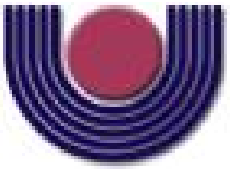

Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Integração do I* ao GERSE para elicitação de requisitos em sistemas embarcados

Maykon Valério da Silva

CASCAVEL
2012

MAYKON VALÉRIO DA SILVA

INTEGRAÇÃO DO I* AO GERSE PARA ELICITAÇÃO DE REQUISITOS EM SISTEMAS EMBARCADOS

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Victor Santander

CASCAVEL
2012

MAYKON VALÉRIO DA SILVA

INTEGRAÇÃO DO I* AO GERSE PARA ELICITAÇÃO DE REQUISITOS EM SISTEMAS EMBARCADOS

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Victor Santander
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Elder E. Shemberger
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Cristiano Ferreira de Souza

Cascavel, 4 de dezembro de 2012

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ER	Engenharia de Requisitos
SE	Sistema Embarcado
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
MDE	Model-Driven Engineer (Engenharia Baseada em Modelo)
UML	Unified Modeling Language (Linguagem de Modelagem Unificada)
SDL	Specification and Description Language (Linguagem de Especificação e Descrição)
SD	Strategic Dependencies (Dependências Estratégicas)
SR	Strategic Rationales (Razões Estratégicas)
NFR	Non-Functional Requirements (Requisitos Não Funcionais)

Sumário

Resumo

Observa-se atualmente um carência de metodologias e técnicas de engenharia de requisitos voltadas aos sistemas embarcados. Desta forma buscou-se neste trabalho estudos que atuem sobre esta necessidade. Foi encontrado assim o Guia para Elecitação de Sistemas Embarcados (GERSE) que visa fornecer um método sistemático para uma sub-área da engenharia de requisitos, a elicitação de requisitos. Visando uma evolução deste guia, este trabalho busca um integração deste com a *framework* i* e o NFR-*Framework*, com o objetivo de facilitar a execução de tarefas que o GERSE propõe. Mostrando formas de se obter as informações requeridas pelo mesmo.

Palavras-chave: Engenharia de Requisitos, Sistemas Embarcados, GERSE, i*, NFR.

Capítulo 1

Introdução

Neste capítulo apresenta-se uma visão geral do trabalho. Na seção 1.1 contextualiza-se a área de sistemas embarcados e engenharia de requisitos, na seção 1.2 é mostrado o problema identificado, na seção 1.3 é mostrada a proposta a ser trabalhada e por fim na seção 1.4 as contribuições esperadas.

1.1 Contexto

A engenharia de requisitos (ER) serve para facilitar todo o desenvolvimento de um software, e possui características diferentes nos diversos tipos de sistemas, sistemas de gerenciamento, sistemas críticos e assim por diante. Neste trabalho será estudada a ER para a categoria de sistemas embarcados (SE's).

A classe de SE's é extremamente importante porque engloba algumas das mais antigas aplicações da área da computação, porém, mesmo assim, não existe um conceito único universal para SE's. Entre os diversos que são encontrados na literatura está o apresentado em [1], que define SE como o dispositivo responsável por controlar um processo. Como o autor mesmo diz, é o responsável por trazer "inteligência"à um ambiente. Ele cita como exemplos os sistemas industriais de controle, sistemas de auxílio de vôo, sistemas de monitoramento de pacientes e diversos outros.

Servindo de complemento ao conceito anterior, em [2] explica-se que os SE's têm como finalidade controlar um ambiente ou dispositivo físico através do envio de sinais de controle à atuadores (saída) reagindo a sinais providos por sensores ou usuários (entrada). Por isto os SE's envolvem desenvolvimento tanto de hardware quanto de software.

Existem também definições mais rigorosas, nas quais além das características citadas acima, são levadas em conta o período de amostragem dos sensores e o *delay* entre uma entrada e a devida reação do sistema. Em [3] o autor usa estes e outros parâmetros para reconhecer se um sistema é embarcado ou não.

O conceito que será adotado neste trabalho é o primeiro dos citados anteriormente pois engloba o sistema que será usado no estudo de caso que será feito para validar a proposta.

Atualmente os SE's voltam a experimentar um crescimento elevado, e isto se deve muito à evolução no hardware, o que os tornam úteis à uma ampla gama de aplicações [1]. O principal dispositivo de hardware responsável por isto são os microcontroladores, e uma evidência disto é que apenas 2% dos processadores fabricados no mundo são destinados a computadores de propósito geral, sendo os outros 98%, que são microprocessadores e microcontroladores de pequeno e médio porte, destinados à SE's [2].

O reflexo deste crescimento está na presença dos SE's nas indústrias de aviação, médica e principalmente automotiva, que é onde os SE's apresentam a maior evolução.

Outra característica interessante, agora no desenvolvimento de SE's, é a multidisciplinaridade que o mesmo envolve, onde os projetos geralmente são conduzidos por engenheiros de hardware, com *expertise* em eletrônica, elétrica e mecânica [2].

É indiscutível a importância de uma boa Engenharia de Requisitos para o sucesso de um software final. No contexto de desenvolvimento de softwares a ER já é bem explorada e estudada. Porém isto não acontece nos SE's.

Segundo [2] existe praticamente uma certa ausência de metodologias de engenharia de requisitos desenvolvidas para os SE's.

Isto acontece devido à algumas características que os diferem de um software comum. Entre estas características, listadas por [1] e [4], estão:

- Envolve desenvolvimento de hardware.
- Os programas são frequêntemente grandes (entre 50000 e 100000 linhas);
- Saídas não são apenas dados, mas também sinais de controle;
- Mudança contínua devido à evolução nos requisitos. Revisões anuais costumam ser do mesmo tamanho que o desenvolvimento original;

- Requisitos não funcionais assumem um papel de extrema importância, como requisitos de segurança e confiabilidade;

Tudo isto está aliado à crescente complexidade das aplicações em que ele está envolvido. Esta complexidade gera um risco muito grande de deficiências e erros não serem detectados, além de um aumento no custo, no tempo e no esforço para o desenvolvimento [3]. O fato de o desenvolvimento de um SE envolver tanto hardware quanto software leva à outra necessidade de apoio na ER, em [5] o autor evidencia isto ao dizer que a total separação entre hardware e software no design de um SE não é mais algo viável.

Um dado que comprova esta carência de metodologias, apresentada em [6], é o de que 50% dos problemas encontrados em SE's ocorrem depois da implantação e são erros conceituais de requisitos, ou seja, são equívocos na captura dos requisitos.

Neste contexto, é importante pesquisar alternativas tanto do uso de técnicas quanto de metodologias da ER aplicáveis no processo de desenvolvimento de SE's.

1.2 Problema

As pesquisas em relação ao desenvolvimento de SE são diversas, o esforço da comunidade do MDE (*Model-Driven Engineer* - Engenharia baseada em modelo) em incorporar ferramentas para especificar esses requisitos não-funcionais de forma a permitir a posterior verificação e validação é apenas um exemplo [7]. Porém em [3] mostra-se a dificuldade que é para as indústrias a adoção destas técnicas, muitas vezes por não se levar em consideração as reais necessidades desta. Ainda em [8] levanta-se uma questão de extrema importância: "Onde está a 'super-cola' que junta todas estas técnicas de uma forma a ajudar os desenvolvedores a contruírem seus sistemas?".

Para tentar esboçar uma solução para estes problemas em [9] apresenta-se o GERSE (Guia de elicitação de Requisitos para Sistemas Embarcados) que, como o próprio nome diz, consiste de um guia com diversos passos e tarefas que no final resultariam nos requisitos necessários para o sistema. Ele trabalha específicamente com a parte da elicitação dos requisitos e deixa como propostas de trabalhos futuros a abordagem das outras partes da ER (documentação, validação e análise) voltadas aos SE's.

Este guía estabelecido pelo GERSE possui duas fases: uma pré-fase, que possui tarefas de nível organizacional e mercadológico, e uma fase principal, onde as tarefas se concentram no nível técnico. A pré-fase busca, através de entrevistas e pesquisas de mercado, requisitos de alto-nível para o produto (características gerais e o que o mercado espera do mesmo) e para a organização que irá desenvolvê-lo (principais stakeholders e riscos de desenvolvimento). A fase principal busca, através dos requisitos de alto-nível gerados na fase anterior, requisitos funcionais e não funcionais de baixo nível, como tipo de alimentação do sistema, tamanho do dispositivo, tipo de comunicação e diversos outros.

Porém o GERSE não possui um processo bem específico de como se chegar aos principais requisitos que suas tarefas visam elicitar, determinando apenas o **o que** elicitar, e nao **como** elicitar. Portanto um processo, que incorporado ao GERSE determine esta maneira de se obter os requisitos e informações traria um ganho muito grande ao guia.

Mais especificamente, percebe-se claramente a necessidade de apoiar o processo do GERSE sob uma perspectiva de análise de intensionalidades de todos os membros integrantes do ambiente de uso do SE, bem como o ambiente organizacional que compõe o desenvolvimento destes SE's.

1.3 Proposta

Tendo em vista esta deficiência do GERSE, propõe-se incorporar ao mesmo o *framework* i* (diz-se "i estrela"), que tem a capacidade de absorver a modelagem organizacional e a definição de *stakeholders* de forma natural, já que surgiu como uma proposta de ser uma ferramenta para modelar os agentes e as intencionalidades destes em um ambiente organizacional [13].

O *framework* i* mapeia os atores (*stakeholders*) em um contexto, colocando suas intencionalidades e dependências em relação aos outros atores, permitindo assim uma análise da organização montada, se seria eficiente, se há uma interação ou dependência crítica para o funcionamento, e ainda levantar alternativas para a mesma.

Com isto o i* pode colaborar em duas partes. Primeiro através da modelagem da interação do produto com os diversos atores que atuarão sobre ele no seu uso, facilitando assim a elicitação de requisitos. E segundo, com a organização de contexto e a definição de *stakeholders* (pré-fase do GERSE). O i* possui elementos e entidades que facilitam este processo, e ainda acrescenta

a possibilidade de uma análise de toda esta organização.

Desta forma pretende-se apoiar as atividades do GERSE na geração de modelos do ambiente de uso do SE, onde serão mapeados os atores envolvidos no uso do mesmo, e do ambiente organizacional de produção e desenvolvimento, onde serão inseridos todos os atores envolvidos no desenvolvimento e na produção do SE, assim como a interação entre eles. Este modelo organizacional permitirá então uma análise sobre esta organização, podendo mostrar características desta não possíveis de ser vistas através da documentação do GERSE.

1.4 Contribuições Esperadas

Busca-se com este trabalho melhoramentos significativos no GERSE, tais como:

1. Total definição de todas as partes principais (organização, *stakeholders*, objetivos, recursos, e diversos outros).

2. Mapeamento das interações chaves entre *stakeholders* necessárias para o desenvolvimento do SE.

3. Melhor especificação das tarefas e passos necessários para se obter os requisitos.

4. Propiciar um entendimento mais global e estratégico do ambiente organizacional no qual um novo SE operará.

1.5 Organização do trabalho

No capítulo 2 são mostrados algumas características da ER para SE's, e alguns trabalhos que abordam estes problemas. No capítulo 3 será mostrado o GERSE, de onde ele surgiu, todas as suas tarefas e a impressão que ele causou nos profissionais da área de SE's. Posteriormente, no capítulo 4 será apresentado a *framework* i*, também de onde surgiu, os seus elementos, os seus diagramas, a análise que ele propicia e algumas aplicações. No capítulo 5 será mostrado o NFR-*Framework*, outra ferramenta inserida no processo do GERSE através desta proposta.Então, no capítulo 6, a proposta é apresentada na forma de diretrizes, juntamente com pequenos exemplos de cada uma delas. No capítulo 7 é apresentado um estudo de caso completo utilizando a pro-

posta. E finalmente, no capítulo 8, serão feitas as considerações finais e propostas de trabalhos futuros.

Capítulo 2

Engenharia de requisitos para sistemas embarcados

Neste capítulo são apresentados características necessárias para a ER em SE's, assim como outros trabalhos que buscam técnicas para melhorar esta deficiência.

2.1 Visão Geral

A ER tem surgido como uma das principais áreas de estudos hoje em dia, e é evidente que os requisitos elicitados têm um grande impacto na qualidade e na usabilidade dos sistemas, e não menos importante, na eficiência e no gerenciamento do seu desenvolvimento [1]. Isto não muda para os SE's, mas a ER, neste caso, precisa de um foco um pouco maior em certos aspectos que não possuem tanta importância em sistemas convencionais de software. Esses aspectos estão principalmente na categoria de requisitos não-funcionais como os de performance (tempo-real, confiabilidade e disponibilidade), os físicos (peso, volume, consumo de potência, tamanho de memória), segurança e diversos outros [2] [1].

O foco maior nestes fatores vem do contexto em que os SE's geralmente são utilizados. Como dito na seção 1.1, os SE's estão intimamente ligados à controle de dispositivos físicos, e geralmente envolvem pessoas nos processos que controlam. Como melhor exemplo pode-se citar o sistema de direção eletrônica de um automóvel, onde um SE ajuda o motorista adicionando estabilidade, leveza e facilidade no controle do carro. Uma falha neste sistema pode trazer problemas sérios. No pior caso pode causar acidentes graves, e no melhor caso, quando o defeito é encontrado, envolve enormes perdas financeiras com o *recall* dos carros, como explicitado em [10]. Além disto, estes fatores afetam diretamente o *design* da arquitetura a ser desenvolvida.

Dependendo do requisito de tempo-real, que diz quanto tempo o sistema tem para tomar determinadas atitudes, deve ser decidido a capacidade de processamento, a fragmentação ou não em *threads*, como será feito o gerenciamento dessas *threads* e outras tarefas, e assim por diante.

A tabela 2.1, obtida em [2] sumariza as principais diferenças entre ER para SE's e para sistemas convencionais.

Tabela 2.1: Diferenças entre a ER para SE's e para sistemas convencionais [2]

Características	**Sistemas Embarcados**	**Sistemas de informação convencionais**
Participação de *stakeholders* com perfil técnico (engenheiros)	Muito freqüente	Pouco freqüente
Requisitos de tempo-real	Comum, tendo grande impacto no produto	Incomum
Utilização de hardware padronizado	Incomum. Normalmente o hardware é projetado especificamente para o produto em desenvolvimento	É comum a utilização de plataformas padronizadas
Restrições quanto ao tamanho do software executável	Presente na maioria dos projetos de SE	Normalmente não há
Time to market	Janela de tempo curta	Janela de tempo longa
Dimensões físicas do sistema (volume, peso, ergonomia)	RNF muito importante a ser considerado	Não se aplica
Consumo de energia	RNF muito importante a ser considerado	Não se aplica
Confiabilidade	RNF fundamental para a maioria dos casos, e mandatório em alguns casos	Necessário, mas com impacto menor no projeto global do sistema
Definição de requisitos de hardware	Forte impacto no projeto global	Baixo impacto no projeto global
Atores que interagem com o sistema	Dispositivos físicos como sensores e atuadores, e eventualmente atores humanos	Normalmente atores humanos

Levando isto em conta a maioria dos trabalhos se concentra nestes requisitos não-funcionais

e na definição de requisitos de hardware, buscando melhor integração com o software. Isto pode ser visto em [4], onde busca-se um processo para descrever e analisar requisitos de segurança através de diagramas de sequência e casos de uso, introduzindo duas nova categorias de casos de uso, os incorretos, que são casos de uso que acarretariam em problemas, e os de segurança, que consistem de contra medidas aos casos de uso incorretos. O processo apresentado por [4] consiste de uma representação do sistema através de um diagrama de contexto de objetos, então para cada objeto são feitos casos de uso, destes casos de uso são feitos os casos de uso incorretos, que derivam em casos de uso de segurança que por fim determinam requisitos de segurança. Estes objetos que representam o sistema então vão sendo decompostos em seu objetos internos e o processo então é refeito até que os objetos analisados sejam primitivos.

Muitos trabalhos se dedicam ao uso de linguagens como a UML (*Unified Modeling Language* - Linguagem de Modelagem Unificada) para o *design* de SE's, como em [5], onde utiliza-se UML e SDL (*Specification and Description Language* - Linguagem de Especificação e Descrição) para especificar a arquitetura, a comunicação, as interfaces, e todas as outras partes do sistema. Já em [7] busca-se novas formulações para a UML para a especificação de requisitos não funcionais

Porém os mais relevantes são os que buscam a "super-cola"descrita em [8] e citada na seção 1.2. Em [11] apresenta-se uma proposta neste sentido, pois o autor apresenta um processo sistemático para definição de requisitos em SE's. Para isto ele divide o sistema em duas partes, uma que representa o sistema inteiro (software e hardware) e outra que representa o controlador (apenas o software). Com isto o autor busca descrever possíveis falhas de hardware e considerá-las no controlador. Também é sugerido notações formais, semi-formais e gráficas.

Outro destes trabalhos é o [1], que mostra uma abordagem operacional para a especificação de requisitos em SE's. E tudo feito através da linguagem PAISLey que é capaz de gerar uma especificação de requisitos executável, pois a abordagem visa gerar um modelo do sistema interagindo com o seu ambiente físico.

O GERSE também é um destes trabalhos e o seu processo sistemático, que consiste de um guia, é independente de linguagens e caracteriza-se apenas pelo uso de tarefas e atividades que devem ser seguidas pelos desenvolvedores. Isto pode ser visto como uma vantagem, pois não exige um conhecimento em linguagens de especificação como a UML, algo comum já que,

como mostrado na seção 1.1, a equipe de desenvolvedores de SE geralmente é constituida de engenheiros, que comumente desconhecem a engenharia de software. Este foi um dos fatores para a escolha do GERSE para este trabalho, além da existência de atividades mercadológicas e organizacionais, demonstrando uma preocupação com as reais necessidades das indústrias e empresas que desenvolvem SE's, um fator que, segundo [3], é determinante para a adoção de uma metodologia por estas.

Buscando analisar esta adoção à metodologias por parte das indústrias e empresas, em [2] é realizado um estudo sobre como acontece na prática esta ER no desenvolvimento de SE's, vendo o que é usado atualmente e como os profissionais contornam esta carência de suporte. Cita-se neste que é comum o desenvolvimento de um modelo de sistema antes mesmo de ser feita a elicitação dos requisitos, e quando o fazem, não é de uma maneira formal, com uma documentação ou gerenciamento dos mesmos. Além da já citada falta de técnicas para a ER, [1] aponta também como causas deste comportamento a falta de conhecimento sobre a importância da ER e a relutância em acrescentar custos e tempo de desenvolvimento no início do projeto. A figura 2.1 mostra um gráfico, gerado por [2], onde são evidenciados os métodos e ferramentas que são geralmente empregados pelos profissionais envolvidos em sua pesquisa.

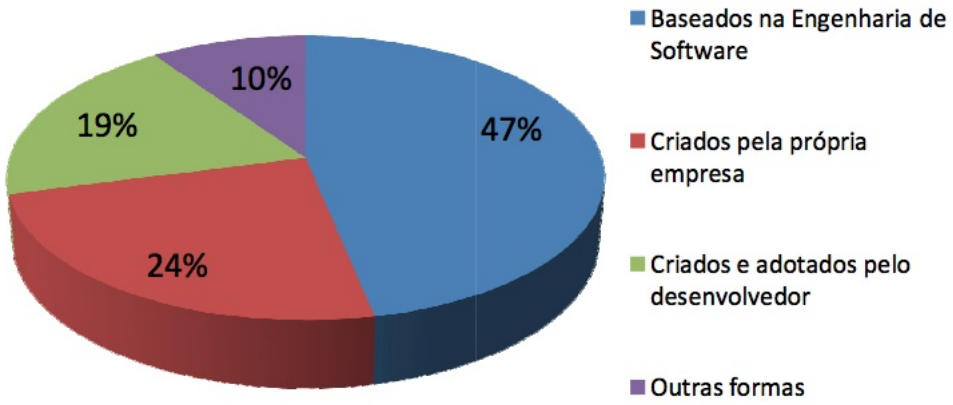

Figura 2.1: Metodologias empregadas para coleta e análise de requisitos [2].

2.2 Considerações finais do capítulo

Como visto a ER para SE's possui algumas características que dificultam a adaptação de metodologias e técnicas da engenharia de software para esta área, e o estudo sobre o estado da prática no desenvolvimento de SE's, apresentado neste capítulo, mostra que esta deficiência se estende para as empresas e indústrias de SE's, mostrando a importância deste trabalho.

Dentre os estudos apresentados neste capítulo destacou-se o GERSE, pelas suas características e avanços na área. Sendo assim ele está no centro deste trabalho, buscando melhorias e avanços no seu método.

O mesmo é apresentado no próximo capítulo.

Capítulo 3

GERSE: Guia de Elicitação de Requisitos para Sistemas Embarcados

Neste capítulo apresenta-se o GERSE, uma das principais bases deste trabalho. Na seção 3.1 é apresentada uma visão geral do GERSE, na seção 3.2 são mostradas as atividades e as tarefas que o compõem. Por fim na seção 3.5 é mostrado a opinião dos profissionais da área de desenvolvimento de SE's sobre o GERSE, mostrando algumas áreas que necessitam de melhoramento.

3.1 Visão Geral

O GERSE, apresentado em [9], surgiu com o objetivo de contribuir para o desenvolvimento de SE's, servindo como um guia para elicitação sistemática de requisitos, buscando uma forma sequencial lógica e organizada de fases e tarefas.

O autor baseou o seu trabalho em três componentes básicos. A IEEE 830-1998, o template *Volere*, e os resultados obtidos da pesquisa sobre o estado da prática em SE's apresentado em [2]. O primeiro consiste de um padrão que fornece recomendações para especificação de requisitos, porém ele é direcionado à softwares. Já o segundo fornece uma estrutura para documentar e organizar requisitos, também direcionada a softwares. A Figura 3.1, obtida em [9], ilustra a interação entre esses componentes.

A documentação de requisitos é uma parte muito importante para a ER, pois esta documentação é utilizada como veículo de comunicação, gerenciamento de mudanças e validação do produto final [1].

