Um Editor para a Linguagem de Especificação de Requisitos RELAX

Luiz Paulo Franz[1], Gabriel B. Moro[2], Rafael Torres[1], João Pablo S. da Silva[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Av.Tiarajú, 810, Alegrete, RS, Brasil

[2]Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500, Porto Alegre, RS, Brasil

{luizpaulofranz, gabrielbronzattimoro.es, torresrafa22, jpabloss}@gmail.com

Abstract. *Self-adaptive systems have unique characteristics that differentiate them from other types of systems. As the main one, we can highlight the situations of uncertainty that such systems must face. Because of this, the usual requirements engineering processes are not satisfactory in this context. One of the proposals to assist in the engineering of requirements for self-adaptive systems is the RELAX language, which aims to highlight the uncertainties inherent in this type of system. In this article, we present the development of a tool that supports the writing and editing of RELAX requirements, exploring its syntax and semantics, and facilitating the use of the language. As a result, we get a functional editor based on the eclipse platform.*

Resumo. *Sistemas autoadaptativos apresentam características únicas, que os diferenciam de outros tipos de sistemas. Como a principal delas, podemos destacar as situações de incerteza que tais sistemas devem enfrentar. Por conta disso, os processos usuais de engenharia de requisitos não são satisfatórios nesse contexto. Uma das propostas para auxiliar na engenharia de requisitos para sistemas autoadaptativos é a linguagem RELAX, que tem como objetivo evidenciar as incertezas inerentes à esse tipo de sistema. Neste artigo, apresentamos o desenvolvimento de uma ferramenta que da suporte à escrita e edição de requisitos RELAX, explorando sua sintaxe e semântica e facilitando o uso da linguagem. Como resultado, obtemos um editor funcional baseado na plataforma eclipse.*

1. Introdução

Sistemas autoadaptativos podem ser definidos como softwares capazes avaliar e mudar seu comportamento em tempo de execução com base nos estímulos que recebem do ambiente e seu estado atual [Macías-Escrivá et al. 2013]. [Whittle et al. 2010] comentam que esses sistemas atuam em condições e contextos diversos, o que dificulta a previsão dos estados pelos quais esse software vai passar ao longo de seu ciclo de vida.

Engenharia de requisitos é um conjunto de processos que tem como objetivo elicitar e avaliar os requisitos que motivam um projeto. De acordo com

[Sommerville et al. 2007], através da engenharia de requisitos, é possivel manter um artefato de documentalção de requisitos, que deve ser seguido pela equipe de desenvolvimento durante a implantação do sistema.

Considerando as situações de imprevisibilidade que os sistemas autoadaptativos enfrentam, fica claro que as abordagens convencionais de engenharia de requisitos não são adequadas para lidar com tais sistemas. Tendo isso em vista, [Whittle et al. 2010] propuseram a linguagem RELAX, com o objetivo de elicitar os requisitos de um sistema autoadaptativo evidenciando suas características de incerteza.

Neste trabalho, temos como objetivo apresentar uma ferramenta para especificação de requisitos que dê suporte a linguagem RELAX, explorando seus aspectos sintáticos e semânticos, tornando seu uso mais transparente.

Este artigo está organizado da seguinte maneira: no capítulo 2 apresentamos os trabalhos relacionados, no capítulo 3 apresentamos em detalhes a linguagem RELAX; no capítulo 4 realizamos a descrição do desenvolvimento da ferramenta; no capítulo 5, a validação da ferramenta, e por fim, no capítulo 6, a conclusão do trabalho.

2. Trabalhos Relacionados

Nesta seção, falaremos de trabalhos que descrevem ferramentas de apoio à engenharia de requisitos para sistemas autoadaptativos propostas por outros autores.

[Ali et al. 2013] propõem um mecanismo de mapeamento da especificação de requisitos em um modelo de metas para identificar inconsistências e requisitos conflitantes de acordo com contexto. Esse mecanismo é acrescentado como funcionalidade para a ferramenta CASE RE-*Context*, desenvolvida pelos autores em um trabalho anterior.

[Hussein et al. 2013] apresentam uma ferramenta de validação de requisitos automatizada baseada em cenários através do processamento da especificação das variantes dos requisitos levantados (características, estado, e assim por diante). A partir disso, a ferramenta gera automaticamente scripts de acordo com o cenário de entrada e os verifica. Após isso, as inconscistências encontradas são apresentadas.

