INTEGRANDO O FRAMEWORK I* AO GUIA PARA ELICITAÇÃO DE REQUISITOS EM SISTEMAS EMBARCADOS (GERSE)

INTEGRATING THE I* FRAMEWORK TO THE REQUIREMENTS ELICITATION GUIDE IN EMBEDDED SYSTEMS (GERSE)

SILVA, Maykon Valério[1]; **SILVA**, Alexandre Luiz de Borba[2]; **SANTANDER**, Victor Francisco Araya[3]; **SILVA,** Ivonei Freitas da[3] ; **SCHEMBERGER**, Elder Elisandro[4]

[1] Pós-graduando em Projetos de Sistemas Embarcados pela Universidade Positivo (UP)
maykonvs@gmail.com

[2]Acadêmico do Curso de Ciência da Informação na UNIOESTE, Câmpus Cascavel
albsilva@outlook.com

[3] Professor Adjunto da UNIOESTE, Câmpus Cascavel
victor.santander@unioeste.br
ivonei.silva@unioeste.br

[4] Professor Assistente da Universidade Tecnológica Federal do Paraná - UTFPR, Câmpus Toledo
elderes@gmail.com

Resumo

Observa-se atualmente uma carência de metodologias e técnicas de engenharia de requisitos voltadas aos sistemas embarcados. Uma das iniciativas visando preencher esta lacuna é o Guia para Elicitação de Sistemas Embarcados (GERSE). Contudo, este guia pode ser consideravelmente melhorado no que tange a sua aplicação e documentação, se utilizarmos outras perspectivas não contempladas no processo original tais como: o uso de modelos organizacionais como proposto pelo Framework i* e uso do NFR-Framework para apoiar o processo de escolha de alternativas de operacionalização de requisitos não-funcionais. Desta forma, considerando as especificidades de sistemas embarcados, propõe-se neste trabalho a integração destes dois Frameworks ao processo proposto pelo GERSE. Para este fim, algumas diretrizes são propostas e utilizadas em um exemplo de um relógio digital de xadrez.

Palavras-chave: engenharia de requisitos, sistemas embarcados, gerse, i*, nfr-framework.

Abstract

There are few works investigating requirements engineering techniques and methodologies for embedded systems. One of these works is the GERSE (guide to requirements elicitation for embedded systems). However, this guide can be better by adopting others perspectives do not considering in the original proposal such as: using organizational models generated by the i* Framework and to use the NFR-Framework to assist the choose of the non-functional requirements operationalization alternatives. Thus, by considering the embedded system characteristics, in this work we propose the integration of i* and NFR Frameworks to the GERSE. To assist this integration, some guidelines are proposed and applied to the chess clock case study.

Key-words: requirements engineering, embedded systems, gerse, i*, nfr-Framework.

INTRODUÇÃO

Nos últimos anos tem se destacado o desenvolvimento de Sistemas Embarcados (SE), os quais devem satisfazer requisitos muito específicos tanto de hardware quanto de software.

As comunidades da computação e engenharias têm proposto a incorporação de ferramentas para especificar os requisitos funcionais e não-funcionais de SE, de forma a permitir a sua posterior verificação e validação (ESPINOZA; SERVAT; GERARD, 2008).

Porém, (SIKORA;TENBERGEN;POHL, 2012) relatam as dificuldades de adoção que a indústria de software enfrenta em relação às técnicas propostas pela comunidade acadêmica, muitas vezes por serem de difícil aplicação e/ou por não permitirem expressar todos os aspectos relevantes associados ao domínio de SE.

Uma proposta que esboça uma solução para estes problemas é o GERSE (Guia de Elicitação de Requisitos para Sistemas Embarcados) (OSSADA; MARTINS, 2010), o qual consiste de um guia com diversos passos e tarefas que ao final resultam nos requisitos necessários para o SE.

Contudo, o GERSE não especifica técnicas ou Frameworks para auxiliar os profissionais envolvidos na elicitação dos requisitos, determinando apenas o que elicitar e não como.

Assim, um processo, que incorporado ao GERSE determine meios de obter os requisitos e informações permitindo representar as necessidades dos stakeholders de forma clara e considerando todos os elementos do ambiente organizacional, traria um ganho significativo ao guia, motivando profissionais da área a segui-lo.

Para este fim, propõe-se neste trabalho o uso do framework de modelagem organizacional i* (YU, 2011), considerando que o mesmo permite representar atores, dependências (requisitos) entre os mesmos bem como a possibilidade de refinar a satisfação das dependências.

Também consideramos que é necessário dar uma atenção especial a operacionalização dos requisitos não-funcionais em SE's. Neste sentido, propõe-se o uso do NFR-Framework (CHUNG, 1999) para apoiar o processo de escolha das melhores alternativas para a construção do SE, considerando os requisitos a serem satisfeitos.

Este trabalho está estruturado conforme segue. Na seção 2 é realizada uma breve descrição do GERSE e são apresentados brevemente os Frameworks I* e o NFR-Framework. Na seção 3, a proposta deste trabalho é descrita e aplicada a um cenário de um relógio digital de xadrez. Finalmente, na seção 4 são realizadas as considerações finais do trabalho.

GERSE E OS FRAMEWORKS DE APOIO PROPOSTOS

O GERSE, apresentado em (OSSADA; MARTINS, 2010), é dividido em duas fases: a pré-fase e a fase principal, conforme apresentado na Figura 1. A pré-fase consiste de tarefas organizacionais e mercadológicas, onde há uma elaboração geral do produto e a sua posição frente

ao mercado.

Já a fase principal consiste das tarefas que resultarão nos requisitos técnicos propriamente ditos, juntamente com uma documentação dos mesmos. Para apoiar algumas das atividades deste Guia, propõe-se neste trabalho a utilização do Framework i* e do NFR-Framework.