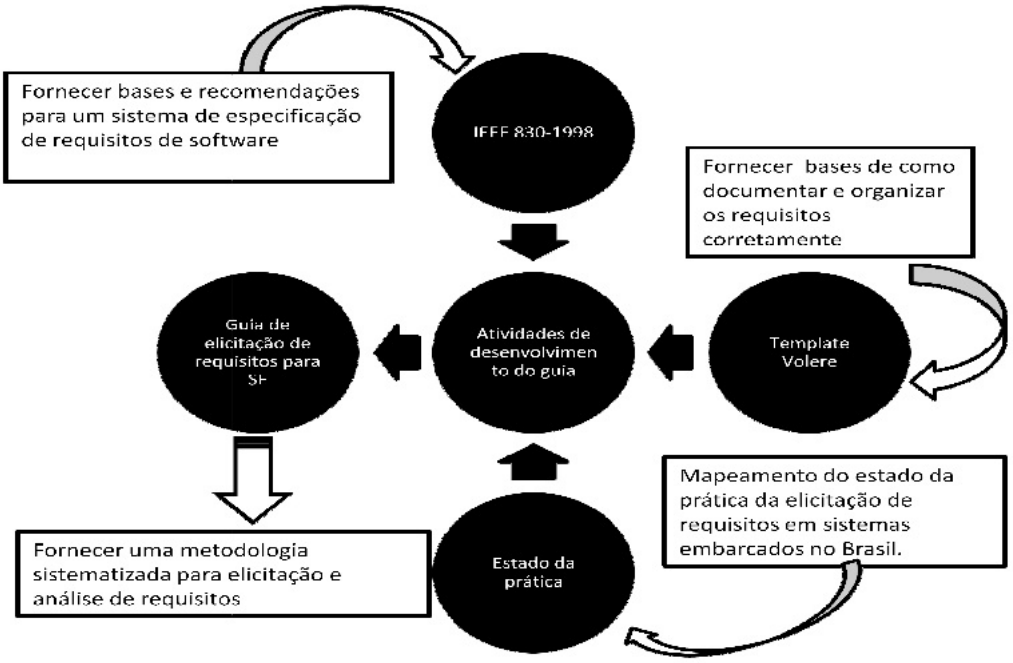

Figura 3.1: Interação entre os componentes em que o GERSE foi baseado [12]

3.2 Atividades

O GERSE é dividido em duas fases, a pré-fase e a fase principal. 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, baseada no template *Volere*.

A pré-fase possui três atividades: Organização de Contexto, Definição de *Stakeholders*, e Requisitos de Alto Nível. Já a fase principal possui quatro atividades: Definição de Hardware, Definição de Software, Identificação de Métricas de Qualidade, e Definição de Requisitos para Produção. Na transição da pré-fase para a fase principal ocorre a chamada tradução, onde os requisitos de alto nível, gerados na primeira, são melhor definidos e passados para os requisitos técnicos, ou requisitos de baixo nível. A figura 3.2 sumariza estas atividades.

Cada atividade possui as suas tarefas, e cada tarefa gera o que [9] chama de artefato de saída, que, além de ajudarem na transição entre uma tarefa e outra, servem de documentação. Cada atividade e suas tarefas são explicadas nas seções a seguir. As tarefas de cada atividade são

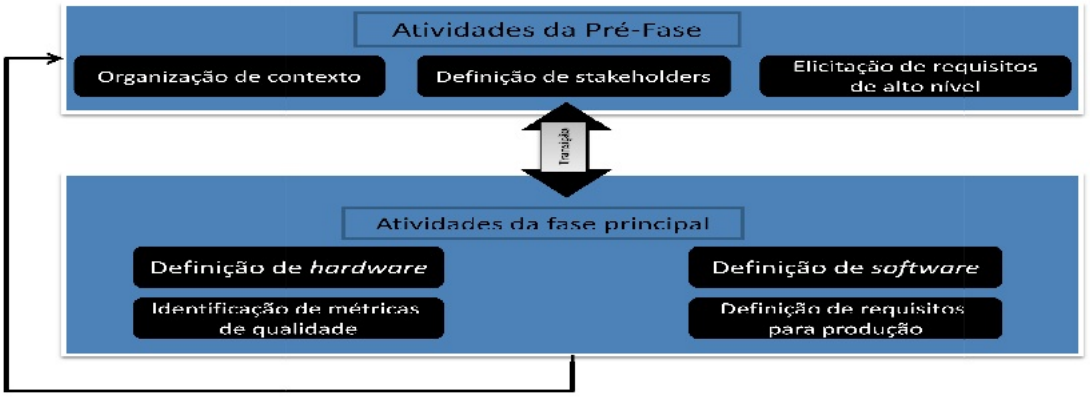

Figura 3.2: Atividades do GERSE [12]

mostradas na forma de tabelas, retiradas de [9].

3.3 Pré-fase

Esta atividade serve para definir uma visão geral do produto e como a empresa deve se preparar para ele. Define principalmente o que a empresa espera com o produto, o que o produto deve ter, quem deverá participar da sua definição e desenvolvimento, e os requisitos de alto nível. As atividades envolvidas são: Organização de Contexto (seção 3.4), Definição de *stakeholders* (seção 3.3.2) e Elicitação de requisitos de alto nível (seção 3.3.3).

3.3.1 Organização de Contexto

A tabela 3.1 contém todas as tarefas relacionadas à esta atividade juntamente com os artefatos de entrada (que permitem a realização das tarefas) e os artefatos de saída (documentos gerados pelas tarefas). Segundo o autor, busca-se através desta atividade entender o problema que o produto irá atingir e a capacidade da empresa de produzí-lo. Isto através da obtenção das diretrizes do projeto, das características gerais do produto, dos impactos organizacionais do desenvolvimento e o público a ser atingido.

3.3.2 Definição de stakeholders

Nas tarefas desta atividade (tabela 3.2) são definidos os principais *stakeholders* do projeto. Existe um grupo de *stakeholders* especialmente importante, e pelo fato de não poderem ser

Tabela 3.1: Tarefas da atividade Organização de Contexto [9]

Entrada	**Tarefa**	**Saída**	**Meta**
Questionários, entrevistas, cenários, políticas organizacionais, e legislação.	**1.1** Obter o propósito e as metas organizacionais do produto frente ao mercado.	Lista com os propósitos e as metas a serem atingidas com o desenvolvimento do produto.	Definir qual o objetivo central do produto a ser desenvolvido.
	1.2 Especificar as características gerais do produto.	Lista das características gerais do produto e de suas prioridades.	Identificação da característica do produto, a ser desenvolvido, e de suas prioridades.
	1.3 Definir os impactos organizacionais com o desenvolvimento do produto.	Lista dos impactos organizacionais com o desenvolvimento do produto	Verificar quais as conseqüências, caso o produto seja desenvolvido.
	1.4 Determinar os impactos negativos causados pelo não desenvolvimento do produto	Lista dos impactos negativos causados pelo não desenvolvimento do produto	Verificar quais as conseqüências, caso o produto não seja desenvolvido.

esquecidos, foi separado uma tarefa para a identificação dos mesmos, que são os especialistas de domínio. Nesta atividade também são identificados os *stakeholders* contrários à realização do projeto e definido o perfil do usuário.

A importância desta etapa é citada em [1], que comenta que o projeto não poderá ser um sucesso completo a não ser que todos os envolvidos no projeto concordem com os requisitos elicitados, portanto, identificá-los é um passo muito importante.

Em [6] é citado a diversidade de *stakeholders* que um projeto de SE envolve, como clientes, analista de negócios, analista de requisitos, engenheiros de hardware e engenheiros de software.

Esta característica de quantidade de *stakeholders* e a interação entre eles favorece o uso do I* como proposto neste trabalho, por possuir representações gráficas para esses elementos.

Tabela 3.2: Tarefas de definição de stakeholders [9]

Entrada	**Tarefa**	**Saída**	**Meta**
Questionários , entrevistas,cenários, políticas organizacionais, legislação e documentação existente.	**2.1** Definir os principais stakeholders	Lista dos stakeholders e os graus de envolvimento e de influência	Definir as pessoas e os grupos, os quais possuem interesse e que influenciarão nos requisitos do produto.
Datasheet, normas, padrões, conformidade e entrevistas.	**2.2** Determinar os stakeholders como especialistas de domínio.	Lista dos stakeholders especialistas.	Indicar os stakeholders que irão contribuir para a definição da arquitetura, a adequação para o cumprimento de normas e padrões, os especialistas de domínios e os fornecedores.
Pesquisa de mercado, análise estrutural da organização, entrevistas e observações.	**2.3** Relacionar os stakeholders contrários ao projeto.	Lista de stakeholders contrários à realização do produto.	Determinar os stakeholders que são contrários ao projeto, os quais não se beneficiarão com a realização ou terão prejuízos.
Pesquisas e entrevistas	**2.4** Caracterizar o perfil do usuário	Lista dos usuários finais e de suas características	Definir quem são os usuários finais, descrevendo as habilidades e os perfis necessários para a utilização do produto.

3.3.3 Elicitação de Requisitos de Alto Nível

Nesta etapa são identificadas as funcionalidades e restrições do produto, segundo os stakeholders elicitados na etapa anterior. São descritos os requisitos funcionais, os não-funcionais, o ambiente em que o produto será utilizado, a interface com o usuário e diversos outros aspectos. Com isto é elaborada uma solução, que consiste de um modelo, diagrama ou protótipo do produto.

Como se trata de uma modelagem de alto-nível, nesta etapa também pode ser usado o I*, agora para modelar a interação do produto com os seus usuários, e também com o seu ambiente, facilitando a execução das tarefas desta etapa (tabela 3.3).

Tabela 3.3: Tarefas de elicitação de requisitos de alto nível [9]

Entrada	**Tarefa**	**Saída**	**Meta**
Cenários, análise de produtos similares, pesquisa de mercado, protótipos, documentação existente e entrevistas.	**3.1** Definir as funções do produto (Requisitos Funcionais)	Relação das funções que o produto deverealizar	Definir todas as funções que deverão ser realizadas pelo produto, enfatiz ando as de entrada e saída.
	3.2 Indicar as restrições do produto (Requisitos não-funcionais)	Listacomas características não desejáveis do produto	Determinar os comportamentos indesejáveis do produto, e as características encontradas em produtos similares, para serem utilizados como um diferencial.
	3.3 Indic ar as restrições físicas do ambiente	Lista com as características de ambiente de trabalho do produto	Caracterizar as condições físicas de ambiento do produto (temperatura, pó, vibrações mecânicas, radiação, eletromagnetismo e umidade)
	3.4 Estabelecer as características de consumo de potência	Formas de alimentação do produto	Definir a forma de alimentação do produto (pilhas, baterias e fonte externa)
	3.5 Identificar as características físicas e mecânicas	Listagemcomas características físicas e mecânicas do produto	Identificar as características físicas e mecânicas (tamanho do produto, designer físico, funções mecânicas de botões e tamanho de sensorese atuadores para a comunicação como usuário)

	3.6 Caracterizar a interface **3.7** Indicar as situações críticas **3.8** Definir o grau de confiabilidade	Lista de interface externa Relação das situações críticas que,eventualmente, possam ocorrer. Nível de confiabilidade	Estabelecer a comunicação externa (USB, C AN, Serial) Quedas, choques e travamentos. Verificar o nível de c onfiabilidade
Solução encontrada, fornecedores, pesquisas , cenários, análise de produtos similares, pesquisa de mercado, protótipos e projetistas.	**3.9** Determinar a solução encontrada	Modelo, Diagrama ou Protótipo com a solução encontrada	Propor uma solução inicial para o produto através de um modelo conceitual ou lógico
Planilha eletrônica	**3.10** Revelar a estimativa de custos	Planilha de custos	Apresentar um orçamento inicial de estimativa de custos de produção

3.4 Fase Principal

Nesta fase ocorre a chamada tradução dos requisitos de alto nível para os requisitos de baixo nível. Desta fase derivam os requisitos técnicos, que é o objetivo final do guia.

3.4.1 Identificação de Hardware

As tarefas desta atividade (Tabela 3.4) visam identificar os requisitos técnicos de hardware necessários, como sensores, atuadores, e ainda portas de comunicação e interrupções. Tudo isto ajudará na definição do microcontrolador ou outros elementos de hardware necessários para o desenvolvimento. Como visto no capítulo 2, a necessidade desta tarefa é uma característica peculiar da ER para os SE's, isto por eles envolverem o desenvolvimento tanto de software quanto de hardware. Estes elementos de harware podem ser elicitados através de um modelo SR, identificando-os como recursos necessário para o desempenho de tarefas internas, que visam satisfazer dependências externas serão requisitos funcionais.

Tabela 3.4: Tarefas de identificação de hardware [9]

Entrada	Tarefa	Saída	Meta
Análise de requisitos de alto nível (pré-fase)	**4.1** Definir os sensores	Listas de sensores e seus atributos (Analógico/Digital)	Definir a lista de sensores a serem usados, com sua s características, para a definição da quantidade de portas, pinos e conversores analógico/digital necessários.
	4.2 Delimitar os atuadores	Relação de atuadores e de seus atributos (Analógico Digital)	Elaborar a lista de atuadores a serem utilizados, com suas características, para a definição da quantidade de portas e de pinos que faltam.
	4.3 Esclarecer a interação com o usuário	Listas de displays, avisos sonoros e visuais para a interação com o usuário.	Definir o HW para a comunicação com o usuário
	4.4 Caracterizar as interrupções deHW	Listagens de interrupções originadas por HW	Indicar as interrupções que serão geradas por HW
	4.5 Identificar os botões	Relação de botões de ação e suas funções	Definir os botões (chave liga/desliga) necessários para a interface com o usuário
	4.6 Indicar as memórias	Lista de memórias externas	Determinar a quantidade de memória externa para o armazenamento das funções e dos dados (Flash/E2PROM)
	4.7 Definiras portas de comunicação externa	Lista de portas de comunicação externa	Esclarecer a necessidade de comunicação externa com outros componentes de HW e o tipo de comunicação a ser empregada (USB/ SERIAL/ ETHERNET/CAN)
	4.8 Fixar os requisitos de com pon en tes	Lista de componentes auxiliares a serem utilizados	Definir a lista de componentes eletrônicos auxiliares, externos ao microcontrolador (cristal, capacitores, resistores, LEDs etc.)
	4.9 Indicar os requisitos de Layout da placa controladora	Listagem de características do layout e da placa de circuito impresso	Determinar o tamanho da placa circuito impresso (Layout) e o tipo de placa (Multilayer)

	4.10 Definir os parâmetros de HW legados	Lista com as características de sistemas legados	Definir as características gerais de HW de sistemas legados
	4.11 Demarcar os parâmetros de COTS especiais	Lista com as características gerais de COTS	Identificar as características gerais de COTS
	4.12 Identificar os microcontroladores	Relação de microcontroladores compatíveis	Listar os microcontroladores que atendam aos requisitos de HW e seus respectivos datasheets

3.4.2 Elicitação de Requisitos de Software

Os requisitos pertencentes ao software são elicitados nesta atividade, como as variáveis de ambiente, as exceções possíveis de acontecer e diversas outras questões. As tarefas desta atividade são mostradas na tabela 3.5.

Esta etapa e a anterior são provavelmente as mais complexas e importantes. A complexidade destas etapas está na íntima ligação que existe entre elas, onde uma afeta a outra. Embora elas estejam mapeadas como etapas diferentes e separadas, cada uma com suas tarefas, elas devem ser feitas paralelamente. Esta necessidade é evidenciada em [5], onde diz-se que a total separação entre hardware e software não pode mais ser uma opção. O modelo SR ajuda igualmente esta parte de requisitos de software, pois estes também podem ser mapeados como recursos necessários à realização de tarefas. O SR ao possui recursos de software e hardware, e apresentar relacionamentos entre estes recursos também facilita esta integração necessária entre essas duas partes, inclusive elicitando alternativas de elementos sendo tanto resolvidos por hardware como por software.

3.4.3 Definição de métricas de qualidade

Nesta etapa são definidos algumas características necessárias de segurança, confiabilidade, manutenção, precisão e desempenho. Estas características influenciam decisões de projeto sobre a solução encontrada. As tarefas desta atividade são mostradas na tabela 3.6.

Tabela 3.5: Tarefas para elicitação dos requistos de software [9]

Entrada	**Tarefa**	**Saída**	**Meta**
Análise de requisitos de alto nível (pré-fase)	**5.1** Definir as variáveis de ambiente	Lista de tipos de variáveis, faixa de valores.	Definir variáveis e faixa de valores enviados pelos sensores
	5.2 Determinar as funções de SW	Relação de funções SW	Delimitar as funções que serão realizadas por intermédio de SW e das telas de mensagens intermediárias.
	5.3 Delimitar as exceções	Lista das exceções de SW	Fixar as funções de SW frente à ocorrência de exceções de SW
	5.4 Definir as funções de interrupções	Listagem de funções de SW em face de interrupções identificadas	Definir as funções e as ações em decorrência de interrupções
	5.5 Caracterizar os requisitos de idioma	Lista de idiomas do produto.	Indicar os idiomas do software embarcado
	5.6 Estabelecer a interfac e de comunicação (software)	Relação das interfaces de comunicação com outros softwares	Estabelecer as variáveis de comunicação com outros softwares embarcados
	5.7 Indicar as funções de monitoramento	Lista com as funções de monitoramento por SW	Delimitar as funções de SW para o monitoramento
	5.8 Definir as funções de armazenamentode dados	Lista com as fun ções de armazenamento e aquisição de dados	Definir as funções de SW e os tipos de dados a serem armazenados

Tabela 3.6: Tarefas de definição de métricas de qualidade [9]

Entrada	**Tarefa**	**Saída**	**Meta**
Análise de requisitos de alto nível (pré-fase)	**6.1** Definir o grau de segurança	Lista com o grau de segurança do produto	Definir o grau de segurança oferecido pelo produto
	6.2 Caracterizar o desempenho	Relação do grau de desempenho	Determinar o grau de desempenho do produto
	6.3 Indicar as métricas de manutenção	Listagem das métricas de manutenção	Indicar as métricas a serem utilizadas para a manutenção do produto

3.4.4 Definição de requisitos de produção

Nesta etapa é levada em consideração a produção do produto, por exemplo, se os componentes presentes necessitam de uma técnica específicas de soldagem, se o tamanho do produto exigirá um layout de placa de multicamadas e diversos outros fatores.

Tabela 3.7: Definição de requisitos de linhas de produção [9]

Entrada	**Tarefa**	**Saída**	**Meta**
Análise de requisitos de alto nível (pré-fase)	**7.1** Definir os aspectos de produção **7.2** Indicar a embalagem	Lista com aspectos para a linha de produção Relação das características da embalagem do produto	Definir os aspectos gerais para a linha de produção Identificar a embalagem do produto

3.5 Opinião de Profissionais

O autor, em [9], elaborou questões sobre o GERSE e as submeteu a quatro projetistas de SE's. As tabelas 3.8, 3.9, 3.10, 3.11 e 3.12 sumarizam as respostas dadas por estes para peguntas sobre, respectivamente, a clareza e a completude das tarefas da pré-fase, das atividades de alto-nível, da identificação de hardware e software, das atividades de métricas de qualidade e produção e ainda uma avaliação geral sobre o GERSE. Através destas respostas o autor chega a conclusão que o guia contribui para o desenvolvimento de SE's de qualidade.

Porém alguns problemas foram levantados pelos entrevistados, observando a tabela 3.8 pode-se ver a necessidade de uma melhor especificação das tarefas relacionadas aos stakeholders.

Já a etapa de requisitos de alto-nível (tabela 3.9) necessita de um estudo para facilitar a transição de requisitos de alto-nível para baixo-nível.

Na definição de harware e software, os entrevistados levantaram a questão já mencionada sobre como o design de hardware influencia o design de software e vice-versa.

O autor também evidencia que a etapa de requisitos de produção causou muitas divergências, os entrevistados citaram que a produção não faz parte do design de um produto, por necessita-

Tabela 3.8: Grupo de questões quanto à clareza e a completude do guia durante a pré-fase [9]

Questões	**Concordo Totalmente**	**Concordo parcialmente**	**Discordo Parcialmente**	**Discordo Totalmente**
As atividades para organização de contexto são suficientemente claras.	100%	0%	0%	0%
As atividades para organização de contexto estão completas.	50%	50%	0%	0%
As atividades para definição de stakeholders são suficientemente claras.	50%	25%	25%	0%
As atividades para definição de stakeholders estão completas.	50%	25%	25%	0%
As atividades para definição de requisitos de alto nível são suficientemente claras.	50%	50%	0%	0%
As atividades para definição de alto nível estão completas.	50%	50%	0%	0%

Tabela 3.9: Grupo de questões quanto as atividades de alto nível [9]

Questões	**Concordo Totalmente**	**Concordo parcialmente**	**Discordo Parcialmente**	**Discordo Totalmente**
As atividades para definição de requisitos de alto nível são suficientemente claras.	100%	0%	0%	0%
As atividades para definição de alto nível estão completas.	50%	50%	0%	0%
Através da análise das atividades de requisitos de alto nível é possível transitar e converter facilmente para as atividades de identificação de hardware e software.	50%	25%	?25%	0%

Tabela 3.10: Respostas do grupo de questões de identificação de hardware e elicitação de funções de software [9]

Questões	**Concordo Totalmente**	**Concordo parcialmente**	**Discordo Parcialmente**	**Discordo Totalmente**
As atividades para identificação de hardware são suficientemente claras.	100%	0%	0%	0%
As atividades para identificação de hardware estão completas.	50%	50%	0%	0%
As atividades para elicitação de funções de software são suficientemente claras.	50%	25%	25%	0%
As atividades para elicitação de funções de software estão completas.	50%	25%	25%	0%

Tabela 3.11: Respostas do grupo de questões de métricas de qualidade e de produção [9]

Questões	**Concordo Totalmente**	**Concordo parcialmente**	**Discordo Parcialmente**	**Discordo Totalmente**
As atividades para definição de métricas de qualidade são suficientemente claras.	100%	0%	0%	0%
As atividades para definição de métricas de qualidade estão completas.	50%	50%	0%	0%
As atividades para definição de métricas de produção são suficientemente claras.	50%	25%	25%	0%
As atividades para elicitação de funções de software estão completas.	50%	25%	25%	0%

rem de métricas totalmente diferentes, e muitas vezes essa produção é terceirizada, não cabendo à empresa desenvolvedora este controle. Também foi citado um problema em relação à embalagem e questões de marketing, que também são questões que não são discutidas pelos projetistas de hardware e software.

Tabela 3.12: Respostas sobre a avaliação geral do GERSE [9]

Questões	**Concordo Totalmente**	**Concordo parcialmente**	**Discordo Parcialmente**	**Discordo Totalmente**
O guia apresentado é claro suficiente para ser utilizado em um projeto de sistemas embarcados de pequeno e médio porte.	50%	50%	0%	0%
O guia apresentado é completo e atende as necessidades para projetos de sistemas embarcados de pequeno e médio porte.	50%	25%	25%	0%
Adotaria o guia apresentado para elicitação de requisitos de projetos futuros.	50%	25%	25%	0%
O guia apresentado é de fácil utilização.	50%	50%	0%	0%
O guia apresentado contribui na melhoria da qualidade de desenvolvimento de sistemas embarcados.	50%	50%	0%	0%
O guia apresentado atende as suas necessidades de definições de requisitos em projetos de sistemas embarcados.	50%	50%	0%	0%

Apesar desses pontos levantados pelos entrevistados, a tabela 3.12 mostra que a avaliação geral do GERSE é positiva. Portanto a contribuição que ele trouxe é válida.

