Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo Comparativo do Uso de Personas para o Desenvolvimento de Sistemas Baseados em Agentes

Carlos Joel Tavares da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2017

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Rodrigo Bonifacio de Almeida

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof.ª Dr.ª Genaina Nunes Rodrigues — CIC/UnB
Dr.ª Vanessa Tavares Nunes — PPGinf/CIC/UnB

CIP — Catalogação Internacional na Publicação

Silva, Carlos Joel Tavares da.

Estudo Comparativo do Uso de Personas para o Desenvolvimento de Sistemas Baseados em Agentes / Carlos Joel Tavares da Silva. Brasília : UnB, 2017.
67 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2017.

1. Sistemas Multiagentes, 2. Sistemas de Informação, 3. Personas, 4. JADE, 5. Tropos, 6. Modelagem Orientada a Objetivos

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo Comparativo do Uso de Personas para o Desenvolvimento de Sistemas Baseados em Agentes

Carlos Joel Tavares da Silva

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof.ª Dr.ª Genaina Nunes Rodrigues
CIC/UnB

Dr.ª Vanessa Tavares Nunes
PPGinf/CIC/UnB

Prof. Dr. Rodrigo Bonifacio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 10 de Julho de 2017

Dedicatória

Dedico esse trabalho para todos aqueles que me ajudaram ao longo de minha formação.

Agradecimentos

Agradeço a todos aqueles que me apoiaram durante minha vida.

Resumo

O avanço das tecnologias de informação e comunicação têm transformado a forma como as pessoas se relacionam com os Sistemas de Informação (SI). Os diferentes perfis dos usuários e o ambiente no qual eles estão inseridos exercem grande influência sobre como um SI deve atender suas necessidades. Umas das formas de representar esse cenário é por meio de Sistemas Multiagentes (SMA), onde o ambiente é trabalhado em termos dos diferentes atores, responsabilidades, objetivos, tarefas e recursos. Imagina-se que personas servem como uma especificação dos usuários do sistema sendo mais especificas que os perfis de usuários. Assim, elas também devem exercer uma grande influência sobre como o SI deve funcionar. Com intuito de verificar a utilidade de personas no desenvolvimento de SMAs apresenta-se um estudo comparativo no cenário de vida assistida entre dois processos de desenvolvimento. Um que se baseia em personas e o outro não. Por meio da comparação dos processos foram encontrados indícios de que as personas provém uma visão mais direcionada sobre as necessidades dos usuários.

Palavras-chave: Sistemas Multiagentes, Sistemas de Informação, Personas, JADE, Tropos, Modelagem Orientada a Objetivos

Abstract

The advances of information and communication technologies has transformed the way people relates to Information Systems (IS). The different users' profiles and the environment in which they are inserted exert great influence on how an IS must meet their needs. One of the ways of representing this scenario is by Multiagent Systems (MAS), where the environment is understood in terms of different actors, responsibilities, objectives, tasks and resources. It is imagined that personas serve epecification of users of the system being even more specific than user profiles. Thus, they must also exert a great influence on how the IS should work. In order to discover how personas can help in the development of MASs we present a comparative study in the assisted living scenario between two development processes. One based on personas and one that isn't. Through the comparison of the processes evidences that personas direct the development to the users necessitys were found.

Keywords: Multiagent Systems, Information Systems, Personas, JADE, Tropos, Goal-Oriented Modeling

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Grande parte dos problemas encontrados no mundo estão relacionados a ambientes dinâmicos e altamente flexíveis, aumentando a complexidade de soluções computacionais [20]. Neste sentido, soluções automatizadas que possibilitem adaptação de processos atendem a demandas emergentes. Adaptação de processos pode ser definida como o ato de customizar um processo para torná-lo aplicável em situações específicas.

Em [13], uma proposta para adaptação dinâmica de processos em sistemas de informação (SI) construída com base no conceito de informações sensíveis a contexto e planejamento automatizado é encontrada. Neste trabalho, argumenta-se que uma situação pode ser caracterizada por uma série de elementos contextuais e deve orientar a tarefa de replanejar o fluxo de instância de um processo, a fim de mantê-lo alinhado aos objetivos pretendidos.

Um exemplo que pode-se utilizar para entender melhor a ideia de adaptação dinâmica de processos em SI é o de um ambiente de vida assistida. Cria-se um sistema que é capaz de monitorar uma pessoa que possui alguma limitação e que precisa de acompanhamento 24 horas por dia, o que pode ser complicado. O sistema possui diversas tecnologias que o permite detectar a localização do usuário. Nem sempre é possível, entretanto, usufruir de todas essas tecnologias e, por isso, o sistema precisa a todo momento verificar o contexto no qual ele está inserido para saber qual é a melhor forma de descobrir sua localização e de assisti-lo.

Nota-se que, por ser um sistema que lida diretamente com a saúde de pessoas, seu bom funcionamento é essencial. Por isso, é de extrema importância garantir que ele funcionará de maneira correta em todos os contextos nos quais possa estar inserido.

Parte das responsabilidades da Engenharia de Software é procurar quais são as melhores formas de se desenvolver um sistema. Independente de qual metodologia se use, o levantamento de requisitos é sempre uma das primeiras e mais importantes fases. Ela

serve para garantir que o sistema realizará o que o usuário espera dele. Assim sendo, é importante que essa etapa entregue os melhores resultados.

1.1 Problema e Motivação

Muito se estuda sobre as melhores formas e práticas para se desenvolver um SMA. Como personas tem sido cada vez mais utilizadas no desenvolvimento de softwares [23], procura-se saber se sua utilização contribui para a área de SMA, analisando principalmente sua contribuição dentro dos levantamentos de requisitos de sistemas orientados a agentes.

Os sistemas estão, de maneira geral, inseridos em um contexto mutável e isso dificilmente é levado em consideração nas primeiras etapas de criação do software. Assim, muitos dos requisitos levantados acabam não englobando todos os casos possíveis, tornando-os suscetíveis a falhas. Pensa-se que existem não só os requisitos que foram levantados, como também outros que são relevantes ao bom funcionamento do sistema, que, por algum motivo, não foram levantados ou foram levantados de uma forma errônea. Muitas metodologias de desenvolvimento de software possuem um ciclo que permite com que etapas seguintes possam retroceder e voltar a etapas iniciais para se definir requisitos que não foram pensados. Percebe-se que, mesmo assim, vários requisitos ainda acabam faltando. Por isso, muitas vezes quando se põe o sistema em execução seu funcionamento não é como o esperado. O conserto desses erros pode ser extremamente caro e, dependendo do tipo do sistema, pode ser que ele tenha consequências irremediáveis.

Foi pensada uma nova forma de se validar requisitos de um sistema, utilizando-se personas em [23]. Personas nada mais são que arquétipos de usuários que são usados para se ter uma definição melhor de quais são as características daqueles que usarão o sistema [22]. Assim, durante o levantamento de requisitos, consegue-se ter uma ideia melhor de como o sistema será usado e se ele, como está modelado atualmente, consegue cumprir com os seus requisitos.

1.2 Objetivos

O objetivo geral deste trabalho é investigar se o uso de um método de levantamento de requisitos orientado a personas traz vantagens ao processo de desenvolvimento de sistemas orientados a agentes. Para tal foram utilizados dois estudos ilustrativos na área de ambiente de vida assistida.

Objetivos específicos

Para viabilizar o alcance do objetivo geral descrito, os seguintes objetivos específicos devem ser alcançados considerando o ambiente de vida assistida:

- realizar um estudo ilustrativo de um sistema baseado em agentes utilizando abordagem orientada a objetivos; e
- realizar um estudo ilustrativo de um sistema baseado em agentes com levantamento de requisitos orientado a personas, onde o usuário interage com o sistema através de uma interface Web.

1.3 Hipóteses

Tem-se como hipóteses definidas nesse trabalho:

- o uso da modelagem orientada a personas auxilia na melhor cobertura dos requisitos que o sistema deve atender uma vez que compreende uma análise mais refinada dos grupos de usuários de um sistema;
- o método orientado a objetivos é eficiente para capturar diferentes personas do sistema para a área de Engenharia de Software Orientada a Agentes (ESOA) [19]; e
- uso da abordagem multiagentes melhor captura conceitualmente os atributos da persona e permite implementação mais acurada por meio do *framework* JADE.

1.4 Metodologia

Durante a realização deste trabalho foram utilizadas várias fases metodológicas. Na fase inicial de estudo foram investigados os conceitos mais importantes relacionados ao tema do trabalho, os quais estão disponíveis na literatura: Engenharia de Software, ESOA, SMA, Agentes de Software, Modelagem Orientada a Objetivos, Metodologia Tropos e Métodos de avaliação empíricos. Além disso, foi realizado o desenvolvimento empírico com dois estudos ilustrativos na área de ambiente de vida assistida, o primeiro com foco em desenvolvimento de SMA e o segundo com método orientado a personas. Por último, foi realizado um estudo comparativo entre as duas abordagens utilizadas nos estudos ilustrativos.

Esta metodologia pode ser apresentada em seis fases conforme segue:

1. estudo dos conceitos de ES, ESOA, SMAs, Modelagem Orientada a Objetivos, Tropos, agentes de software e personas;

2. estudo de caso no contexto de ambiente de vida assistida;

3. desenvolver um protótipo de sistema baseado em agentes utilizando a abordagem orientada a objetivos;

4. desenvolver um protótipo de sistema baseado em agentes com levantamento de requisitos orientado a personas, onde o usuário interage com o sistema através de uma interface Web;

5. comparação dos estudos ilustrativos utilizando o método de *Goal Question Metric* (GQM) [4]; e

6. Análise dos resultados.

1.5 Apresentação do conteúdo

Esta monografia está dividida em cinco capítulos. No Capítulo 2 será apresentada a fundamentação teórica relativa à ES, ESOA, SMAs, Modelagem Orientada a Objetivos, Tropos, agentes de software e personas.

No Capítulo 3 é apresentada a proposta de solução deste trabalho, o método de avaliação empírico, bem como os estudos ilustrativos desenvolvidos para a comparação.

No Capítulo 4 os protótipos desenvolvidos são analisados, bem como as etapas de desenvolvimento para, no Capítulo 5, concluir o trabalho e relacioná-lo a trabalhos futuros.

Capítulo 2

Fundamentação

Nesse capítulo serão apresentados os conceitos estudados e que são necessários para um melhor entendimento do estudo realizado nesta monografia. Começa-se estudando alguns conceitos sobre Engenharia de Software, vê-se também definições importantes sobre Sistemas Multiagentes e por últimos analisa-se o método empírico que será usado para descobrir se alcançou-se ou não o objetivo principal deste trabalho.

2.1 Engenharia de Software

Segundo [15], software é uma entidade composta de três partes: (1) um programa que contém instruções que, quando executadas, provêem algum tipo de serviço desejado; (2) estruturas de dados que permitem a manipulação das informações necessárias a seu funcionamento; e (3) documentação necessárias para operá-lo. Assim, a *The Institute of Electrical and Electronics Engineers* (IEEE) define que Engenharia de Software (ES) é a aplicação de um abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de um software e o estudo dessas abordagens [1].

Tendo em vista essa definição de ES, são criados diversos processos e formas de se desenvolver um software, cada um com um foco específico e feito para atender algum tipo de necessidade requerida. É discutido em [15] a existência de cinco etapas independentemente do tipo de processo que se deseja utilizar: Comunicação, Planejamento, Modelagem, Construção e Implantação. O que muda de um processo para o outro é a forma em que se pode organizar essas etapas.

Se o processo for linear, as etapas acontecem uma depois da outra. Se ele for iterativo, pode-se voltar em etapas anteriores para refazer algo caso perceba-se que deve-se mudar ou refazer alguma coisa. Se ele for evolucionário, no final da etapa de Implantação volta-se para a etapa de Comunicação com o software em uma nova versão incrementada até que ele esteja pronto. Ele também pode ser paralelo, onde algumas dessas etapas ocorrem

concomitantemente. Essas são algumas das possíveis formas de organizar as etapas de desenvolvimento de software, mas existem ainda outros processos que as organizam de uma forma diferente, como o processo em espiral [20].

No final de cada etapa do desenvolvimento, costuma-se ter como resultado os chamados artefatos. Eles são subprodutos concretos produzidos durante o desenvolvimento dos softwares. Eles podem ser diagramas, modelos ou documentos que ajudam a descrever a função, arquitetura e o *design* do software. Um estudo detalhado da área de ES pode ser encontrado em [20] e [15].