O Framework i* permite descrever aspectos de intencionalidade e motivações envolvendo atores em um ambiente organizacional. Para descrever estes aspectos são propostos dois modelos:

O Modelo de Dependências Estratégicas (SD) e o Modelo de Razões Estratégicas (SR). O Modelo SD é composto entidades que realizam ações para obter objetivos no contexto do ambiente organizacional, denominadas atores.

Esses atores dependem uns dos outros para atingir objetivos, realizar tarefas, e obter recursos no ambiente organizacional. O modelo SR é um modelo complementar ao modelo de dependências estratégicas.

Este modelo permite compreender e modelar de forma mais detalhada as razões associadas com cada ator e suas dependências.

Este modelo auxilia no processo de Engenharia de Requisitos, permitindo que elementos de processos e as razões por detrás dos mesmos sejam expressos (YU, 2011).

O NFR-Framework visa representar requisitos não-funcionais, os quais são vistos como softgoals (objetivos que não possuem um critério bem definido para decidir se sua condição é alcançada).

Este Framework usa os requisitos não-funcionais para guiar todo o processo de projeto e oferece uma estrutura SIG (Grafos de Interdependência de Softgoals) para descrever a interdependência entre softgoals e como os mesmos são decompostos, (CHUNG, 1999).

Figura 1 – Fases, atividades e tarefas do GERSE.

PROPOSTA

Esta proposta consiste de quatro passos contendo diretrizes que orientam a execução de cada passo. O Passo 1 visa identificar, por meio de um modelo SD, os usuários e componentes que terão comunicação com o SE, os ambientes em que o SE será utilizado e os requisitos que todos estes possuem sobre o SE.

O Passo 2 busca a tradução dos requisitos do Passo 1, que estão em alto nível, para requisitos de baixo nível, como requisitos mecânicos, de hardware e software. Isto é realizado através do refinamento do modelo SD do Passo 1 em um modelo SR.

O Passo 3 visa trabalhar com requisitos de qualidade (não-funcionais) de forma a satisfazê-los e definir completamente o SE, sendo que para este fim é utilizada a abordagem proposta pelo NFR-Framework.

E finalmente no Passo 4, busca-se elicitar os requisitos de produção e desenvolvimento do SE, por meio de um SR.

Para facilitar o entendimento da proposta, para cada diretriz será demostrado um exemplo de aplicação da mesma. O exemplo consiste da elicitação de requisitos para um relógio digital de xadrez, que tem como função principal marcar o tempo entre as jogadas de um jogo de xadrez, de modo a estabelecer um limite de tempo máximo entre cada uma das jogadas.

Esta aplicação foi inicialmente apresentada em (OSSADA; MARTINS, 2010) e adaptada para a presente proposta.

Passo 1 – Construir o modelo SD do produto inserido em seu contexto de uso

A construção do SD do produto inserido em seu contexto de uso visa mapear a interação do produto com todos os atores organizacionais externos ao mesmo, satisfazendo as necessidades que os atores do ambiente organizacional possuem em relação ao produto. Neste Passo, são elicitados e representados na forma de atores, os usuários e componentes externos ao SE, e na forma de dependências, os requisitos funcionais, os não-funcionais, requisitos de ambiente e o tipo de alimentação do sistema embarcado pretendido, cobrindo as tarefas 2.4, 3.1, 3.2, 3.3, 3.4 e a atividade 6 propostas pelo GERSE (ver fig. 1).

Diretriz 1: Inserir o produto como um ator no modelo. Inserir o produto como um ator no SD permite analisar toda a interação deste com os diversos outros atores que serão inseridos nos Passos posteriores. Exemplo: No caso do relógio digital de xadrez, insere-se no modelo um ator com o nome “Relógio digital de xadrez”. A Figura 2 apresenta este ator inserido no modelo.

Diretriz 2: Identificar os usuários do produto e mapeá-los como atores. Nesta diretriz deve-se pensar em quais são os usuários do produto, pois cada um deles terá requisitos (dependências) diferentes sobre o produto, e este deve satisfazer essas necessidades. Exemplo: Os usuários do relógio digital de xadrez são: jogador de xadrez e o jogador profissional de xadrez. Portanto são inseridos dois novos atores com

o nome de cada um desses usuários (ver figura 2).

Diretriz 3: Identificar outros componentes que terão interface com o produto. Esta diretriz busca satisfazer os projetos nos quais pretende-se desenvolver um produto que integre uma rede de outros produtos. Estes componentes também serão mapeados como atores no modelo, mapeando a interação entre os mesmos. Exemplo: Algo comum em SE's é a utilização de computadores para a atualização do firmware do dispositivo. O computador é um novo ator no modelo, como apresentado na Figura 2.

Diretriz 4: Identificar os requisitos funcionais do produto para cada um dos usuários. Estes serão os objetivos do usuário (depender) em relação ao produto (dependee). Esta diretriz auxilia o profissional da área na elicitação propriamente dita dos requisitos funcionais. A análise aqui acontece em alto nível, e consiste em elicitar e representar as funções que o produto deve ter para satisfazer as necessidades de seus usuários. Exemplo: O jogador depende do relógio para iniciar e parar a contagem de tempo a cada turno do jogador, e também para dar bônus de tempo, nos casos em que isso for aplicável. Sendo assim, insere-se dois objetivos representando esses requisitos funcionais. Já o jogador profissional deseja que as regras de campeonatos oficiais já estejam pré-cadastradas no dispositivo, sendo assim, um novo objetivo no modelo, porém agora entre o jogador profissional e o SE. Estas dependências do tipo objetivo podem ser visualizadas na Figura 2.