Capítulo 4

Framework I*

Nesta capítulo é apresentado o i*, ferramenta que servirá para trazer algumas melhorias necessárias ao GERSE (visão geral e análise organizacional). Na seção 4.1 é apresentada uma visão geral sobre o i*, na seção 4.2 é apresentado o modelo de dependências estratégicas, na seção 4.3 é mostrado o modelo de razões estratégicas e por fim, na seção 4.5 são feitas algumas considerações finais sobre o capítulo.

4.1 Visão geral

O I* é apresentado por Eric Yu em [13] como um ferramenta auxiliar ao processo de ER. O autor identifica como um problema o fato da análise tradicional de requisitos tomar uma visão de atividades e entidades como totalmente conhecidas e preditíveis. Portanto no I* adota-se uma visão social do mundo, onde existem atores e estes atores possuem intenções e são apenas parcialmente previsíveis. Isto é claramente visto na citação: "Se buscamos requisitos que atendam aos desejos e necessidades de *stakeholders*, precisamos visualizar os *stakeholders* como atores tendo interesses estratégicos, não apenas um envolvimento operacional com o sistema proposto."(traduzido de [13])

O I* é orientado a objetivos e atores, que podem ser tanto humanos quanto sistemas tecnológicos, portanto, como dito em [14], pode-se incluir o sistema como um ator dentro da organização, onde os outros atores possuem dependências e intencionalidades sobre o mesmo.

O I* é contituído de dois modelos, o de dependências estratégicas (SD) e o de razões estratégicas (SR). O SD tem como objetivo capturar e descrever as intenções por trás de um processo. Já o SR, que é complementar ao SD, busca detalhar as relações entre os atores ao descrever as

razões por trás de cada elemento de um processo [13] [14].

Na figura 4.1 são mostradas as entidades do I*. Estas entidades são usadas tanto no modelo SD quanto no SR e o contexto em que elas são utilizadas será explicado nas seções seguintes, onde serão detalhados o modelo SD (seção 4.2) e o SR (seção 4.3).

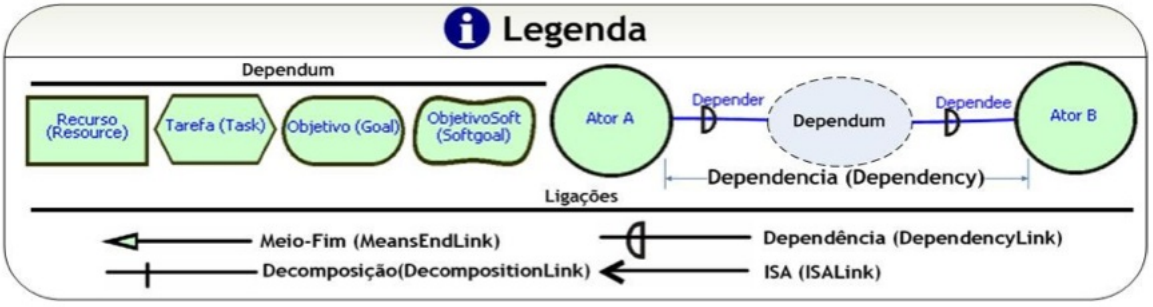

Figura 4.1: Entidades dos modelos do I* [14]

4.2 Modelo de Dependências Estratégicas (SD)

Este modelo provê ferramentas e um nível de abstração para modelar apenas os relacionamentos externos entre os atores [14]. Os atores são entidades que possuem conhecimentos (*know-how*) para atingir determinados objetivos, desta forma um ator busca relacionamentos com outros atores para atingir objetivos que ele não conseguiria atingir (pelo menos não de forma fácil), ou por não possuir *know-how*, ou recursos necessários, ou por não conseguir realizar uma determinada tarefa sozinho [13].

Para representar isto, o SD se utiliza de uma rede com relações de dependências entre atores, onde os atores são representados como nós e as ligações são as dependências entre os mesmos. Dependendo da posição de um ator em relação à outro em uma dependência ele pode ser classificado em uma de duas categorias: ***Depender*** e ***Dependee***. O *Depender* é o ator que depende de outro para atingir um objetivo, realizar uma tarefa ou obter um recurso. O *Dependee* é o ator sobre o qual o *Depender* possui uma dependência, ou seja, é o ator do outro lado da relação. Estes objetivos, tarefas ou recursos, que são o centro da dependência, são chamados de ***Dependum*** [13]. Desta forma as relações terão a seguinte forma: *Depender* -> *Dependum* -> *Dependee*, como resumido em [14].

Sendo assim existe no I* quatro categorias de dependências: os objetivos, as tarefas, os recursos e os *softgoals*, que trazem a noção de requisitos não-funcionais. Estes quatro elementos também são mostrados na figura 4.1.

Cada uma destas categorías de dependências levam à uma situação diferente, como explicado em [13]. Nas dependências de objetivos, o depender necessita do dependee para atingir um objetivo sobre o qual ele se preocupa apenas com o resultado e não como será feito, ficando a cargo do dependee o *know-how* para isto.

Nas dependências de tarefas o depender possui o *know-how* e escolhe como será feita a tarefa, porém depende de outro ator (o dependee) para realizá-la. Em [13], o autor diz que as dependências de tarefas devem ser vistas como restrições no desempenho das mesmas pelo depender.

Nas dependências de recursos, o depender necessita de um recurso que outro ator (o dependee) pode fornecer.

Já nas dependências de *softgoals* o depender depende do dependee para realizar tarefas que atingirão determinado requisito não-funcional.

É importante frizar que ao ser depender em uma ou mais relações o ator torna-se vulnerável, já que os *dependee's* dessas relações podem falhar, seja em disponibilizar um recurso, em realizar uma tarefa ou em atingir um objetivo.

A figura 4.2, retirada de [15], mostra um modelo SD de exemplo. Este exemplo refere-se à uma situação muito comum, onde o sistema auxilia no processo de compra e venda em uma empresa.

Neste exemplo existem três atores, o cliente, o funcionário e o sistema. O cliente tem o objetivo de fazer uma compra, e ele depende do funcionário para isto, que deve saber os processos e técnicas da empresa para vender um produto para o cliente (*know-how*), portanto mapeado como uma dependência de objetivo. No entanto, para realizar a venda, o funcionário necessita de um recurso informacional que são os dados para pagamento, que deverão ser fornecidos pelo cliente [13].

O funcionário também possui objetivos que dependem do sistema, como consultar produtos, efetuar vendas, e emitir notas fiscais. O sistema deverá saber os processos internos necessários para atingir tais objetivos para o funcionário. O funcionário também deseja que o sistema seja

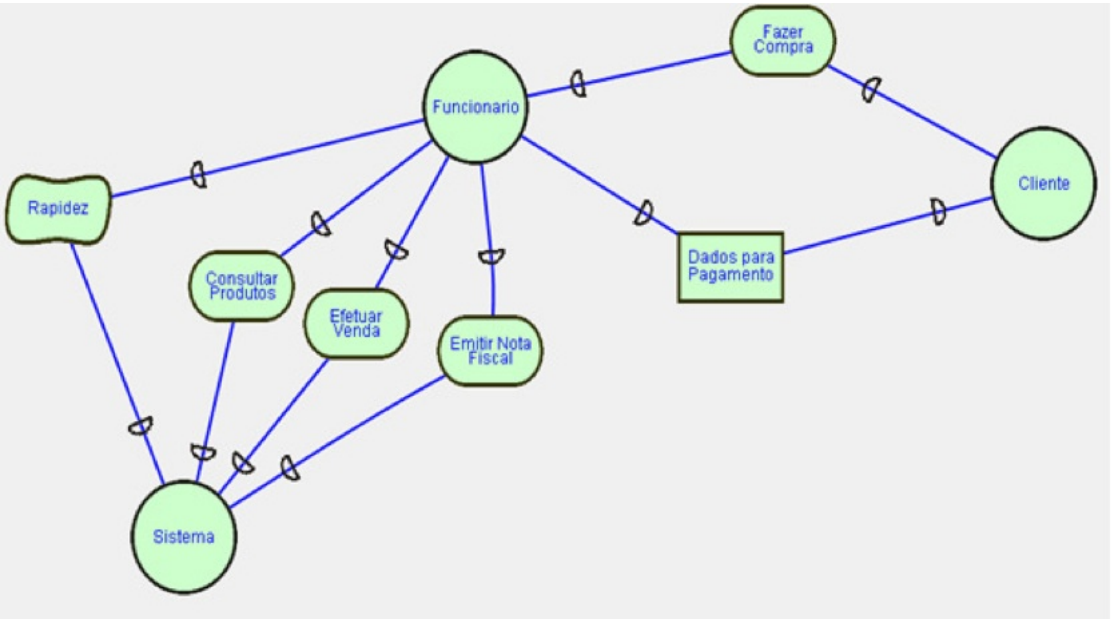

Figura 4.2: Modelo SD onde um sistema auxilia no processo de compra e venda [15]

rápido para não causar inconvenientes ao cliente. Esta necessidade de ser rápido do sistema é um requisito não-funcional, portanto modelado como um *softgoal* [13].

4.2.1 Análise

Em [13] também são apresentadas algumas análises que podem ser feitas sobre o modelo SD para se obter um melhor entendimento sobre os processos. Como dito na seção 4.2, um ator, ao fazer uma relação de dependência, mesmo se tornando vulnerável, busca atingir um objetivo que ele por sí só não conseguiria, abrindo assim diversas oportunidades à esse ator. Desta forma o SD pode ser usado para propor novas relações entre os atores presentes no modelo, relações estas que podem significar mais eficiência, menos vulnerabilidade ou melhoramento em qualquer outro fator de um processo, ou até novos processos que antes não eram vistos como possíveis. Estas novas relações surgem da análise dos desejos de um ator com as capacidades de outro, podendo trazer diversas novas oportunidades a cada um deles.

A análise pode levar a mudanças na configuração de modo a diminuir vulnerabilidades. Em [13] diz-se que uma dependência é forte se o depender pode causar falhas em dependências do dependee. Por exemplo, no modelo apresentado na figura 4.2, o cliente depende do funcionário para realizar o seu objetivo de fazer compra, porém o funcionário (ou a empresa que ele está

representando) depende do cliente para se manter no mercado. Esta mútua dependência torna a relação forte, o que torna a possibilidade dos objetivos de ambos serem atingidos, muito grande.

Em [13] diz-se que uma análise do SD também revela quem deverá lidar com os problemas e quais objetivos são afetados caso uma dependência não seja satisfeita. Por exemplo, no modelo de exemplo (figura 4.2), o objetivo do cliente de realizar uma compra é severamente prejudicado caso o sistema não consiga satisfazer os objetivos do funcionário de consultar produtos ou realizar a venda, cabendo ao funcionário ter alternativas caso isto aconteça.

Portanto através da análise do SD pode-se mudar a configuração das dependências de modo a ter um processo mais seguro e mais completo.

4.3 Modelo de Razões Estratégicas (SR)

O modelo SR é complementar ao SD e visa modelar as razões associadas a cada dependência de cada ator, permitindo assim a identificação de elementos de processos . Sendo assim o SR é muito útil para a ER, pois através desse é possível ver como o sistema estará inserido na rotina dos atores [15]. No SR são inseridos elementos de processos que representam as razões, os passos e as alternativas para um processo. Desta forma pode-se gerar e explorar sistematicamente diversas alternativas, de modo a encontrar novas formas para os processos, formas que favoreçam aos interesses dos atores [13].

As entidades que permitem essa expressividade do SR são basicamente as mesmas que representam os *dependum's* no SD (objetivo, tarefa, recurso e *softgoal*), porém agora ligadas entre si, e com *links* (ligações) diferentes (*means-ends* e *decomposition*, também presentes na figura 4.1). O *link means-ends* (meio-fim) representa a relação entre um fim e o meio de satisfazê-lo. Como fim podemos ter objetivos, tarefas, recursos ou *softgoals*, e os meios são geralmente expressos como tarefas. Já o *link decomposition* (decomposição) é uma ligação entre uma tarefa e seus componentes. Dependendo dos nós envolvidos em uma relação de decomposição ela assume uma entre quatro configurações: *subgoal* (sub-objetivo - o elemento ligado à tarefa é um objetivo), *subtask* (sub-tarefa - o elemento ligado à tarefa é outra tarefa), *resourceFor* (recurso-para - o elemento ligado à tarefa é um recurso) e *softgoalFor* (*softgoal*-para - o elemento ligado à tarefa é um *softgoal*).

Os nós (objetivos, tarefas, recursos e *softgoals*) possuem significado semelhante aos seus

correspondentes no SD. Um objetivo é um condição desejada, sobre o qual não há preocupação em como ele será atingido, uma tarefa é um meio específico de se fazer algo, um recurso é uma entidade física ou informacional que não é considerada problemática para o ator, a não ser que ela seja obtida de outro ator através de uma dependência externa mapeada no SD (os *links* do SR podem estar ligados à dependências externas ao ator), e um *softgoal* é uma condição desejada onde não se tem uma clareza de como pode ser atingida [13].

A figura 4.3 mostra o modelo SR gerado a partir do modelo SD presente na figura 4.2, e, assim como esta, foi retirado de [15].

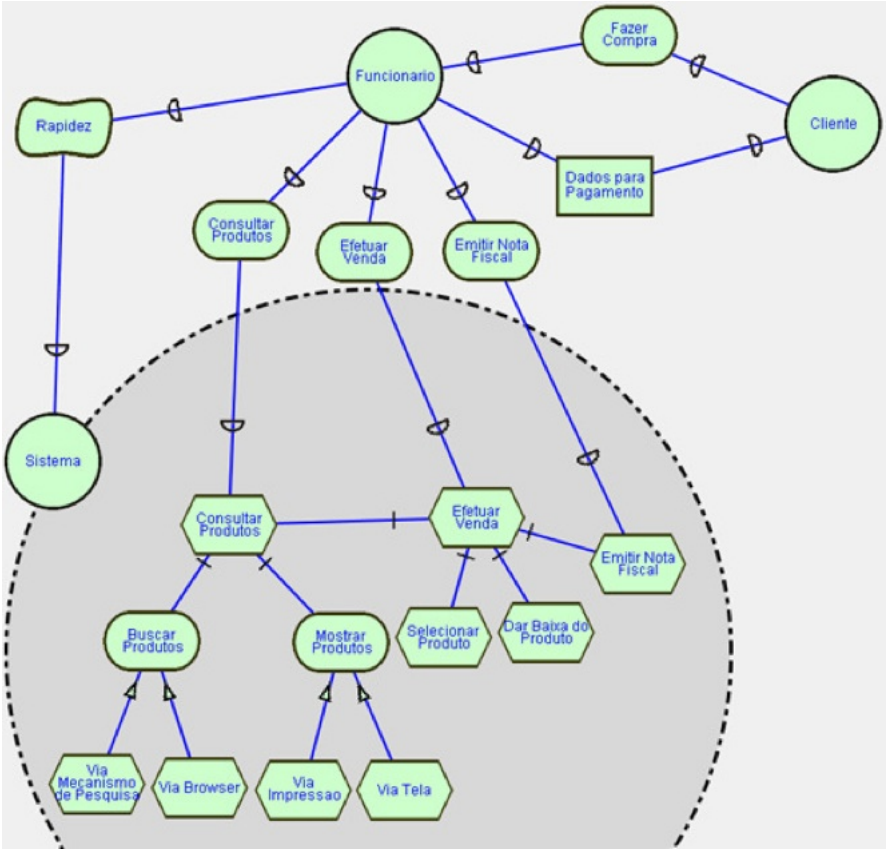

Figura 4.3: Modelo SR de exemplo, gerado a partir do modelo SD da figura 4.2 [15].

O círculo que contém em sua borda o ator representa a "região"do ator (traduzido de [13] - *actor boundary*), dentro dela estão as rotinas que ele possui para satisfazer as dependências dos outros atores. Cada dependência que possui este ator como *dependee* agora possui uma tarefa correspondente à ela dentro da região do ator. Estas tarefas são decompostas em objetivos, tarefas, recursos ou *softgoals* através de relações de decomposição, e o mesmo vai sendo feito com

os novos elemento até chegar a uma tarefa final, que representa um meio de atingir determinado fim (relação meio-fim).

Isto pode ser visto na figura 4.3, onde efetuar uma venda envolve consultar produtos, selecionar produtos, dar baixa nos produtos e emitir a nota fiscal. Já a tarefa de consultar produtos envolve buscar os produtos e exibí-los. Buscar produtos pode ser feito através de mecanismo de pesquisa ou via *browser*, e a exibição pode ser feita através da tela ou via impressão. Mostrando assim os passos e as alternativas para satisfazer determinados objetivos.

4.4 Agentes, papéis e posições

Há também nos modelos do i* a noção de agentes, papéis e posições, que são especializações dos atores, e permitem uma distinção entre as dependências que são atribuíveis aos mesmos. Os agentes podem ocupar posições ou fazer algum papel dentro de uma organização. Isto traz duas novas relações: *occupies* e *plays*.

Requisitos de qualificações necessárias geralmente pertencem à posições e papéis, enquanto que treinamento e conhecimento pertencem à agentes, já que estes acompanham o agente [13].

A figura 4.4 contém a representação gráfica destes elementos.

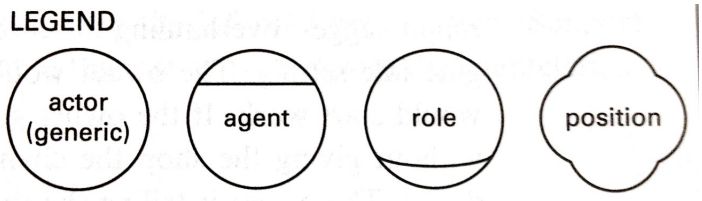

Figura 4.4: Representação gráfica dos elementos agentes, papéis e posições [13].

4.5 Considerações finais

Esta capítulo mostrou as entidades do i* e como elas são montadas em um modelo de forma a representar uma organização. Estas entidades e modelos serão usadas, no capítulo 6 de proposta, para modelar o ambiente de uso e o ambiente organizacional do SE, para apoiar as atividades de pré-fase e fase do GERSE.

Capítulo 5

NFR Framework

Neste capítulo será apresentado o NFR Framework (Non-Function Requirements Framwork), proposto por Chung em [16]. Este framework será utilizado na proposta para a análise dos requisitos não funcionais envolvidos. Na seção 5.1 é apresentado uma visão geral sobre o framework e na seção 5.2 é apresentado o gráfico SIG e alguns passos de como construí-los.

5.1 Visão geral

O NFR Framework visa descrever **como** o sistema deve fazer algo e nao **o que** deve fazer [16]. O NFR Framework trata os requisitos não funcionais como requisitos de qualidade, que devem ser satisfeitos. Uma das principais características é a noção de contribuição entre os elementos, que podem contribuir positivamente, negativamente ou parcialmente [16]. Formando um conjunto de ferramentas que permitem analisar e operacionalizar requistitos não-funcionais. Operacionalizar um requisito não-funcional significa gerar um conjunto de elementos tangíveis que devem satisfazer ou contribuir com relação ao requisito.

Os elementos que permitem esta expressão no NFR Framework são: os softgoals e as interdependências [17].

Os softgoals são os principais elementos, e o centro de toda a análise e design inerente do framework. Eles possuem três especializações, softgoals de requisitos não-funcionais, softgoals de operacionalização (ou hardgoals) e softgoals declarativos. Os primeiros trazem a noção de restrições ao sistema, já o segundo fornece mecanismos para para elicitar elementos mais concretos relacionados aos softgoals de requisitos não funcionais. Esses são as operações, pro-

cessos, representação de dados, estruturação, limitação e agentes no sistema. E finalmente o último, fornece forma de justificar decisões tomadas na etapa de design [17].

Os softgoals também podem ter marcações qualitativas, indicando criticidade, aceitação ou rejeição dos mesmos. A figura 5.1 mostra os softgoals e as marcações qualitativas.

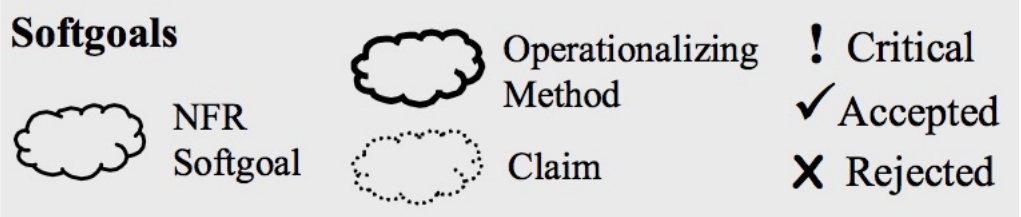

Figura 5.1: Tipos de softgoal do NFR-*Framework* [18]

As interdependências são as relações entre os softgoals mostrados anteriormente. Elas visam refinar os softgoals de forma a torná-los mais claros, com uma carga de subjetividade menor, até se chegar aos hardgoals, que é quando ocorre a operacionalização do softgoal inicia. O NFR Framework possui três tipos de refinamentos: a decomposição, a operacionalização e a argumentação. Na decomposição a interdependência refina o softgoal em outros softgoals tentando melhorar a compreensão sobre este softgoal. Na operacionalização o requisito não funcional é refinado em um hardgoal, que representa um componente concreto e definido. Esta transição é a mais importante do framework, pois representa a possibilidade de se satisfazer o softgoal. Por fim a argumentação serve para anotar razões de projeto [17].

As interdependência de refinamento consistem de ligação AND e OR, mostrando como esses refinamentos se relacionam com o softgoal "pai". Um softgoal refinado em dois softgoals com a relação AND, significa que para ele ser satisfeito, ambos os softgoals "filhos"devem ser satisfeitos. Já um refinamento em softgoals com a relação OR, significa que qualquer um dos "filhos"sendo satisfeitos já satisfaz o pai.

As interdependências de operacionalização são as ligações "make", "help", "hurt e "break". A relação "make"representa que um softgoal satisfaz totalmente outro softgoal. A relação "help"representa que um softgoal satisfaz parcialmente outro softgoal. A relação hurt, significa que um softgoal prejudica parcialmente outro softgoal. E a relação "break"significa que um softgoal prejudica completamente outro softgoal.

A figura 5.2 mostra esses relacionamentos, juntamente com as relações explicitas e implíci-

tas. Que como o próprio nome diz, representa se os softgoals se relacionam explicitamente ou implicitamente.

