UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

GOAL-REQUIREMENTS LANGUAGE BASEADO EM METRICAS

VINICIUS GAMA VALORY FRAUCHES
CARLOS EDUARDO CORREA BRAGA
EDUARDO ARRUDA

RENATA S. S. GUIZZARDI

ABRIL, 2013

1. Introdução

Organizações tendem a monitorar continuamente o contexto no qual estão inseridas, a fim de encontrar oportunidades e riscos que tem o potencial de impactar no alcance de seus objetivos. Nesse contexto, insere-se o conceito de *Key Performance Indicator* (KPI). KPI é um termo para uma medida, ou métrica, que avalia o desempenho em relação a algum objetivo. Indicadores são comumente usados para mensurar o grau de êxito e qualidade no cumprimento de objetivos estratégicos, processos, ou entrega de produtos e serviços. Para escolher os indicadores corretos para um determinado objetivo – seja ele um *goal*, um processo ou produto –, é necessário ter um bom entendimento acerca do que é importante para a organização. Essa importância é, de maneira geral, dependente do contexto. Por exemplo, indicadores úteis para o domínio de finanças podem ser inapropriados para o domínio de vendas (BARONE et al., 2011).

O presente trabalho tem como objetivo apresentar GRL (*Goal-oriented Requirement Language*) conforme adotado no padrão URN (*User Requirements Notation*) definido pela ITU (*International Telecommunication Union*). Além disso, é apresentada a ferramenta jUCMNav – primeira a apoiar contemplar a modelagem utilizando URN. Este trabalho está dividido em 4 seções, além da presente introdução, à saber: Seção 2 – descrevendo os elementos e relacionamentos básicos de GRL; Seção 3 – descrevendo indicadores no contexto de GRL; Seção 4 – apresentação da ferramenta jUCMNav; Seção 5 – expondo considerações finais acerca do tema.

2. A Linguagem GRL

Goal-oriented Requirement Language (GRL) é uma linguagem orientada a objetivos para suporte a modelagem inicial dos requisitos, especialmente não-funcionais, e sua comunicação. Esta linguagem faz parte da User Requirments Notation (URN) (ITU, 2011) criada e recomendada pela International Communication Union, para analise e comunicação de requisitos utilizando objetivos e cenários. Também faz parte da URN, a User Case Maps (UCM) que oferece notações para cenários relacionados a requisitos operacionais, requisitos funcionais e elementos de performance e arquiteturais (Amyot *et al.*, 2010). Vale ressaltar que as partes GRL e UCM, apesar de encontradas definidas separadamente, são integráveis entre si. UCM é uma importante parte da URN mas foge ao escopo deste trabalho.

A linguagem GRL é baseada nos conceitos dos frameworks i* (Yu, 1997) e NFR (Chung *et al.*, 2000). Um diagrama GRL apresenta os objetivos de alto nível - também chamados de estratégicos- e NFRs de um interessado (*stakeholder*), e suas decomposições a fim de alcançar os objetivos de alto nível. A seguir estão descritos os conceitos de GRL com seus respectivos termos da linguagem concreta.

2.1. Ator

Um ator é uma entidade com intenções e que executa ações a fim de alcançar seus objetivos. Pode representar tanto *stakeholders* quanto sistemas. Definições de ator podem ou não conter elementos intencionais – que serão apresentados posteriormente. Uma alternativa é modelar o domínio usando

somente atores e dependências entre eles, sem a utilização de elementos em seu escopo (*boundary*). A partir disso, pode-se adicionar elementos e indicadores ao escopo dos atores de maneira a especificar porque há alguma dependência entre eles e como essa dependência é atendida. A Figura 1 apresenta um "*Telecom Provider"* como um ator colapsado e um ator com escopo.

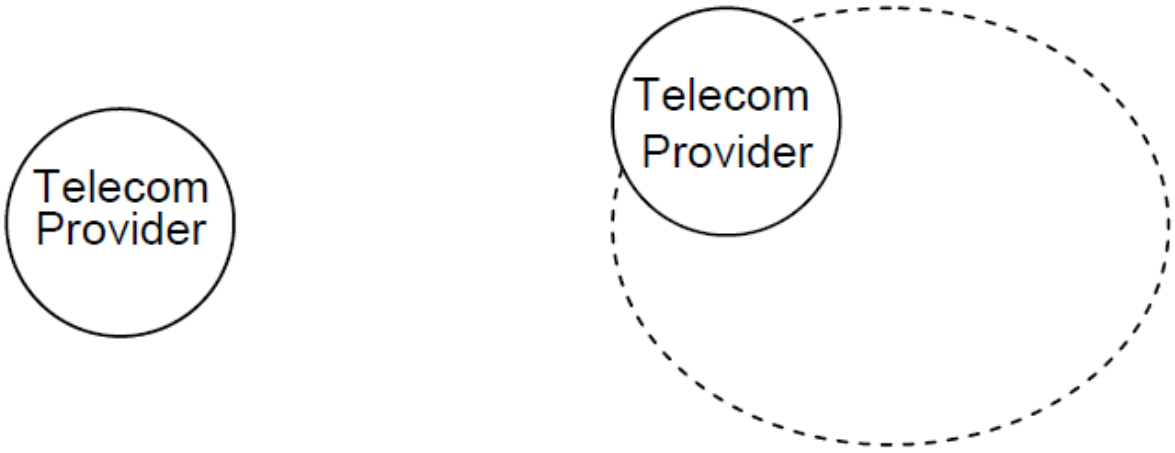


Figura 1 – Ator colapsado e ator com escopo (*boundary*)