Diretriz 5: Identificar e mapear os requisitos (dependências) entre os componentes (identificados na diretriz 3) e o SE. Exemplo: Como mencionado no exemplo da diretriz 3, uma situação comum é a utilização de um computador para atualização de firmware do SE. A atualização de firmware é um objetivo do SE (ver Figura 2), que depende do computador para ser satisfeito.

Diretriz 6: Mapear os ambientes como Agentes/Papéis e seus requisitos com o SE. O ambiente é algo de muita influência em um SE, pois o funcionamento adequado deste, depende fortemente do reconhecimento das características do ambiente onde o SE atuará. O SE atua sobre o ambiente através de atuadores, respondendo aos estímulos lidos por sensores. Já o ambiente expõe o SE às diversas situações sobre as quais este deve manter-se em funcionamento. São estas situações que esta diretriz busca elicitar. Para este fim, propõe-se as subdiretrizes abaixo:

Subdiretriz 6.1: Inserir um novo papel chamado "Ambiente". Este papel será ocupado por diversos agentes que serão os diversos ambientes em que o produto será usado.

Subdiretriz 6.2: Identificar os diversos ambientes em que o produto será usado, e colocá-los como agentes no modelo. Cada um destes agentes terá uma relação de plays com o papel Ambiente. Com isso, serão identificados todos os ambientes de uso do produto, e então mapeados para o modelo. Estes serão os ambientes que irão impor situações e restrições ao SE. Exemplo: Os ambientes de uso

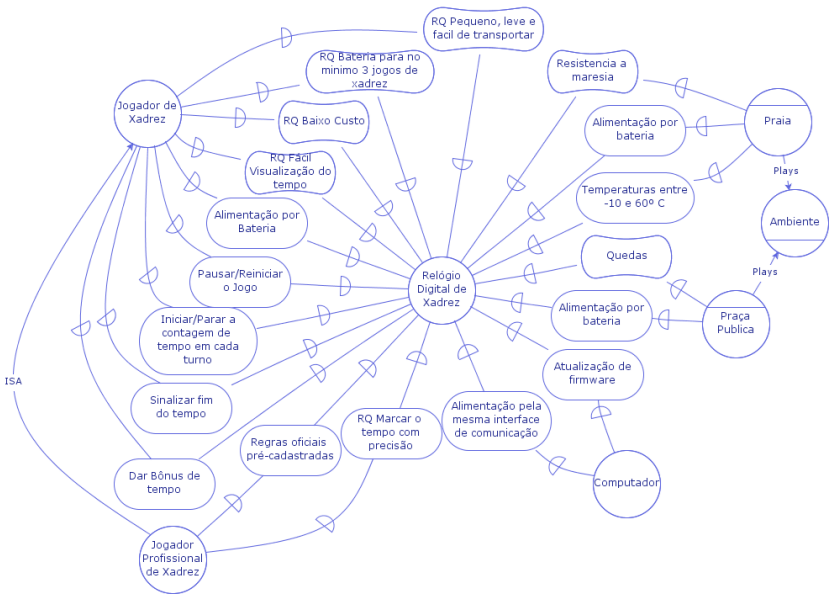
de um relógio de xadrez são os mesmos em que acontecem as partidas de xadrez, como por exemplo a praia e praças públicas (ver Figura 2).

Subdiretriz 6.3: Definir para cada um dos agentes da diretriz anterior, as restrições físicas do ambiente que podem ser impostas ao produto, como: temperatura, humidade, resíduos, vibrações, etc. Nesta subdiretriz são elicitados os requisitos do ambiente. Se existe uma forte carga de subjetividade em relação ao grau de satisfação de um requisito, deve-se mapear o mesmo como softgoal. Contudo, se o requisito for bem determinado, e sua satisfação não inclui subjetividade, propõe-se mapeá-lo como objetivo. Exemplo: O relógio, ao ser usado na praia durante o dia, será exposto às temperaturas altas. Se a faixa de temperatura for determinada, o requisito é mapeado como um objetivo, mas se for mantido no âmbito qualitativo, tratando o requisito como "Temperatura alta" ele deve ser mapeado como um softgoal. O mesmo acontece nas praças públicas, onde existe um risco de queda que se deve considerar (ver Figura 2).

Diretriz 7: Definir para cada um dos atores e agentes elicitados se há um tipo de alimentação ideal para os mesmos. Com esta diretriz busca-se as formas de alimentação necessárias para o produto que geralmente limitam-se à bateria ou rede elétrica (tomadas). Exemplo: Começando pelos ambientes, que são os mais determinantes neste requisito de alimentação, a praia e a praça pública têm como alimentação ideal as baterias, pela falta de tomadas nesses ambientes. Já o ator computador, prefere que a alimentação aconteça através da mesma interface de comunicação, facilitando assim o uso dos dois em conjunto (ver Figura 2).

Diretriz 8: Identificar para cada usuário (ator ou agente descoberto nas diretrizes anteriores) os requisitos de qualidade, que são desejos do usuário em relação ao produto. Estes serão softgoals entre o usuário (depender) e o produto (dependee). Para

Figura 2 – Diretrizes do Passo 1 e o modelo SD do produto.



diferenciá-los de outros softgoals, usar o prefixo RQ para nomeá-los. Exemplo: Baixo Custo e Fácil Visualização do tempo são alguns exemplos de requisitos de qualidade para SE's (ver Figura 2).

Passo 2 – Construir o modelo SR do produto

O modelo SD gerado no Passo 1 possui todos os requisitos de alto nível, e o refinamento deste modelo em um SR mostra-se uma forma eficaz para transformar estes requisitos em requisitos de baixo nível, que são os requisitos de hardware, de software e da parte mecânica de um SE.