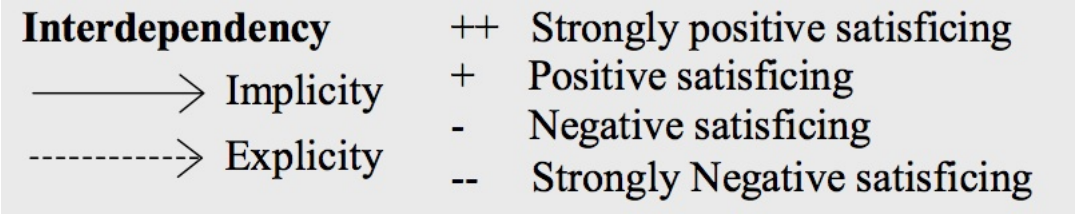

Figura 5.2: Tipos de interdependências do NFR-*Framework* [18]

5.2 SIG - Softgoal Interdependency Graph

O SIG (Softgoal Interdependency Graph - Gráfico de interdependências de Softgoals) é o modelo que permite gerar e analisar operacionalizações para softgoals elicitados. Para isto ele se utiliza dos elementos descritos na seção anterior (5.1).

Para a construção destes modelos em [18] são enumerados 7 passos que servem para auxiliar este processo. Estes passos estão descritos abaixo:

1. Desenvolver os softgoals não funcionais como objetivos, juntamente com suas decomposições em uma hierarquia, utilizando as interdependência AND e OR.
2. Definir métodos de operacionalização através de hardgoals para cada um dos softgoals do passo anterior.
3. Identificar correlações entre os softgoals.
4. Desenvolver as criticidades dos objetivos.
5. Analizar as razões de design.
6. Identificar cenários de operacionalização que melhor satisfazem os requisitos
7. Relacionar as decisões tomada como requisitos funcionais no sistema em questão.

A figura 5.3, retirada de [18] mostra um exemplo de um gráfico SIG.

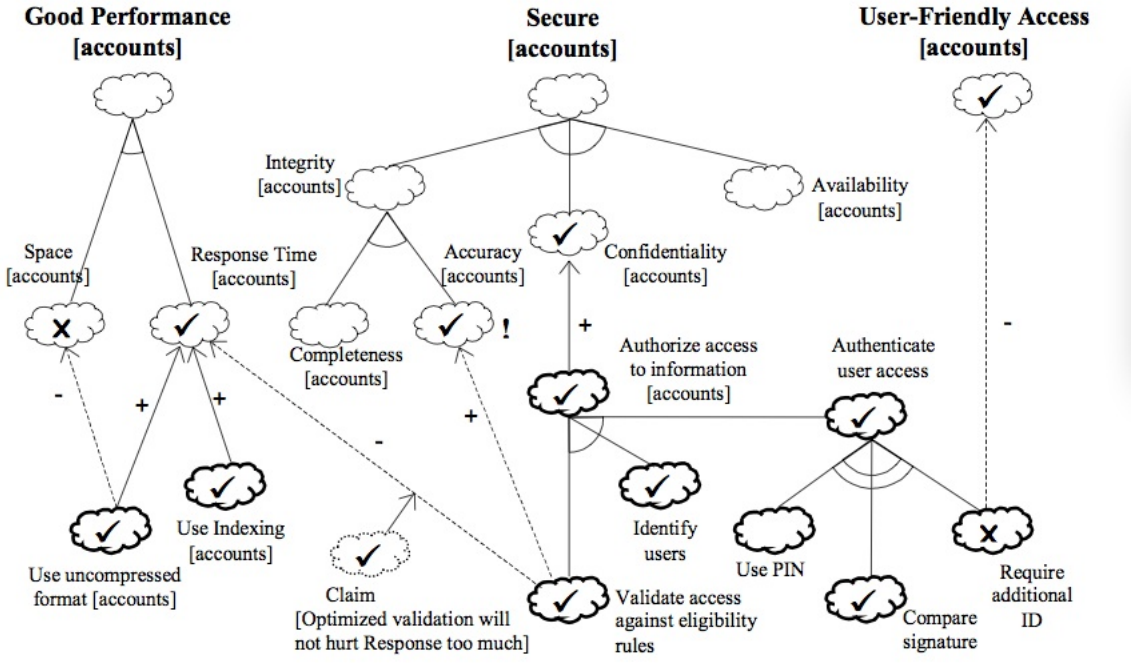

Figura 5.3: Exemplo de gráfico SIG [18]

Capítulo 6

Proposta

Neste capítulo será apresentado a proposta de elicitação de requisitos para sistemas embarcados através dos *frameworks* i*(istar) e NFR(Non-Functional Requirements) . Cabe ressaltar que este dois *frameworks* são utilizados para apoiar o processo de engenharia de requisitos proposto em [9]. A idéia é auxiliar o GERSE na elicitação de requisitos, mostrando como obter determinadas informações por meio da construção de modelos aqui propostos.

A proposta será apresentada na forma de diretrizes, que estão divididas em passos, que representam alguns objetivos da proposta. A cada diretriz é demonstrado um exemplo de aplicação da mesma. Estes exemplos foram retirados do estudo de caso que será apresentado na íntegra no capítulo 7. Este estudo de caso 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 delas. Este estudo de caso foi inicialmente apresentado em [9] e adaptado na presente proposta.

Na seção 6.1 é apresentado uma visão geral da proposta. Nas seções 6.2, 6.3, 6.4 e 6.5, são apresentados os passos 1, 2, 3 e 4 respectivamente, juntamente com suas diretrizes e exemplos. E por fim, na seção 6.6 são feitas as considerações finais do capítulo.

6.1 Visão geral

Como mostrado no Capítulo 3, o GERSE consiste de diversos passos que ao final resultam nos requisitos para desenvolvimento do SE, porém o mesmo não apresenta ferramentas para geração de diagramas e modelos que possam auxiliar nesta elicitação. O mesmo também tem um foco maior nas categorias de requisitos que se devem gerar (qualidade, hardware, software,

etc) e nao em como elicitá-las, foco desta proposta.

Esta proposta possui quatro passos principais, cada um com suas diretrizes. A tabela 6.1 apresenta esses passos, juntamente com o objetivo principal de cada um deles. O passo 1 visa identificar 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, tudo isto através de um modelo SD. 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 através da derivação 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, para isto é utilizada a abordagem do NFR *Framework*. E finalmente o passo 4, onde busca-se elicitar os requisitos de produção e desenvolvimento do SE, por meio de um SR.

Tabela 6.1: Passos da proposta

Passos	**Objetivos**
Passo 1 - Construir o modelo SD (Dependências Estratégicas) do produto inserido em seu contexto de uso.	Identificar usuários e dispositivos que se relacionarão com produto e elicitar requisitos funcionais, de ambientes, de qualidade e de alimentação.
Passo 2 - Construir o modelo SR (Razões Estratégicas) do produto.	Gerar formas e alternativas de se implementar os requisitos de alto nível do Passo 1 e elicitar requisitos de baixo nível, como requisitos de hardware, de software e mecânicos.
Passo 3 - Construção de um modelo NFR com requisitos de qualidade.	Escolher entre as alternativas geradas no Passo 2.
Passo 4 - Construir o modelo SR do produto no contexto de desenvolvimento e produção.	Elicitar requisitos de produção e desenvolvimento.

Nas seções a seguir são explicados cada um dos passos da proposta, juntamente com suas diretrizes, apresentando exemplos para ajudar no entendimento da aplicação das mesmas.

6.2 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. No capítulo 4 foram descritos os elementos que integram o i* e que agora devem ser utilizados para representar estes relacionamentos. Neste passo são elicitados e representados na forma de dependências, os requisitos funcionais, os não funcionais, requisitos de ambiente e o tipo de alimentação do sistema embarcado pretendido, cobrindo parte da atividade de elicitação de requisitos de alto nível proposto pelo GERSE. A tabela 6.2 mostra as diretrizes deste passo. Cada uma destas diretrizes são explicadas e exemplificadas a seguir.

Tabela 6.2: Diretrizes do passo 1.

Diretrizes
1 - Inserir o produto como um ator no modelo.
2 - Identificar os usuários do produto e mapeá-los como atores no modelo.
3 - Identificar outros componentes que terão interface com o produto e mepeá-los como atores no modelo.
4 - Identificar os requisitos funcionais do produto para cada um dos usuários. Estes serão objetivos do usuário (depender) em relação ao SE (dependee).
5 - Identificar para os componentes, requisitos (dependências) entre eles e o SE. Estes serão objetivos do componente (depender) em relação ao SE (dependee).
6 - Identificar os requisitos relacionados ao ambiente (subdiretrizes).
7 - Definir para cada um dos agentes e atores qual a alimentação ideal que o SE deve ter que seria ideal para eles.
8 - Identificar para cada usuário os requisitos de qualidade. Estes serão *softgoals* entre ele e o SE (subdiretrizes).

Diretriz 1: Inserir o produto como um ator no modelo.

Isto é parte fundamental do processo, pois 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 6.1 mostra este ator, o mesmo é o terceiro da esquerda

para a direita.

Diretriz 2: Identificar os usuários do produto e mapeá-los como atores. Fazer as devidas associações entre os mesmos, por exemplo, se um usuário é uma especialização de outro usuário, pode-se usar a associação ISA (ver capítulo 4).

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. O jogador profissional consiste de um tipo especial de jogador de xadrez, portanto usa-se a relação ISA, para mostrar que todos os requisitos (dependências) que envolvem o jogador de xadrez também envolvem o jogador profissional, porém este possui alguns específicos somente a ele. Este exemplo pode ser visualizado na figura 6.1, onde existem dois atores (os dois primeiros da direita para a esquerda) representando os dois usuários.

Diretriz 3: Identificar outros componentes que terão interface com o produto. Estes também serão mapeados como atores no modelo.

Esta diretriz busca satisfazer os projetos onde busca-se desenvolver um produto que integre um rede de outros produtos. Ou simplesmente quando se tem um interface alternativa ao SE. Ao inserir esses outros componentes como atores, pode-se mapear a interação entre os mesmos. Foge ao foco deste trabalho, mas isto também abre portas para a elicitação de requisitos destes outros componentes, caso eles já não estejam desenvolvidos.

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, sendo o último da esquerda para a direita na figura 6.1.

Diretriz 4: Identificar os requisitos funcionais do produto para cada um dos usuários. Estes serão 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. Como não importa ao usuário como as dependências serão satisfeitas, cabendo ao produto decidir como realizá-las, elas são inseridas como dependências do tipo objetivo entre o usuário, que é o depender, e o

produto, que desempenha o papel de dependee no relacionamento.

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 são inseridos 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. As dependências de objetivos no modelo SD da figura 6.1 representam os requisitos elicitados neste passo.

Diretriz 5: Identificar para os componentes, requisitos (dependências) entre eles e o SE. Estes podem ser mapeados utilizando-se os elementos propostos pelo *framework* i*.

Aqui são mapeados os requisitos entre os componentes e o SE. Aqui não existe uma determinação de quais elementos do i* utilizar, pois muitas vezes necessita-se um recurso informacional, ou uma tarefa que depende do outro para ser realizada.

Exemplo: Como dito 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, que depende do computador para ser satisfeito.

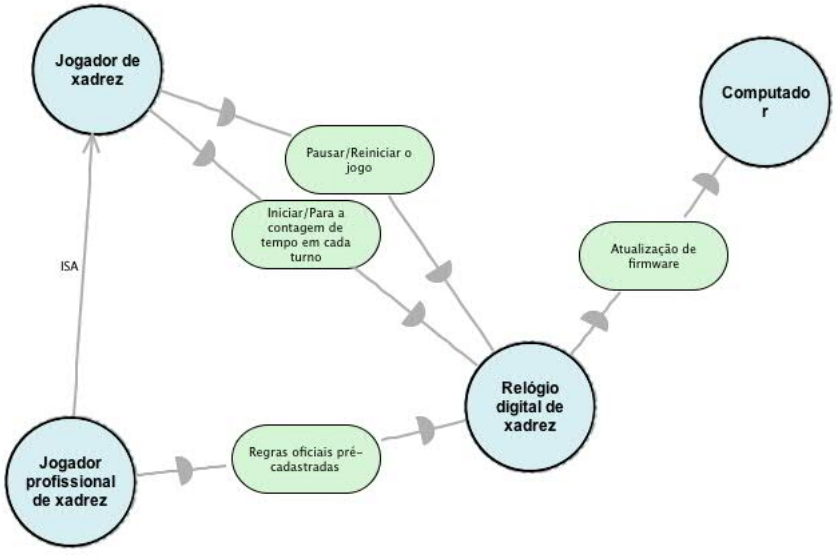

Figura 6.1: Modelo gerado após a diretriz 5 (interação entre o SE e outros componente).

Diretriz 6: Mapear os requisitos relacionados ao ambiente. Estes requisitos serão dependências entre o ambiente e o produto.

O ambiente é algo de muita influência em um SE, pois eles possuem uma interação muito forte um com o outro. O SE atua sobre o ambiente através de atuadores, respondendo a estímulos lidos por sensores. Já o ambiente expõe o SE a diversas situações sobre as quais o SE deve manter funcionamento. E são estas situações que esta diretriz busca elicitar. E as subdiretrizes abaixo ajudam nesta elicitação:

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

O papel é um elemento do i* que representa uma entidade com características comuns à qualquer agente que for tomar o seu lugar. Esta relação entre o papel e o agente que for desempenhar este papel caracteriza uma associação *plays*. Este papel ambiente será um constante na proposta, pois independente do projeto ele estará presente. Ele servirá para identificar os diversos ambientes de uso do SE.

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.

Nesta diretriz são identificados todos os ambientes de uso do produto, e então mapeados para o modelo. Estes serão os ambientes que irão impor situações ao SE.

Exemplo: Os ambientes de uso de um relógio de xadrez, são os mesmos em que acontecem as partidas de xadrez, como exemplo pode-se citar a praia e praças públicas.

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 diretriz 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 a temperaturas altas. Se a faixa de temperatura for determinada, o requisito é mapeado como um objetivo, mas se foi mantido no âmbito qualitativo, tratando o requisito como "Temperatura alta"ele é mapeado como um *softgoal*. O mesmo acontece nas praças públicas, onde existe um risco de queda.

A figura 6.2 mostra o modelos SD da figura 6.1 incrementado com os requisitos de ambiente elicitados nesta diretriz.

Diretriz 7: Definir para cada um dos atores e agentes até aqui elicitados se existe 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 a bateria ou rede elétrica (tomadas), mas podem existir outras situações, como em SE's em automóveis, que na maioria das vezes são alimentados pela bateria do carro.

Exemplo: Começando pelos ambientes, que são os mais determinantes neste requisito de alimentação, a praia e a praça pública têm como ideal alimentação através de 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. O modelo SD com o resultado deste diretriz é mostrado na figura 6.3.

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 diferenciá-los de outros *softgoals*, usar o prefixo RQ para nomeá-los.

Os requisitos de qualidade são os requisitos não-funcionais, e determinam parâmetros para escolha de alternativas e validação do produto final. Para facilitar a elicitação destes requisitos pode-se prestar atenção especial nos requisitos de qualidade da maioria dos sistemas embarcados, conforme expresso nas subdiretrizes abaixo:

Subdiretriz 8.1: Identificar requisitos de qualidade para o consumo de potência.

Exemplo: Duração mínima da bateria.

Subdiretriz 8.2: Identificar requisitos de qualidade das partes físicas e mecânicas (tamanho, peso, resistência, etc).

Exemplo: Deve ser fácil de transportar.

Subdiretriz 8.3: Identificar requisitos de usabilidade.

Exemplo: Deve ser fácil de utilizar.

Subdiretriz 8.4: Identificar requisitos de custo.

Exemplo: Deve ser de baixo custo.

Subdiretriz 8.5: Identificar requisitos de precisão.

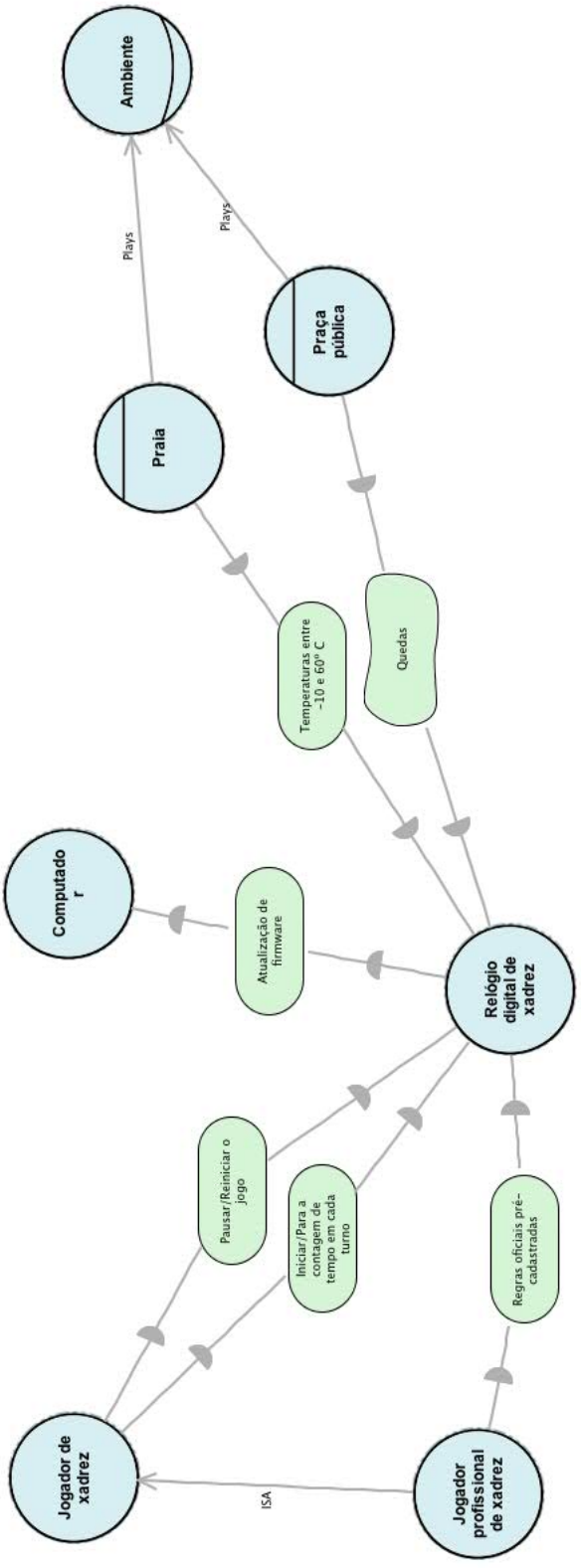

Figura 6.2: Modelo SD do passo 1 incrementado com os requisitos de ambiente.

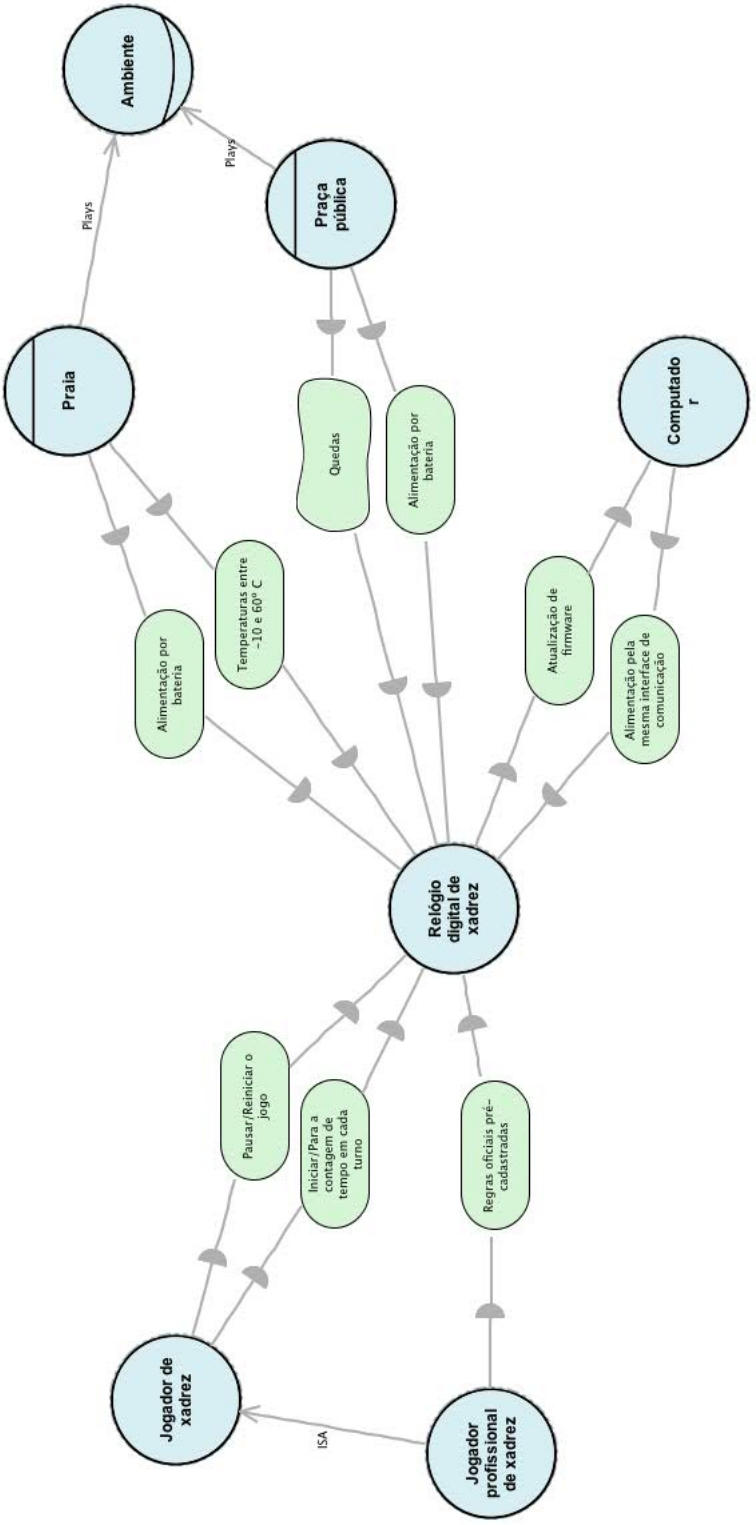

Figura 6.3: Modelo SD com o tipo de alimentação ideal para cada ator.

Exemplo: Marcar o tempo com precisão.

A figura 6.4 apresenta o modelo SD ao final da diretriz 8, que é a última do passo 1.

Ao final deste passo, com o modelo SD foi possível elicitar os requisitos funcionais, os requisitos não-funcionais e os requisitos de ambiente. Estes fazem parte das tarefas 3.1, 3.2 e 3.3 do GERSE respectivamente. Como adicional os requisitos não-funcionais elicitados neste processo incluíram requisitos de consumo de potência e requisitos com respeito a parte mecânica, que são as tarefas 3.4 e 3.5 do GERSE. Nesta proposta os requisitos não-funcionais já são vistos como requisitos de qualidade. O GERSE elicita esses requisitos de qualidade em sua atividade 6.