2.2. Elementos Intencionais

Elementos intencionais são aqueles usados em modelos que permitem responder questões como o porquê de um comportamento particular, aspectos informacionais e estruturais terem sido escolhidos para serem incluídos nos requisitos do sistema, quais alternativas foram considerados, quais critérios foram utilizados para optar entre as alternativas e quais razões para escolher uma alternativa em detrimento de outra.

2.2.1. Objetivo (*Goal*)

Um objetivo é uma condição ou estado do mundo (*state of* affairs) que *stakeholders* gostariam de alcançar. Como o objetivo é alcançado não é especificado, permitindo que sejam consideradas alternativas. Um *goal* pode ser um objetivo de negócio (*business goal*) ou um objetivo de sistema (*system goal*). O primeiro expressa objetivos a respeito do negócio ou estado de negócio que um indivíduo ou organização deseja atingir. O segundo, por sua vez, expressa objetivos que um sistema deve cumprir e descreve de maneira geral seus requisitos funcionais. A Figura 2 apresenta a representação visual de um objetivo.

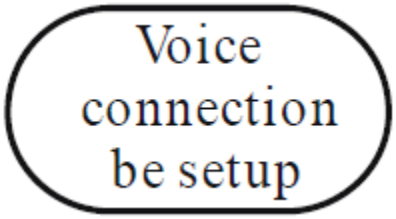


Figura 2 – Representação de objetivo

2.2.2. *Softgoal*

Um *Softgoal* é uma condição ou estado do mundo que *stakeholders* gostariam de alcançar, mas, ao contrário do que ocorre no conceito de *goal*, não há um critério bem definido para decidir se a condição é alcançada. Além disso, permite interpretação e julgamento subjetivos do modelador para decidir se um estado do mundo alcança suficientemente um *softgoal* modelado. *Softgoals* são frequentemente utilizados para descrever qualidades e aspectos não funcionais como, por exemplo, segurança, confiabilidade, desempenho, usabilidade, entre outros. A Figura 3 apesenta a representação visual de um *softgoal*.

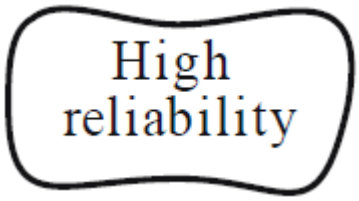


Figura 3 – Representação de *softgoal*

2.2.3. Tarefa (Task)

Uma tarefa especifica um jeito particular de fazer algo. Quando uma tarefa é parte de uma decomposição, ela restringe a tarefa de mais alto nível àquele particular curso de ações. Tarefas podem ser vistas também como as soluções em um dado sistema, ou seja, elas são a operacionalização de objetivos e *softgoals*.

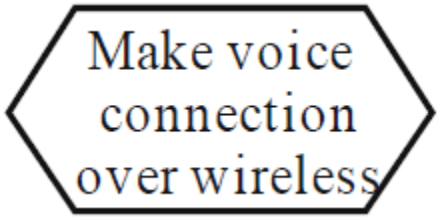


Figura 4 – Representação de tarefa

2.2.4. Recurso (Resource)

Um recurso é uma entidade física ou informacional para a qual a principal preocupação é acerca de sua disponibilidade.

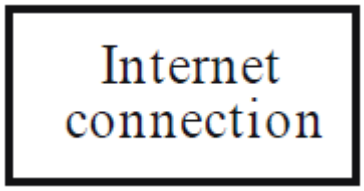


Figura 5 – Representação de recurso

2.2.5. Crença (Belief)

Uma crença é usada para representar *design rationale*. Crenças tornam possível que características do domínio sejam consideradas e refletidas de maneira apropriada no processo de tomada de decisão, facilitando assim uma posterior revisão, justificação e mudança no sistema, além de promover uma melhoria na rastreabilidade.

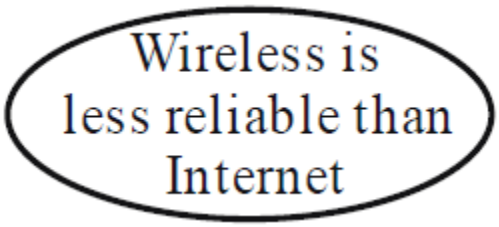


Figura 6 – Representação de crença

2.3. Relacionamentos Intencionais

2.3.1. Contribuição (*Contribution)*

Uma relação de contribuição descreve como um elemento contribui para a satisfação de outro elemento. Uma contribuição se trata de um desejo primário durante a modelagem.