[Fredericks et al. 2014] relatam uma abordagem que automatiza a análise de modelos objetivos e utiliza como entrada modelos de metas construído em KAOS (Keep All Objectives Satisfied). A partir disso a ferramenta desenvolvida pelos autores gera possíveis soluções em modelos Relax, onde contém operadores da própria linguagem como também funções que caracterizam o objetivo definido no modelo de entrada.

[Moro 2015] apresenta a primeira versão da RelaxEditor, com a proposta de uma ferramenta que dê suporte à requisitos RELAX. No entanto, optamos por re-desenvolver a ferramenta utilizando uma versão mais atual da plataforma Eclipse, que oferece uma série de recursos: *syntax high-light*, *auto-complete*, *hovering*, entre outros.

3. Linguagem RELAX

Apresentada por [Whittle et al. 2010], a linguagem RELAX é uma linguagem natural estruturada que tem como objetivo evidenciar características de incerteza, que por sua vez, são inerentes à sistemas autoadaptativos. A RELAX utiliza operadores para organizar o requisito de forma mais formal, porém ainda com trechos em linguagem natural. A

classificamos, portanto, como linguagem semi-formal. Os operadores RELAX são apresentados na Tabela 1 e a semântica na Tabela 2.

Table 1. O vocabulário da Linguagem RELAX.

Operadores	Descrição
Operadores Modais	
SHALL	Um requisito deve ser fornecido/mantido.
MAY... OR MAY	Um requisito especifica uma ou mais alternativas.
Operadores Temporais	
EVENTUALLY	Um requisito deve ser fornecido/mantido eventualmente.
UNTIL	Um requisito deve ser fornecido/mantido até uma posição futura.
[BEFORE — AFTER]	Um requisito deve ser fornecido/mantido antes ou depois de um evento em particular.
IN	Um requisito deve ser fornecido/mantido em/durante um particular intervalo de tempo.
AS [EARLY — LATE] AS POSSIBLE	Um requisito especifica algo que deveria ser fornecido/mantido tão cedo quanto possível ou deveria ser postergado por tanto tempo quanto possível.
AS CLOSE AS POSSIBLE [frequência]	Um requisito especifica algo que acontece repetidamente mas a frequência pode ser relaxada tão próxima quanto possível de uma frequência.
Operadores Ordinais	
AS CLOSE AS POSSIBLE [quantidade]	Um requisito especifica uma quantidade contável, mas o número exato pode ser relaxado tão próxima quanto possível de uma quantidade.
AS [MANY — FEW] AS POSSIBLE	Um requisito especifica uma quantidade contável, mas o número exato pode ser relaxado o maior número possível ou o menor número possível.
Fatores de Incerteza	
ENV	Define um conjunto de propriedades que definem o ambiente do sistema.
MON	Define um conjunto de propriedades a serem monitoradas pelo sistema.
REL	Define a relação entre as propriedades de ENV e MON.
DEP	Identifica as dependências entre os requisitos relaxados e invariantes.

O exemplo apresentado na Tabela 3 ilustra um requisito em RELAX. Podemos perceber nele as características básicas da linguagem, tais como operadores e fatores de incerteza. Também podemos perceber que a especificação do requisito se torna mais expressiva, mesmo sendo escrita em uma estrutura declarativa e em linguagem natural.

4. Desenvolvimento do RelaxEditor

Na Figura 1, apresentamos um *screenshot* da tela do RelaxEditor. Por ter sido desenvolvido na plataforma Eclipse, a interface se assemelha à IDE Eclipse. Na imagem temos os arquivos no formato *.relax* no canto esquerdo, onde são escritos os requisitos, no centro o conteúdo do arquivo (editor), e a direita, a expressão resultante do requisito *tractor01*.

4.1. Xtext

Para realizarmos o desenvolvimento da ferramenta, optamos pela utilização do Xtext. Xtext é um framework pertencente à plataforma Eclipse para o desenvolvimento de *Domain Specific Language* (DSL), bem como linguagens de propósito geral

Table 2. A semântica da Linguagem RELAX.