2.1.1 Engenharia de Requisitos

As etapas de Comunicação, Planejamento e Modelagem levantam e definem quais são os requisitos do sistema. Segundo [21], Engenharia de Requisitos (ER) está preocupada com a elicitação, avaliação, especificação, análise e evolução dos objetivos, funcionalidades, qualidades e restrições a serem alcançados por um software. Assim, os requisitos seriam os objetivos, funcionalidades, qualidades e restrições. Todas as definições dessa Subseção foram retiradas de [21].

Objetivos como Requisitos de Software

Objetivos de software incluem uma declaração prescritiva de intenções sobre o que o sistema deve satisfazer por meio da cooperação de seus agentes, sendo eles humanos ou entidades de software. Diferencia-se objetivos de propriedades do domínio, sendo que esta últimas estão relacionadas com as características do ambiente. Isso significa que os objetivos de um sistema podem ser modificados, alterados e priorizados, enquanto as propriedades de domínio são imutáveis. Além disso, [21] ainda propõe diversas características e categorias para os objetivos. Eles podem possuir um granularidade alta, tendo um caráter mais estratégico de alto nível, ou baixa granularidade, tendo um caráter mais técnico de baixo nível. Desta forma, percebe-se que quanto mais baixo o nível de granularidade dos objetivos mais simples eles serão, sendo necessários menos agentes para cumpri-los.

Dentre os tipos de objetivos, tem-se os comportamentais e os leves. O primeiro tipo serve para descrever um comportamento que o sistema deve seguir, enquanto o segundo tenta modelar as possíveis preferências de estados dentro dos possíveis comportamentos.

Exemplificando, tem-se que um objetivo comportamental é direto no sentido de definir quantos possíveis estados um agente pode possuir. Além disso, ele pode ser verificado de uma forma simples como sim e não. Eles também definem um conjunto de comportamentos necessários. O objetivo de "manter portas fechadas enquanto o trem está em movimento" pode ser dividido em diversos comportamentos que vão desde o momento em

que o trem começa a parar com as portas fechadas até o momento em que ele volta a andar com as portas fechadas por exemplo.

Os objetivos comportamentais também podem ser divididos em subcategorias: alvos e desejáveis/indesejáveis. Os alvos são objetivos que descrevem condições que serão alcançadas dada uma condição anterior. Por exemplo, se foi feita a requisição de um livro (condição atual), cedo ou tarde, ele será emprestado pela biblioteca (condição alvo). Os objetivos desejáveis são aqueles que representam condições comportamentais desejáveis do sistema, enquanto requisitos previstos. Já os objetivos indesejáveis são os que não devem ocorrer.

Além dos comportamentais também tem-se os leves. Eles servem para descrever preferências entre possíveis comportamentos. E ao contrário dos comportamentais eles não podem ser verificados de uma forma simples. Eles podem ser classificados em funcionais, qualitativos e de desenvolvimento, sendo que a fronteira entre essas categorias não é 100% definida. Os funcionais descrevem serviços que devem ser providos pelo sistema. Os qualitativos servem para indicar uma qualidade do sistema, como segurança, desempenho, usabilidade, entre outros. E, por último, os de desenvolvimento são sobre a qualidade do desenvolvimento, como custos, prazos, reuso, etc. Todos esses tipos são utilizados para criar heurísticas mais sofisticadas para a análise do sistema.

Por último é mostrado como os objetivos são usados no processo de ER. Primeiro entende-se que a granularidade dos objetivos serve para criar uma estruturação hierárquica entre eles. E com base nessa hierarquia é possível fazer uma análise de completude, uma vez que os mais simples são mais fáceis de serem verificados e garantem a validade dos mais complexos, de forma a se validar o sistema como um todo. Além disso é possível fazer uma análise de pertinência deles, verificando se são de fato necessários para o sistema ou se eles já são cumpridos por outros.

É importante ressaltar que ter um orientação a objetivos não é o mesmo que uma abordagem *top-down*, indo do objetivo principal do sistema e dividindo-o em outros menores. É possível também ter uma visão *bottom-up* na construção. Também é possível fazer um paralelo entre eles, agentes e cenários. Onde os objetivos são responsabilidades dos agentes, que interagem com cenários que por sua vez contemplam os objetivos, criando o chamado triângulo mágico da ER.

Agentes como Requisitos de Software

Para SMA, um agente é uma entidade do software capaz de perceber e agir no ambiente de forma autônoma, possuindo características como: proatividade, adaptabilidade, comunicação, decentralização. Já na ER, [21] define agente como sendo um componente

ativo do sistema responsável diretamente pela satisfação de um objetivo. Isso é, ele pode ser um software, um dispositivo eletrônico ou uma entidade humana.

É de extrema importância ressaltar a diferença entre esses dois agentes e em quais contextos eles são utilizados. A principal diferença entre eles é que o agente do SMA é inteligente e será implementado e o outro pode ser uma entidade de software não inteligente ou até humano. Assim, o agente inteligente é uma especialização do agente requisito de software. Em ambos, eles são capazes de perceber e atuar, estão conectados com um modelo de objetivos, possuem uma relação de dependência, desejos e crenças, e servem para compor um sistema maior a fim de resolver um problema. Além disso, um SMA é compostos desses agentes modelados. Já em ER, o agente serve para levantar os requisitos do sistema e não necessariamente será utilizado, implementado ou necessário para o desenvolvimento final do sistema. Outro detalhe importante de ser ressaltado é que na ER os agentes e os objetivos estão ligados um a um diretamente como iremos apresentar na subseção Modelagem, porém em SMA, cada agente possui sua própria modelagem de objetivos que é atribuída unicamente a ele. Logo é importante não confundir agentes definidos em ER e agente de um SMA.

Dentro da ER, agentes e objetivos estão intimamente relacionados, e por isso serão detalhadas um pouco mais as características desses agentes. É importante ressaltar que por mais que existam diferenças entre um agente de um SMA e um agente como requisito de software, as características descritas aqui valem para os dois.

Para ER, um agente monitora uma variável quando ele é capaz de ver seu valor e que ele controla uma variável se ele consegue estabelecer o valor dele. Além disso, [21] ressalta que um agente monitora o estado de uma variável que é controlado por outro agente e que, a fim de evitar interferências entre agentes, variáveis devem ser controladas por um único agente. Percebe-se que o autor tenta deixar claro que os agentes estão sempre se comunicando, seja diretamente ou não, uma vez que, por mais que eles não troquem mensagens, o impacto de um agente acaba sendo captado por outro.

Além disso, os agentes possuem responsabilidades, isso quer dizer que será considerado um agente responsável por um objetivo quando suas instâncias são as únicas necessárias para satisfazer o objetivo. Isso significa que é possível definir uma sequência de estados de transição, cujas diferenças estão unicamente nas variáveis controladas e monitoradas pelo agente, que garantem os comportamentos esperados pelo objetivo daquele agente. Isso implica diretamente na noção de um agente conseguir ou não alcançar seu objetivo, caso ele não consiga monitorar nem controlar as variáveis necessárias para cumpri-lo, o objetivo não será alcançado.

Além disso, também define-se que um agente deseja alcançar um objetivo quando ele tem planos para cumpri-lo. No caso da ES esses desejos são atribuídos unicamente para

agentes humanos, porém para SMA, como os agentes são inteligentes, eles também são capazes de possuir desejos e agir com base neles por meio de um modelo mentalístico denominado *Belief-Desire-Intention* (BDI). O BDI foi definido por [7], sendo citado na literatura como uma teoria do raciocínio prático humano [16]. Eles também podem acreditar em fatos uma vez que eles estejam em suas memórias. Dizemos que eles acreditam no fato A caso A esteja em sua memória e que eles sabem A quando A está em sua memória e A é verdade.

De acordo com [21], para a ER existem diversas formas de se modelar um agente. A primeira seria o Diagrama de Agentes, que explicita quais são as capacidades e responsabilidades do agente. A segunda seria o Diagrama de Contextos, que é responsável por mostrar as interfaces que os agentes tem uns com os outros, isso é, quais variáveis que um controla que são monitoradas por outros. E por último temos o Diagrama de Dependências que mostra quais objetivos conectam os agentes.

Além disso, agentes também possuem um grau de granularidade o que permite fazer uma relação com os objetivos. Objetivos simples podem ser realizados por agentes simples e atômicos, enquanto objetivos complexos precisam de múltiplos agentes ou de agentes com maior grau de racionalidade.

Modelagem

Tendo em vista que agora sabemos exatamente o que é um agente e o que são objetivos e que ambos tem como função fazer o levantamento de requisitos do sistema para a ER, precisa-se modelá-los.

Como a modelagem que foi utilizada no trabalho foi a Tropos, será detalhado sobre como ela é futuramente. Porém, devido a sua relação com SMA, seu detalhamento acontecerá na Seção 2.2. Portanto nessa seção só será discutido o que ela deve conter e a relação entre a modelagem de objetivos e a modelagem de agentes.

Como comentado, os objetivos possuem um grau de granularidade. Esse fato será utilizado para realizar a modelagem. O princípio básico é pensar qual é o objetivo principal. E a partir dele pensa-se em objetivos menores, ou seja, com uma granularidade menor que ajuda ou que são necessários para realizar o objetivo principal. Dessa forma, liga-se os objetivos até ter uma árvore de objetivos.

Para modelar os agentes dentro da ER, faz-se o mesmo processo que o de modelar os objetivos. Pensa-se em um agente que seja responsável em cumprir o objetivo principal, e depois disso em agentes menores que auxiliam o "agente principal"a alcançar aquele objetivo principal com objetivos secundários. Ao final, tem-se também uma árvore que relaciona os agentes do sistema.

Pode-se perceber que o produto final das modelagens são parecidos, um com uma abordagem dos objetivos do sistema e o outro com uma abordagem dos agentes. E de fato existe essa relação direta entre os nós dos nossos grafos, onde um agente está diretamente ligado a um objetivo.

2.1.2 Engenharia de Software Orientada a Agentes

Aplicações industriais de softwares vem se tornando cada vez mais complexas e cada vez maiores, o que torna o seu processo de construção difícil. A fim de tentar resolver esse problema, engenheiros de software criaram os mais diversos paradigmas de desenvolvimento para tentar facilitar esse processo (programação estruturada, programação declarativa, programação orientada a objetos e baseada em componentes).

Além disso, um bom processo de software precisa prover um menor tempo de desenvolvimento com o menor risco possível [20].

A área de ES busca um paradigma de desenvolvimento de software orientado a agentes que seja mais adequado aos aspectos de comportamento dos agentes.

ESOA é um paradigma que tenta aplicar as melhores práticas no desenvolvimento de SMAs complexos trazendo o foco para os agentes e em sua organização como comunidade como sendo o ponto de abstração principal. [19]

Com a avanço de computação autônoma, pervasiva e inspiradas em conceitos biológicos, as vantagens e as necessidades de tecnologias baseadas em agentes e sistemas multiagentes se tornaram mais óbvias. Infelizmente, as metodologias atuais de ESOA são dedicadas ao desenvolvimento de SMA com um grau de complexidade pequeno.

Obviamente, muitos SMAs usarão do mesmo tipo de técnicas, adaptações e abordagens. Por isso esse é um campo com diversas oportunidade para explorar conceitos como: redução de custos e melhorar o *time-to-market*. Além disso procura aperfeiçoar cada vez mais a tecnologia baseada em agentes para torná-la cada vez mais utilizável na indústria.

2.2 Sistemas multiagentes

Segundo [25], um agente é como sendo uma entidade computacional que é capaz de realizar uma ação independente em prol de seus usuários ou proprietários. Um SMA seria um sistema que é constituído de diversos agentes que são capazes de interagir entre si.

Para [17], agentes precisam perceber o ambiente no qual estão inseridos e depois de analisá-lo devem ser capazes de realizar uma ação. Podemos exemplificar um agente simples como sendo um termostato, se ele perceber que a temperatura está muito alta ele a diminui.

Agentes de um SMA são agentes inteligentes, porém pode ser difícil definir o que exatamente é um agente inteligente, para isso algumas características são esperadas, sendo elas:

- Reatividade: perceber e reagir às alterações que ocorrem no ambiente para satisfazer seus objetivos;
- Proatividade: tomar a iniciativa para alcançar seus objetivos;
- Habilidade Social: conseguem interagir com outros agentes para satisfazer seus objetivos.