Este Passo tem como correspondentes no GERSE as atividades 4 (Definição de Hardware) e 5 (Definição de Software), juntamente com a tarefa 3.5, que visa a elicitação de requisitos mecânicos.

A vantagem da utilização deste modelo para a elicitação destes requisitos é que os mesmos vão sendo inseridos de acordo com o requisito funcional que eles devem satisfazer (presente no SD do Passo 1), facilitando a tradução destes (alto-nível) em requisitos de baixo nível, além da possibilidade de geração de alternativas que posteriormente podem ser escolhidas para melhor satisfazer outros requisitos relacionados (o que será melhor definido no Passo 3). A seguir são apresentadas as diretrizes e subdiretrizes para este Passo.

Diretriz 9: Abrir a região do ator que representa o SE, e iniciar o processo elicitação de requisitos de baixo nível. Na região de um ator são inseridos elementos representando o know-how deste ator. O know-how do SE consiste de suas tarefas, objetivos e recursos internos que deverão satisfazer as dependências das quais o mesmo é dependee. As subdiretrizes a seguir elicitam os requisitos de baixo nível e devem ser seguidas para cada uma das dependências do SE, com exceção dos requisitos de qualidade.

Subdiretriz 9.1: Criar uma tarefa nova, com o mesmo nome da dependência externa a ser satisfeita, e então ligá-la à mesma. Se o produto já possuir uma tarefa que satisfaça a dependência, basta ligar a dependência a esta tarefa sem a necessidade de criar uma nova. Exemplo: Na Figura 3 pode-se observar que, para satisfazer a dependência "Iniciar/Parar a contagem de tempo em cada turno", foi criado no ator "Relógio Digital de Xadrez" uma tarefa com o mesmo nome da dependência.

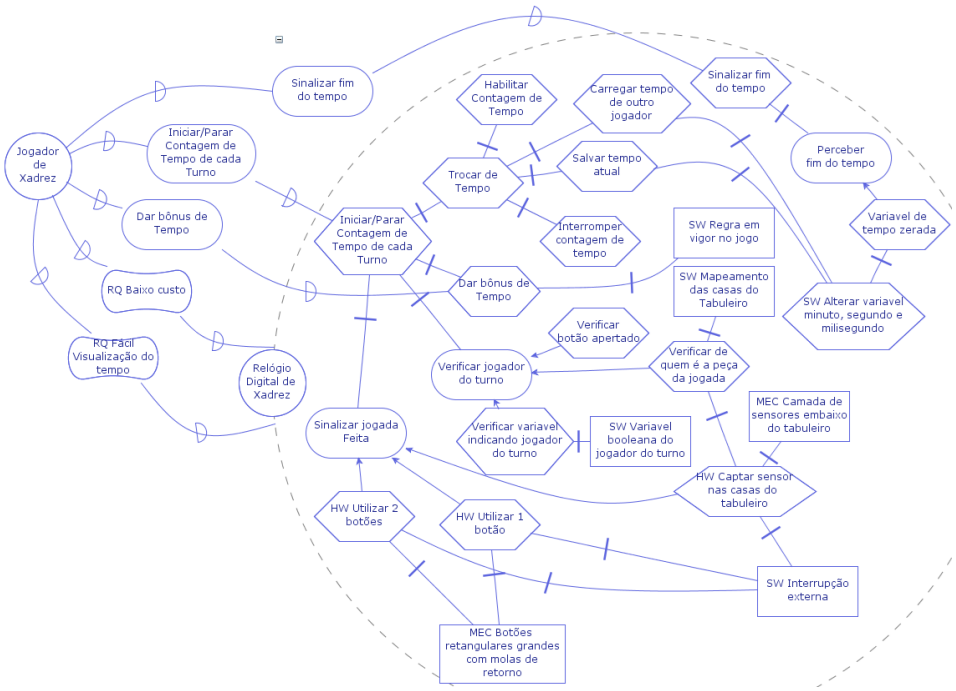
Subdiretriz 9.2: Dividir o novo elemento em subelementos (tarefas, objetivos ou recursos) necessários para satisfazê-lo, ligando-os através de task-decomposition ou, se são maneiras diferentes de se fazer algo, através de means-ends. Estes subelementos podem possuir novos subelementos, até que todas as atividades da tarefa estejam bem definidas. Uma tarefa pode ser considerada bem definida, quando já está claro como implementá-la. Exemplo: A tarefa gerada na subdiretriz anterior envolve quatro atividades: "Sinalizar jogada feita", "Verificar jogador do turno", "Dar bônus de tempo" e "Trocar de tempo". A primeira é uma espécie de "gatilho" que desencadeia a tarefa. A segunda serve de verificação, para não parar a contagem em horas

indevidas. A terceira é simplesmente a tarefa de dar o bônus de tempo quando isto se aplica, e finalmente, a quarta troca o contexto da contagem de tempo, e começa a contar o tempo do outro jogador (ver Figura 3). A tarefa "Dar bônus de tempo", satisfaz a dependência externa de mesmo nome, portanto, ao analisar esta dependência, não há a necessidade de criar uma nova tarefa, bastando ligar a dependência a esta tarefa.