Expressão RELAX	Formalização FBTL
SHALL ϕ	AG ϕ
MAY ϕ1 OR MAY ϕ2	AG (ϕ_1 ou ϕ_2)
EVENTUALLY ϕ	A F ϕ
ϕ_1 UNTIL ϕ_2	A(ϕ_1 UNTIL ϕ_2)
BEFORE *e* ϕ	A$\chi <_{e_d}\phi$ onde e_d é a duração até a próxima ocorrência de e
AFTER *e* ϕ	A$\chi >_{e_d}\phi$
IN t ϕ	(AFTER $t_{começo}$ $\phi \bigwedge$ BEFORE t_{fim} ϕ) onde $t_{começo}$, t_{fim}, são eventos que denotam o início e o fim do intervalo *t*, respectivamente
AS EARLY AS POSSIBLE ϕ	A$\chi_{\geq_d}\phi$ onde *d* é a duração difusa definida de tal forma que seus membros de função tem o seu máximo em 0 (ou seja, M(0)= 1) e diminui continuamente para os valores >0
AS LATE AS POSSIBLE ϕ	A$\chi_{\geq_d}\phi$ onde *d* é a duração difusa definida de tal forma que seus membros de função tem o seu valor mínimo em 0 (ou seja, M(0)= 0) e aumenta continuamente para os valores >0
AS CLOSE AS POSSIBLE TO *f* ϕ	A ($\chi=_d\phi \bigwedge \chi=_{2d}\phi \bigwedge \chi=_{3d}\phi \bigwedge$...) onde *d* é a duração difusa definida de tal modo que a sua função de adesão tem o seu valor máximo, no período definido por *f* (ou seja, M(d)= M(2d)=...=1) e diminui continuamente para valores inferiores e superiores a *d* (e 2*d*, ...)
AS CLOSE AS POSSIBLE TO *q* ϕ	AF(($\Delta(\phi)$-q) ϵ S) pertence a S, onde S é um conjunto fuzzy cuja função de composição tem um valor no zero (M(0)= 1) e diminui continuamente em torno de zero. $\Delta(\phi)$ "conta"o quantificável, que será comparado com *q*
AS MANY AS POSSIBLE ϕ	AF ($\Delta(\phi)$ ϵ S), onde S é um conjunto fuzzy cuja função de pertinência tem 0 valor no zero(M(0)= 0) e aumenta continuamente em torno de zero
AS FEW AS POSSIBLE ϕ	AF ($\Delta(\phi)$ ϵ S), onde S é um conjunto fuzzy cuja função de pertinência tem um valor no zero(M(0)= 1) e diminui continuamente em torno de zero

Table 3. Exemplo de requisito escrito usando a Linguagem RELAX

O refrigerador DEVE gerenciar o plano de dieta de forma que esse mantenha o consumo de calorias TÃO PRÓXIMO QUANTO POSSÍVEL do consumo diário ideal.

ENV:	Alimentos.
MON:	Leitores RFID; Sensores de Peso.
REL:	Leitores RFID monitoram os Alimentos para prover informações sobre propriedades calóricas; Sensores de Peso monitoram os Alimentos para prover informação sobre consumo de calorias.
DEP:	Não se aplica.

[Foundation 2017]. Além da facilidade que o framework oferece para desenvolver linguagens, ele ainda oferece recursos avançados de forma simples, sendo alguns deles: Syntax Highlight, evidenciando palavras reservadas, comentários e erros de sintaxe; Con-

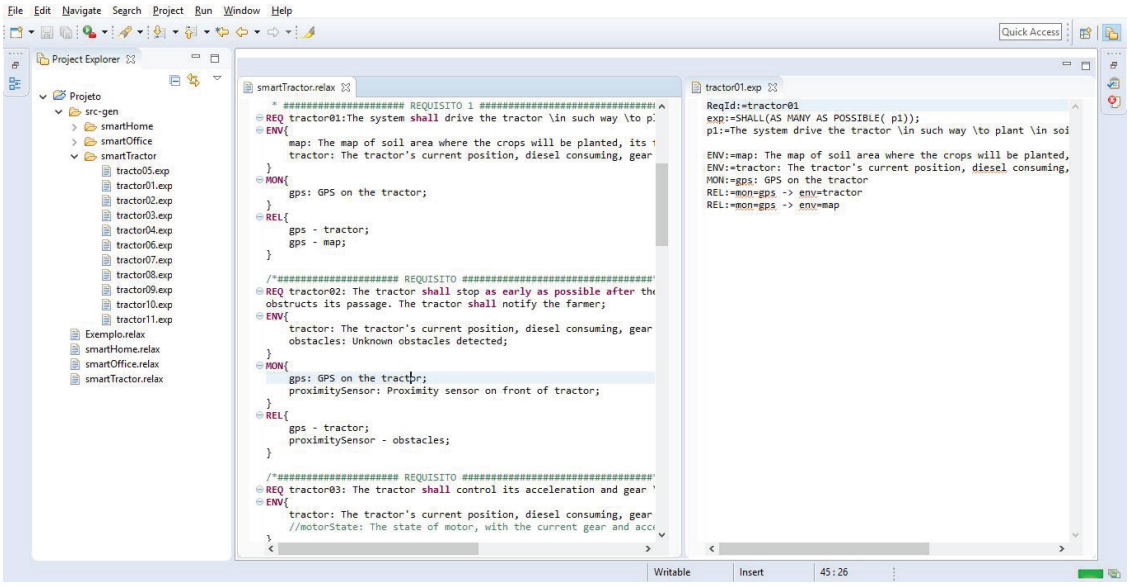

Figure 1. Screenshot da tela do RelaxEditor

tent assist, opção de auto-complete e Hyperlinking, possibilitando saltar para a declaração de algum bloco ou variável.