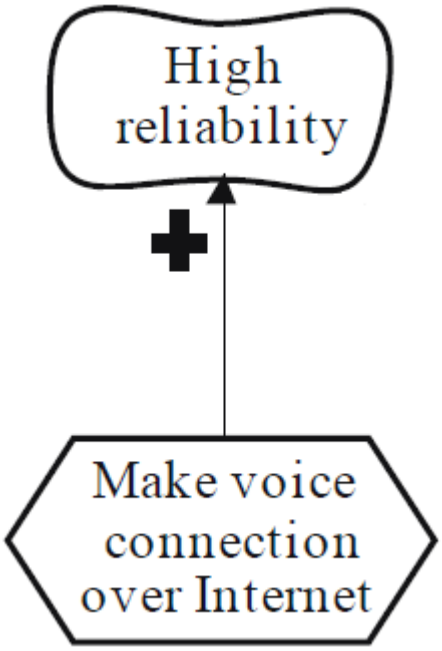


Figura 7 – Exemplo de contribuição entre tarefa e *softgoal*

2.3.2. Correlação (*Correlation*)

A relação de correlação é similar à contribuição, mas, nesse caso, o impacto não é desejável, ou seja, trata-se de um efeito colateral.

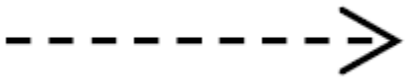


Figura 8 – Representação visual de correlação

2.3.3. Tipos de Contribuição e Correlação

Essas relações podem ser de diversos tipos, de acordo com o que é apresentado na Figura 9.

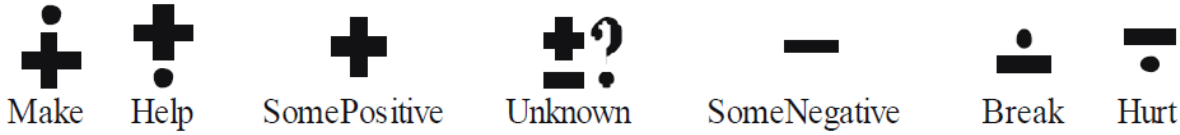


Figura 9 – Representação dos tipos de contribuição e correlação

- **Make**: positiva e suficiente, ou seja, o cumprimento do elemento fonte garante o cumprimento do elemento destino.

- **Help**: positiva, mas não suficiente. O cumprimento do elemento fonte ajuda, mas não garante o cumprimento do elemento destino.
- **SomePositive**: o cumprimento do elemento fonte ajuda no cumprimento do elemento destino, mas não se sabe em qual grau é essa ajuda.
- **Unknown**: o cumprimento do elemento fonte tem impacto no cumprimento do elemento destino, mas não se sabe se é um impacto positivo ou negativo e, além disso, não se sabe grau do impacto.
- **SomeNegative**: o cumprimento do elemento fonte tem impacto negativo no cumprimento do elemento destino, mas não se sabe o grau desse impacto.
- **Break**: negativa e suficiente. O cumprimento do elemento fonte garante que o elemento destino não será cumprido.
- **Hurt**: negativa, mas não suficiente. O cumprimento do elemento fonte prejudica o cumprimento do elemento destino, mas não garante que ele não será cumprido.

É importante ressaltar que essa se trata de uma avaliação qualitativa. A avaliação pode ser ainda quantitativa, variando de -100 a 100. Nesse sentido, um impacto -100 seria equivalente ao break e o impacto 100 ao make.

2.3.4. Dependência (*Dependency*)

Uma dependência descreve como um ator fonte (*depender*) depende de um ator destino (*depende*) em relação a um elemento intencional ou indicador (*dependum*). O *dependum* especifica sobre **o que** é a dependência. Com um elemento fonte da dependência, o *depender* pode especificar **porque** existe a dependência. Com um elemento destino da dependência, o *dependee* pode especificar **como** é cumprida a satisfação da dependência. São permitidas diversas configurações da relação de dependência. Nas figuras abaixo são apresentados alguns exemplos dessas configurações.

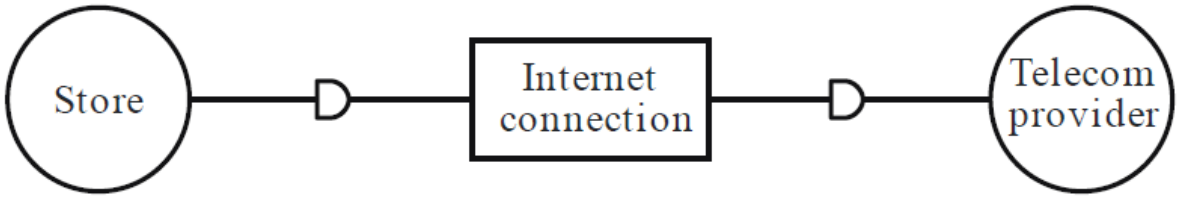


Figura 10 – Exemplo i

i) Essa configuração foca somente em dependência estratégica entre atores. **Porque** e **como** o *dependum* é provido são desconhecidos.

