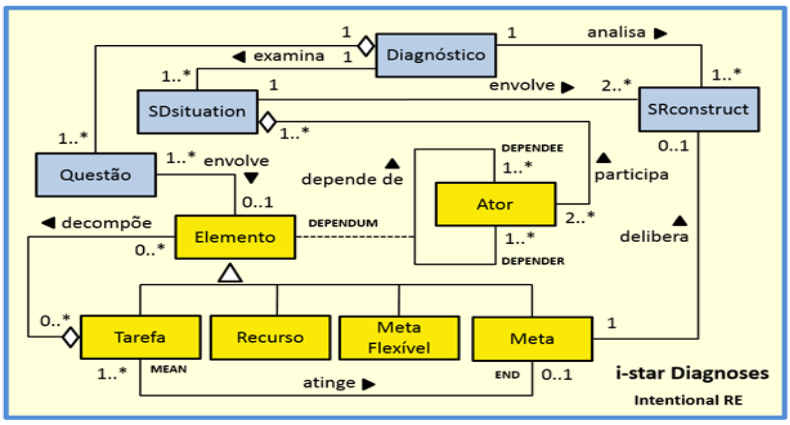

Figura 4 – Diagrama de classes da ferramenta iStar Diagnoses

A ferramenta permite ao engenheiro fazer ou modificar o diagnóstico de cada SDsituation respondendo as questões do "template" através do UC 4 – Responder Questionário de uma SDsituation, fazer ou modificar o diagnóstico de cada SRconstruct respondendo as questões do "template" através do UC 5 – Responder Questionário de uma SRconstruct.

O engenheiro pode trabalhar com diagnósticos aplicando templates de SDsituations e SRconstructs em duas línguas: português e inglês, através do UC 6 – Escolher Idioma da Interface.

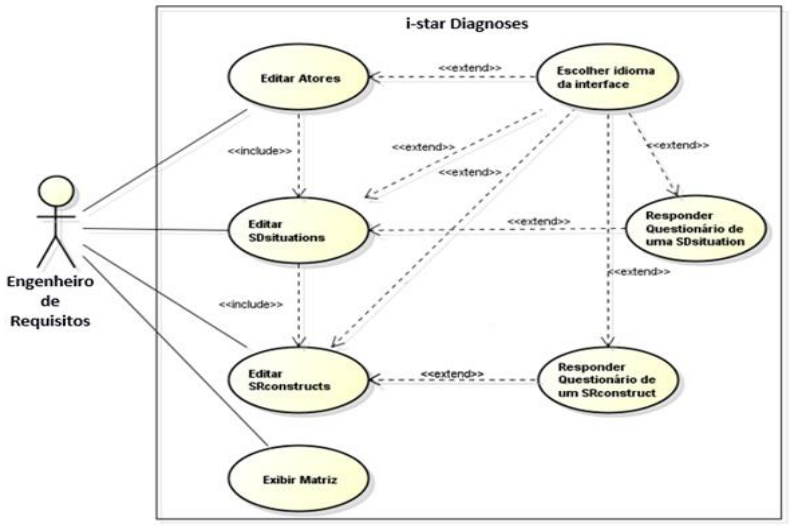

Figura 5 – Diagrama dos casos de uso da iStar Diagnoses

4.4. As interfaces da ferramenta

A Figura 6 apresenta a tela para adição e edição de SDsituations. A Figura 7 apresenta a tela referente a adição e edição de SRconstructs. O usuário irá adicionar tarefas-meio referentes a uma meta-final, assim como suas decomposições. A Figura 8 apresenta a tela referente ao questionário montado para a SDsituation que foi editada e mostrada na Figura 3.