4.2. Desenvolvimento da gramática

Para o desenvolvimento da ferramenta, estendemos a gramática original da Linguagem RELAX e adicionamos alguns operadores: o operador REQ para demarcar o início de um novo requisito; os operadores ENV, MON, REL e DEP que, por mais que tenham sido apresentadas no artigo de [Whittle et al. 2010], não haviam sido formalmente definidos.

Como terminal da Linguagem RELAX utilizado para o desenvolvimento da ferramenta, temos o conjunto T. Nele, temos os operadores da linguagem definidos, o terminal ID, que representa os caracteres alfanuméricos para os trechos em linguagem natural, e o simbolo ϕ representando uma expressão RELAX.

T = {REQ, ID, ϕ, true, false, SHALL, MAY... OR MAY..., EVENTUALLY, UNTIL, BEFORE, AFTER, IN, AS CLOSE AS POSSIBLE TO, AS CLOSE AS POSSIBLE TO, AS {EARLY; LATE; MANY; FEW} AS POSSIBLE, ENV, MONS, RELS, DEP}

Como variáveis da gramática, temos V = {ϕ, ϕ1, ϕ2, e, f, q, t, p}, onde ϕ representa uma expressão RELAX (ou seja, tratando a recursividade da linguagem), *e* representa um evento, *f* representa frequência, *q* representa quantidade, *t* representa tempo e *p* representa uma frase em linguagem natural.

Para a regra maior da linguagem, definimos que ela será composta pelo operador REQ (ou req), um identificador único para o requisito, uma sentença RELAX, suas propriedades, e encerrará com ponto e virgula. Temos, portanto, a seguinte regra de construção:
P = {S→REQ p: ϕ ; Prop | true | false,
Prop→ ENV | MON | ENV | ENV MON REL | ENV MON REL DEP | DEP | ENV DEP | MON DEP | ε,
ENV→ ENV { (p : p;)$^+$ } ,

MON→ MON { (p : p;)$^+$ } ,
DEP→ DEP: p; | (p,)$^+$ p; ,
REL→ REL { (p : p;)$^+$ }}

4.3. Geração de expressões

Segue um trecho de código para exemplificar como o Xtext nos permite manipular informações:

```
def dispatch compileRest(ShallOperator s) {
this.operators += "SHALL("
s.elements.compileGeneral
}

def dispatch compileRest(EventuallyOperator e) {
this.operators += "EVENTUALLY("
e.element.compileGeneral
}

def dispatch compileRest(BeforeOperator b) {
this.operators += "BEFORE("
this.e = b.event
}
```

No trecho acima, os atributos *elements* e *event* que foram definidos nas regras da gramática representam o texto em linguagem natural que vem logo após os operadores, e a partir daí, montamos as expressões da maneira que desejamos.

A seguir, apresentamos um exemplo de requisito escrito utilizando a Linguagem RELAX e depois a expressão que é gerada a partir dele.

```
REQ prenat01 : The smartphone SHALL send a help message \to
another authorized user when the pregnant woman is \in a
hospital ;
ENV { currentPosition : Current position of the pregnant woman ; }
MON {gps: The pregnant woman 's smartphone GPS ;}
REL {gps - currentPosition ;}
```

Expressão gerada:

```
ReqId := prenat01
exp := SHALL ( p1);
p1 := The smartphone send a help message \to another authorized
user when the pregnant woman is \in a hospital
ENV := currentPosition : Current position of the pregnant woman
MON := gps: The pregnant woman 's smartphone GPS
REL := mon=gps -> env= currentPosition
```

5. Testes

Para realizarmos os testes da ferramenta, desenvolvemos testes baseados em contextos distintos e cobrindo várias das possibilidades oferecidas pela linguagem RELAX, bem

Table 4. Resultado dos testes

Cenário de teste	Testes	Erros
SmartCampus	12	1
SmartCar	12	1
SmartHome	6	0
SmartTractor	11	0
Total	41	2

como todos os operadores que ela apresenta junto às suas propriedades. Na Tabela 4 podemos observar os resultados dos testes elaborados.