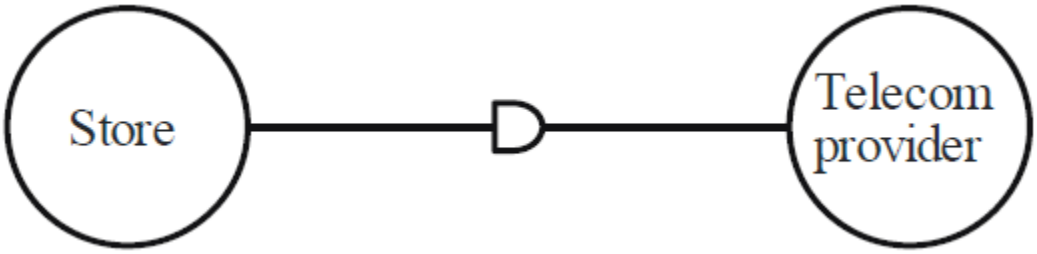


Figura 11 – Exemplo ii

ii) Essa configuração é típica de modelos preliminares que precisam de um maior nível de refinamento posterior. Nesse caso, uma dependência é identificada, mas **o que**, **porque** e **como** são ainda desconhecidos.

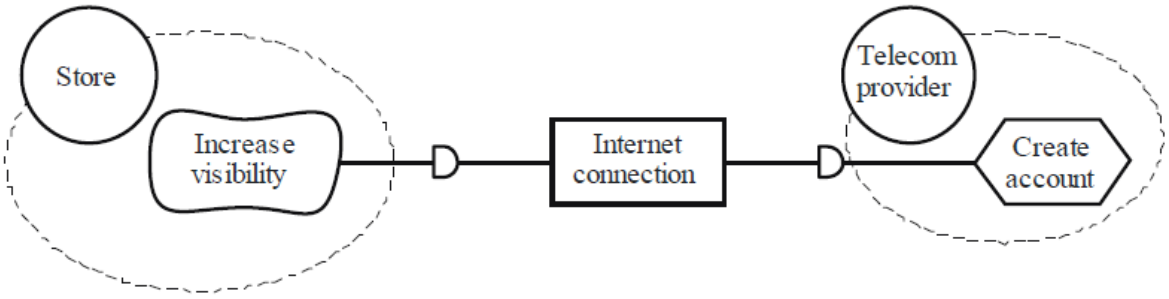


Figura 12 – Exemplo iii

iii) Essa configuração descreve tanto o *dependum*, quanto o **porquê** dele ser requerido e **como** ele é provido.

2.3.5. Decomposição (*Decomposition*)

Um relacionamento de decomposição provê a habilidade de definir quais elementos intencionais fonte precisam ser satisfeitos ou estarem disponíveis de maneira que o elemento intencional destino seja satisfeito. Três tipos de decomposição podem ser especificados: AND, XOR, IOR.

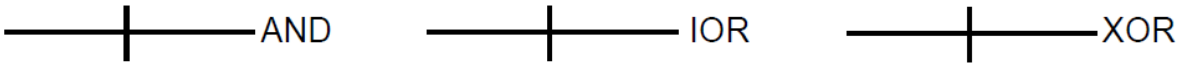


Figura 13 – Representação das relações de decomposição

- AND: é necessária a satisfação de todos sub-elementos para a satisfação do elemento alvo.

- XOR: a satisfação de um e somente um dos sub-elementos é necessária para a satisfação do elemento alvo.
- IOR: a satisfação de um sub-elemento é suficiente para a satisfação do elemento alvo, mas vários sub-elementos podem ser satisfeitos simultaneamente.

2.4. Indicador

Indicador é um elemento usado em modelos para expressar medições do mundo real. Ele pode explicitar tanto valores qualitativos quanto quantitativos. A Figura 14 apresenta o exemplo de um indicador expresso em GRL.

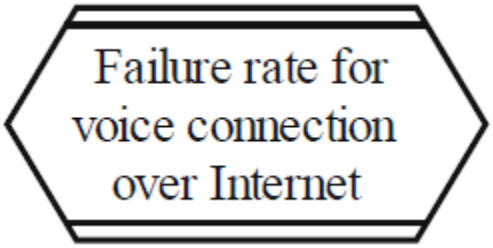


Figura 14 – Indicador de taxa de falha

3. Indicadores

Conforme discutido anteriormente, um indicador é uma medida, quantitativa ou qualitativa, que expressa o grau de aderência aos objetivos de uma organização (Barone *et al.*, 2011). Portanto, um indicador é definido para *avaliar* um objetivo, e para isso *mede* algo. Ou seja, um indicador é utilizado para avaliar a eficiência de algo que está sendo medido com respeito a um objetivo organizacional. Assim cada indicador tem um *current value*, um *target*, um *threshold* e um *worst values*. Onde, *current value* corresponde ao valor atual da medida, *target* é o valor que indica o alcance ao objetivo, o *threshold* é um divisor entre valores bons e ruins, ele simboliza quando a medida está agradando ou não. O *worst,* por fim, refere-se a valores os quais simbolizam um risco. Estes valores são utilizados em uma função que transforma o current value em um valor de avaliação (*Evaluation*) do indicador, conforme demonstra a Figura 15 (Barone *et al.*, 2011) e será melhor discutido adiante.