6.3 Passo 2 - Construir o modelo SR do produto

O modelo SD gerado no passo anterior possui todos os requisitos de alto nível, e o derivamento deste modelo em um SR mostra-se uma forma evidente de como transformar estes requisitos em requisitos de baixo nível, que são os requisitos de hardware, os de software e os requisitos da parte mecânica. As diretrizes e subdiretrizes mostrados na tabela 6.3 formam um processo que tem como objetivo esta elicitação.

Tabela 6.3: Diretrizes do passo 2.

Diretrizes
9 - Elicitar requisitos de baixo nível (subdiretrizes).

Diretriz 9: Abrir a região do ator que representa o SE, e iniciar o processo elicitação de requisitos de baixo nível.

A região de um ator é onde 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. Ou seja, na região do SE são elicitados os elementos que o mesmo deve possuir. Estes serão os requisitos de baixo nível. As diretrizes a seguir visam elicitar estes requisitos e devem ser seguidas para cada uma das dependências do SE, menos para os requisitos de qualidade.

Subdiretriz 9.1: Criar uma tarefa nova para a dependência externa e então ligá-la à mesma.

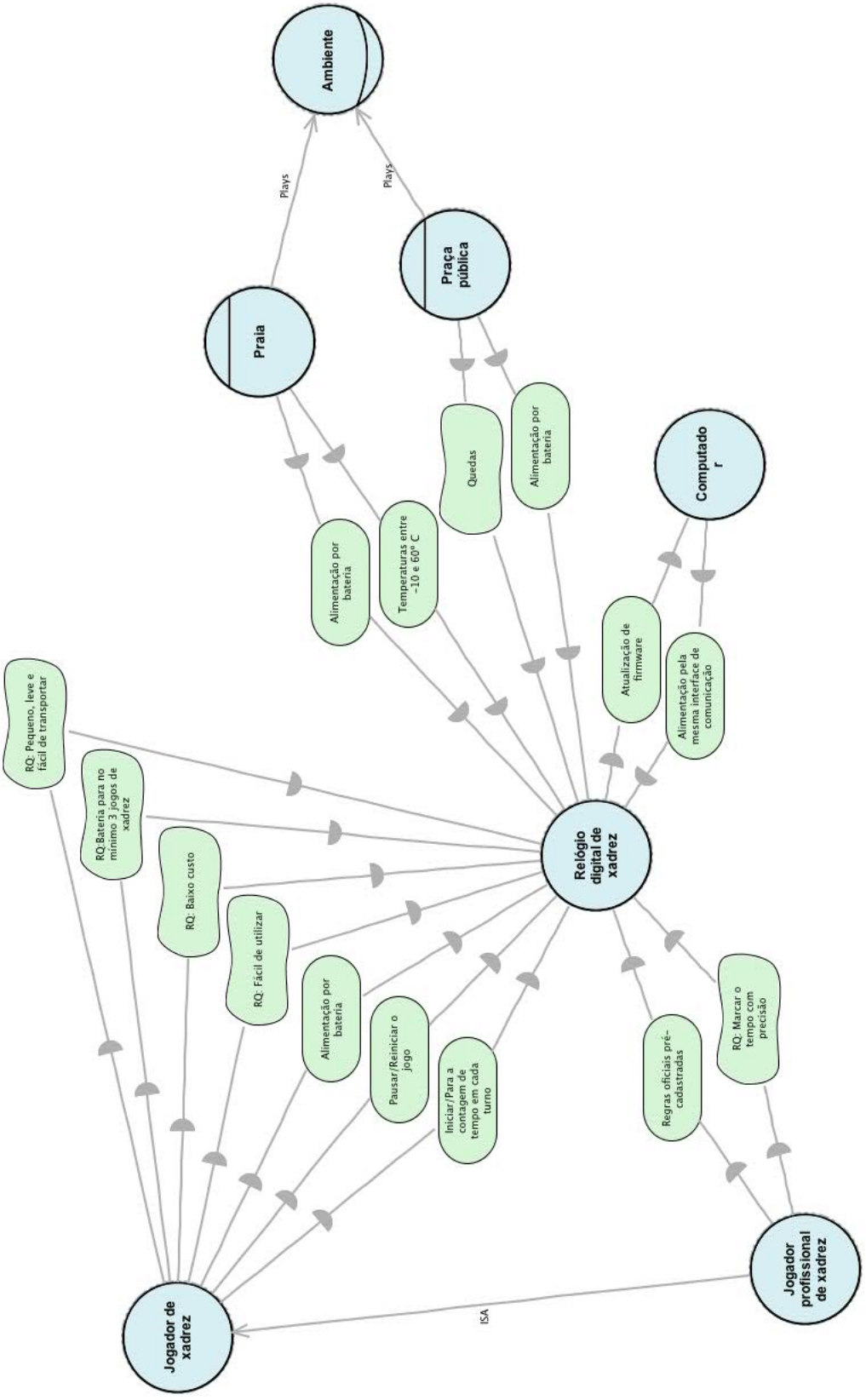

Figura 6.4: Modelo SD final, gerado pelo passo1.

Se o produto já possuir uma tarefa que satisfaça a dependência, basta ligar a dependência à esta tarefa sem a necessidade de criar uma nova.

Subdiretriz 9.2: Dividir o novo elemento em subelementos (tarefas, objetivos ou recursos) necessários para satisfazê-los, 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 se está claro como implementá-la.

Subdiretriz 9.3: Analisar para cada elemento deste requisito a necessidade de ele ser resolvido por hardware ou por software. Para estes elementos inserir recursos representando esta necessidade. Sempre pensar nos recursos de hardware como meios para se atingir um fim, portando sempre ligá-los utilizando o means-ends, isto favorece à 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.

Subdiretriz 9.4: Analisar quais elementos da dependência possuem um requisito mecânico. Estes também serão recursos. Para diferenciá-los dos outros recursos, utilizar o prefixo MEC na sua nomenclatura. Um recurso mecânico, descreve partes mecânicas necessárias, juntamente com características das mesmas.

Subdiretriz 9.5: Para cada recurso de hardware analisar recursos mecânicos ou de software relacionados a ele, estes serão novos recursos no modelo, ligados através de *task-decomposition*.

Exemplo: Uma dependência externa, que ao ser operacionalizada no SR demonstra muito bem esta diretriz é a dependência "Iniciar/Parar a contagem de tempo em cada turno". Isto é mostrado na figura 6.5. Pela figura pode-se ver que foi criado no ator "Relógio Digital de Xadrez"uma tarefa para satisfazer a dependência "Iniciar/Parar a contagem de tempo em cada turno". Esta tarefa 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.

A principal atividade a se estudar neste exemplo é o objetivo "Sinalizar jogada feita", este objetivo pode ser atingido através de três alternativas, que consistem de recursos de hardware. 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 sinalizar 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 uma outra ação do jogador (como apertar um botão por exemplo). Estes três recursos de hardware dependem da interrupção externa, que dá a noção de eventos ao SE, ou seja, ao acontecer a jogada, o SE responde à ela realizando a sua função. Esta interrupção é um recurso de software.

Finalizando o processo para esta tarefa analisa-se a necessidade de requisitos mecânicos. Os dois primeiros recursos de hardware ("HW: 2 botões"e "HW: 1 botão") devem ser botões grandes, resistentes e com molas para retorno, e isto é um novo recurso mecânico inserido. Já o recurso de hardware "HW: Sensores nas casas do tabuleiro"necessita que estes sensores estejam presentes em uma camada abaixo das casas do tabuleiro, senso este um novo recurso mecânico.

Se, por exemplo, o próximo requisito (dependência) a ser analisado fosse "Dar bônus de tempo", não haveria a necessidade de se criar uma nova tarefa para satisfazê-lo, pois o SE já possui a tarefa "Dar bônus de tempo", que é uma subtarefa da tarefa "Iniciar/Parar a contagem de tempo em cada turno, que satisfaz o mesmo.

Ao final deste passo foi possível elicitar requisitos de hardware e software. Atividades 4 e 5 do GERSE. Adicionalmente também foi elicitado requisitos mecânicos, que não são trabalhados no GERSE a nível de características específicas, como é trabalhado nesta proposta.

A principal característica deste passo é incentivar a geração de alternativas para as diversas tarefas do SE. Esta geração sistemática de alternativas aumenta as chances de se gerar requisitos que satisfaçam todos os requisitos de qualidade. Porém não foi determinado quais das diversas alternativas serão realmente implementadas. Para isto vêm o passo seguinte.

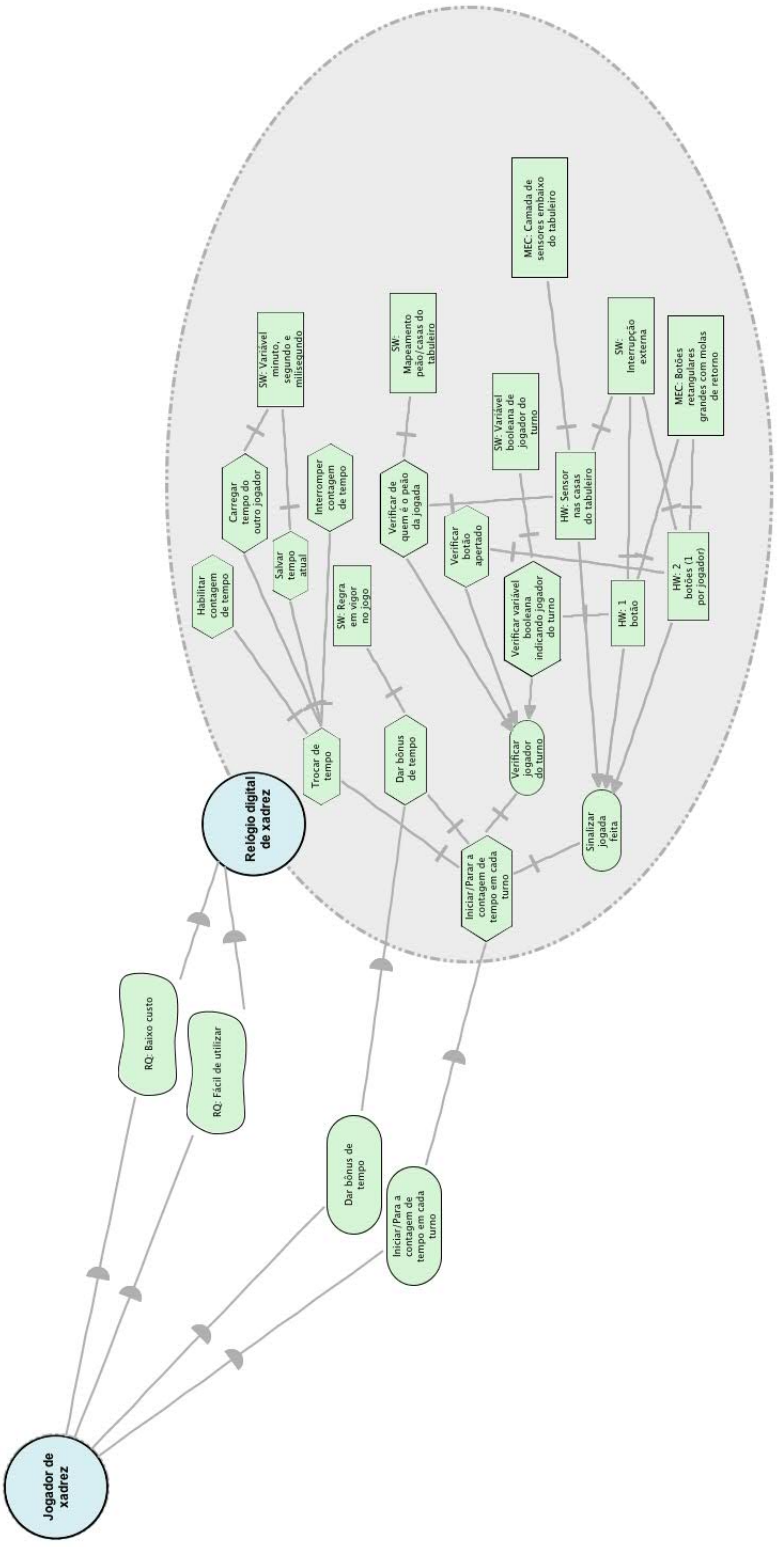

Figura 6.5: Exemplo de modelo SR gerado no passo 2.

6.4 Passo 3 - Construção de um modelo NFR com requisitos de qualidade

O SR modelado no passo 2 gera diversas alternativas para a realização de tarefas, evidenciadas através de ligações *means-ends*. Este passo busca através do NFR escolher entre essas alternativas. O NFR possui uma expressividade que permite a operacionalização dos requisitos de qualidade, de modo a melhor definí-los. E ainda permite analisar as influências de cada elemento de baixo nível nos elementos de alto nível, justificando assim o seu uso para a escolha entre alternativas. No NFR os elementos de baixo nível são representados por *hardgoals* e são elementos concretos, e os de alto nível por *softgoals* que são elementos de qualidade ou que não possuem uma clareza. A tabela 6.4 mostra as diretrizes deste passo.

Tabela 6.4: Diretrizes do passo 3

Diretrizes
10 - Colocar inicialmente os *softgoals* de qualidade elicitados na diretriz 8 do passo 1.
11 - Inserir *softgoals* que representam características de SE's (subdiretrizes).
12 - Inserir *hardgoals* com os meios das ligações means-ends do modelo SR do passo 2.
13 - Juntamente com os *hardgoals* da diretriz anterior trazer os elementos ligados à eles no modelo SR do passo 2. Estees também serão *hardgoals*.
14 - Analisar quais *hardgoals* melhor satisfazem os *softgoals* das diretrizes 1 e 2. Se tornando as escolhas entre as alternativas.
15 - Escolher o microcontrolador do projeto.
16 - Escolher modelos reais dos *hardgoals* de hardware e inserí-los no modelo.
17 - Analisar a integração entre todo *hardgoals* escolhidos.

Diretriz 10: Colocar inicialmente os *softgoals* de qualidade elicitados no modelo SD.

Estes *softgoals* serão a base para as escolhas entre as alternativas. A Figura 6.6 mostra na parte superior um destes *softgoals* de qualidade inserido no NFR.

Diretriz 11: Juntar com estes, outros *softgoals* que representam características específicas de SE's. Porém a necessidade de cada um deles pode variar entre cada projeto. Para identificar se existe a necessidade ou não, verificar as seguintes subdiretrizes.

Subdiretriz 11.1: Se a quantidade de memória for bem limitada em relação ao necessário, insere-se um *softgoal* "Otimização do uso da memória".

Comum em projetos de SE's que já possuem uma arquitetura definida, geralmente de sistemas legados, e devem obedecer a restrições no tamanho da memória.

Subdiretriz 11.2: Se a quantidade de energia for limitada, como em casos em que se usa uma bateria, insere-se um *softgoal* "Otimização do consumo de energia".

Comum em projetos de SE's alimentados a bateria.

Subdiretriz 11.3: Se a performance é importante, insere-se um *softgoal* "Otimização de performance".

Comum em projetos de SE's que controlam variáveis físicas, como por exemplo em controles de *grids* de redes-elétricas, onde o SE deve responder rapidamente à queda de uma subestação de forma a balancear a carga.

Diretriz 12: Colocar em um nível abaixo, como *hardgoals*, 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. Por 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 do passo 2. Cada um deles será um *hardgoal* com uma ligação help entre ele e o *hardgoal* que representa o meio.

Diretriz 14: Através dos *softgoals* e das ligações do NFR *framework*, analisar quais *hardgoals* 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 diretrizes.

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 *hardgoals*, de forma que satisfazê-lo é muito importante. Portanto o mesmo será o ponto de partida da análise.

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.

Subdiretriz 14.3: Verificar quais dos *hardgoals* influenciam no *softgoal* sendo analisado.

Se o *hardgoal* tem influência positiva, ligá-lo com um some+, se for influência negativa, usar o some-, se o *hardgoal* foi suficiente para satifazer o *softgoal*, usar o make, se ele prejudicar completamente, usar o hurt. Também pode ocorrer a situação em que o *hardgoal* ajuda muito na satisfação de um *softgoal*, mas nao satisfaz completamente. Para este caso, pode-se usar o help.

Subdiretriz 14.4: Os *hardgoals* 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 *hardgoals*. Utilizar o símbolo conhecido como "*check*"para simbolizar essa marcação como satisfeito. Na figura 6.6, o *hardgoal* "Display de LCD", possui essa marcação.

Subdiretriz 14.5: Seguir as ligações que estes *hardgoals* possuem com os outros *hardgoals*, e também marcá-los como escolhidos (satisfeitos).

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.

Exemplo: A figura 6.6 mostra o resultado deste processo na escolha dos *hardgoals* retirados da tarefa "Visualizar tempo". O *softgoal* mais importante escolhido foi o "RQ: Fácil de utilizar". O *hardgoal* que colabora positivamente com este *softgoal* é o "HW: Tela de LCD". Este, portanto, é marcados como satisfeito, sendo preferido sobre o outro *hardgoal* "Display 7 segmentos".

Diretriz 15: Baseado nos *hardgoals* 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.

Diretriz 16: Nos *hardgoals* de hardware que foram escolhidos, derivar um novo *hardgoal*

com um modelo real do dispositivo. A relação entre esses dois será de um make.

Exemplo: Como a tela de LCD foi escolhida, é buscada um modelo de tela que possua as características mecânicas necessárias. Neste caso será a tela de modelo LCR-U12864GSF-WH.

Diretriz 17: Após isto, pode-ser fazer uma última análise, agora com os *hardgoals* escolhidos, de forma a levantar novos requisitos que integram todos os *hardgoals* escolhidos, isto principalmente em relação a parte mecânica. O objetivo é modelar de forma mais completa possível os requisitos do produto final.

A parte principal desta última diretriz é a já dita integração, aqui o produto é totalmente definido, interligando todos os recursos de hardware.

Exemplo: Para exemplificar esta diretriz pode-se usar o processo que aconteceu após a escolha da bateria como alimentação, com possibilidade de carregamento via USB ou rede elétrica. Para evitar a necessidade de duas interfaces de conexão para o carregamento (USB e tomada) pode-se usar uma fonte que possui como interface de saída de energia com o conector do USB. Esta fonte é comum em celulares.

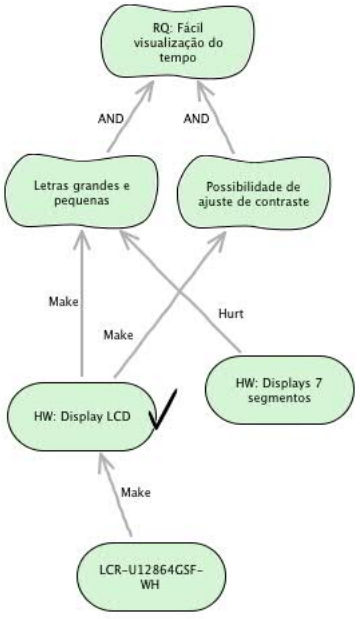

Figura 6.6: Exemplo de modelo NFR gerado no passo 3.

Este passo representa a principal diferença entre esta proposta e o GERSE. Elicitar os requisitos de qualidade no primeiro passo deste processo permite que estes sejam usados para a

escolha entre as alternativas geradas no passo 2, sempre buscando a satisfação destes requisitos, aumentando assim a qualidade final do produto.

Ao final deste passo temos em definitivo os requisitos de hardware, software e mecânicos. Ao derivar os *hardgoals* de hardware em modelos de dispositivos reais permite que seja feita uma estimativa do custo do produto (passo 3.10) do GERSE.

6.5 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, além de identificar os atores que estarão envolvidos neste processo. A tabela 6.5 lista as diretrizes deste passo.

Tabela 6.5: Diretrizes do passo 4.

Diretrizes
18 - Trazer o ator produto presente no SR do passo 2 e inserí-lo no novo modelo.
19 - Excluir os meios da ligações means-ends que não foram escolhidos no passo 3.
20 - Inserir os novos elementos de integração elicitados na diretriz 8 do passo 3 e fazer as devidas ligações com os outros elementos.
21 - Inserir os modelos reais elicitados na diretriz 7 do passo 3 e ligá-los ao recurso de hardware que eles satisfazem.
22 - Inserir atores que serão responsáveis pela produção da parte física do SE (eletrônica e mecânica).
23 - Para cada um dos requisitos de hardware e requisitos mecânicos, ver os requisitos que eles exigem dos atores de produção.
24 - Verificar se todos os requisitos de software estão bem definidos, os que não estiverem, verificar quem podería esclarecê-los. Estes podem ser novos atores.
25 - Inserir um ator representando a equipe de desenvolvimento.
26 - Verificar se existe algum elemento na região do ator que necessita de uma competência específica da equipe de desenvolvimento. Adicionar estas competências como objetivos destes elementos com o ator representando a equipe de desenvolvimento.

Diretriz 18: Trazer o ator produto presente no SR do passo 2, juntamente com todos elementos que estão na sua região.

Diretriz 19: Excluir os meios-fins que não foram escolhidos no NFR do passo 3.

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 recursos de hardware que eles satisfazem.

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 levar a necessidade, pode ser inseridos novos atores responsáveis pela produção, como por exemplo uma fabricante de chicotes elétricos.

Diretriz 23: Para cada um dos requisitos de hardware e parte mecânica, ver os requisitos que eles exigem de cada um dos atores de produção.

Exemplo: O microcontrolador escolhido no exemplo é um componente SMD, exigindo assim da montadora de placas uma linha de montagem desses componente.

Diretriz 24: Verificar se todos os requisitos de software estão bem definidos. Os que não estiverem, verificar quem poderia esclarecê-los. Estes podem ser novos atores.

Exemplo: Foi identificado que a regra em vigor no jogo é uma variável necessária ao SE, porém não está esclarecido quais campos fazem parte de uma regra de jogo de xadrez, e a melhor pessoa para conseguir esta informação é um membro da federação brasileira de xadrez, já que ele pode informar as regras de campeonatos oficiais e o que as compõe. A figura 6.7 mostra isto.

Diretriz 25: Inserir um ator representando a equipe de desenvolvimento.

Diretriz 26: Verificar se existe algum elemento na região do ator que necessita de uma competência específica da equipe de desenvolvimento. Adicionar estas competências como objetivos destes elementos com o ator representando a equipe de desenvolvimento.

Exemplo: Como deseja-se aumentar a durabilidade da bateria trabalha-se muito com interrupções, já que com elas permite-se colocar em diversos momento o microcontrolador em um estado dormente, em que ele consome menos energia. Os programadores da equipe de desenvolvimento devem ter conhecimento em como lídar com estas características.

Neste passo foram gerados os requisitos de produção e desenvolvimento do produto, que é o passo 7 do GERSE.