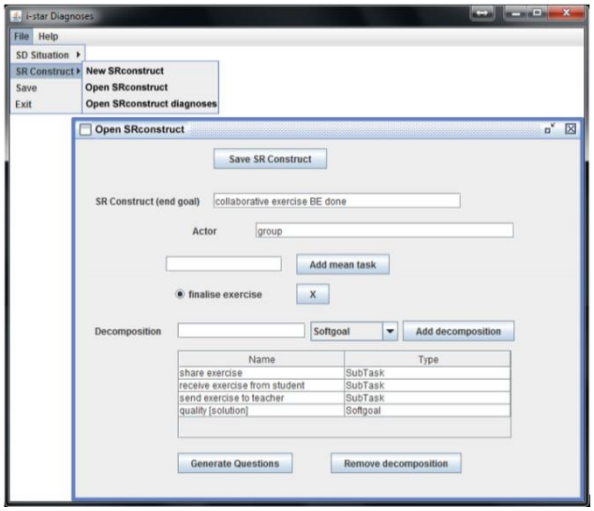

Figura 6 – Tela do UC 2 – Editar SDsituations

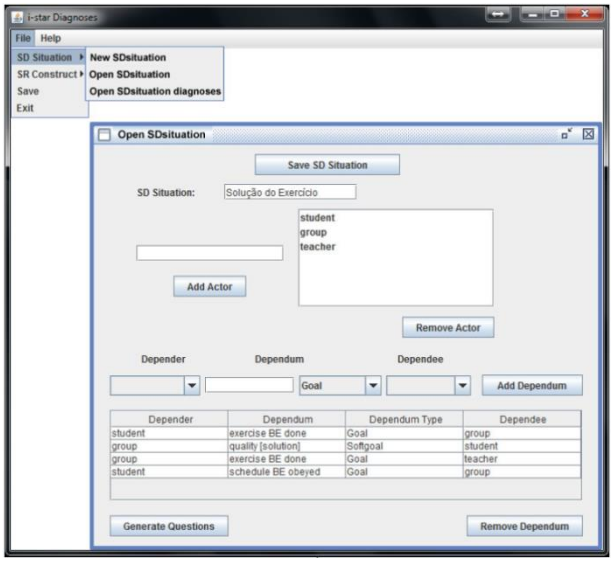

Figura 7 – Tela do UC 3 – Editar SRconstructs

4.5. Linguagem de Programação e SGBD

A ferramenta de diagnósticos foi desenvolvida na linguagem Java. O Java foi escolhido para a implementação por ser orientada a objetos, ou seja, permite a codificação por meio de classes e, através de suas instâncias, as chamadas de métodos são agilizadas. Além disso, a organização do sistema por meio de pacotes e classes permite que haja um melhor entendimento de cada parte do código e suas funções.

Por ser uma das mais populares para desenvolvimento em Java, a IDE utilizada foi o NetBeans. Ela permite a organização da interface gráfica com dinamismo e clareza.

Para que as informações que os usuários eventualmente queiram salvar e reutilizar fossem devidamente armazenadas, utilizou-se o SGBD MySQL. Este foi escolhido por ser um SGBD gratuito, muito conhecido e utilizado por empresas e desenvolvedores de software de todo o mundo.

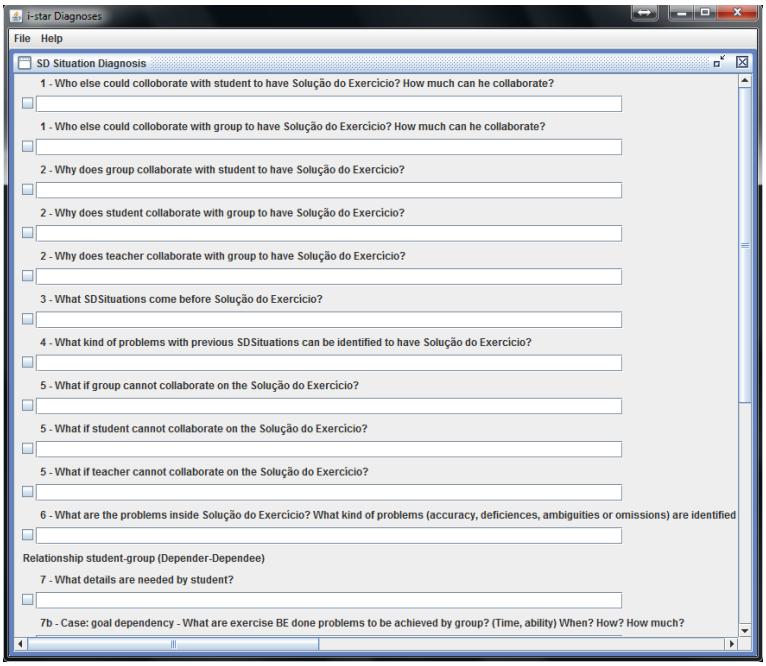

Figura 8 – Tela do UC 4 – Responder Questionário de uma SDsituation

5. Conclusões

Neste artigo apresentamos a ferramenta iStar Diagnoses[1] que apoia a etapa de análise de requisitos do método ERi*c. A análise através de diagnósticos trabalha usando duas estruturas canônicas definidas pelo método ERi*c. O bom emprego das estruturas canônicas (SDsituation e SRconstruct) propiciam dividir o modelo em partes pequenas do problema com o benefício da administração da complexidade dos modelos i* para o melhor entendimento pelos interessados ("stakeholders"). Nós acreditamos fortemente que a estratégia de diagnósticos fornece uma base para a análise de modelos i* de modo a assegurar uma melhor qualidade final. Sobre trabalhos que promovem a verificação de modelos i*, apenas encontramos a técnica de Easterbrook et al. [15] que lida com a validação modelos i* aplicado a ideia de pontos de vista. Nosso trabalho trabalha na linha de tema do trabalho de Potts [14].