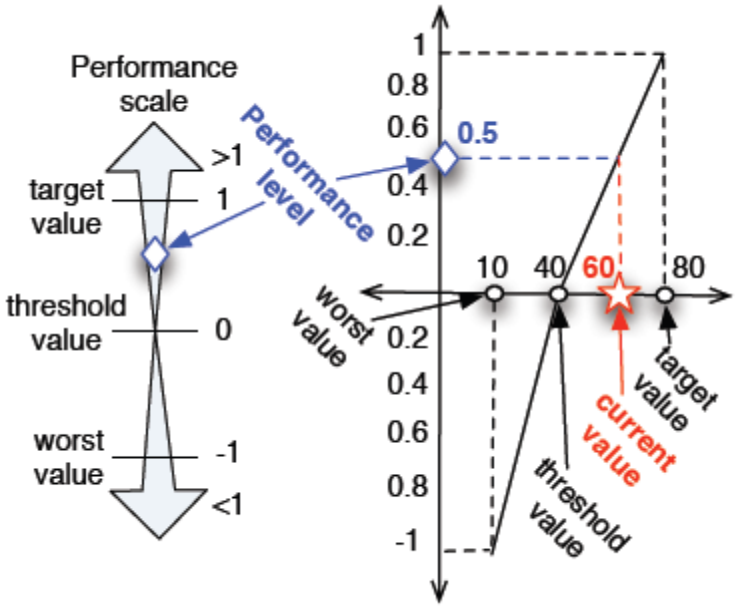


Figura 15 – Exemplo de interpolação para calcular níveis de desempenho (Barone *et al.*, 2011)

O cálculo da transformação demonstrada pela figura acima, é realizada da seguinte forma. Se o valor de *currentValue* é maior ou igual ao *thresholdValue*, então o *Evaluation* do indicador recebe o valor de |*currentValue* - *threshoouldValue*| / |*targetValue* - *threshouldValue*|. Caso contrário, ou seja, se o valor de *currentValue* for menor que o valor de *threshouldValue*, então o valor de *Evaluation* do indicador será -|*currentValue* - *threshoouldValue*| / |*wrostValue* - *threshouldValue*|.

O conceito de *evaluation* refere-se à avaliação de um elemento intencional, isso inclui o próprio indicador. É um valor que varia de 100 a -100 e que indica o grau de satisfação com relação a este elemento. Por exemplo, um *evaluation* -100 para uma *task* indica que a tarefa foi executada de forma precária, bem como um valor de *evaluation* 100 para um objetivo, seria interpretado como o sucesso do mesmo. Assim, para a avaliação automática de objetivos, e em geral elementos intencionais, os Evaluations são propagados através de relacionamentos intencionais. A recomendação ITU (2011) regulamenta atualmente o padrão GRL. Ela foca principalmente em como os termos são apresentados e estruturados. Porém não há uma recomendação quanto ao algoritmo utilizado para a propagação de *evaluations*, desta forma, fica a critério do implementador do algoritmo o comportamento da propagação.

Como o foco principal deste trabalho são os KPIs, nele será apresentado um algoritmo quantitativo, o qual leva em consideração a gama de possibilidades maior dos *Evaluations* quantitativos em relação aos qualitativos. Neste algoritmo, o e*valuation* de um elemento intencional é calculado através dos *evaluations* de outros elementos intencionais ligados a este. Utilizando cálculos diferentes para relacionamentos diferentes, um evaluation é calculado por uma função dependente dos *intentional relations* e *evaluations* relacionados.

Inicialmente, a relação de decomposição pode ser dividida em *AND* e *OR*, conforme discutido anteriormente. Uma relação de decomposição *AND* entre um elemento intencional e um grupo-parte de elementos intencionais, é tomada como uma função de mínimo. Ou seja, o elemento intencional decomposto terá a mesma satisfação que a menor satisfação dentro do grupo-parte de elementos intencionais. Por outro lado, na decomposição OR, o valor de satisfação do composto terá o mesmo valor da maior satisfação do grupo-parte.

A relação de contribuição por sua vez é calculada como um quantificador da satisfação. Assim, o valor quantitativo da contribuição é multiplicado pela satisfação do elemento relacionado e dividido por 100. Após calculada cada contribuição para um determinado elemento intencional, é produzida uma média destas contribuições, afim de achar sua satisfação. A relação de dependência, por fim, é utilizada no agrupamento de contribuições que acabamos de definir. Esta relação calcula a contribuição de um elemento para com o dependente através de sua satisfação, assim, caso a satisfação seja negativa, a contribuição terá o mesmo valor, caso a satisfação não seja negativa, a contribuição é zero.