Levando em conta os casos testados, obtivemos uma baixa quantidade de erros, sendo que todos eles estão relacionados ao operador *MAY... OR MAY...* .

Os requisitos que apresentaram mau funcionamento são apresentados abaixo.

```
REQ campus02: The app MAY notify students and teachers when
a class \or meeting time approaches OR MAY register
the student attendance \in case he's already \in
the classroom;

ENV{smartphone : The student's smartphone that is running
the app;
people : Students and teachers;
}
MON{mainSystem : the SmartCampus main system;
roomSensor : The sensor that is going \to monitor \and
    identify people;
}
REL{mainSystem - smartphone;
roomSensor - people;
}

REQ car02: The Smartcar MAY turn on air conditioner OR MAY
open the windows;

ENV{temperature: Smartcar's internal temperature;}

MON{temperatureSensor: Thermometer \to monitoring the
Smartcar's temperature;}

REL{temperatureSensor-temperature;}
```

Ambos os erros ocorrem por erro de sintaxe, ressaltados pelo recurso de *syntax highlight*, como pode ser visto na Figura 2.

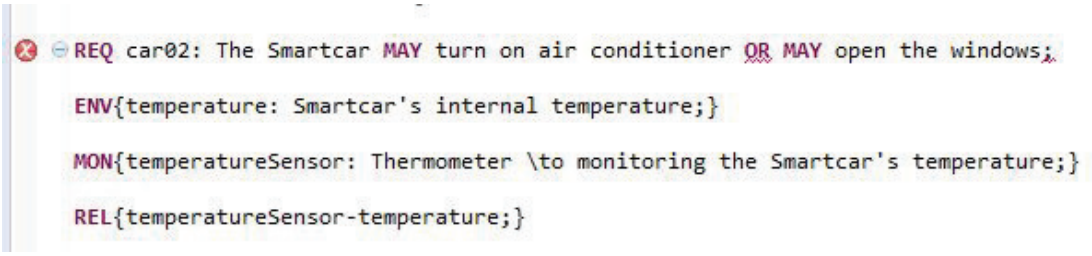

Figure 2. Screenshot de um erro de sintaxe sendo apontado pela ferramenta.

6. Conclusão

O principal objetivo deste trabalho é criar uma ferramenta para elicitação de requisitos com suporte à linguagem RELAX proposta por [Whittle et al. 2010]. Baseado no que foi apresentado, podemos concluir que cumprimos o objetivo de produzir tal ferramenta. Com ela podemos escrever sentenças complexas utilizando os operadores RELAX e extrair as expressões formais delas. Entretanto, a ferramenta ainda apresenta limitações relacionadas ao operador *MAY... OR MAY...*. Como trabalhos futuros, propomos a melhoria da ferramenta em relação ao operador que apresenta problema e ampliar ainda mais os cenários de teste para aumentar a confiabilidade na ferramenta.

References

[Ali et al. 2013] Ali, R., Dalpiaz, F., and Giorgini, P. (2013). Reasoning with contextual requirements: Detecting inconsistency and conflicts. *Information and Software Technology*, 55(1):35–57.

[Foundation 2017] Foundation, E. (2017). Xtext - reference documentation. Disponível em: `https://eclipse.org/Xtext/documentation/index.html`. Acesso em: 02 de junho de 2017.

[Fredericks et al. 2014] Fredericks, E. M., DeVries, B., and Cheng, B. H. (2014). Autorelax: automatically relaxing a goal model to address uncertainty. *Empirical Software Engineering*, 19(5):1466–1501.

[Hussein et al. 2013] Hussein, M., Han, J., Yu, J., and Colman, A. (2013). Scenario-based validation of requirements for context-aware adaptive services. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 348–355. IEEE.

[Macías-Escrivá et al. 2013] Macías-Escrivá, F. D., Haber, R., Del Toro, R., and Hernandez, V. (2013). Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications*, 40(18):7267–7279.

[Moro 2015] Moro, G. B. (2015). Uma ferramenta de apoio à especificação de requisitos para sistemas autoadaptativos.

[Sommerville et al. 2007] Sommerville, I. et al. (2007). *Engenharia de software*, volume 8. Addison Wesley São Paulo.

[Whittle et al. 2010] Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H., and Bruel, J.-M. (2010). Relax: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering*, 15(2):177–196.