Esta parte da proposta abre porta a novos estudos, como a separação da equipe de desen-

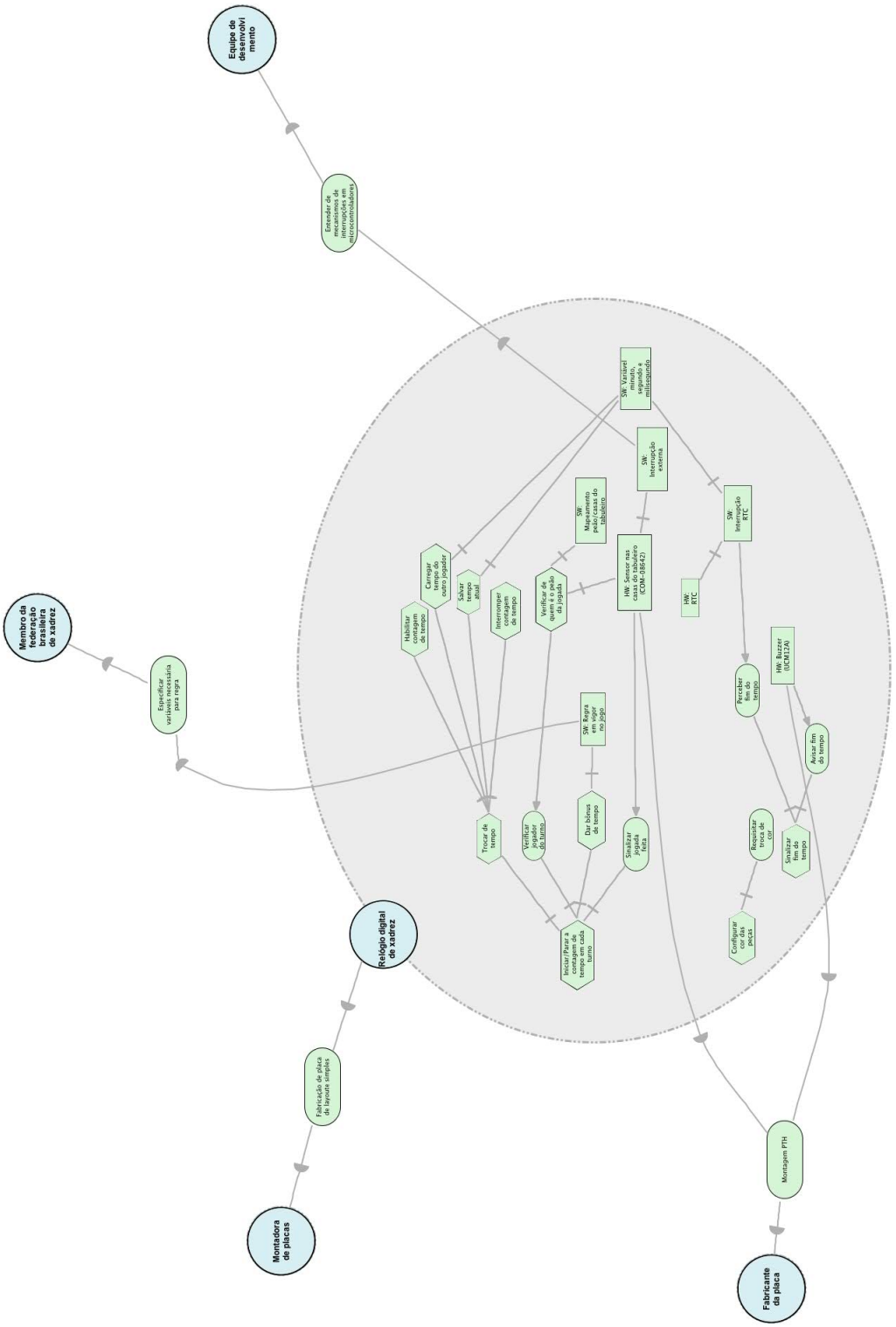

Figura 6.7: Exemplo de modelo SR gerado no passo 4.

volvimento em elementos menores, como por exemplo o testador, e ver a interação destes com o produto durante o desenvolvimento. Isto dá a possibilidade de um estudo sobre os riscos no desenvolvimento, estimativas de tempo, de custo de desenvolvimento e diversos outros.

6.6 Considerações finais do capítulo

Os passos e diretrizes aqui propostos guiam uma elicitação sistemática de requisitos para SE's. Cobrindo os requisitos de ambiente, de hardware, de software e mecânicos, características inerentes deste tipo de sistema, além de levar em consideração as limitações que eles possuem.

De forma a validar a proposta, no próximo capítulo é realizado um estudo de caso, de forma a demonstrar o uso completo da proposta no desenvolvimento de um SE.

Capítulo 7

Estudo de caso

Neste capítulo será apresentado um estudo de caso de forma a validar a proposta. O estudo de caso será o mesmo apresentado parcialmente no Capítulo 6 para exemplificar os passos e as diretrizes da proposta. A tabela 6.1 contém os passos, e as tabelas 6.2, 6.3, 6.4, e 6.5 contém as diretrizes dos passos 1, 2, 3, e 4 respectivamente.

Na seção 7.1 é apresentado o SE que busca-se desenvolver neste estudo de caso. Nas seções 7.2, 7.3, 7.4, e 7.5 o estudo de caso é desenvolvido nos passos 1, 2, 3, e 4 respectivamente.

7.1 Relógio digital de xadrez

O relógio digital de xadrez consiste, como o próprio nome diz, de um relógio digital para marcação de tempo entre as jogadas de um jogo de xadrez. Em [9] conta-se que a necessidade de um dispositivo para contagem de tempo em um jogo de xadrez surgiu no século XIX, quando em 1881, em um campeonato de xadrez em Nova York, o público reclamou da demora para serem feitas as jogadas, que chegavam a demorar duas horas. Desde então começaram a surgir novas modalidades de xadrez que levavam em consideração o tempo entre as jogadas, e, juntamente com estes, dispositivos para a marcação de tempo segundo as regras dessas modalidades. Primeiramente o relógio era um dispositivo mecânico, porém com o avanço da tecnologia digital eles passaram a ser digitais. Eles geralmente consistem de um visor onde é mostrado o tempo restante para cada jogador, e dois botões, um para cada jogador. Ao apertar este botão o jogador interrompe a contagem regressiva do seu tempo e inicia a contagem do outro jogador. As modalidades variam na forma da contagem de tempo, onde o tempo configurado pode ser para cada jogada (ele é reiniciado assim que a jogada é feita), ou é para o jogo inteiro (após a

jogada ser feita ele é pausado, reiniciando quando for novamente a vez do jogador). Existem também modalidades que dão bônus para jogadas feitas em um determinado intervalo de tempo.

Hoje os relógios de xadrez são usados tanto em competições oficiais quanto em partidas amadores, englobando enxadristas amadores e profissionais.

O estudo de caso se dará na elicitação de requisitos para o projeto de um relógio digital de xadrez. A escolha deste estudo de caso veio pelo fato de ser o mesmo estudo de caso feito com o GERSE em [9], permitindo uma boa comparação entre ambos os processos.

Na seção a seguir (7.2) a proposta começará a será aplicada para elicitação de requisitos no projeto do relógio digital de xadrez.

7.2 Passo 1. Construir o modelo SD (Dependências Estratégicas) do produto inserido em seu contexto de uso.

Os resultados de todas as diretrizes do passo 1, mostradas a baixo, podem ser observadas na figura 7.3.

Diretriz 1: Inserir o produto como um ator no modelo.

O produto em questão é o relógio digital de xadrez. Portanto um ator com este nome é inserido no modelo.

Diretriz 2: Identificar os usuários do produto e mapeá-los como atores. Fazer as devidas associações entre os mesmos, por exemplo, se um usuário é uma especialização de outro usuário, pode-se usar a associação ISA (ver capítulo 4).

Os usuários do relógio digital de xadrez são: Jogador de xadrez, jogador profissional de xadrez e o ârbitro de partidas de xadrez. Como o jogador de profissional de xadrez é uma especialização do jogador de xadrez o mesmo é relacionado ao jogador de xadrez com uma associação ISA.

Diretriz 3: Identificar outros componentes que terão interface com o produto. Estes também serão mapeados como atores no modelo.

Como componente que terá interface com o relógio digital de xadrez foi identificado o computador, que poderá ser usado para atualizar o sistema do relógio.

Diretriz 4: Identificar os requisitos funcionais do produto para cada um dos usuários. Estes serão objetivos do usuário (depender) em relação ao produto (dependee).

Cada um dos usuários possui os seus requisitos funcionais em relação ao relógio. Que são os seguintes:

1. Jogador de xadrez

    Marcar o tempo.

    Iniciar/Parar a contagem de tempo em cada turno.

    Dar bônus de tempo.

    Visualizar o tempo.

    Configurar regras.

    Configurar cor das peças.

    Configurar o dispositivo.

    Pausar/Reiniciar o jogo.

2. Jogador profissional de xadrez

    Regras oficiais pré-cadastradas.

3. Árbitro de xadrez

    Configurar as regras.

    Pausar/Reiniciar o jogo.

    Visualizar o tempo.

    Regras oficiais prê-cadastradas.

    Salvar Regras

Diretriz 5: Identificar para os componentes, requisitos (dependências) entre eles e o SE. Estes podem ser mapeados utilizando-se os elementos propostos pelo framework i*.

O requisito entre o componente e o relógio é o de atualização de software.

Diretriz 6: Mapear os requisitos relacionados ao ambiente. Estes requisitos serão dependências entre o ambiente e o produto.

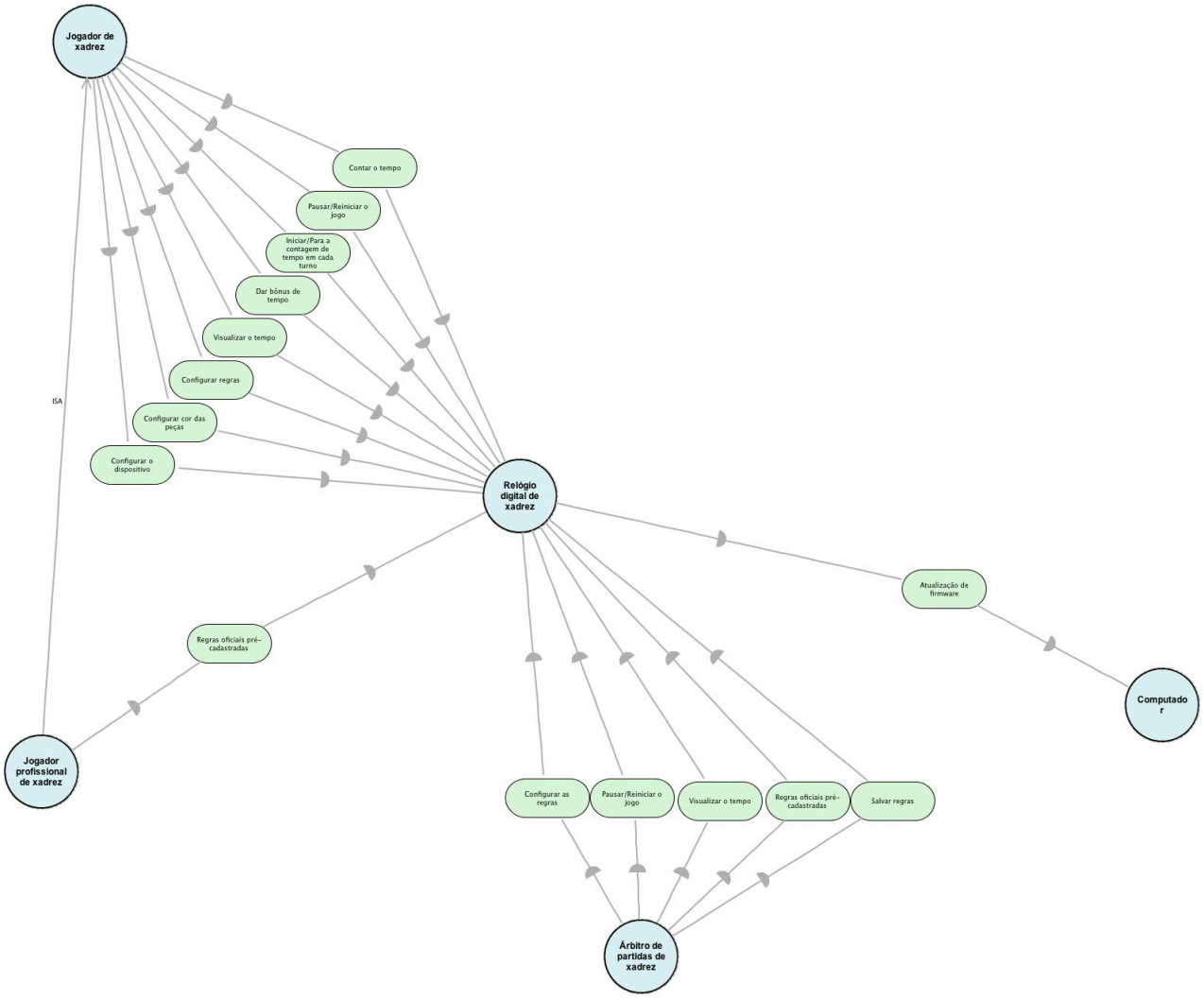

Figura 7.1: Exemplo de modelo SD do passo 1 gerado até a diretriz 5.

O ambiente é algo de muita influência em um SE, pois eles possuem uma interação muito forte um com o outro. O SE atua sobre o ambiente através de atuadores, respondendo a estímulos lidos por sensores. Já o ambiente expõe o SE a diversas situações sobre as quais o SE deve manter funcionamento. E são estas situações que esta diretriz busca elicitar. E as subdiretrizes abaixo ajudam nesta elicitação:

Subdiretriz 6.1. Inserir um novo papel chamado "Ambiente". Este 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.

Os ambientes identificados como potenciais ambientes de uso do relógio são: Praia, Praça pública e Casa.

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).

As restrições físicas que cada ambiente impõe são as seguintes:

1. Praia

    Ventos

    Altas temperaturas

    Maresia

    Resíduos sólidos

2. Praça pública

    Quedas

3. Casa

    Pó

    Resíduos líquidos

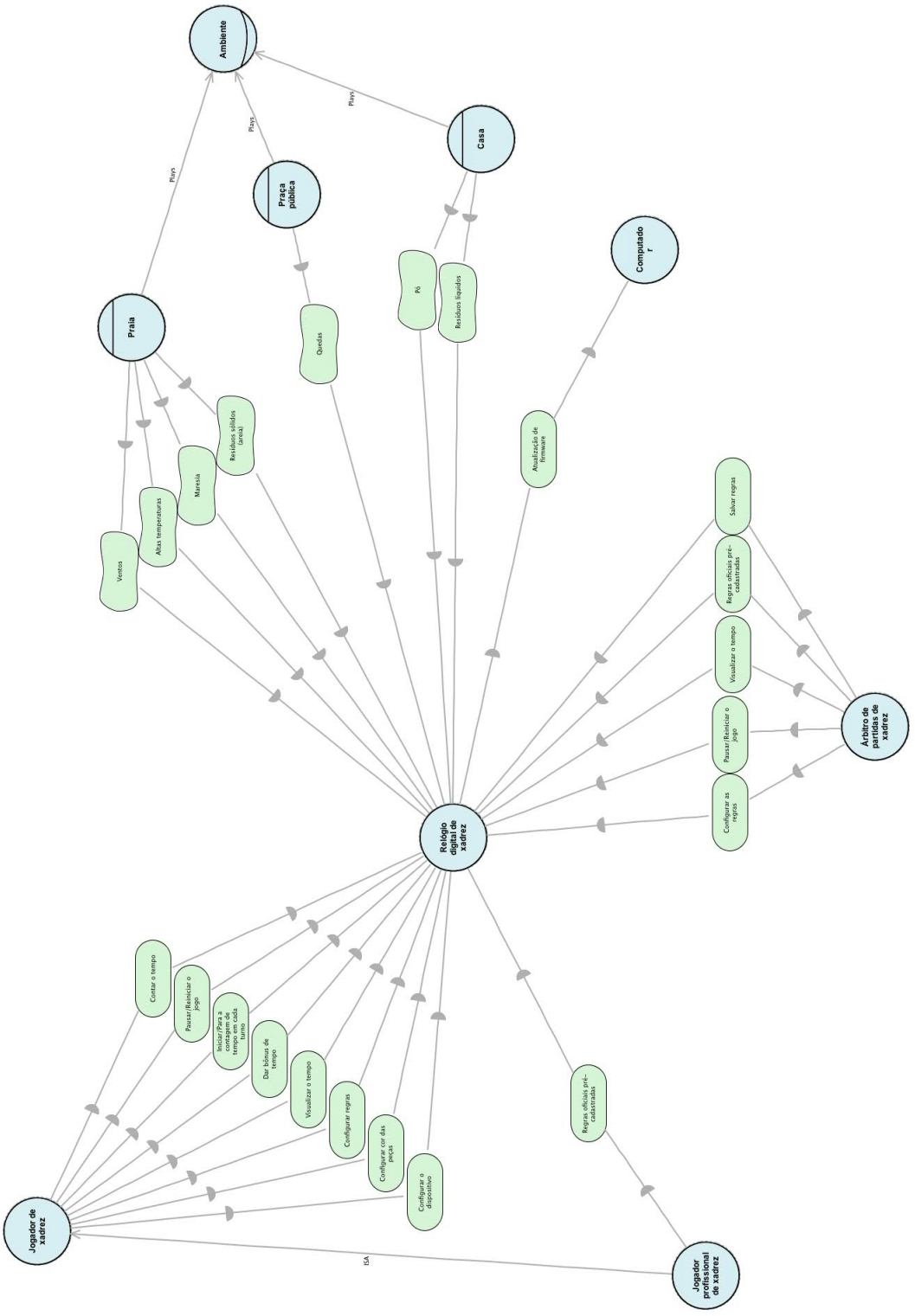

Figura 7.2: Exemplo de modelo SD gerado até a diretriz 6, com os requisitos de ambiente.

Diretriz 7: Definir para cada um dos atores e agentes até aqui elicitados se existe um tipo de alimentação ideal para os mesmos.

Por falta de tomadas a alimentação ideal para a praia e praça pública é a bateria. O mesmo tipo de alimentação é ideal também para o jogador de xadrez, já que este pretende que o relógio seja um dispositivo móvel independente de tomada. A alimentação ideal para o componente computador é aquela que utiliza a mesma interface de comunicação que será usada entre os dois.

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 diferenciá-los de outros softgoals, usar o prefixo RQ para nomeá-los.

Os requisitos de qualidade são os requisitos não-funcionais, e determinam parâmetros para escolha de alternativas e validação do produto final. Para facilitar a elicitação destes requisitos pode-se prestar atenção especial nos requisitos de qualidade da maioria dos sistemas embarcados, conforme expresso nas subdiretrizes abaixo:

Subdiretriz 8.1: Identificar requisitos de qualidade para o consumo de potência.

O jogador de xadrez espera que a bateria dure no mínimo 3 jogos de xadrez de tempo médio.

Subdiretriz 8.2: Identificar requisitos de qualidade das partes físicas e mecânicas (tamanho, peso, resistência, etc).

O usuário pretende que o relógio seja um dispositivo móvel, portanto o mesmo deve ser fácil de transportar.

Subdiretriz 8.3: Identificar requisitos de usabilidade.

O jogador de xadrez não deseja ter que ler um extenso manual para poder começar a usar o produto, portanto o mesmo deve ser fácil de utilizar. O jogador também deseja que o tempo esteja bem visível por ele. E finalmente o mesmo deseja que o relógio seja fácil de transportar.

Já o árbitro necessita visualizar o tempo a uma determinada distância, e que as suas funções estejam com fácil acesso no relógio.

Subdiretriz 8.4: Identificar requisitos de custo.

Como o produto visa também jogadores que não são profissionais o mesmo deve ser de baixo custo, para não afastar esta clientela.

Subdiretriz 8.5: Identificar requisitos de precisão.

O jogador profissional pretende que o relógio marque o tempo com precisão, de modo ao jogo ser justo para ambos os jogadores de uma partida.

Este primeiro passo elicitou as tarefas que o relógio deve desempenhar, bem como requisitos de ambiente, que o tornarão mais robusto. Também foram elicitados os requisitos de qualidade, que deverão ser satisfeitos para validação final do produto.

De posse deste modelo pode-se passar para o passo 2, onde serão elicitados os elementos internos do relógio para satisfazer essas dependências.

7.3 Passo 2. Construção do modelo SR (Razões Estratégicas) do produto.

As figuras 7.4 e 7.5 apresentam o modelo SR resultante do passo 2. Portanto, possuem os elementos que serão elicitados na diretrizes a baixo. A primeira possui os requisitos de baixo nível relacionados ao usuários, e a segunda possui os requisitos relacionados aos ambientes e componentes.

Diretriz 9: Abrir a região do ator que representa o SE, e iniciar o processo elicitação de requisitos de baixo nível. As subdiretrizes abaixo devem ser seguidas para cada dependência externa ligada ao SE.

A primeira dependência analisada foi a "Pausar/Reiniciar o Jogo". Esta dependência determina que o relógio deve permitir que os usuários parem o jogo a qualquer momento, para poderem depois continuá-lo.

Subdiretriz 9.1: Criar uma tarefa nova para a dependência externa e então ligá-la à mesma. Se o produto já possuir uma tarefa que satisfaça a dependência, basta ligar a dependência à esta tarefa sem a necessidade de criar uma nova.

Na região do produto é criada uma tarefa com o mesmo nome da dependência, esta será a tarefa responsável por satisfazer a dependência externa.