Com esse algoritmo é possível calcular o alinhamento dos objetivos definidos, através dos indicadores. Algumas ferramentas de GRL utilizam a captura automática de *currentValues*, contribuindo para uma avaliação de objetivos em tempo real.

4. jUCMNav

jUCMNav é um editor gráfico para *User Requirements Notation (URN)* e, portanto, utilizado para modelar objetivos de acordo com a linguagem GRL. Esta ferramenta foi desenvolvida como um plugin do IDE Eclipse (Ambiente de Desenvolvimento Integrado) e possui licença grátis para sua utilização.

4.1 Requisitos

O jUCMNav possui compatibilidade com a maioria dos sistemas operacionais, porém este documento só abordará a versão Windows. Por se tratar de um plugin do Eclipse é preciso instalar esta ferramenta primeiro. Recomendamos que instalem a **versão 3.7** e **pacote Modeling Tools** do Eclipse, mesmo que qualquer versão a partir da **3.5** seja permitida para instalar o plugin. Além disso, é preciso instalar o GraphViz, uma ferramenta que auxilia na funcionalidade de AutoLayout do jUCMNav.

4.2 Instalação

Com os requisitos satisfeitos, inicia-se a etapa de instalação do plugin. Segue abaixo os passos deste processo:

- Selecione Ajuda -> Instalar novo Software... -> Adicionar
    - Nome: jUCMNav
    - Link: http://jucmnav.softwareengineering.ca/jucmnav/updatesite/
- Recomendamos, após instalar o jUCMNav, fazer atualização do Eclipse.

4.3 Visão Geral

Como toda aplicação no eclipse, primeiro é necessário criar um projeto. Após a criação de um projeto, para criar um modelo GRL basta seguir os passo abaixo:

- Arquivo -> Novo -> Outros
- Selecionar jUCMNav , e marcar o *checkbox GRL*.

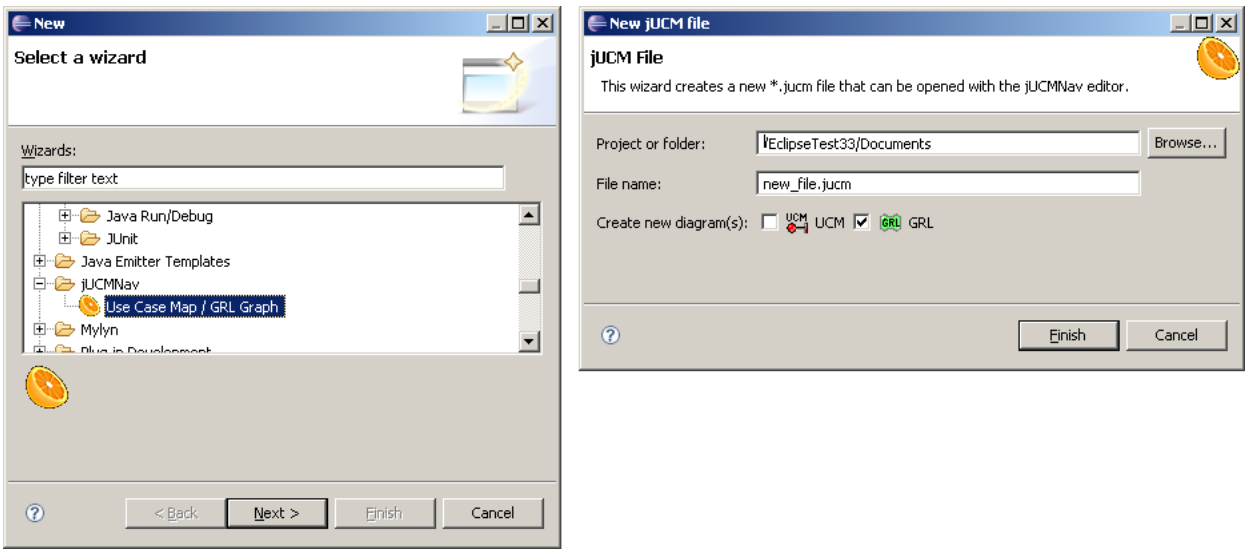
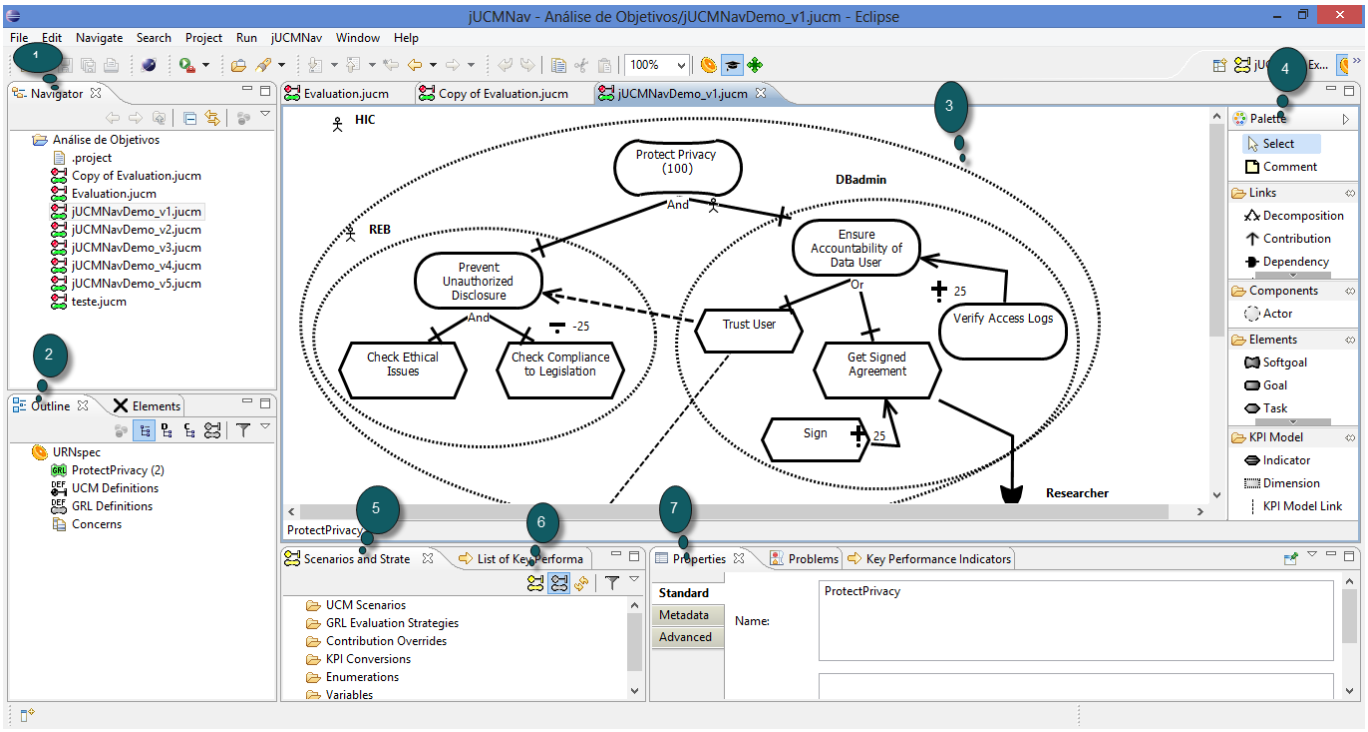