Com base nessas características e o que a ESOA define, as seguintes propriedades precisam ser consideradas a fim de construir um software orientado a agente:

- Autonomia, a capacidade de agir de forma independente sem intervenção direta de humanos ou de outros agentes [5]. É uma propriedade inerente aos agentes que implica em um software mais robusto;
- Deliberatividade, a capacidade de um agente tomar decisões considerando informações oriundas do ambiente onde ele se situa, bem como as informações sobre experiências prévias;
- Reatividade, a capacidade dos agentes de perceber seu ambiente e responderem aos estímulos externos;
- Organização, estabelecimento de um comportamento coeso para um grupo de agentes, onde o comportamento de cada um é restrito por um conjunto de normas sociais [9];
- Socialização, a capacidade que os agentes possuem de cooperarem e coordenarem suas atividades com os demais agentes do ambiente. Os agentes são capazes de comunicar suas crenças, objetivos e planos uns com os outros; e
- Interação, a habilidade de se comunicar com o ambiente e com outros agentes. Segundo [5], a comunicação é um dos componentes chaves dos SMA, uma vez que para atingir um objetivo, muitas vezes os agentes precisam ser capazes de se comunicar com usuários, recursos ou outros agentes.

No entanto, o paradigma de agentes apresenta desafios e obstáculos relacionados a manter o equilíbrio entre comportamento proativo e reativo, considerando que os agentes precisam interagir continuamente com o ambiente e perseguir seus objetivos. Para que este equilíbrio aconteça é necessário que a tomada de decisão seja sensível ao contexto,

isto é, as pessoas e ao ambiente com as quais eles interagem e as situações que acontecem, o que resulta em um grau significativo de imprevisibilidade sobre quais objetivos o agente perseguirá, em quais circunstâncias e quais os métodos que serão usados.

O projeto de como os agentes raciocinam, agem e interagem entre si e com o ambiente é essencial para o desenvolvimento de SMAs que atendam aos objetivos e expectativas dos usuários.

Essas características podem ser simples de serem definidas dependendo do sistema, ou podem se tornar muito complexas. Isso acarreta também em uma simplicidade ou complexidade na modelagem do nosso agente. Podemos ter agentes simples que funcionam de uma forma reativa, onde ele só percebe e age. Ou então temos agentes que precisam armazenar o estado do ambiente passado e que suas ações dependam de ações e de ambientes passados, onde teríamos um chamado agente com estado. Também existem os agentes orientados a objetivos, onde são definidos os objetivos do agente, que então se preocupa em determinar um plano para alcançá-lo. Um agente ainda mais complexo seria um onde não é implementado como que ele deve fazer e sim só o que deve ser feito, assim ele deve por meios de métricas e utilidades definidas achar a melhor forma possível de alcançar seus objetivos. Os agentes capazes de aprender são os mais complexos de todos. Eles utilizam técnicas de aprendizado de máquina para agir.

Portanto, agentes são entidades de software que podem ser simples ou complexos mas que em todos os casos possuem um objetivo a ser cumprido e para isso eles devem ser capazes de interagir com o ambiente. Existem algumas formas de se modelar esses agentes, para isso precisamos definir exatamente quais são as ações e percepções que esse agente tem do ambiente, como é o ambiente e quais são suas características e quais são seus objetivos ou formas de metrificar seu desempenho. Isso é chamado de PEAS (*Performance, Environment, Actions, Sensors*) do agente e será abordada com mais profundidade em seguida.

Para definir o objetivo do agente, utiliza-se uma modelagem orientada a objetivos (MOO) que será mais detalhada adiante. Existe diversas metodologias para se realizar essas modelagens. A que aborda todas as fases de desenvolvimento de um software, desde seus requisitos iniciais até a implementação é a modelagem Tropos que também será abordada na Seção 2.2.2.

Temos que o agente é um parte fundamental de um SMA, porém só eles não é o suficiente. Para termos um SMA, ainda precisamos que esses agentes sejam capazes de interagir entre si utilizando trocas de mensagens. Além disso, eles devem trabalhar em prol de um objetivo final. Por mais que nem todos os agentes cooperem diretamente entre si, eles foram projetados para desenvolver um sistema que seja capaz de resolver um problema, assim, de alguma forma as tarefas devem ser distribuídas entre eles.

Assim, além de definir como será cada agente do nosso sistema, precisamos definir como que esses agentes interagem entre si, se eles cooperam ou competem, se existe algum hierarquia entre eles. Todas essas informações são definidas na arquitetura de um SMA. Onde são detalhadas as relações entre os objetivos dos nossos agentes.

Ainda precisamos definir também quais são as mensagens que esses agentes trocam entre si para se comunicar, para isso precisamos definir não só uma linguagem para isso, como também fazer o diagrama de sequência UML para mostrar exatamente como e quais são essas mensagens. Uma possível linguagem para se fazer essa comunicação entre os agentes é a *Agent Communication Language* (ACL) definida em [14].

2.2.1 Ambiente de Tarefas

Para definir um agente também é preciso definir seu PEAS, isso é, para definir um agente precisa-se dizer se ele possui alguma forma de medir seu desempenho, em qual ambiente ele está inserido, suas ações dentro desse ambiente e seus sensores. [17] define esse conjunto de itens como sendo o Ambiente de Tarefas do agente, diz ainda que é a primeira etapa do projeto de um agente e deve ser o mais detalhado possível.

O desempenho é definido de acordo com o objetivo que se tem para aquele agente, se ele for por exemplo um aspirador de pó cujo objetivo é limpar a sala independente do custo, define-se seu desempenho como sendo as vezes em que ele limpou. Porém se deseja-se fazer essa limpeza com o menor custo, será preciso definir um modelo de metrificação que leva em conta não só o número de limpezas, mas como também a energia utilizada.

Para definir o ambiente é preciso mais do que só informar onde ele está, é preciso atribuir diversas características a ele que serão utilizadas quando formos desenvolver os agentes. Essas características são: se ele é acessível ou não, determinísticos ou estocástico, sequencial ou episódico, estático ou dinâmico, discreto ou contínuo, de um agente ou multiagente.

2.2.2 Modelagem Orientada a Objetivos

A MOO é aquela que foca nos objetivos que devem ser alcançados ao invés das ações tomadas. Ela pode fazer parte tanto da ES para se definir as relações e entre os objetivos do sistema [21] quanto para definir o comportamento de um agente inteligente e autônomo dentro de uma abordagem orientada a agentes [25]. Para ES, MOO é importante pois por meio dessa é possível fazer uma análise dos contextos e dos cenários nos quais o sistema estará inserido, tentando garantir que os requisitos que serão levantados a partir dela sejam os mais precisos e corretos. Dentro dos SMAs essa modelagem também é importante

pois por meio dela definimos boa parte dos nossos agentes, uma vez que é com base nos objetivos que os agentes definem o que elas farão, determinando seu comportamento.

Existem diversas notações para se modelar os objetivos de um sistema. i* [26], GRL [10], KAOS [12] e até por meio de diagramas de casos de uso da UML [6]. Porém foi criada outra notação de modelagem de objetivos com base no i*, o Tropos, voltada para o desenvolvimento de SMA que abrange todas as etapas de desenvolvimento de um software, desde o levantamento de requisitos até a implementação.

Metodologia Tropos

A metodologia de modelagem Tropos [8] fornece construções para analisar objetivos hierarquicamente, em uma abordagem *top-down*, e descobrir requisitos e conjuntos alternativos de tarefas que o sistema pode executar para realizar as funcionalidades necessárias à satisfação dos objetivos.

Os modelos são construídos para capturar as intenções dos interessados (usuários, proprietários, etc.) e adotam o *framework* i* (ISTAR) para representar os conceitos de atores (agentes, posições ou papéis), objetivos (quantitativos e qualitativos), tarefas ou planos, recursos e suas interdependências através de uma notação própria. Essa representação está disponível em ferramentas associadas ao Tropos e pode ser vista na Figura 2.1, sendo que a utilizada neste trabalho é denominada TAO4ME [2].

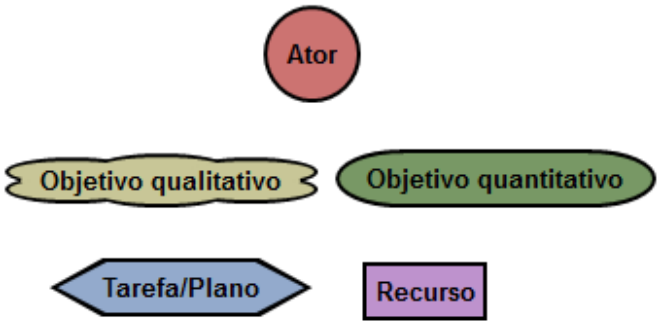

Figura 2.1: Notação gráfica do i* na ferramenta TAOM4E.

Um projeto de SMA utilizando a metodologia orientada a objetivos do Tropos pode ser dividida em cinco etapas conforme a Figura 2.2: requisitos iniciais, requisitos finais, design arquitetural, design detalhado e implementação.

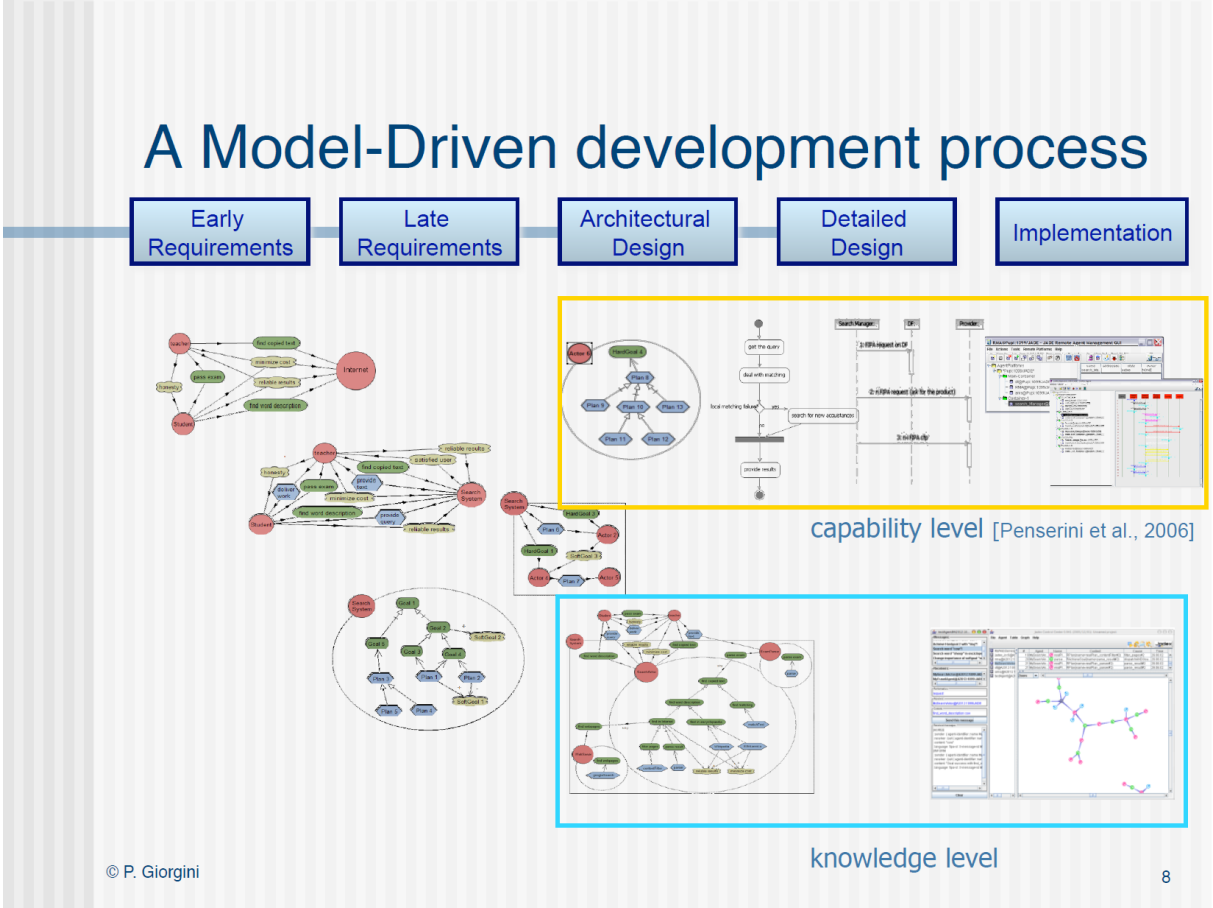

Figura 2.2: Etapas da metodologia Tropos.

- Requisitos Iniciais: busca-se o entendimento do problema através da compreensão do contexto organizacional. Durante essa fase, os engenheiros de requisitos modelam os interessados como atores sociais que dependem uns dos outros e que têm intenções de atingir objetivos, realizar tarefas e fornecer recursos [10].