Subdiretriz 9.2: Dividir o novo elemento em subelementos (tarefas, objetivos ou recursos) necessários para satisfazê-los, 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

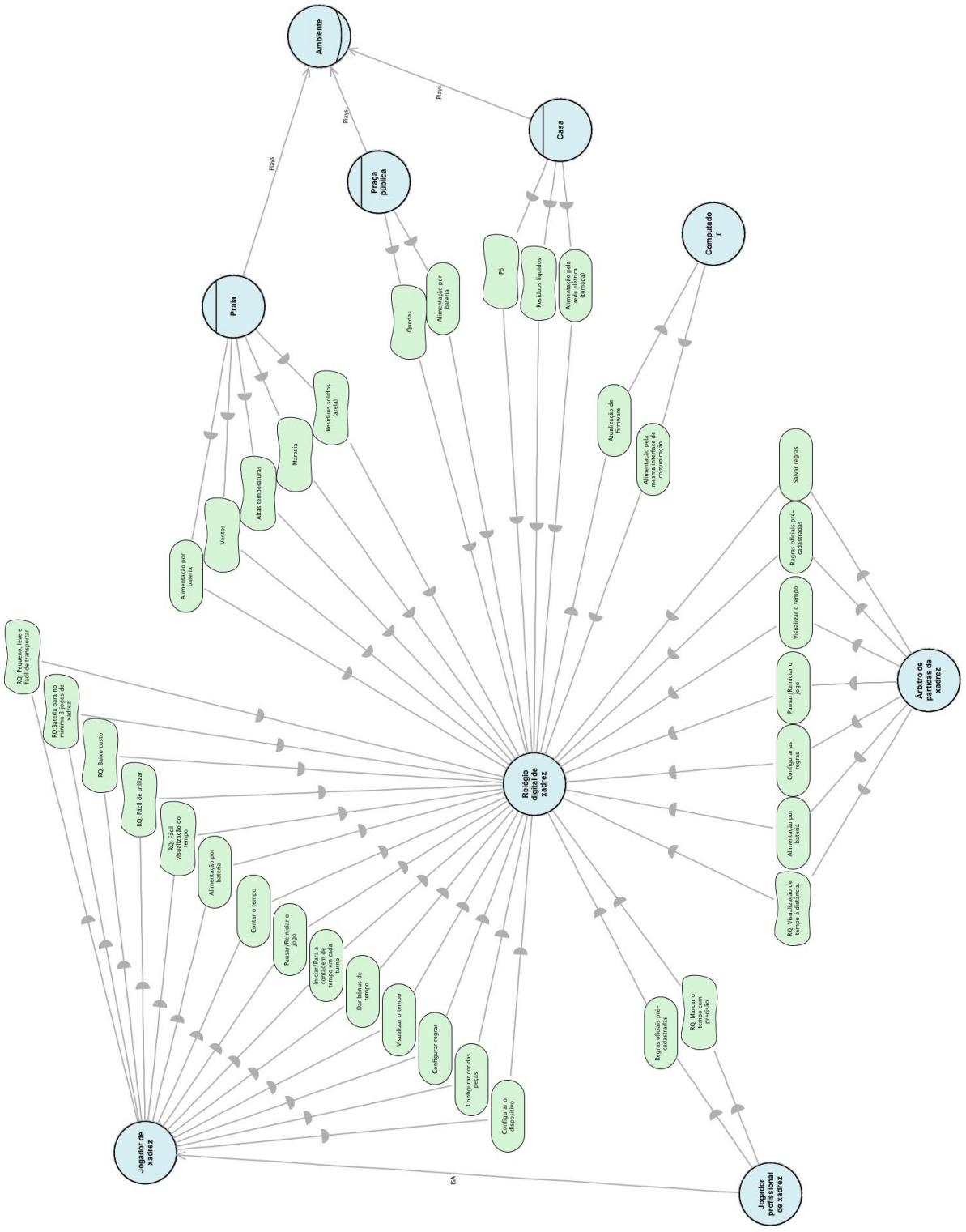

Figura 7.3: Exemplo de modelo SD gerado ao final do passo1.

considerada bem definida, quando já se está claro como implementá-la.

Para pausar ou reiniciar o jogo primeiro o jogador deve requisitar essa ação, para então o relógio reagir, parando ou voltando a contar o tempo.

Subdiretriz 9.3: Analisar para cada elemento deste requisito a necessidade de ele ser resolvido por hardware ou por software. Para estes elementos inserir recursos representando esta necessidade. Sempre pensar nos recursos de hardware como meios para se atingir um fim, portando sempre ligá-los utilizando o means-ends, isto favorece à 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.

O objetivo de requisitar a pausa ou reinício do jogo pode ser feita através de um botão, que é um recurso de hardware. Como ele representa uma forma de se conseguir algo, ele é relacionado com o objetivo "Requisitar pausa/reinício"através de um means-ends. O mesmo acontece com a tarefa "Habilitar/Interromper interrupção que atualiza o tempo". Esta é uma forma de se atingir o objetivo "Habilitar/Interromper contagem de tempo", portanto também relacionada através de um means-ends com este último.

Subdiretriz 9.4: Analisar quais elementos da dependência possuem um requisito mecânico. Estes também serão recursos. Para diferenciá-los dos outros recursos, utilizar o prefixo MEC na sua nomenclatura. Um recurso mecânico, descreve partes mecânicas necessárias, juntamente com características das mesmas.

Esta tarefa não possui requisitos mecânicos relacionados com ela.

Subdiretriz 9.5: Para cada recurso de hardware analisar recursos mecânicos ou de software relacionados a ele, estes serão novos recursos no modelo, ligados através de task-decomposition.

O recurso de hardware elicitado para esta tarefa não necessita de um requisito mecânico.

Estes mesmos passos são feitos para todos os requisitos externos ligados ao relógio, retirando os requisitos de qualidade (estes continuam apenas ligados ao ator). Com o SR concluído pode-se observar as diversas tarefas que necessitam ser implementadas, e ainda as diversas alternativas que muitas delas possui. No SR gerado neste estudo de caso muitas tarefas possuem

recursos em comum, o recurso que possui mais ligações é o recurso de software "Variável minuto, segundo e milisegundo". E isto faz sentido, pois o SE trata-se de um relógio, portanto a maior parte de suas tarefas estão correlacionadas com o tempo, neste caso, o tempo restante para a jogada.

As principais funções que o relógio deve desempenhar, é contar o tempo, e iniciar e parar esta contagem a cada turno de cada jogador. Em um sistema embarcado esta contagem de tempo pode ser feita por duas forma, uma é utilizando o timer do microncontrolador, que gera uma interrupção a cada intervalo de tempo determinado pelo programador. Neste caso a função que trataria esta interrupção atualizaria as variáveis de tempo. Outra forma é através da utilização de um RTC (Real Timer Clock - Relógio de tempo real). O RTC é um recurso disponível em alguns microcontroladores, e tem como principal função contar o tempo. Ele dispensa configurações de frequência de clock, pois ele já é pré-determinado a contar o tempo utilizando horas, minutos, segundo e milisegundos, e ele atualiza os seus registradores internos automaticamente, se tornando assim um ótima alternativa para desempenhar esta tarefa.

Já para iniciar e parar a contagem de tempo, o relógio deve perceber a troca de turno. Esta troca de turno pode acontecer com o usuário participando de forma ativa ou passiva. Na forma ativa o usuário informa o relógio diretamente que ele fez uma jogada, isto através de botões, porém neste processo, não se está condicionado o fato de o jogador ter feito sua jogada ou não, podendo levar a problemas na contagem de tempo caso o jogador aperte o botão, mas não tenha realmente realizado a jogada. Na forma passiva o relógio percebe automaticamente esta troca de turno, já condicionada a realização da jogada, ou seja, o relógio percebe que uma jogada foi feita, portanto passa o turno para outro jogador. Isto representa uma alternativa excelente para esta tarefa, pois além de melhorar o processo, agrega um valor muito grande ao produto.

Com estes elementos internos elicitados, dá-se início ao passo 3, onde é realizada as escolhas entre as alternativas geradas neste passo.

7.4 Passo 3. Construção de um modelo NFR para escolha entre as alternativas.

O SR do passo 2 representa um espaço de alternativas para construção do relógio, no entanto precisa-se definir quais serão os elementos que realmente farão parte do mesmo. E a ferramenta

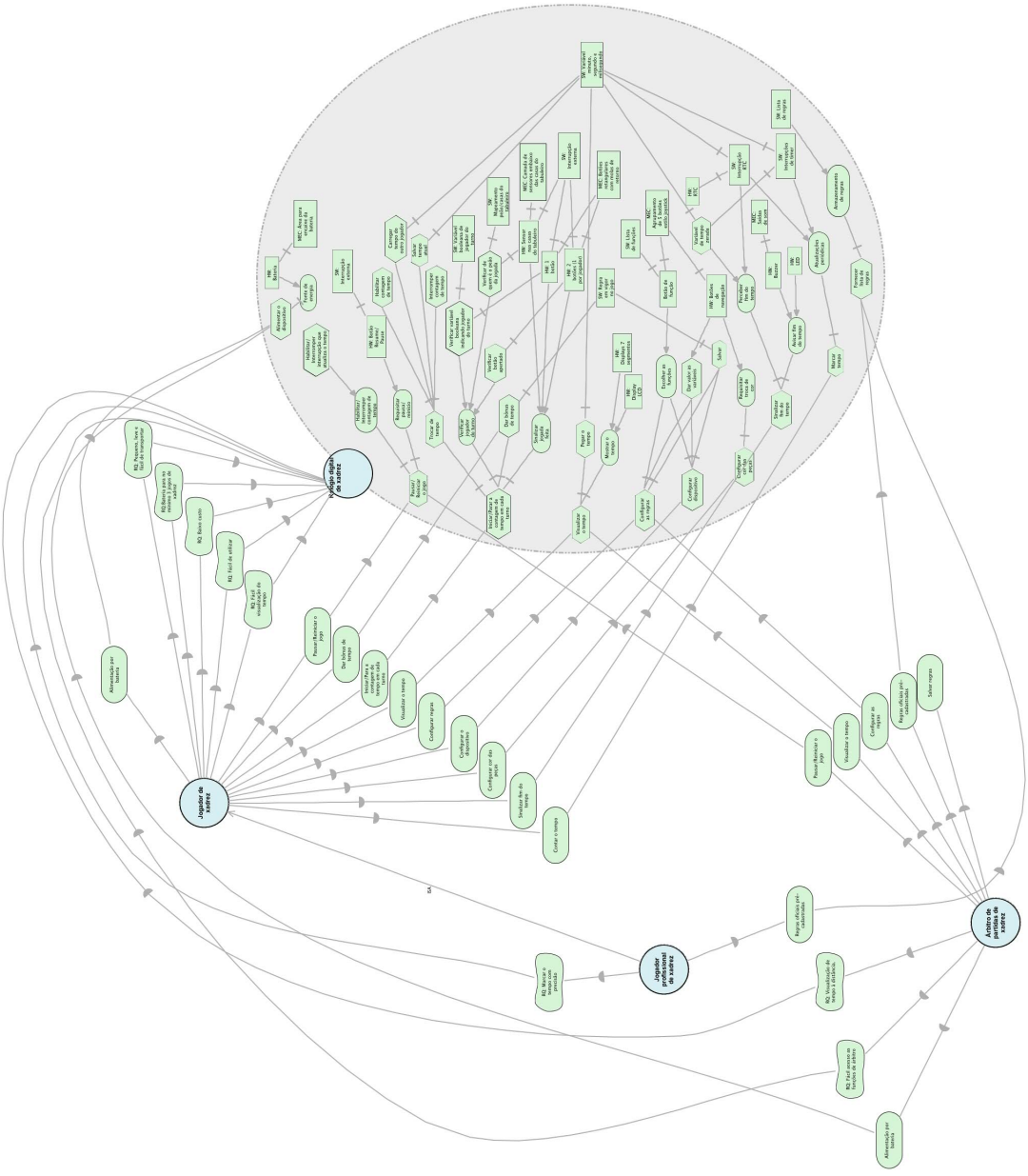

Figura 7.4: Parte do modelo SR gerado ao final do passo 2. Neste constam os requisitos relacionado aos usuários.

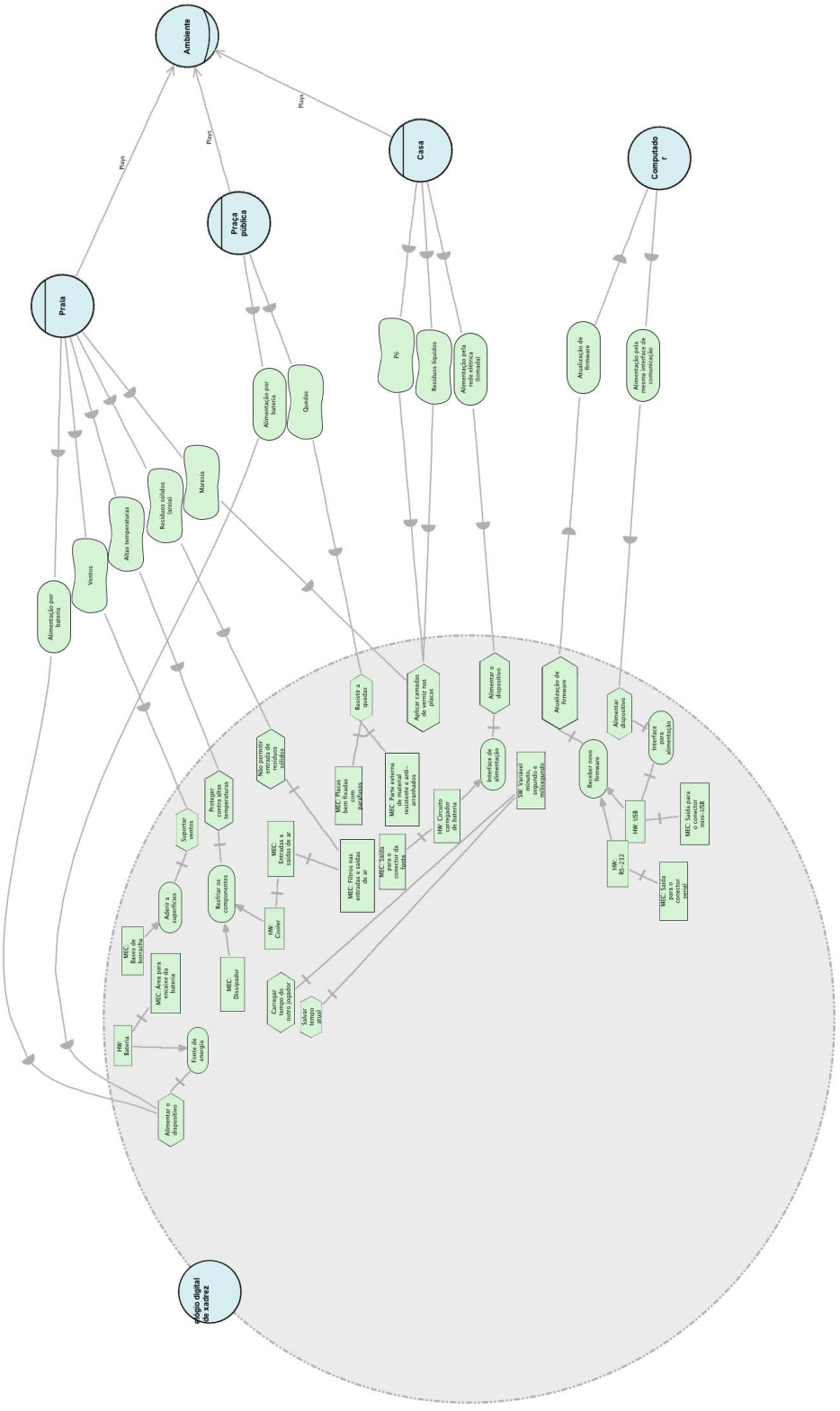

Figura 7.5: Parte do modelo SR gerado ao final do passo 2. Neste constam os requisitos relacionado aos ambientes e componentes.

utilizada para isto é um modelo NFR. Através deste modelo NFR, as diversas alternativas são confrontadas com os requisitos de qualidade, e suas forças e fraquezas em relação a cada um destes podem ser analisadas.

Este modelo NFR está presente nas figuras 7.6 e 7.7. Estas, portanto, podem ser utilizadas para acompanhar os resultados das diretrizes do passo 3. O modelo foi dividido em duas figuras devido ao seu tamanho. Os softgoals superiores estão repetidos em ambas as figuras, mas no modelo original eles aparecem apenas uma vez.

Diretriz 10: Colocar inicialmente os softgoals de qualidade elicitados no modelo SD.

Estes softgoals para o caso do relógio digital de xadrez são: Fácil de transportar, Bateria para no mínimo 3 jogos de xadrez, Baixo custo, Fácil de utilizar, Fácil visualização do tempo, Marcar o tempo com precisão, Visualização de tempo a distância, Fácil acesso as funções de árbitro e Marcar o tempo com precisão.

Diretriz 11: Juntar com estes, outros *softgoals* que representam características específicas de SE's. Porém a necessidade de cada um deles pode variar entre cada projeto. Para identificar se existe a necessidade ou não, verificar as seguintes subdiretrizes.

Subdiretriz 11.1: Se a quantidade de memória for bem limitada em relação ao necessário, insere-se um softgoal "Otimização do uso da memória".

Este não é um requisito no caso do relógio, pois a sua restrição de tamanho é bem flexível, portanto, aumentar o tamanho e colocar mais memória não é um problema. Por isto, este softgoal não é inserido no modelo.

Subdiretriz 11.2: Se a quantidade de energia for limitada, como em casos em que se usa uma bateria, insere-se um softgoal "Otimização do consumo de energia".

Como elicitado no passo 1, o relógio será alimentado por bateria, portanto necessita de um bom gerenciamento nesta parte.

Subdiretriz 11.3: Se a performance é importante, insere-se um softgoal "Otimização de performance".

O relógio não controla nenhuma variável física, e também não possui uma restrição rigorosa de tempo de resposta. Portanto a performance não se caracteriza como um requisito para este SE.

Diretriz 12: Colocar em um nível abaixo, como hardgoals, 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. Por 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.

Todos os mens-ends são inseridos, porém muitos deles consistem da única alternativa para a realização de uma tarefa, portanto estes já estão automaticamente escolhidos. No entanto eles não foram marcados já de início, apenas quando entraram em foco no processo de análise, diretriz 14.

Mas com o NFR pretende-se escolher entre as seguintes alternativas para realizar as seguintes tarefas:

- Escolher entre "Sensor nas casas do tabuleiro", "2 botões (1 por jogador)"e "1 botão", para sinalizar que a jogada foi feita.
- Escolher entre "Display 7 segmentos"e "Display LCD", para mostrar o tempo.
- Escolher entre "RTC"e "Timer"para contar o tempo.
- Escolher entre "Led"e "Buzzer"para sinalizar que o tempo acabou.

Diretriz 13: Juntamente com os meios, também representar no NFR as tarefas, recursos e objetivos ligados aos mesmos no SR do passo 2. Cada um deles será um hardgoal com uma ligação help entre ele e o hardgoal que representa o meio.

Cada um dos elementos que servem como recursos, tarefas ou objetivos para os hardgoals trazidos para o NFR também são trazidos para o mesmo. Como por exemplo o "Buzzer", que traz junto o recurso mecânico "Saídas de som".

Com todos os elementos já no NFR, pode-se começar a análise, descrita na diretriz a seguir.

Diretriz 14: Através dos softgoals e das ligações do NFR Framework, analisar quais hardgoals 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 diretrizes.

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 hardgoals, de forma que satisfazê-lo é muito importante. Portanto o mesmo será o ponto de partida da análise.

O softgoal considerado mais importante foi o "Fácil de utilizar", pois foi considerado para este caso, que se o produto satisfaz todas as suas dependências, e ainda é fácil de utilizar, provavelmente irá conquistar uma clientela fiel.

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.

Foi considerado que o softgoal está claro o suficiente para ser feito a análise, portanto não foram inseridos nenhum novo softgoal.

Subdiretriz 14.3: Verificar quais dos hardgoals influenciam no softgoal sendo analisado. Se o hardgoal tem influência positiva, ligá-lo com um some+, se for influência negativa, usar o some-, se o hardgoal foi suficiente para satifazer o softgoal, usar o make, se ele prejudicar completamente, usar o hurt. Também pode ocorrer a situação em que o hardgoal ajuda muito na satisfação de um softgoal, mas nao satisfaz completamente. Para este caso, pode-se usar o help.

O hardgoal Sensores nas casas do tabuleiro foi considerado de influência positiva no softgoal em questão, devido ao fato de automatizar o processo de troca de turno, portanto o hardgoal "Verificar de quem é o peão da jogada"também é considerado de influência positiva, já que está intimamente ligada ao hardgoal anterior.

O display de lcd também foi considerado positivo, devido à facilidade que ele traz de se criar menus mais elaborados para interação com o usuário. Já o display de 7 segmentos possui uma influência negativa, pela limitação que ele impõe na interface possível de ser criada.

Os hardgoals "Botão de função"e "Botões de navegação"foram considerados positivos, por já serem um conceito definido e aceito na parte de SE's, com diversos produtos de exemplo com esta mesma interação.

O buzzer tem influência positiva, primeiro pelo fato de o display de lcd poder fazer a função que o led desenpenharia, e outro, por avisar que o tempo está acabando, sem chamar a atenção da visão do jogador.

O USB tem influência positiva pela sua característica de Plug'n Play, não necessitando de instalações ou configurações complexas, além da rapidez de comunicação, fato que não acontece

com o RS-232, que já é mais complicado de usar, portanto, com influência negativa. Finalizando assim esta subdiretriz.

Subdiretriz 14.4: Os hardgoals 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 hardgoals. Utilizar o símbolo conhecido como "check"para simbolizar essa marcação como satisfeito. Na figura 6.6, o softgoal "HW: Buzzer", possui essa marcação.

Os seguintes hardgoals, baseados na diretriz anterior (13.3) foram escolhidos: "Sensores nas casas do tabuleiro", "Display LCD", "Botão Função", "Botões de navegação", "Buzzer"e "USB".

Subdiretriz 14.5: Seguir as ligações que estes hardgoals possuem com os outros hardgoals, e também marcá-los como escolhidos (satisfeitos).

Os hardgoals que possuem ligações com estes que foram escolhidos na diretriz anterior também são marcados, como por exemplo, o "Sensores nas casas do tabuleiro", que ao ser escolhido, provocou a escolha dos hardgoals "Interrupção externa"e "Camada de sensores em baixo das casas do tabuleiro".

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.

O segundo softgoal mais importante escolhido, foi "Marcar tempo com precisão", e toda a diretriz 14, com suas subdiretrizes é repetida, agora com foco neste softgoal. A seqüência dos próximos softgoals analisados foi: Fácil visualização de tempo, Bateria para no mínimo 3 jogos de xadrez, Baixo custo, Fácil de transportar, Visualização de tempo a distância e Fácil acesso as funções de árbitro. A análise do softgoal otimização do consumo de energia não foi necessária, pois o softgoal de bateria para no mínimo 3 jogos de xadrez já o satisfaz.

Diretriz 15: Baseado nos hardgoals escolhidos, escolher o microcontrolador do projeto.

Com os hardgoals escolhidos, pode-se escolher o microcontrolador que será usado. Pesquisou-se um microcontrolador que possuí-se RTC, USB implementado em hardware, e driver de LCD. Porém os drivers de LCD implementados diretamente em microcontroladores, são para displays pequenos, não correspondentes ao que é necessário para o projeto. Portanto

foi excluído essa necessidade na pesquisa, e chegou-se ao microcontrolador MSP430FS5638, que possui os elementos buscados.

Diretriz 16: Nos *hardgoals* de hardware que foram escolhidos, derivar um novo *hardgoal* com um modelo real do dispositivo. A relação entre esses dois será de um make.

Foi buscado para cada um dos hardwares escolhidos modelos reais dos mesmos, e inseridos no modelo. Como exemplo pode-se usar o UCM12A, que é modelo de um buzzer.

Diretriz 17: Após isto, pode-ser fazer uma última análise, agora com os hardgoals escolhidos, de forma a levantar novos requisitos que integram todos os hardgoals escolhidos, isto principalmente em relação a parte mecânica. O objetivo é modelar de forma mais completa possível os requisitos do produto final.