Figura - Criação do Modelo GRL

A figura abaixo mostra visão geral da utilização do jUCMNav e seus principais conceitos:

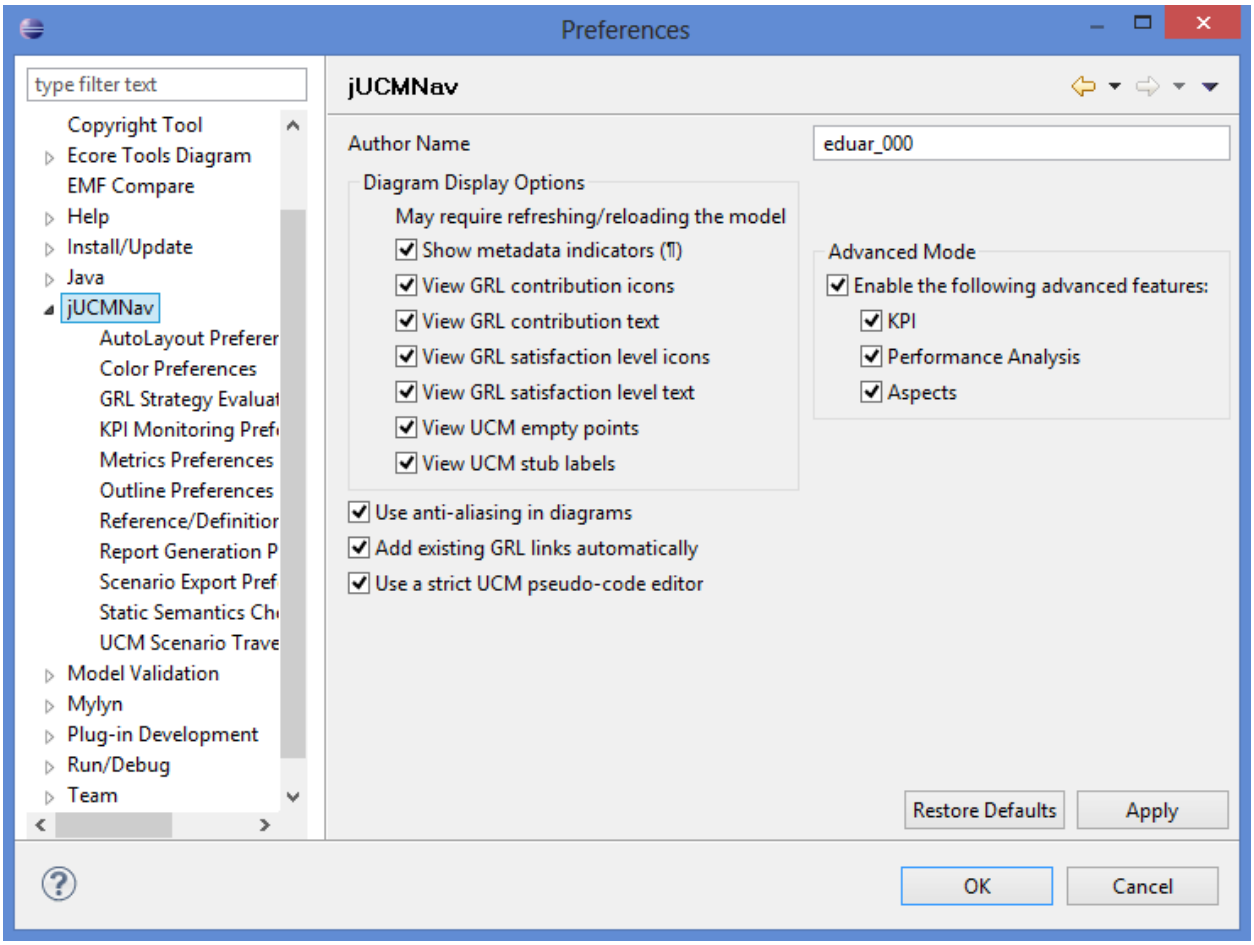


A tela principal é dividida em 7 áreas principais. São elas:

- Área 1: Aba de Navegação – Nela são listados os arquivos .jucm (extensão do jCMNav) criados no projeto
- Área 2: Outline – Nesta aba pode-se visualizar um resumo do modelo, como por exemplo os conceitos GRL criados (elementos intencionais, atores)
- Área 3: Área visual do editor, onde o modelo pode ser visualizado
- Área 4: Menu com as ferramentas necessários para se criar o modelo
- Área 5: Cenários do Modelo: Ao clicar em um cenário e escolher a opção “Switch to Execution View”
- Área 6: Lista dos Indicadores (KPIS)
- Área 7: Lista de Propriedades: Nesta área poderá ser modificado os valores de um elemeno (tipo de decomposição, tipo de elemento, evaluation, importance).

4.4 Configurações

Nas preferências da ferramenta é possível editar preferências relacionadas ao design do modelo, escolher algoritmos de avaliação a serem usados no modelo GRL, configurar o web service responsável por alimentar os dados de um indicador e outros. A figura abaixo nos mostra o painel de preferências:



Para editar o algoritmo de avaliação do modelo GRL, basta clicar em *GRL Strategy Evaluation*. Para utilizar um webservice, como o Cognos por exemplo, deverá colocar a sua referência na guia KPI Monitoring Preference.

4.5 Guia Básico de Utilização

Este documento não contemplará o guia básico para utilizar a ferramenta, pois concluímos que o website do jUCMNav possui ampla documentação e vídeos. Mostrar um guia como este seria redundância de artigo e o foco deste artefato é apresentar a linguagem GRL e sua ferramenta, deixando a cargo do leitor o aprofundamento na ferramenta. Segue abaixo boas referências (ambas do website oficial da ferramenta) para serem estudadas e que auxiliam na utilização do jUCMNav:

- jUCMNav Website
- Introdução a GRL no jUCMNav (vídeo)
- Utilizando KPI e webservice no jUCMNav (vídeo)

5. Discussão

Tendo em vista tudo que foi abordado, percebe-se que o uso de modelagem de objetivos de forma isolada deixa margem para uma dificuldade de compreensão exata do estado de uma organização, o que pode ser resolvido com a adição de indicadores. Nesse sentido, o entendimento do estado atual bem como o processo de tomada de decisão acaba sendo facilitado.

Referência Bibliográfica

AMYOT, D.; GHANAVATI, S.; HORKOFF, J.; MUSSBACHER, G.; PEYTON, L.; YU, E. Evaluating Goal Models within the Goal-Oriented Requirement Language. **INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS,** v. 25, p. 841 - 877, 2010.

BARONE, D.; JIANG, L.; AMYOT, D.; MYLOPOULOS, J. Reasoning with Key Performance Indicators. **The Practice of Enterprise Modeling,** v. 92, p. 82 - 96, 2011.

CHUNG, L.; NIXON, B. A.; YU, E.; MYLOPOULOS, J. Non-functional requirements in software engineering. Kluwer, 2000, Dordrecht, the Netherlands. p.412.

ITU, T. S. S. O. **SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS**. Formal description techniques (FDT) – User Requirements Notation (URN) 2011.

YU, E. S. K. Towards modelling and reasoning support for early-phase requirements engineering. In: ENGINEERING, R., Proceedings of the Third IEEE International Symposium on, 1997, Annapolis, MD. p.226 - 235.