- Requisitos Finais: esta fase está preocupada com o refinamento dos modelos produzidos na fase anterior. Os componentes devem ser detalhados para apresentar as razões que estão por trás de suas dependências. Suas tarefas e objetivos precisam ser revisados, analisados e detalhados através de ligações de decomposição e de meios-fins;

- Projeto Arquitetural - busca-se definir a arquitetura do sistema, modelando sua estrutura em termos de subsistemas, interconectados através de fluxos de controle de dados. Tropos representa os subsistemas como atores e interconexões são representadas como dependências entre os atores;

- Projeto detalhado - esta fase envolve o uso de plataformas de desenvolvimento específicas e depende das características da linguagem de programação adotada. Assim,

esta etapa geralmente está estritamente relacionada com as escolhas da implementação. Para documentar o projeto, pode-se ainda incluir a especificação da estrutura de comunicação e comportamento dos agentes adotando modelos UML, utilizando diagramas de classe, sequência e comunicação;

- Implementação - esta fase envolve a implementação do sistema projetado utilizando uma linguagem de programação.

2.3 Personas

Personas são definidas com base nos Projetos Centrados no Usuário (PCU) [22]. Ele não é exclusivo da tecnologia onde já é um projeto muito comum dentro da área de Design. Encontramos em [22] uma tentativa, por meio da PCU, de encorajamento aos desenvolvedores a usar como foco do desenvolvimento do sistema a experiência do usuário e uma das formas de se fazer isso é por meio das personas.

Dentro do contexto utilizado nessa monografia, personas nada mais são que usuários fictícios dos sistemas que servem para definir diferentes tipos de comportamentos, objetivos, vontades e características. Como cada uma delas serve para representar uma pessoa, elas possuem informações como: idade, sexo, escolaridade, profissão, passatempo, família, entre outras informações que forem julgadas necessárias. Além disso, é preciso atribuir a ela uma história que servirá para mostrar a vida dessa Persona como seu dia-a-dia, que será usado para definir o contexto no qual ela está inserida. Por último vamos ressaltar os objetivos da Persona. Assim, usa-se a história da Persona para saber onde que o sistema a ser desenvolvido pode se encaixar na vida dela e o usaremos os objetivos dela para saber como o sistema pode diretamente ajudá-la a cumpri-los.

Elas são normalmente representadas por cartões que possuem essas informações, como se nota na Figura 2.3. Note que estão descritos o nome, sexo, educação, profissão, salário, história e objetivos da Persona.

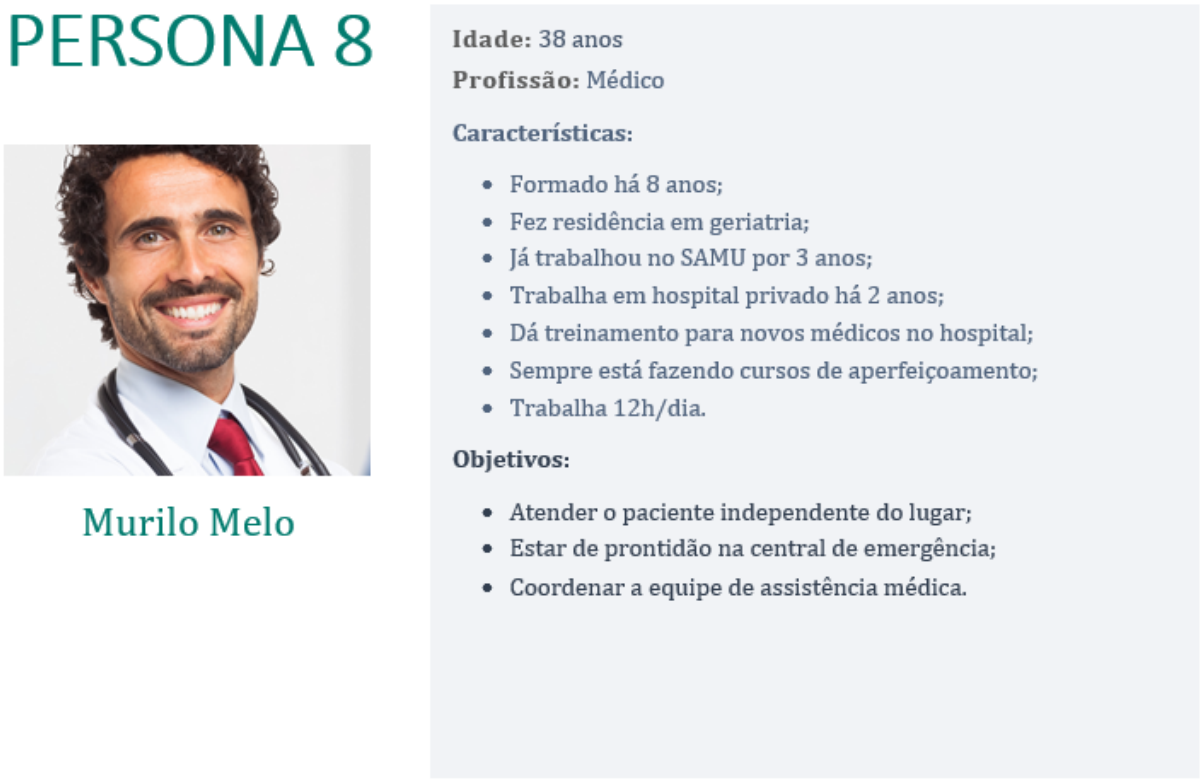

Figura 2.3: Exemplo de uma persona retirada de [23].

2.3.1 Levantamento de requisitos utilizando personas

Em [23] foi definida uma forma de se levantar requisitos utilizando personas. Esse levantamento leva em consideração as etapas existentes da Metodologia Tropos dos requisitos iniciais e finais. Ela introduz uma nova etapa entre elas sendo que as três etapas são feitas de forma que se pode voltar nas etapas passada caso se veja necessário.

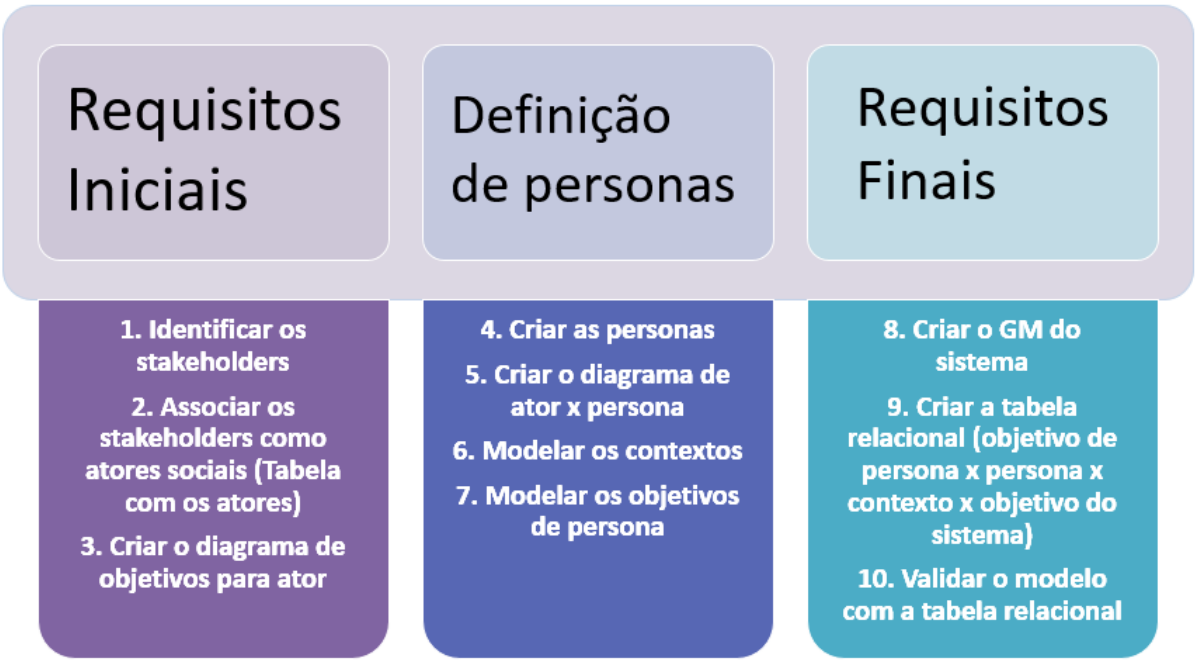

Figura 2.4: Etapas do levantamento dos requisitos retirada de [23].

Vê-se na Figura 2.4 dez passos divididos entre as três etapas. Os três primeiros passos concernem a primeira etapa, a de requisitos iniciais. Neste passo é feita a elicitação das entidades relacionadas com o sistema, os *stakeholders*, e os atores que os representam dentro da modelagem do nosso sistema e como que estão relacionados por meio de seus objetivos.

Os quatro passos seguintes são sobre a definição das personas. Começa-se criando as personas definindo quem são, quais são suas características, historia e objetivos colocando todas essas informações em cartões. Depois, é feita a relação entre os atores criados na etapa passada com as personas. Essa relação é definida por meio da ligação dos objetivos da persona com o ator, sendo que tal ligação é definida por contextos que serão modelados no próximo passo. São nesses passos em que se define o que as personas esperam do sistema e quais são os fatos e sentenças que devem ser cumpridos para que isso ocorra. O próximo passo, então, é modelar os contextos que representam os objetivos das personas e como que eles estão ligado com os atores. Para fazer a modelagem dos contextos usa-se a notação definida em [3] Por último é feita a modelagem dos objetivos de cada persona criada. Nesse passo modela-se o sistema levando em consideração somente a persona em questão e quais são os objetivos que o sistema deve ter para satisfazer os objetivos dela.

Por ultimo temos os três passos finais relacionados com os requisitos finais do sistema. Começamos fazendo o modelo do objetivos do sistema, onde une-se todos os modelos de objetivos das personas criados na etapa passada. Então, cria-se um tabela relacional referente aos objetivos das personas, as personas, os contextos e os objetivos do sistema.

E então é feita a validação do modelo com a tabela, onde verifica-se se todos os objetivos do sistema estão sendo cobertos por alguma persona e contexto.

2.4 Método de Avaliação Empírico

Medir a eficiência de um processo pode ser complicado, porém é de extrema importância quando procura-se alcançar processos eficientes e eficazes. Sem isso, se torna impossível saber no que e em quanto um processo melhorou, o que é feito normalmente na base dos sentimentos das pessoas em relação a como era o processo antes e como ele está agora. Além disso, outro problema é que, mesmo quando percebe-se que a medição é algo importante, saber o que e como medir ainda é um processo confuso quando é feita sem planejamento e sem um objetivo claro.

A fim de solucionar esse problema o GQM foi proposto por [24]. Ele é um método empírico de planejamento de medições de forma a deixar claros os objetivos específicos a serem alcançados com as medições. Ele se baseia em três níveis de medição, um conceitual (objetivo), onde se define primeiro qual o objetivo que se pretende alcançar, por exemplo, o objetivo pode ser melhorar a eficiência dos processos automatizados de testes de software.

Depois disso tem-se o nível operacional (pergunta), onde se definem as perguntas que irão definir se o nosso objetivo foi ou não alcançado. Utilizando o exemplo do objetivo, afirma-se que tem-se uma melhora na eficiência dos processos automatizados de testes de software se soubermos dizer qual é a eficiência desse processos. Ou seja, nossa pergunta seria "Qual a eficiência dos processos automatizados de testes de software?"

E por último, o nível quantitativo (métrica), onde se definem as métricas utilizadas para responder as perguntas, considerando o exmemplo, poderia-ser definir como métricas, os tempo de intervalo entre as falhas que acontecem no software, número de erros que são repostados no período de homologação, número de falhas que são repostadas pelos usuários quando o software está em produção, etc.

Capítulo 3

Estudo Ilustrado

Nesse capítulo será apresentado o detalhamento do desenvolvimento de um SMA com e sem a utilização da abordagem de personas na fase de ER, ambos experimentos no ambiente de vida assistida.

Começa-se ilustrando qual é o contexto no qual nosso sistema estará inserido. Será apresentada uma visão geral do sistema. Na sequência, será descrito o processo de desenvolvimento do primeiro exemplo, começando pela definição do PEAS dos agentes e depois pela definição das etapas definidas na metodologia Tropos. Por último, será descrito o processo do segundo exemplo, onde usa-se a metodologia definida na parte de personas no Capítulo 2.

3.1 Detalhamento da Solução

Ambiente de Vida Assistida

Visa-se o desenvolvimento de um sistema capaz de auxiliar pessoas adultas que morem sozinhas e que precisem de acompanhamento médico, seja por motivo de idade ou por necessidades especiais. Para tal, o cenário que vislumbra-se inclui um dispositivo que é capaz de verificar se o paciente caiu. Ele também possui um botão que pode ser apertado em casos de emergências notificando uma Central de Atendimento que será responsável por determinar qual é a providência necessária dependendo da situação. Junto dele existe um comunicador que é posicionado em um local da casa do paciente. Ele serve para entrar em contato com a Centra de Atendimento caso seja detectada alguma queda e permite a interação por meio da voz entre o paciente e a Central.