Subdiretriz 9.3: Analisar para cada elemento interno ao SE obtido na subdiretriz anterior, a necessidade do mesmo ser resolvido por hardware ou por software. Para estes elementos, inserir tarefas ou objetivos representando a necessidade de obtenção destes recursos. Sempre pensar nesses recursos de hardware e software como meios para se atingir um fim, portanto sempre ligá-los utilizando a ligação do tipo means-ends. Isto favorece a criação de alternativas, melhorando assim o resultado do processo. Para diferenciar os recursos de hardware dos de software, utilizar o prefixo HW para os primeiros e SW para os outros. Um recurso de hardware consiste de um sensor, atuador ou circuito-integrado que realiza uma função específica. Um recurso de software pode ser uma variável, uma função, um driver, uma interrupção ou qualquer outro recurso implementável em software. A atividade que melhor demonstra esta subdiretriz é o objetivo "Sinalizar jogada feita". Este objetivo pode ser atingido através de três alternativas representadas pelas tarefas "HW: Utilizar 2 botões", ˜HW: Utilizar 1 botão e "HW: Captar sensor nas casas do tabuleiro", ligadas ao objetivo "Sinalizar jogada feita", via ligação means-ends, representando os recursos de hardware que podem ser utilizados para atingir o referido objetivo. A primeira e mais óbvia opção é a presença de dois botões, um para cada jogador, onde o jogador faz a sua jogada e o aperta, sinalizando o fim do seu turno. A segunda representa a opção de se ter apenas um botão, que serve para ambos os jogadores sinalizarem o fim de seus turnos. A terceira, e mais elaborada, é a de se ter sensores nas casas do tabuleiro, de modo a identificar movimentos nos peões e perceber quando uma jogada aconteceu automaticamente, sem a necessidade de outra ação do jogador (como apertar um botão, por exemplo) (ver Figura 3).

Subdiretriz 9.4: Analisar quais elementos internos ao SE, que refinam a dependência externa analisada, possuem um requisito mecânico. Estes requisitos serão representados como recursos no Framework i*. A representação deste tipo de recurso deve utilizar o prefixo MEC na sua nomenclatura. Um recurso mecânico descreve partes mecânicas necessárias, juntamente com características das mesmas. Exemplo: As duas tarefas "HW: Utilizar 2 botões" e "HW: Utilizar 1 botão" que representam a possibilidade de uso de dois recursos de hardware para satisfazer o objetivo "Sinalizar jogada feita", devem se levadas a cabo usando botões retangulares grandes, resistentes e com molas para retorno, sendo este um novo recurso mecânico. Já o recurso de hardware "HW: Captar sensor nas casas do tabuleiro" necessita que estes sensores estejam

Figura 3 - SR do produto em seu contexto de uso.



presentes em uma camada abaixo das casas do tabuleiro, sendo este um novo recurso mecânico.

Subdiretriz 9.5: Para cada elemento no modelo que representa um hardware, ou seja, possui o prefixo HW na sua nomenclatura, analisar os recursos mecânicos ou de software relacionados ao mesmo, sendo que estes serão novos recursos no modelo, ligados através de task-decomposition. Exemplo: Os três recursos de hardware elicitados na subdiretriz 9.3 ("HW: Utilizar 2 botões", "HW: Utilizar 1 botão" e "HW: Captar sensor nas casas do tabuleiro"), dependem de interrupções externas. Este tipo de interrupção é um recurso de software.

Passo 3 – Construção de um modelo NFR com requisitos de qualidade

O SR modelado no Passo 2 (ver figura 3) gera diversas alternativas para a realização de tarefas, evidenciadas por meio das ligações means-ends. O Passo 3 busca, fazendo uso do NFR, escolher entre essas alternativas. Este Passo diferencia do GERSE, pois o mesmo não incentiva a geração de alternativas, portanto não trabalha com as mesmas. A geração de alternativas aumenta a possibilidade de geração de requisitos que satisfaçam a maioria dos requisitos de qualidade, sendo este o objetivo principal deste Passo. Ao final deste Passo, também são definidos completamente os requisitos mecânicos e de hardware, complementando a atividade 4 (Definição de Hardware) do GERSE.

Diretriz 10: Colocar inicialmente, no primeiro nível do NFR, os softgoals de qualidade elicitados no modelo SD. Estes softgoals serão a base para as escolhas entre as alternativas. A Figura 4 mostra na parte superior o softgoal "RQ: Fácil visualização do tempo" que é um dos softgoals de qualidade inseridos no NFR vindos do SD do Passo 1 (ver Figura 2).

Diretriz 11: Juntar com estes, outros softgoals que representam características específicas de SE's. "Otimização do uso da memória", "Otimização do consumo de energia" e "Otimização da performance" são exemplos de características importantes no projeto de um SE, e estes poderiam ser softgoals inseridos no modelo NFR.

Diretriz 12: Colocar em um nível abaixo, como Sofgoals de Operacionalização, todos os meios das ligações means-ends do produto (ver diretriz 9.3). Agrupá-los de acordo com as tarefas que cada um deve desempenhar. Exemplo: Se dois recursos de hardware são alternativas para uma mesma tarefa, colocá-los um ao lado do outro, pois um dos mesmos deve ser escolhido.

Diretriz 13: Juntamente com os meios, também representar no NFR as tarefas, recursos e objetivos ligados aos mesmos no SR gerado no Passo 2. Cada um dos mesmos será um Sofgoal de Operacionalização com uma ligação help entre ele e o Sofgoal de Operacionalização que representa o meio. Exemplo: o recurso mecânico "MEC: Saída para o conector mini-USB" está ligado ao recurso de hardware "HW: USB" no SR do Passo 2 (ver Figura 3). Portanto, o mesmo é trazido ao NFR e ligado ao Sofgoal de Operacionalização "HW: USB" através" de uma ligação help.

Diretriz 14: Através dos softgoals e das ligações do NFR Framework, analisar quais Sofgoals de Operacionalização melhor satisfazem os requisitos de qualidade e os de SE's. Existem várias formas de se fazer esta análise. Para este fim sugere-se usar as seguintes subdiretrizes.