No momento, pretendemos dar prosseguimento ao nosso trabalho da abordagem de diagnósticos nesta mesma direção atual, usando a estratégia de leitura para fazer a inspeção de modelos i* como também a possível edição automática de questões para promover a qualidade das questões atuais. Pretendemos também aplicar a ferramenta para problemas de maior escala com a possível aplicação na indústria.

6. Referências

1. MELGAÇO, Amanda Garcez. SOUZA, Jonathan Dias Lima; Ferramenta de diagnósticos de Modelos i* aplicando a Engenharia de Requisitos Intencional (ERi*c). Projeto final (Bacharelado em Ciência da Computação) - Instituto de Matemática e Estatística, UERJ, Rio de Janeiro, 2014.
2. YU, E. Modelling Strategic Relationships for Process Reengineering. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada - 1995.
3. Chung, L.; Nixon, B.; Yu, E.; MYLOPOULOS, J.; Non-Functional Requirements in Software Engineering – Kluwer Academic Publishers 2000 – Massachusetts, USA.

4. LEITE, Julio C. S. P.; FRANCO, Ana P. M. A Client Strategy for Conceptual Model Acquisition, Proceedings of the International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego (1993), pp. 243-246.

5. OLIVEIRA, A. Padua A.; CYSNEIROS, L. M.; "Defining Strategic Dependency Situations in Requirements Elicitation" The IX Workshop on Requirements Engineering; Rio, Brazil - July/2006.

6. OLIVEIRA, A. Padua A.; LEITE, J.; CYSNEIROS, L.; CAPPELLI, C.; "Eliciting Multi-Agents Systems Intentionality: From Language Extended Lexicon to i* Models", Proceedings of the 26th International Conference of the Chilean Computer Science Society. IEEE Computer Society, 2007.

7. OLIVEIRA, A. Padua A.; LEITE, J.; CYSNEIROS, L., LUCENA, C.; i* Diagnoses: A Quality Process for Building i* Models pp. 9-12, CAiSE'08 Forum Proceedings of the Forum at the CAiSE'08 conference.

8. GRAU, G.; FRANCH, X.; MAIDEN, N.; PRiM: An i*-based process reengineering method for information systems specification; ScienceDirect - Information and Software Technology 50 (2008).

9. OLIVEIRA, A. Padua A.; (2008) Intentional Requirements Engineering - A Method for Requirements Elicitation, Modeling, and Analysis. 261p. Doctoral Thesis; Computer Science Department, PUC-Rio.

10. OLIVEIRA, A. Padua A.; LEITE, J. C. S.; CYSNEIROS, L. M.; Método ERi*c - Engenharia de Requisitos Intencional. The XI Workshop on Requirements Engineering - WER 2008 – Barcelona.

11. OLIVEIRA, A. Padua A.; LEITE, J.; CYSNEIROS, L. (2010) Using i* Meta Modeling for Verifying i* Models. In: iStar 2010 4th International i* Workshop, Hammamet, Tunisia. CEUR - Workshop Proceedings. Aachen : rwth-aachen, 2010. v. 586. p. 76-80.

12. CHESBROUGH, Henry. Open Innovation: The New Imperative for Creating and Profiting from Technology, Boston: Harvard Business School Press, 2003.

13. CHRISTENSEN, Clayton M. e Raynor, Michael E, The innovator's solution; O crescimento pela inovação: Como crescer de

forma sustentada e reinventar o sucesso – Tradução de Serra, Afonso C.C – Elsevier – RJ - 2003

14. POTTS, Colin; TAKAHASHI, Kenji; ANTÓN, Annie I.; "Inquiry-Based Requirements Analysis", IEEE Software, Volume 11, Issue 2 (March 1994), Pages: 21 - 32

15. EASTERBROOK, S; YU, E; ARANDA, J; FAN, Y; HORKOFF, J, M; Do Viewpoints Lead to Better Conceptual Models? An Exploratory Case Study - Requirements Engineering, 2005. Proceedings. 13th IEEE Conference on Requirements Engineering.

16. PINTEC2011 - www.pintec.ibge.gov.br/downloads/InstrucoesPINTEC2011.pdf

17. LEI DE INOVAÇÃO TECNOLOGICA http://www.lexml.gov.br/urn/urn:lex:br:federal:lei:2004-12-02;10973

18. TIGRE, Paulo Bastos; Gestão da Inovação: A Economia da Tecnologia no Brasil, Rio de Janeiro: Elsevier, 282 páginas, 2006.

19. OLIVEIRA, A. Padua A.; LEITE, J. C. S.; Building Intentional Models Using the ERi*c Method http://www.e-publicacoes.uerj.br/index.php/cadinf/issue/view/375/showToc – Cadernos do IME – Série Informática - v. 32 (2011).