O sistema possui seis formas de saber qual a localização do paciente. Ele possui um GPS, que é capaz de se conectar a redes Wi-Fi, o Comunicador, que realiza a comunicação via voz, um alto falante e um sistema inteligente *Bread Crumbs*. Caso o sistema

tenho acesso ao GPS, Wi-Fi ou ao Comunicador ele consegue detectar a localização do paciente. Caso exista algum problema nessas tecnologias, mas o paciente se encontra perto do comunicador e consciente, ele pode informar sua localização diretamente via voz a central. Caso não seja possível usar nenhuma dessas formas, o *Bread Crumbs* então utiliza informações passadas sobre o estado e localização do paciente para tentar achar uma possível localização do paciente e o alto falante emite um som para que ele possa ser encontrado.

O objetivo desse trabalho é o desenvolvimento de um protótipo de SMA que terá que monitorar informações como sinal de GPS, sinal de Wi-Fi e sinal do comunicador para tentar determinar a localização do Paciente e então entrar em contato com a Central de Atendimento.

PEAS

Conforme citado na Seção 2.2.1, o PEAS é o ambiente de tarefas dos agentes de um SMA, que representa o primeiro passo dentro de um projeto, podendo ser considerado como um pré-projeto. Nele precisamos definir quais são os agentes que farão parte do sistema. Faz-se necessário definir qual é o ambiente no qual eles estarão inseridos e quais são as características dele. Depois disso, por meio dos objetivos descobre-se as responsabilidades de cada um desses agentes e o que eles devem fazer. Com o ambiente e objetivos definidos define-se como que eles interagem com o ambiente por meio dos sensores e atuadores para alcançar seus objetivos.

Essa etapa então será dividida em 5 partes:

- definir os possíveis agentes do SMA;
- definir o ambiente de cada agente;
- definir os objetivos de cada agente;
- definir os sensores de cada agente; e
- definir os atuadores de cada agente.

Metodologia Tropos

Depois de ter-se uma ideia de como serão os agentes do sistema, pode-se começar a modelá-lo utilizando a metodologia Tropos.

A primeira etapa é a de Requisitos Iniciais. Nela analisa-se o contexto social e organizacional no qual o sistema atuará. Com base nessa análise modela-se os atores mais importantes dentro desse contexto e a relação entre eles dentro do domínio de operação

do sistema. A segunda etapa é a de Requisitos Finais, nela introduz-se a ideia do sistema que será desenvolvido e a relação dele com os outros atores. Assim, fica claro quais as interfaces que o sistema tem com os atores e quais objetivos dos atores que o sistema resolverá. Na terceira etapa, de Design Arquitetural, faz-se o design do sistema de fato. Para isso, os agentes definidos no PEAS são utilizados como insumo para dividir o sistema, introduzido na segunda etapa, em sub-atores delegando as tarefas e objetivos. Durante a etapa do Design Detalhado usa-se UML para definir as capacidades, protocolos, planos e tarefas dos agentes. Por último são codificados os agentes.

Essa etapa então será dividida em:

- Requisitos iniciais:
  - Modelar os atores mais importantes e a relação entre eles dentro do domínio de operação.
- Requisitos finais:
  - Introduzir o sistema a ser desenvolvido como um novo ator.
- Design Arquitetural:
  - Dividir o sistema em sub-atores delegando tarefas e objetivos.
- Design Detalhado:
  - Utilizar diagramas UML para melhor detalhar as capacidades, protocolos, planos e tarefas de cada agente.
- Implementação:
  - Codificar cada agente utilizando uma linguagem de programação ou uma plataforma de desenvolvimento de SMA, sendo que nesse trabalho foi utilizado o *framework* JADE.

Foi apresentado o método de especificação do projeto de SMA, denominado primeiro exemplo ilustrativo. Na sequência, o segundo exemplo ilustrativo será apresentado.

Metodologia orientada a personas

Para a metodologia orientada a personas foi utilizado como base o trabalho de [23] com uso da modelagem Tropos, conforme descrito no item Metodologia Tropos 3.1. Nos Requisitos Iniciais, procura-se definir os *stakeholders*, associá-los aos atores sociais para

então definir objetivos para eles. E então, ao invés de ir para os Requisitos Finais, define-se as personas, associando-as aos atores que foram definidos na primeira etapa, modela-se os contextos e os objetivos das personas. Nos Requisitos Finais monta-se o *goal model* do sistema e depois relaciona-se os objetivos das personas, as personas, os contextos e os objetivos do sistema.

Essa etapa então será dividida em:

- Requisitos iniciais:
  - Identificar os *stakeholders*;
  - Associar os *stakeholder* a atores sociais;
  - Criar o diagrama de objetivos para os atores sociais.
- Definição das personas:
  - Criar as personas;
  - Criar o diagrama de Ator x Persona;
  - Modelar os contextos;
  - Modelos os objetivos das personas.
- Requisitos finais:
  - Criar o modelo orientado a objetivos do sistema;
  - Criar a tabela relacional objetivos de personas x contexto x objetivos sistema.
- Design Arquitetural:
  - Dividir o sistema em sub-atores delegando tarefas e objetivos.
- Design Detalhado:
  - Utilizar diagramas UML para melhor detalhar as capacidades, protocolos, planos e tarefas de cada agente.
- Implementação:
  - Codificar cada agente utilizando uma linguagem de programação ou uma plataforma de desenvolvimento de SMA, sendo que nesse trabalho foi utilizado o *framework* JADE.

Com essa abordagem, procura-se analisar uma metodologia que seria capaz de prover uma modularização dos requisitos do sistema, isso é, ao invés de se ter um sistema grande com várias funcionalidades que servem para o usuário, tem-se um sistema dividido em funcionalidades com base nos arquétipos de seus usuários, uma vez que dificilmente todos os usuários usam todas as funcionalidades do sistema. Isso já ocorre quando divide-se o sistema por perfis de usuários, porém procura-se mostrar que utilizar personas torna esse processo ainda mais meticuloso. Isso pode facilitar a validação, manutenção e modificação do sistema.

3.2 Exemplos Ilustrativos

Vamos dividir essa seção em duas partes, a descrição do desenvolvimento do primeiro exemplo e a do segundo.

3.2.1 Primeiro Exemplo

No primeiro exemplo, procura-se desenvolver um protótipo que envolvesse o sistema de vida assistida, um usuário que interagisse com o sistema e a central de atendimento. Foi feito assim pois, como não se tem o foco em personas e, por consequência, em usuários, a forma de testar a interação do sistema com os usuários será simples, focando somente nas interfaces entre o sistema e o usuário, e não em suas necessidades. Assim, ficamos com a seguinte modelagem do nosso sistema:

PEAS

Conforme o que se espera do sistema imaginou-se então três agentes: um paciente, o sistema de vida assistida e a Central de Atendimento. Com seus respectivos PEAS:

- Paciente:
  - Desempenho (*Perfomance*): ser atendida em caso de quedas onde ela não consegue levantar sozinha;
  - Ambiente (*Enviroment*): O paciente se encontra em um ambiente com três possíveis estados: em casa, do lado de fora da casa ou em um lugar longe da casa. Conforme descrito na Seção 2.2.1, o ambiente pode ser classificado através de suas características específicas para auxiliar no desenvolvimento dos agentes, neste caso o ambiente é: parcialmente observável, estocástico, sequencial, dinâmico, discreto e multiagente;
  - Ações (*Actions*): pode cair, se levantar e avisar ao Sistema que caiu;

  - Sensores (*Sensors*): consegue sentir se está em pé ou caída.

- Sistema:

  - Desempenho (*Perfomance*): interagir de forma rápida com a Central de Atendimento para assistir as necessidades da Maria em casos de queda;
  - Ambiente (*Enviroment*): o ambiente do sistema é o ambiente no qual Maria está inserida, uma vez que ela está sempre com o dispositivo no qual o Sistema está embarcado. Ele possui as mesmas características do paciente;
  - Ações (*Actions*): enviar mensagens para a Central de Atendimento;
  - Sensores (*Sensors*): GPS, WiFi, Comunicador, *BreadCrumbs*, *AudioBeacon*.

- Central de Atendimento:

  - Desempenho (*Perfomance*): atender prontamente o paciente em caso de queda;
  - Ambiente (*Enviroment*): não considerou-se o ambiente da Central de Atendimento como sendo o local onde ela está fisicamente, mas sim como o local onde ela atua, ou seja, diremos que seu ambiente seria aquele nos quais o paciente pode cair e que a Central de Atendimento teria que enviar ajuda, sendo, assim, o mesmo ambiente de Maria;
  - Ações (*Actions*): enviar ajuda para o paciente;
  - Sensores (*Sensors*): receber mensagens do Sistema.

Requisitos Inicias

Considerando a descrição do PEAS dos agentes foi desenvolvida a modelagem. Nela procura-se manter em foco somente a relação do paciente com a Central de Atendimento.

O paciente tem como objetivo principal ser assistida e para isso é preciso que sua queda seja detectada e que a ajuda seja enviada a sua posição. Além disse ele tem como um *softgoal* não se frustrar com a tecnologia. Para conseguir detectar a queda do paciente precisamos que a queda seja detectada ou que o paciente aperte o botão de emergência por meio do dispositivo. Para que ela seja ajudada é preciso que a ajuda enviada identifique-a.

A Central de Atendimento tem como objetivo principal ajudar seus pacientes. Para isso ela precisa receber a notificação caso algo aconteça, determinar qual é a melhor forma de ajudar e enviar a ajuda. Receber a notificação pode implicar tanto em possuir ou não a localização do paciente. Além disso, consideramos que ela enviou ajuda ao paciente quando é enviada uma ambulância para o local depois que foi determinada a rota que ela fará. E para determinar o melhor plano ela precisa descobrir qual é o paciente, buscar seus dados e então determinar o melhor plano.

A relação entre os agentes é explicitada por meio de dois outros objetivos da Central, que é enviar ajuda ao paciente e detectar queda do paciente.

A modelagem dos Requisitos Iniciais se encontra na Figura 3.1.

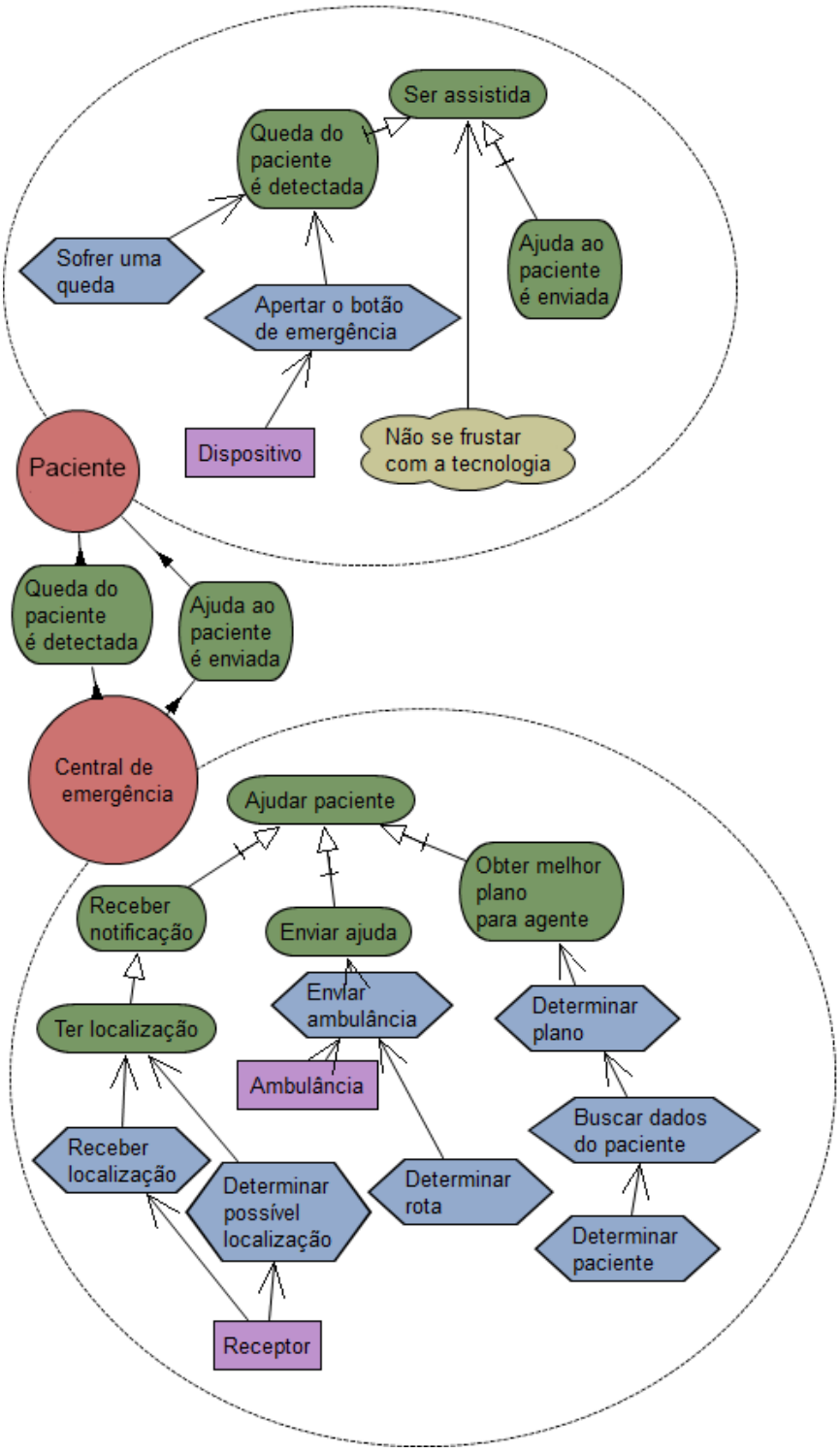

Figura 3.1: Requisitos Inicias.