Subdiretriz 14.1: Escolher entre os requisitos de qualidade o considerado mais importante. Anotar este softgoal como prioritário usando o símbolo "!" proposto no NFR. Esta priorização significa que este softgoal tem prioridade na escolha dos Sofgoals de Operacionalização, de forma que satisfazê-lo é muito importante. Portanto, o mesmo será o ponto de partida da análise. Exemplo: O requisito "RQ: Fácil visualização de tempo", presente na Figura 4, foi escolhido como o mais importante, portanto marcado com o símbolo "!".

Subdiretriz 14.2: Se o softgoal não estiver muito claro, pode-se decompor o mesmo em outros softgoals mais refinados, utilizando as relações AND e OR. Um softgoal decomposto em outros dois com relações AND, significa que os dois softgoals precisam ser satisfeitos para o original também ser satisfeito. Se fossem ligações OR, qualquer um dos dois sendo satisfeitos, já satisfaz o original. Exemplo: O softgoal "RQ: Fácil Visualização do tempo" é muito subjetivo, portanto necessita-se uma decomposição. São inseridos então dois novos softgoals: "Letras grandes e pequenas" e "Possibilidade de ajuste de contraste". Estes dois novos softgoals são ligados ao softgoal "pai" com relações AND.

Subdiretriz 14.3: Verificar quais dos Sofgoals de Operacionalização influenciam no softgoal sendo analisado. Se o Sofgoal de Operacionalização tem influência positiva, ligá-lo com um some+, se for influência negativa, usar o some-, se o

Sofgoal de Operacionalização for suficiente para satisfazer o softgoal, usar o make, se ele prejudicar completamente, usar o hurt. Também pode ocorrer a situação em que o Sofgoal de Operacionalização ajuda muito na satisfação de um softgoal, mas não satisfaz completamente. Para este caso, pode-se usar o help. Exemplo: Na Figura 4, o Sofgoal de Operacionalização "HW: Display LCD" satisfaz os dois softgoals superiores ("Letras grandes e pequenas" e "Possibilidade de ajuste de contraste"), portanto possui relações make com os mesmos. Já o Sofgoal de Operacionalização "HW: Display 7 segmentos" não possui a possibilidade de ter letras grandes e pequenas, tornando impossível satisfazer este softgoal através do mesmo, tendo então uma relação hurt.

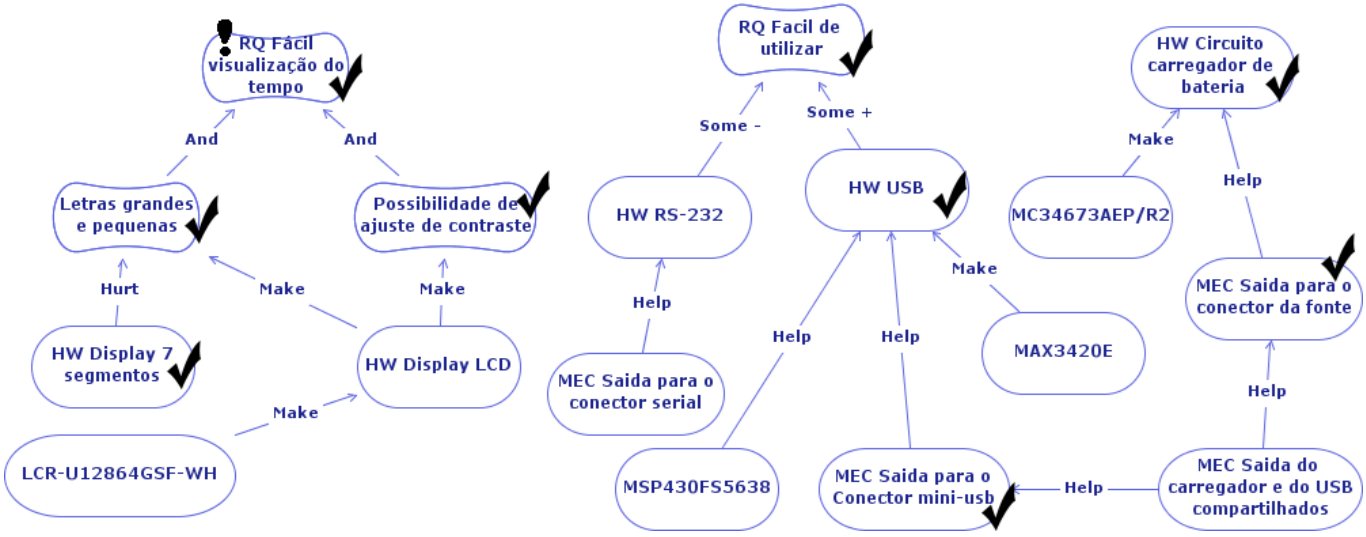
Subdiretriz 14.4: Os Sofgoals de Operacionalização que possuem influência positiva, ou ajudam (help) ou satisfazem o softgoal sendo analisado, portanto colaboram com o mesmo, já podem ser marcados como satisfeitos, evidenciando a escolha destes Sofgoals de Operacionalização. Utilizar o símbolo "✓", para simbolizar essa marcação como satisfeito. Exemplo: Na Figura 4, o Sofgoal de Operacionalização "HW Display de LCD", possui essa marcação, devido à contribuição positiva que ele possui com o softgoal sendo analisado.

Subdiretriz 14.5: Seguir as ligações que estes Sofgoals de Operacionalização possuem com outros Sofgoals de Operacionalização e softgoals, e também marcá-los como escolhidos (satisfeitos). Exemplo: Com o Sofgoal de Operacionalização "HW: Display LCD" escolhido, deve-se marcar com um "□" os softgoals "Letras grandes e pequenas", "Possibilidade de ajuste de contraste" e "RQ: Fácil visualização de tempo".