Para buscar a integração entre os hardgoals escolhidos, foi elicitado por exemplo o hardgoal mecânico, "Saída do carregador e do usb compartilhados", de forma a não necessitar de mais uma entrada, sendo a do usb responsável também por receber a fonte de energia para recarregar a bateria.

Porém o mais interessante está na parte do display LCD. O fato de ser escolhido a alternativa dos sensores, traz o hardgoal de camada de sensores em baixo das casas do tabuleiro, isto faz com que esta parte esteja altamente acoplada com o tabuleiro, para facilitar a comunicação destes sensores como os displays, pode-se colocá-los ao lado no tabuleiro, também acoplando-os ao tabuleiro. Desta forma, o SE deixa de ser um simples relógio para marcar tempo e torna-se um tabuleiro automatizado, com o relógio totalmente embutido nele, isto faz com que o tabuleiro passa a fazer parte do produto, e portando, da fabricação.

Com isto o produto está definido, com os seus requisitos de hardware, de software e mecânicos, que possibilitam a satisfação das suas dependências. O próximo passo, a partir deste, é identificar requisitos de produção e desenvolvimento. Descrito no próximo passo.

7.5 Passo 4. Construção de um SR do produto no contexto de produção e desenvolvimento para elicitação de requisitos para este processo.

O modelo SR final deste passo está presente na figura 7.8, contendo os resultados das diretrizes aqui apresentadas.

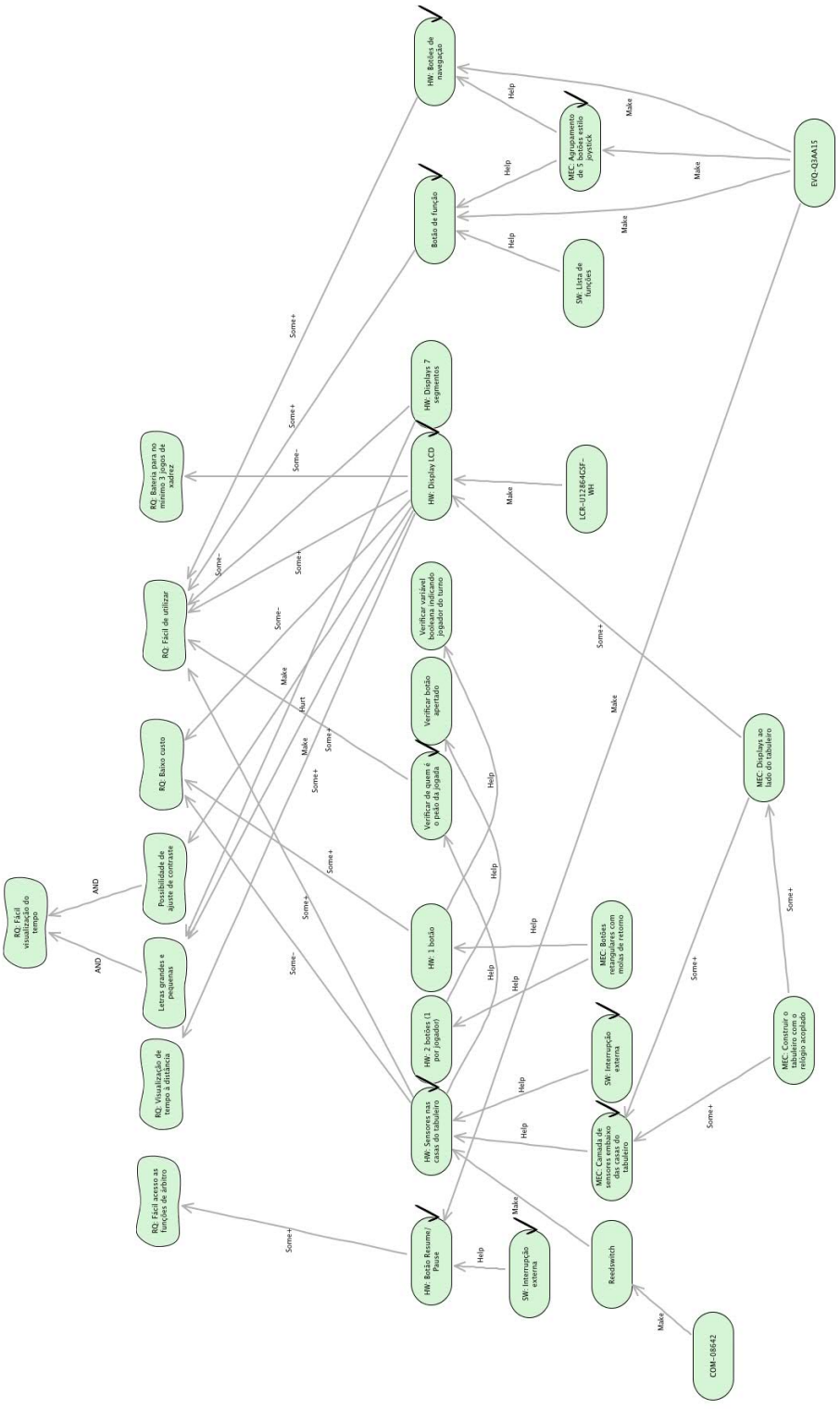

Figura 7.6: Parte do modelo NFR gerado no passo3. Contém os requisitos principais (LCD e Sinalizar jogada).

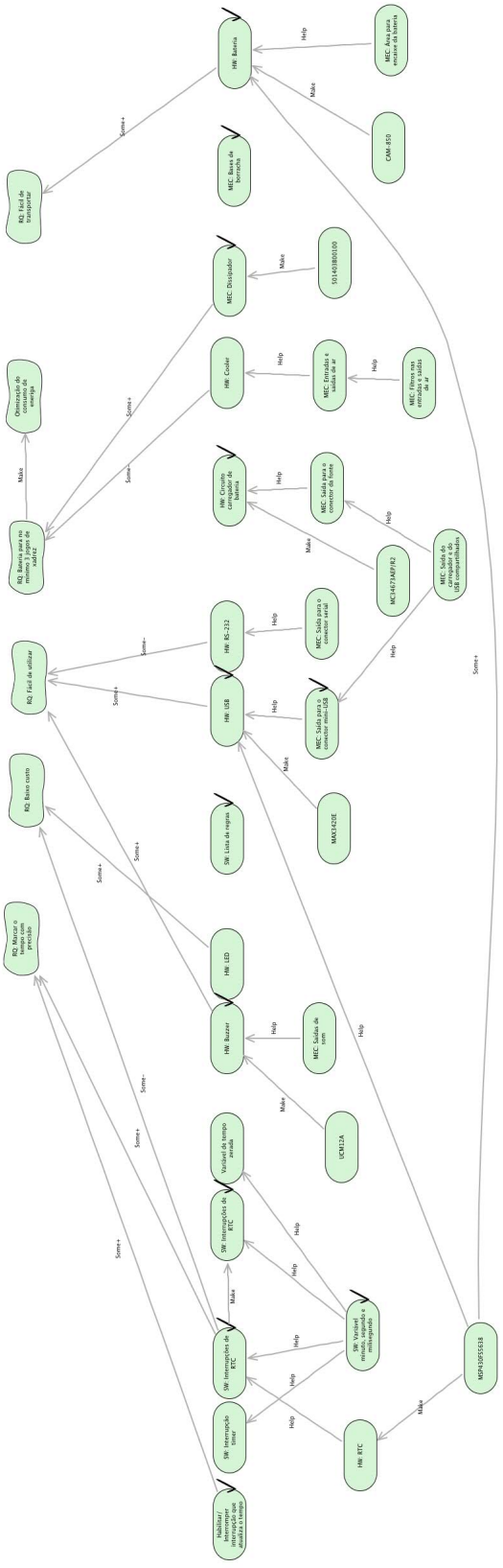

Figura 7.7: Parte domodelo NFR gerado no passo3. Contém requisitos secundários.

Diretriz 18: Trazer o ator produto presente no SR do passo 2, juntamente com todos elementos que estão na sua região.

O ator relógio digital de xadrez, do SR do passo 2, é trazido para um novo SR, que será trabalhado neste passo.

Diretriz 19: Excluir os meios-fins que não foram escolhidos no NFR do passo 3.

Aqui são excluídos, as alternativas nao escolhidas no NFR do passo 3. Como por exemplo o recurso de software "Interrupção de timer", que foi preterido em relação ao RTC.

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

São inseridos os recursos: "Displays ao lado do tabuleiro", "Construir o tabuleiro com o relógio acoplado"e "Saída do carregador e do usb compartilhados".

Diretriz 21: Os modelos reais de dispositivo são inseridos como recursos e ligados aos recursos de hardware que eles satisfazem.

Cada modelo de dispositivo de hardware elicitado no NFR é passado como um recurso de hardware para o ator do relógio no novo SR. Com isto o ator relógio digital está com todos os elementos que serão usados, ou seja, ele representa a essência do que será desenvolvido e produzido.

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 levar a necessidade, pode ser inseridos novos atores responsáveis pela produção, como por exemplo uma fabricante de chicotes elétricos.

Nesta parte são apenas inseridos os elementos citados na diretriz, nenhum ator adicional foi necessário.

Diretriz 23: Para cada um dos requisitos de hardware e parte mecânica, ver os requisitos que eles exigem de cada um dos atores de produção.

O microcontrolador escolhido é um componente SMD (Surface Mounted Device - Dispositivo montado em superfícies), que significa que ele será soldado na superfície da placa e não transpassando a mesma, como acontece com componentes PTH (Pin Through Hole - Pino atravessando um furo). Desta forma a empresa que irá montar as placas deve ter uma linha de montagem capaz de montar esses componentes SMD. No entanto o buzzer é um componente

PTH, necessitando que a montadora também tenha uma linha de montagem para estes componentes.

Não foi elicitado nenhum requisito relacionado ao layout das placas do relógio, portanto foi considerado que será usado um layout simples, sem requisitos adicionais.

Já a fabricante da parte mecânica deverá ter a capacidade de produzir um tabuleiro de xadrez, juntamente com suas peças, que exigem um detalhamento muito grande.

Diretriz 24: Verificar se todos os requisitos de software estão bem definidos. Os que não estiverem, verificar quem poderia esclarecê-los. Estes podem ser novos atores.

Além das variáveis de minuto, segundo e milisegundo, outro recurso de software importante é a Regra. Que representa as regras do jogo de xadrez, porém não sabe-se do que se trata os campos deste recurso, ou seja, não sabe-se do que consiste essa regra. Para esclarecê-la, pode-se manter contato com um membro da federação brasileira de xadrez, que além de enviar quais campos fazem parte deste recurso, também pode explicá-los.

Diretriz 25: Inserir um ator representando a equipe de desenvolvimento.

Diretriz 26: Verificar se existe algum elemento na região do ator que necessita de uma competência específica da equipe de desenvolvimento. Adicionar estas competências como objetivos destes elementos com o ator representando a equipe de desenvolvimento.

Pode-se notar pelos elementos do relógio, que as interrupções estão altamente presentes neste SE, portanto a equipe de preferência deve ter experiência no paradigma de programação que isto envolve, que traz noções da orientação a eventos.

Portanto assim, o processo é finalizado, e as informações elicitadas devem ser utilizadas no desenvolvimento do produto.

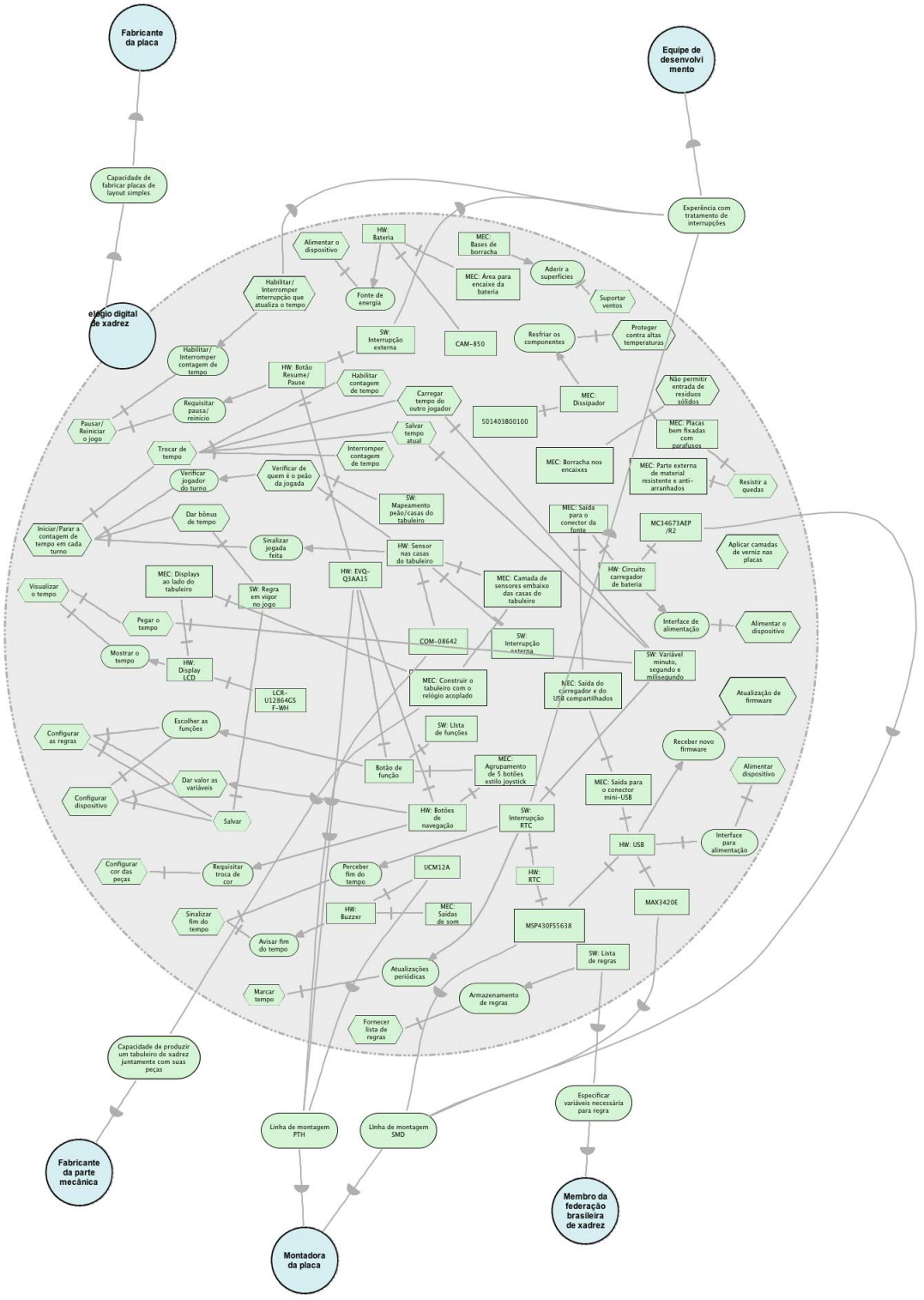

Figura 7.8: Exemplo de modelo SR do produto inserido no seu ambiente de produção e desenvolvimento.

Capítulo 8

Considerações Finais

A proposta de integração do I* ao GERSE, por meio de um conjunto de diretrizes se demonstrou através do estudo de caso um método robusto, capaz de gerar requisitos de forma completa para um sistema embarcado. Ele englobou as partes de requisitos funcionais, não-funcionais, requisitos de ambiente, de software, de hardware, requisitos mecânicos, requisitos de qualidade e requisitos de produção e desenvolvimento do GERSE. Porém ao tratá-los como elementos do I* foi possível gerar um processo orientado a atores e objetivos, de forma a elicitar os requisitos diferenciados que cada um destes possui sobre o SE, mostrando um processo de como adquirir as informações que cada atividade do GERSE visa elicitar

O modelo SD demonstrou poder de expressivade capaz de representar praticamente toda a parte de requisitos de alto nível do GERSE (atividade 3).

Já o SR permitiu relacionar os requisitos de baixo nível com os seus respectivos requisitos de alto nível. Melhorando assim um aspecto levantado nas entrevistas realizadas com profissionais da área de desenvolvimento de SE's (Ver seção 3.5). Segundo estes profissionais, o GERSE não deixa claro como realizar a transição entre os requisitos de alto nível para os de baixo nível. O passo 2 desta proposta mostra um processo de como realizar esta tarefa.

O processo descrito no passo 2, ao se utilizar de recursos de hardware e software, também facilita o *design* da integração entre estas partes do SE, outro problema do GERSE segundo os profissionais pesquisados.

Neste passo também há o incentivo à geração de alternativas para cada um dos requisitos, aumentando assim a possibilidade de se gerar uma gama completa de requisitos para o SE.

A incorporação do NFR-Framework se demonstrou uma excelente ferramenta no processo de escolha entre alternativas, com o adicional de elicitação de novos requisitos para a total

integração entre os requisitos já elicitados.

O segundo SR gerado tem um caráter mais organizacional, descrevendo os relacionamentos entre os diversos atores que estarão envolvidos no processo de desenvolvimento e produção do SE. Gerando requisitos que facilitam na escolha dos agentes reais que tomarão o lugar desses atores, ou seja, facilita a escolha de profissionais para fazerem parte da equipe de desenvolvimento, ajuda na escolha de empresas terceirizadas que irão produzir o produto e colabora com a elicitação de *stakeholders* externos cuja participação será importante no processo de desenvolvimento, como os *stakeholders* especialistas de domínio.

Os modelos gerados servem também como documentação, principalmente o SR do passo 4. Pois neste constam todos os requisitos que realmente farão parte do produto, e os atores que farão parte do desenvolvimento e produção do mesmo. Esta documentação através de modelos também representa um melhoramento em relação ao GERSE, pois estes modelos possuem mais informações, e estas informações são obtidas mais facilmente através da análise dos mesmos.

8.1 Trabalhos futuros

Como trabalhos futuros sugere-se um melhoramento no processo de escolha de alternativas através do NFR-Framework (diretriz 14). Também sugere-se um estudo sobre a aplicabilidade de metodologias de gerenciamento de projetos aplicáveis ao SR gerado no passo 4, como gerência de riscos, estimativa de custos, gerência de equipes e acompanhamento de projetos.

Este trabalho concentrou-se na elicitação de requisitos, apenas uma parte da engenharia de requisitos, portanto tanto abre-se espaço para estudos na parte de estudo de viabilidade, análise e negociação, documentação, validação e gerência de requisitos.

Glossário

Stakeholders	Termo usado para descrever as partes interessadas nas práticas corporativas da empresa.
Threads	É uma corrente única de controle sequencial dentro de um programa. Dividir um programa em threads siginifica dividí-lo em processos que serão executados paralelamente ou concorrentemente.
Know-how	É o conhecimento de como executar alguma tarefa. Também é chamado de conhecimento processual.

Referências Bibliográficas

[1] ZAVE, P. An operational approach to requirements specification for embedded systems. *IEEE Transactions on Software Engineering*, IEEE Computer Society, Los Alamitos, CA, USA, v. 8, p. 250–269, 1982. ISSN 0098-5589.

[2] OSSADA, J. C.; MARTINS, L. E. G. Um estudo de campo sobre o estado da prática da elicitação de requisitos em sistemas embarcados. In: HADAD, G. D. S.; DIESTE, O.; CARVALLO, J. P. (Ed.). *WER*. [S.l.: s.n.], 2010. ISBN 978-956-319-941-3.

[3] SIKORA, E.; TENBERGEN, B.; POHL, K. Industry needs and research directions in requirements engineering for embedded systems. *Requir. Eng.*, v. 17, n. 1, p. 57–78, 2012.

[4] MARKOSE, S.; LIU, X.; MCMILLIN, B. A systematic framework for structured object-oriented security requirements analysis in embedded systems. In: *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*. [S.l.: s.n.], 2008. v. 1, p. 75 –81.

[5] JONG, G. de. A uml-based design methodology for real-time and embedded systems. In: *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*. [S.l.: s.n.], 2002. p. 776 –779.

[6] OSSADA, J. C.; MARTINS, L. E. G. Elaboração de um guia para elicitação de requisitos de sistemas embarcados. In: *WETS*. [S.l.: s.n.], 2009. p. 53–58.

[7] 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. ISBN 978-1-60558-038-8. Disponível em: <http://doi.acm.org/10.1145/1370062.1370078>.

[8] KORDON, F. Design methodologies for embedded systems: Where is the super-glue? In: *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*. [S.l.: s.n.], 2008. p. 358 –359.

[9] OSSADA, J. C. *GERSE: Guia de Elicitação de Requisitos para Sistemas Embarcados de Pequeno e Médio Porte*. Dissertação (Mestrado) — Universidade Metodista de Piracicaba, Piracicaba, SP, 2010.

[10] KRAMMER, M. et al. Improving methods and processes for the development of safety-critical automotive embedded systems. In: *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. [S.l.: s.n.], 2010. p. 1 –4. ISSN 1946-0740.

[11] LATTEMANN, F.; LEHMANN, E. A methodological approach to the requirement specification of embedded systems. In: *Formal Engineering Methods., 1997. Proceedings., First IEEE International Conference on*. [S.l.: s.n.], 1997. p. 183 –191.

[12] 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.

[13] YU, E. et al. *Social Modeling for Requirements Engineering*. Mit Press, 2011. (Cooperative Information Systems). ISBN 9780262240550. Disponível em: <http://books.google.com.br/books?id=ceNA3l1jOeAC>.

[14] SANTANDER, V. F. A. Avaliando a utilização da técnica i* no processo de ensino e aprendizagem na engenharia de requisitos Ð um relato de experiência. In: *IV Fórum de Educação em Engenharia de Software*. [S.l.: s.n.], 2011. v. 1.

[15] BRISCHKE, M.; SANTANDER, V. F. A.; SILVA, I. F. da. Melhorando a ferramenta jgoose. In: *XV Workshop de Engenharia de Requisitos*. [S.l.: s.n.], 2012. v. 1.

[16] CHUNG, L. et al. *Non-Functional Requirements in Software Engineering (THE KLUWER INTERNATIONAL SERIES IN SOFTWARE ENGINEERING Volume 5)*. 1st. ed. Springer, 1999. Hardcover. ISBN 0792386663. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0792386663>.

[17] COUTO, A. de A.; MARTINS, L. E. G. Um processo de validação de requisitos não-funcionais baseado no nfr-framework. In: AYALA, C. P.; SILVA, C. T. L. L.; ASTUDILLO, H. (Ed.). *WER*. [S.l.: s.n.], 2009. ISBN 978-956-319-941-3.

[18] PAIM, F. R. S.; CASTRO, J. Enhancing data warehouse design with the nfr framework. In: PASTOR, O.; DIAZ, J. S. (Ed.). *Anais do WER02 - Workshop em Engenharia de Requisitos, Valencia, Espanha, Novembro 11-12, 2002*. [S.l.: s.n.], 2002. p. 40–57. ISBN 84-96023-01-X.