Requisitos Finais

No desenvolvimento dos requisitos finais inseriu-se o Sistema de Vida Assistida (SVA) cujo objetivo então é assistir o paciente. Para isso ela deve monitorar o estado do paciente, detectar a ocorrência de emergências e enviar o estado do paciente para a Central de Atendimento. Vamos considerar que o estado dela está sendo monitorado quando o sistema verificar os sensores que ele possui. Ele detectará emergências quando ocorrer uma queda e o estado do paciente for detectado por meio do comunicador que tentará entrar em contato com o paciente. Por último será considerado que o objetivo de enviar estado do paciente para a central foi cumprido quando uma mensagem for enviada a central com as informações do paciente.

Depois de modelar o sistema tivemos que modelar a relação dele com os outros atores. Para isso introduziu-se um novo objetivo para o paciente que é o de ter o estado monitorado pelo SVA, que será cumprido uma vez que ela enviar seus dados ao SVA, seja diretamente por meio do botão indicando uma emergência, ou indiretamente tendo o estado monitorado pelos sensores.

Teve-se que remodelar a Central de Atendimento. Nela adicionou-se um *softgoal* para garantir a segurança do paciente relacionado a ajudar o paciente. Além disso também viu-se a necessidade de refinar o cumprimento do objetivo de se ter a localização do paciente. Agora ele será cumprido quando for recebido o sinal do comunicador, uma localização exata, o sinal do botão de emergência, um sinal do detector de queda e então usar essas informações para determinar o possível local de queda.

Após essas alterações fizeram-se as conexões entre os agentes. Entre o paciente e a Central não houveram alterações. O SVA é responsável por ajudar a Central a garantir a segurança do paciente e por receber a notificação de uma emergência, enquanto isso a Central é responsável por receber as mensagens do SVA. O paciente relaciona-se com o sistema pois tem seus estado monitorado.

A modelagem dos Requisitos Finais se encontra na Figura 3.2.

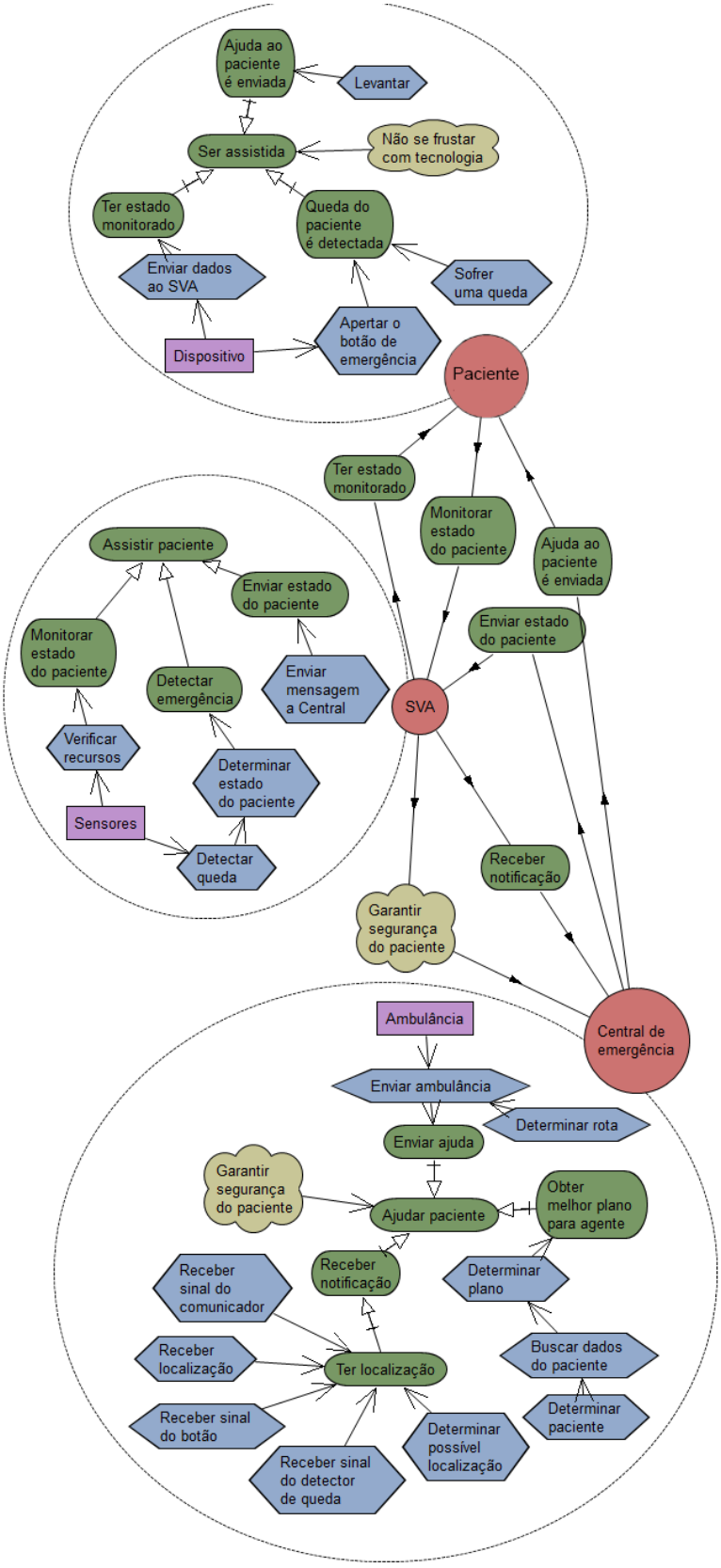

Figura 3.2: Requisitos Finais.

Design Arquitetural

Para modelar a arquitetura do sistemas precisou-se dividir o sistemas em outros agentes dividindo suas responsabilidades. Assim dividiu-se o SVA em um Gerenciador de Ações responsável por receber as informações do paciente e tentar descobrir qual é a melhor ação a se tomar e um Notificador responsável por entrar em contato com a Central de Atendimento.

Assim o Gerenciado de Ações obtém informações sobre a paciente e quando percebe que houve uma emergência notifica o Notificador que enviar uma mensagem a Central.

A modelagem dos arquitetura do sistema se encontra na Figura 3.3.

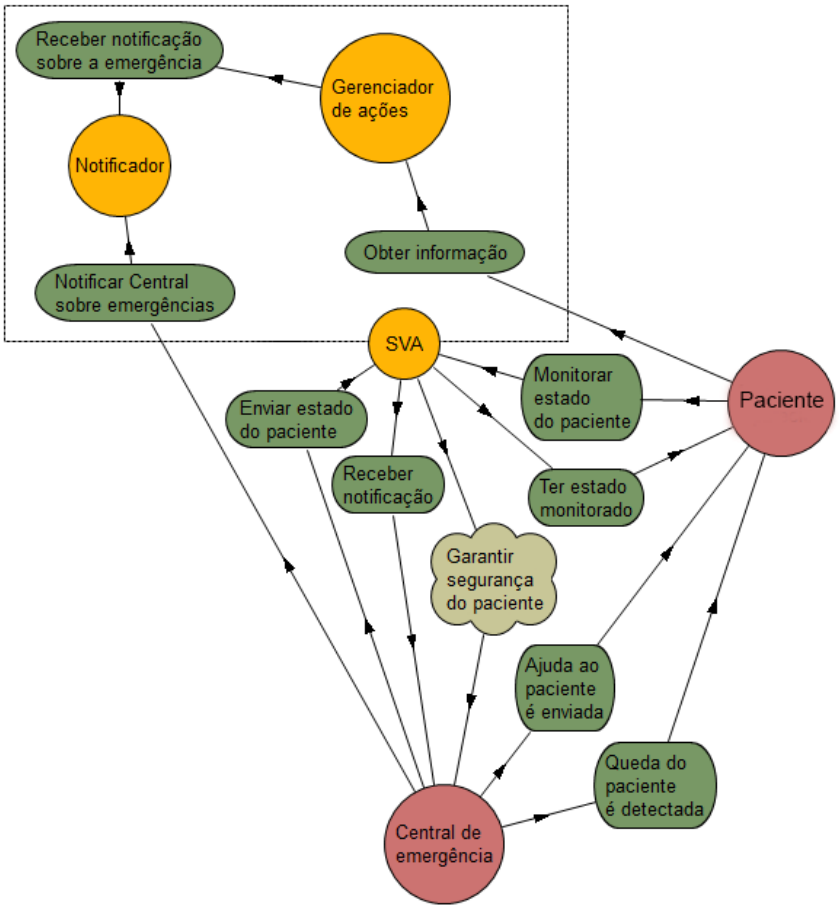

Figura 3.3: Design Arquitetural.

Design Detalhado

O Design Detalhado envolve demonstrar em detalhes como que os agente funcionam, para isso usa-se UML para descrever: como seriam as trocas de mensagem entre os agentes para a realização das tarefas, diagrama de atividades que define o ciclo de vida dos agentes e um diagrama de classes dos agentes.

Analisando a Figura 3.4 percebe-se que o paciente começa em uma atividade chamada Vive, que simularia o cotidiano de sua vida. Dele seria possível ela realizar três atividades que representariam emergências, ela cair dentro de casa, ela cair do lado de fora da sua casa e ala cair em algum lugar longe de sua casa e onde não existe sinal GPS, Wi-Fi nem do comunicador. De qualquer um desses ela realiza a atividade de espera da ajuda e por último ela se levanta.

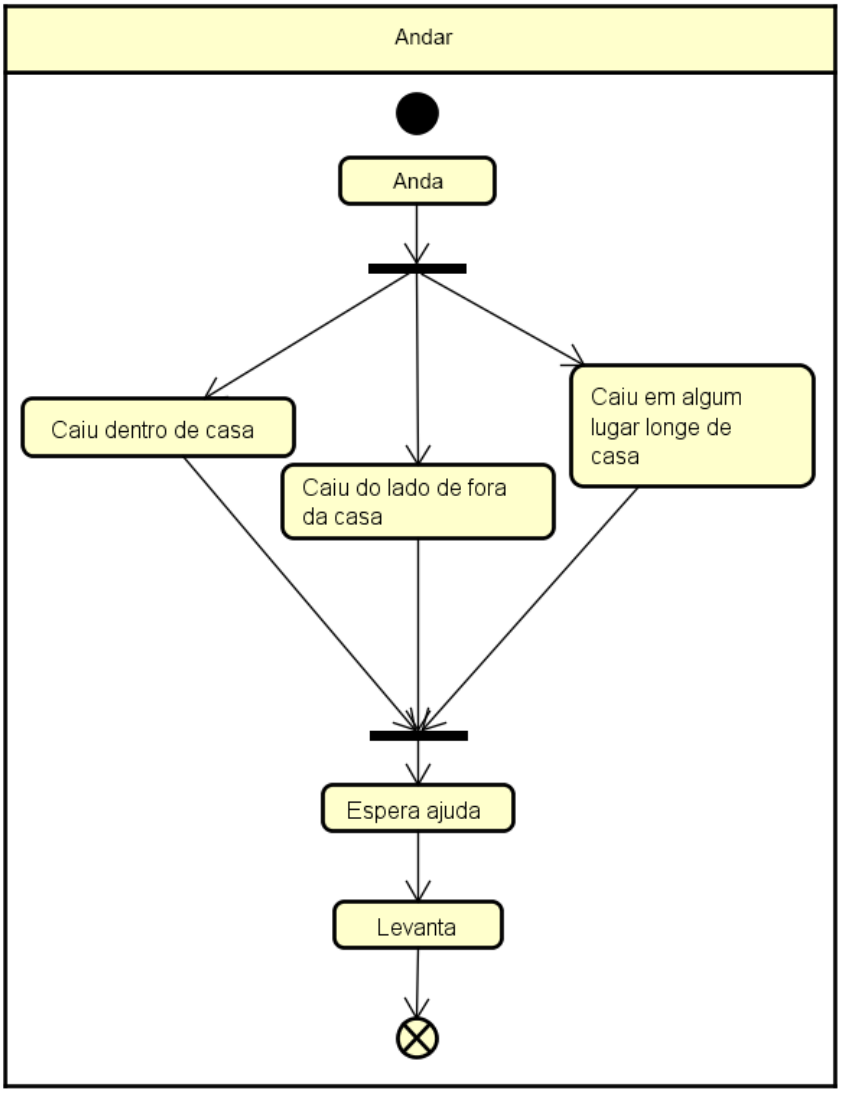

Figura 3.4: Design Detalhado - Diagrama de Atividade do Agente Paciente.