Subdiretriz 14.6: Escolher o próximo softgoal mais importante e então voltar à diretriz 14.1, agora com foco neste softgoal. Repetir este processo até que todos os softgoals tenham sido analisados. Esta análise é de extrema importância para o sucesso da proposta, e a metodologia explicada acima é uma sugestão que leva a um resultado satisfatório.

Diretriz 15: Baseado nos Sofgoals de

Figura 4 – NFR após o processo de escolha entre as alternativas



Operacionalização escolhidos, escolher o microcontrolador do projeto. O microcontrolador é o coração de um SE, e ele será determinado de acordo com os recursos de hardware escolhidos, pois o microcontrolador pode suprir diversos desses recursos, e quanto mais destes recursos integrados no microcontrolador, mais fácil o desenvolvimento. Exemplo: Na diretriz anterior foi escolhida a alternativa tela de LCD. Levando isto em consideração, um microcontrolador que possua drivers de LCD embutidos facilitaria o desenvolvimento dessa tarefa. Portanto foi escolhido o microcontrolador MSP430FS5638.

Diretriz 16: Nos Sofgoals de Operacionalização de hardware que foram escolhidos, derivar um novo Sofgoal de Operacionalização com um modelo real do dispositivo. Esses dois elementos são ligados usando a relação make. Exemplo: Como a tela de LCD foi escolhida, é buscado um modelo de tela que possua as características mecânicas necessárias. Neste caso será a tela de modelo LCR-U12864GSF-WH.

Diretriz 17: Fazer uma última análise com os Sofgoals de Operacionalização escolhidos, de forma a levantar novos requisitos que integram todos os Sofgoals de Operacionalização escolhidos, isto principalmente em relação à parte mecânica. O objetivo é modelar de forma mais completa possível os requisitos do produto final. Um exemplo de um Sofgoal de Operacionalização que surgiu para esta integração é o Sofgoal de Operacionalização "MEC: Saída do carregador e do USB compartilhados", presente na Figura 4. Considerando os Sofgoals de Operacionalização "MEC: Saída para o conector da fonte" e "MEC: Saída para o conector mini-USB", pode-se integrar ambas como a mesma entrada, algo comum em celulares.

Passo 4 – Construir o modelo SR do produto no contexto de desenvolvimento e produção

O objetivo deste Passo é elicitar requisitos de produção e desenvolvimento (veja atividade 7 do GERSE apresentado na figura 1), além de identificar os atores que estarão envolvidos neste processo. Na Figura 5 são apresentados os atores e requisitos para a aplicação do relógio digital de xadrez.

Diretriz 18: Trazer o ator produto presente no SR do Passo 2, juntamente com todos elementos que estão na sua região. Exemplo: Na Figura 5 está o ator "relógio digital de xadrez", que se originou do modelo gerado no Passo 2.

Diretriz 19: Excluir os means-ends que não foram escolhidos no NFR do Passo 3. Exemplo: No objetivo interno "Sinalizar jogada feita" do "Relógio digital de xadrez" existiam ligações means-ends (ver Figura 3), porém no Passo 3 foi escolhido o recurso "HW Captar sensor nas casas do tabuleiro", portanto os outros dois são excluídos do modelo em questão.

Diretriz 20: Inserir os novos requisitos elicitados no Passo 3 e fazer as devidas ligações com os outros elementos.

Diretriz 21: Os modelos reais de dispositivo são inseridos como recursos e ligados aos elementos

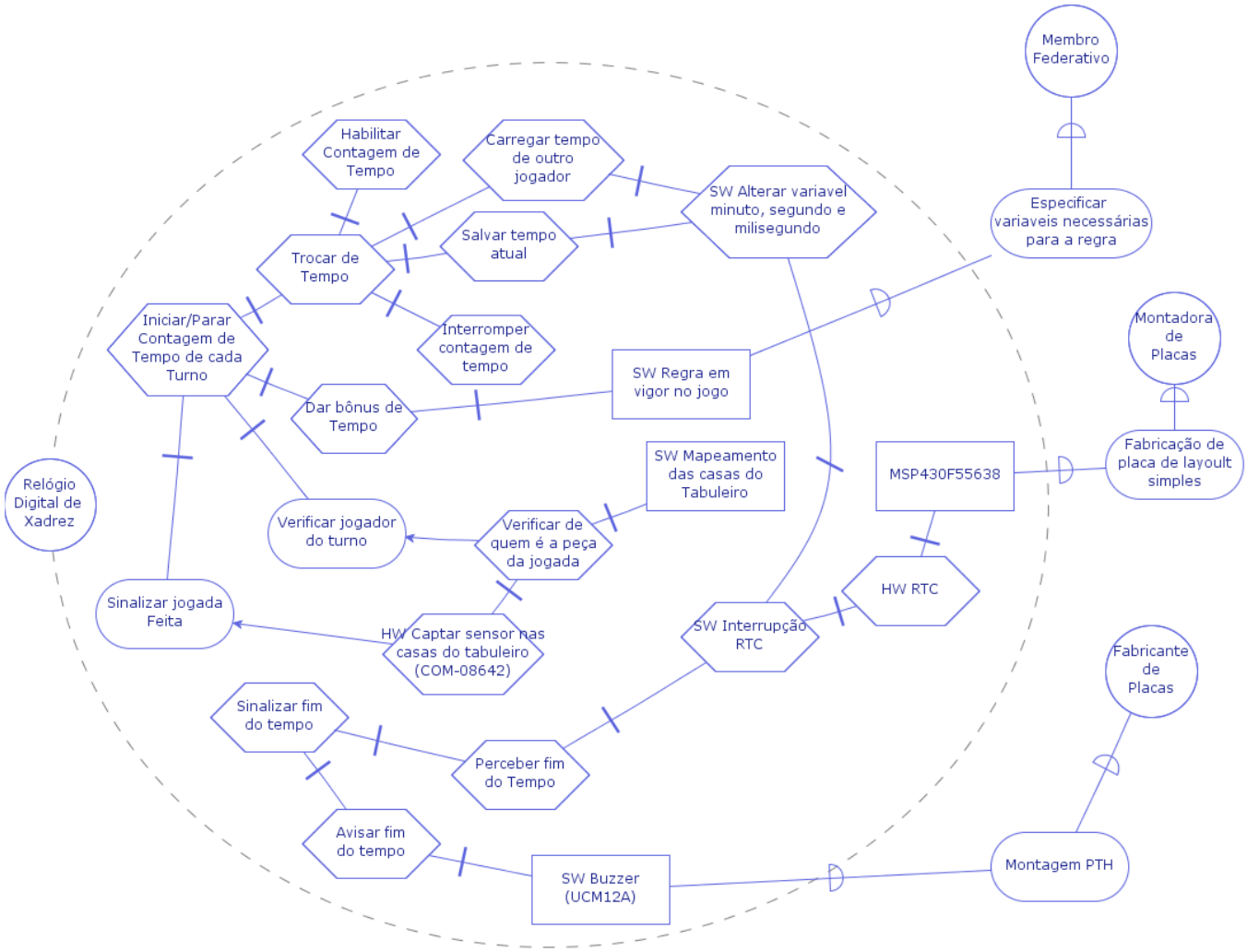
de hardware que eles satisfazem. Exemplo: Na Figura 5, pode-se observar o recurso de hardware Buzzer. No Passo 3, este Buzzer foi definido como o dispositivo de modelo UCM12A, portanto esta informação é inserida no nome do elemento.