Além disso, pode-se notar na Figura 3.5 um exemplo de como deve funcionar a ação de levantar o paciente. No caso será uma mensagem do tipo *Request* que simulará a ambulância ir até o paciente e ajudá-lo de fato.

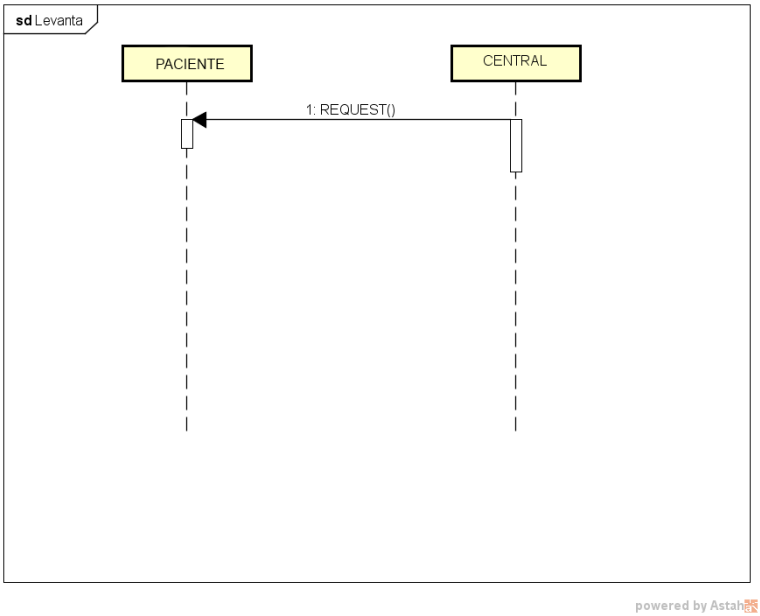

Figura 3.5: Design Detalhado - Diagrama de Sequência.

Por último mostra-se na Figura 3.6 como serão divididas e separadas as classes dos agentes quando forem implementados no *framework*.

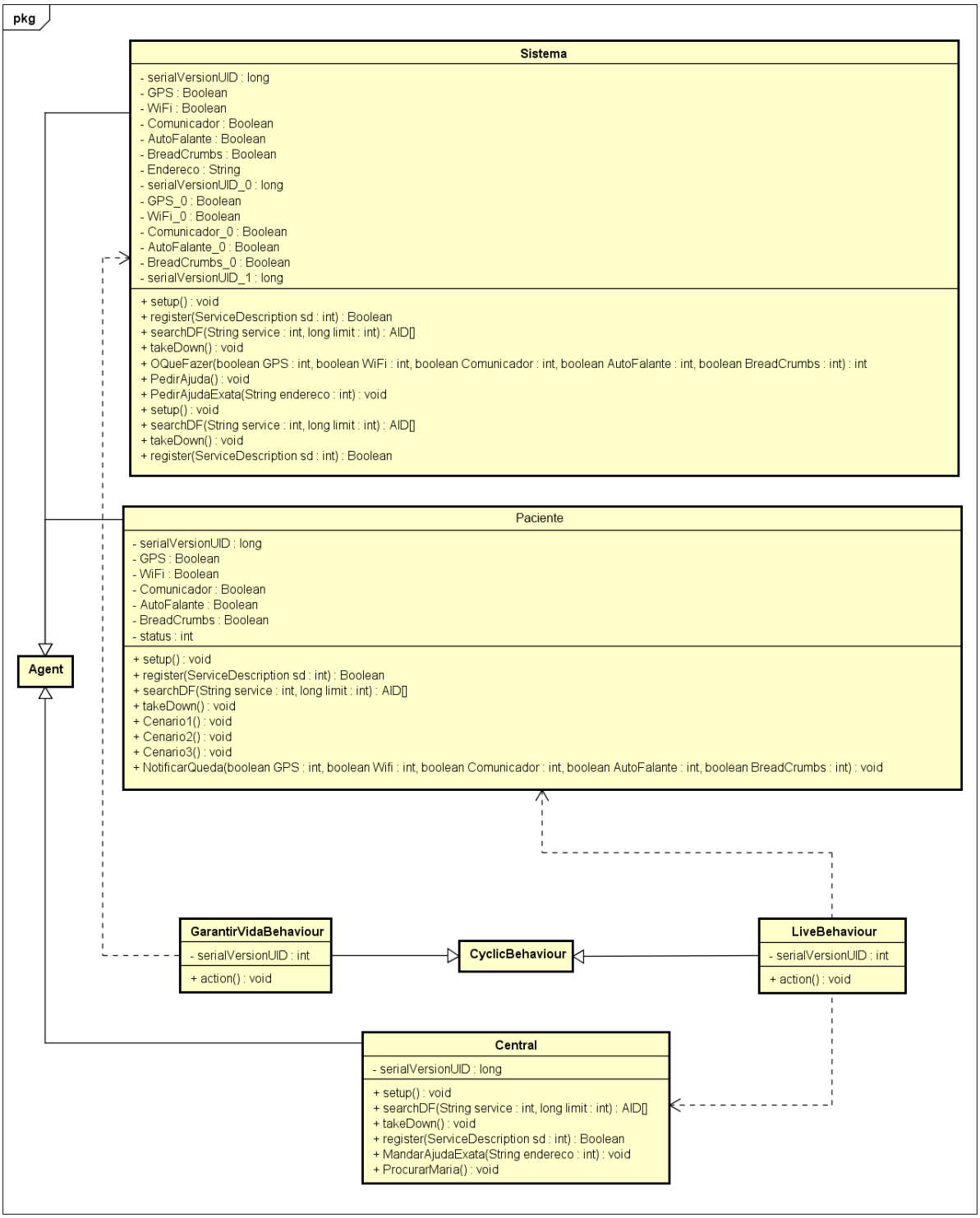

Figura 3.6: Design Detalhado - Diagrama de Classes.

Implementação

Para implementar nosso agente utilizou-se o *framework* JADE e os dividimos conforme o diagrama de classes da Figura 3.6. Pode-se ver um exemplo do código do Gerenciador

de Ações que faz parte da classe Sistema na Figura 3.7

```
49              if(msg != null) {
50                  switch(msg.getPerformative()) {
51                      case ACLMessage.INFORM: {
52                          String[] argumentos = msg.getContent().split("-");
53                          if(argumentos[0].equals("Maria caiu")){
54                              System.out.println("O sistema percebeu que Maria caiu");
55                              GPS = Boolean.parseBoolean(argumentos[1]);
56                              WiFi = Boolean.parseBoolean(argumentos[2]);
57                              Comunicador = Boolean.parseBoolean(argumentos[3]);
58                              AutoFalante = Boolean.parseBoolean(argumentos[4]);
59                              BreadCrumbs = Boolean.parseBoolean(argumentos[5]);
60
61                              int action = OQueFazer(GPS, WiFi, Comunicador, AutoFalante, BreadCrumbs);
62
63                              if(action == 1){
64                                  System.out.println("O sistema sabe que Maria está em casa.");
65                                  PedirAjudaExata(Endereco);
66                              }if(action == 2){
67                                  System.out.println("O sistema sabe onde Maria está.");
68                                  PedirAjudaExata("707 Bloco B Casa 28");
69                              } else if(action == 3){
70                                  System.out.println("O sistema não sabe onde Maria está.");
71                                  PedirAjuda();
72                              }
73                          }
74
75                      } break;
76                  }
```

Figura 3.7: Parte do código do SVA.

Umas das funcionalidade oferecidas pelo *framework* é mostrar as trocas de mensagens ACL entre os agentes. Um exemplo das trocas de mensagem de uma execução pode ser vista na Figura 3.8 retirada do *sniffer* provido pela ferramenta JADE. Em azul escuro, tem-se uma mensagem do tipo *inform*, que é utilizada quando um agente quer informar algo para outro agente. Ela representa os sensores indicando para o sistema que o paciente caiu e quais são os sinais que ele tem disponível para descobrir a localização do paciente. Além disso tem-se em vermelho uma mensagem do tipo *request*, que é utilizada quando um agente quer requisitar algo de outro agente. Ela representa o sistema requisitando para a Central ajuda para o paciente.

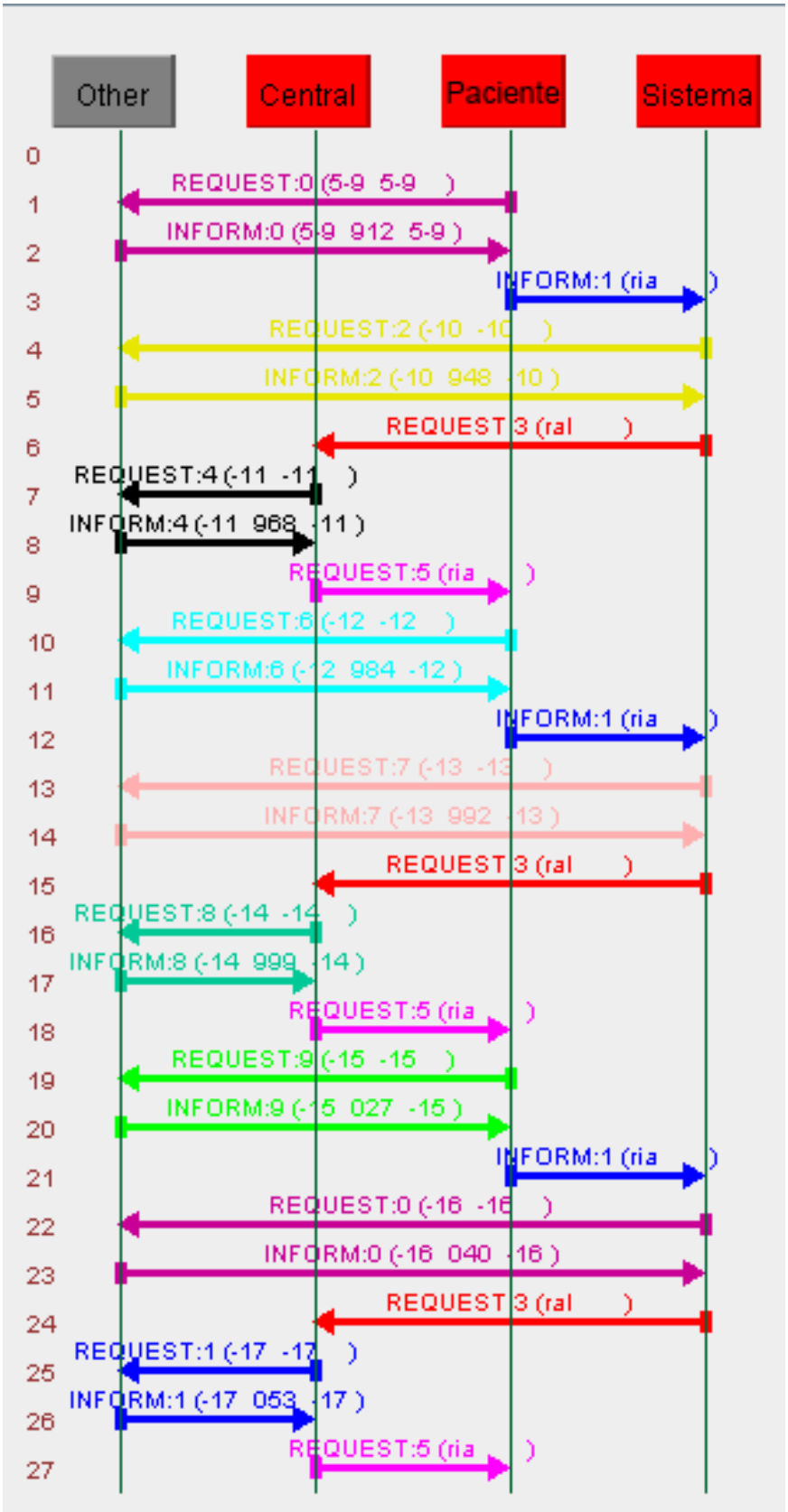

Figura 3.8: Troca de mensagens entre os agentes.