Diretriz 22: Inserir atores que serão responsáveis pela produção da parte física do produto (eletrônica e mecânica). Estes são o fabricante das placas, a montadora das placas, e a fabricante das partes mecânicas. Se o projeto necessitar, podem ser inseridos novos atores responsáveis pela produção, como por exemplo, uma fabricante de chicotes elétricos. Exemplo: Os atores inseridos neste Passo são: "Montadora de placas" e "Fabricante de placas" (ver Figura 5).

Diretriz 23: Para cada um dos requisitos de hardware e parte mecânica, ver os requisitos que os mesmos exigem de cada um dos atores de produção. Exemplo: O microcontrolador escolhido no exemplo adotado é um componente SMD, exigindo assim da montadora de placas uma linha de montagem desses componentes.

Diretriz 24: Verificar se todos os requisitos de software estão bem definidos. Os que não estiverem,

Figura 5 – SR do produto em seu contexto de desenvolvimento e produção.



verificar quem poderia esclarecê-los. Esta análise pode gerar novos atores. Exemplo: Foi identificado que a regra em vigor no jogo de xadrez é uma variável necessária ao SE, porém não está claro quais campos fazem parte de uma regra de jogo de xadrez, e a melhor pessoa para fornecer esta informação é um membro da federação brasileira de xadrez. Este membro foi agregado ao modelo SD na Figura 5 como o ator "Membro Federativo".

CONSIDERAÇÕES FINAIS

Os passos e diretrizes propostos neste trabalho apoiam o processo de elicitação dos requisitos de ambiente, de hardware, de software e mecânicos de SEs, seguindo as atividades propostas pelo GERSE.

Atualmente, uma das grandes dificuldades no uso deste guia, está em que o mesmo não propõe técnicas específicas para elicitar estes requisitos.

Neste sentido, a integração do i* e do NFR-Framework ao GERSE surge como alternativa de como, por meio de diretrizes, realizar o processo de elicitação. Umas das vantagens de usar estas técnicas é que as mesmas fazem parte da engenharia de requisitos orientada a objetivos (GORE – Goal Oriented Requirements Engineering), a qual considera que as atividades da engenharia de requisitos devem ser orientadas pelas intencionalidades (objetivos) dos stakeholders.

Desta forma, neste trabalho, são propostas diretrizes para construir os modelos associados aos Frameworks i* e NFR, considerando as intencionalidades dos atores envolvidos, de forma a elicitar os requisitos diferenciados que cada um destes possui sobre o SE.

A proposta deste artigo abre portas para investigações futuras tais como explorar outros papéis e intencionalidades da equipe de desenvolvimento de SEs, estudar riscos no desenvolvimento bem como utilizar os modelos gerados para realizar estimativas de tempo e de custo, entre outros.

REFERÊNCIAS

CHUNG, L. et al. Non-Functional Requirements in Software Engineering (THE KLUWER INTERNATIONAL SERIES IN SOFTWARE ENGINEE- RING Volume 5). 1st. ed. Springer, 1999. Hardcover.

ESPINOZA, H.; SERVAT, D.; GÉRARD, S. Leveraging analysis-aided design decision knowledge in uml-based development of embedded systems. In: Proceedings of the 3rd International Workshop on Sharing and Reusing Architectural Knowledge. New York, NY, USA: ACM, 2008. (SHARK '08), p. 55–62.

OSSADA, J. C.; MARTINS, L. E. G. Gerse: Guia para elicitação de requisitos de sistemas embarcados. In: I Workshop de Sistemas Embarcados. [S.l.: s.n.], 2010. v. 1, p. 117–130.

SIKORA, E.; TENBERGEN, B.; POHL, K. Industry needs and research directions in requirements engineering for embedded systems. Requirements Engineering, v. 17, n. 1, p. 57–78, 2012.

YU, E. et al. Social Modeling for Requirements Engineering. Mit Press, 2011. (Cooperative Information Systems)

Artigo submetido em: 31;08.2014

Artigo aceito para publicação em: 23.12.2014