Além disso, também usou-se o console para indicar o que estava acontecendo nas execuções do sistema, ele demonstra o cenário no qual o paciente se encontra e as reações do sistema e da Central.

Tem-se um exemplo na Figura 3.9. Nela encontra-se os três cenários possíveis. No primeiro o paciente cai dentro de casa, o sistema percebe isso e informa a central por meio do próprio comunicador. No segundo o paciente cai do lado de fora de sua casa, o sistema detecta a queda e por meio do GPS e do Wi-Fi descobre onde ela está e informa a central. Por ultimo tem-se o caso onde o paciente cai em um lugar onde não é possível detectar a localização. Assim a ambulância enviada pela central deve procurar onde ocorreu o acidente para então conseguir ajudar.

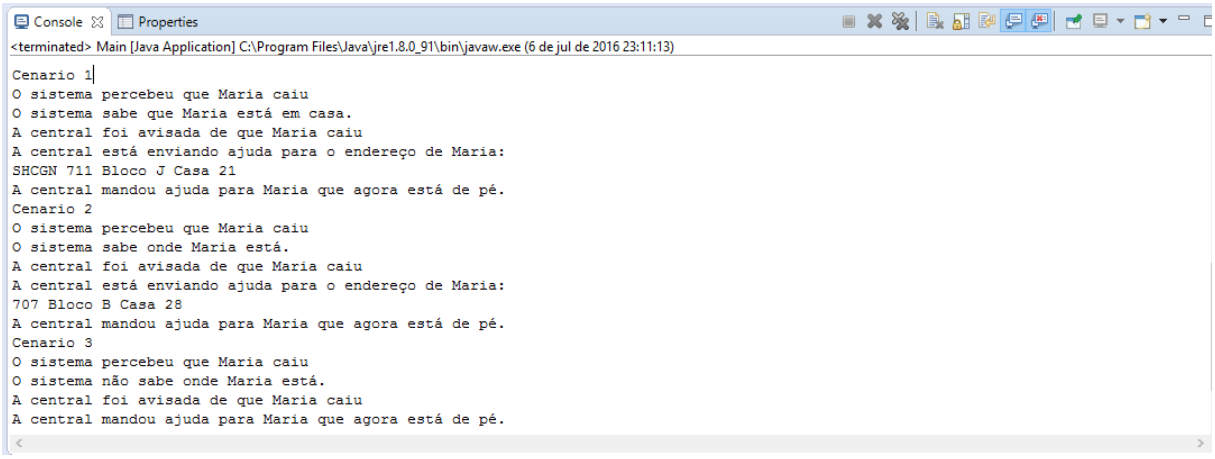

Figura 3.9: Exemplo da saída do console.

Assim se termina o desenvolvimento do primeiro exemplo ilustrativo.

3.2.2 Segundo Exemplo

Nosso segundo exemplo tem como foco as personas, quer-se tentar melhorar a experiência do usuário. Portanto, a ideia inicial do exemplo é desenvolver um sistema que ao invés de simular a interação com um usuário, será de fato utilizado por um. Ao invés de desenvolver agentes Paciente e Central de atendimento que trocaram mensagens com o sistema, iremos desenvolver interfaces que terão entradas fornecidas pelo o usuário de fato.

Requisitos Iniciais

O primeiro passo dos requisitos iniciais é identificar quais são os *stakeholders* do sistema. Pode-se ver que as principais entidades relacionadas ao sistema são os usuários

pacientes do sistema, o próprio sistema e as centrais de atendimento de emergências. Depois disse fez-se a tabela que relaciona esses *stakeholders* aos atores sociais do sistema que pode ser vista na Tabela 3.1.

Tabela 3.1: Tabela de Relação *Stakeholder* e Ator Social

Stakeholder	Ator
Usuário	Paciente
Sistema de Vida Assistida	SVA
Centra de Atendimento	Central

Depois disso criou-se o diagrama de objetivos para os atores sociais. Tem-se que o principal objetivo de um Paciente é ser assistido e que o principal objetivo da Central é conseguir atender emergências. O objetivo do nosso sistema seria servir de interface entre esses dois atores por meio de seus objetivos, assim, o SVA seria responsável por assistir o paciente notificando emergências para a Central. Pode-se verificar esse diagrama na Figura 3.10

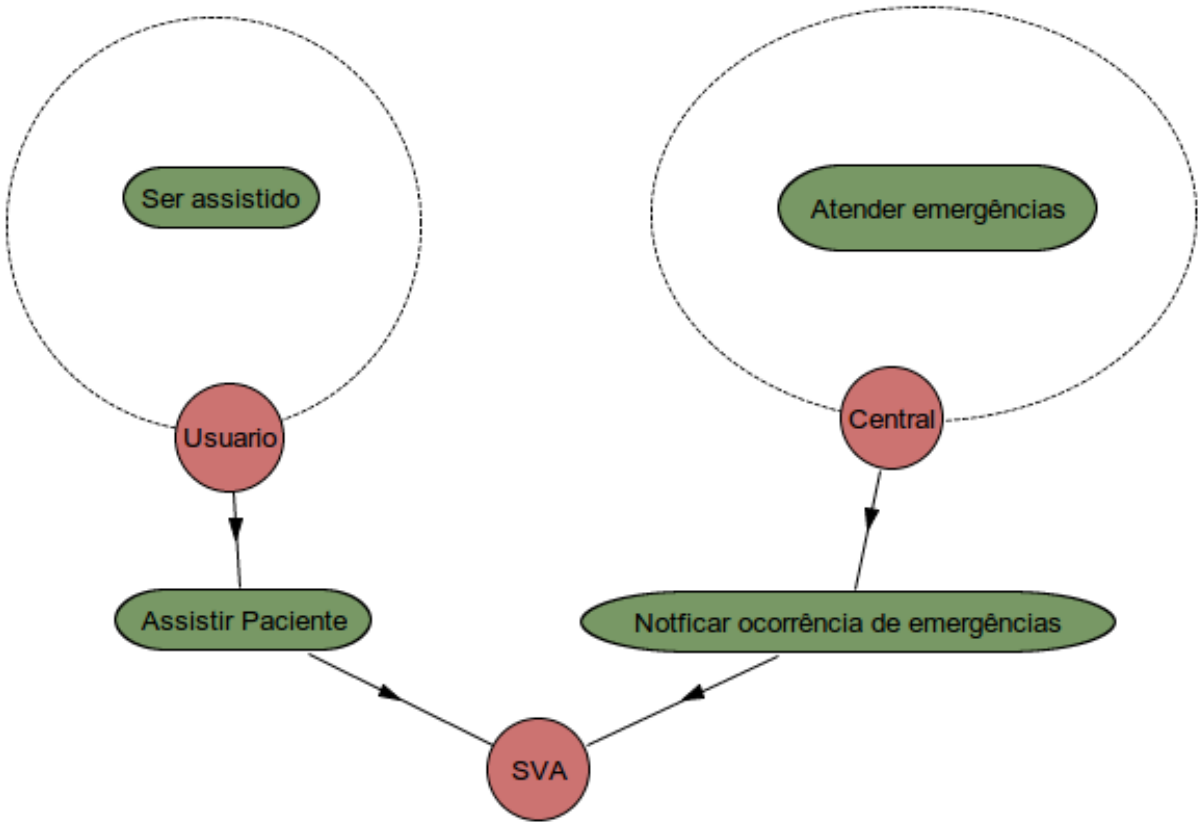

Figura 3.10: Diagrama de objetivos para os atores sociais.

Definição das personas

Depois do requisitos iniciais começou-se a etapa de definição de personas. Como a preocupação é o paciente, criou-se uma persona somente para o Paciente, porém personas que representam a Central de Atendimento também poderiam ser criadas, como foi feito em [23].

A persona utilizada será uma definida em [23]. Ela é Maria Aparecida, uma mulher de 70 anos aposentada. Ela mora sozinha em uma pequena casa e não possui uma empregada, mas possui uma diarista de 15 em 15 dias. Assim ela mesma realiza grande parte das tarefas domésticas. Possui um registro de quedas dentro de casa e tem Osteoporose. Ela tem hábitos majoritariamente diurnos mas tende a acordar até duas vezes a noite para ir ao banheiro. Tem dois filhos adultos que moram sozinhos. Ela não possui Wi-Fi em casa. Ela procura não se frustrar com tecnologia, por isso precisa de algo simples para lhe atender. Além disso ela não quer preocupar seus filhos, quer se sentir segura dentro de sua própria casa e quer ter qualidade de vida. Podemos ver um cartão que a representa na Figura 3.11

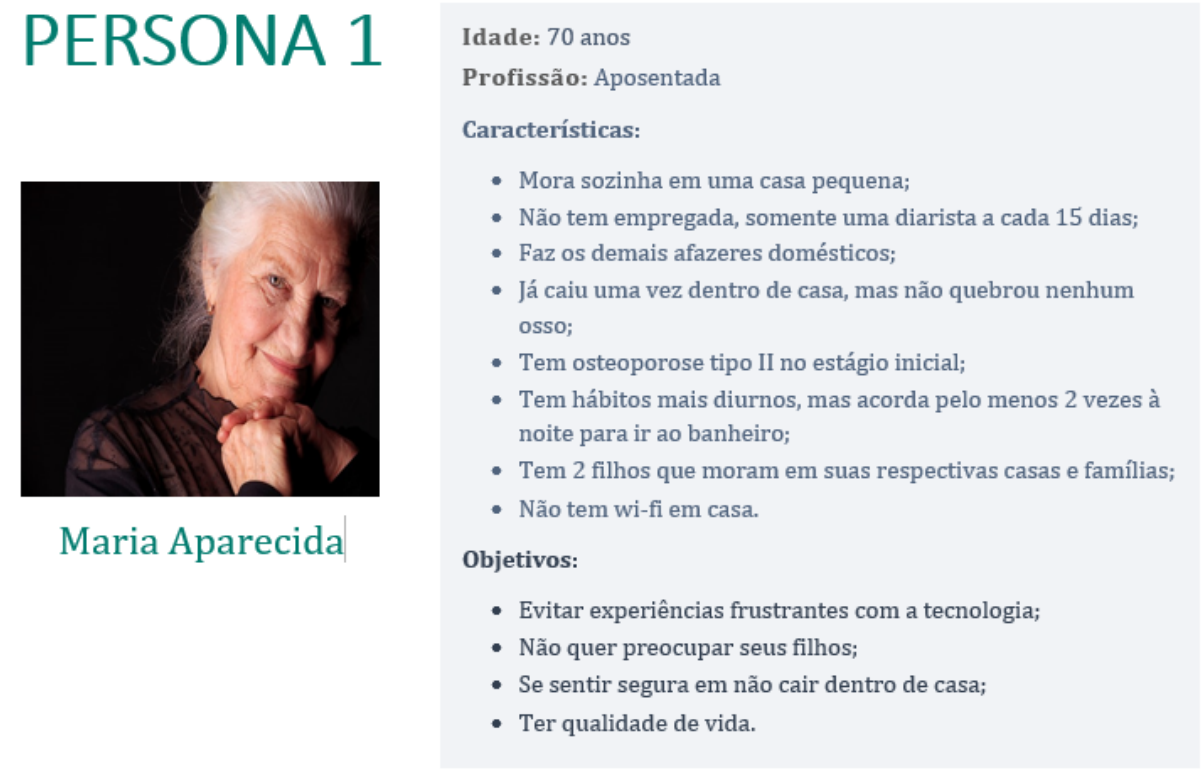

Figura 3.11: Cartão da Maria Aparecida.

Depois de definir a persona fez-se o diagrama Ator X Personas, apresentado na Figura 3.12. Além do objetivo de "Ser assistido" nota-se que Maria também introduz outros três objetivos para o ator Usuário. Não preocupar os familiares, sentir-se segura em casa e

evitar experiências frustrantes com a tecnologia. Cada objetivo do usuário que é mapeado a uma persona é caracterizado por um contexto, representados na Figura por C1, C2 e C3.

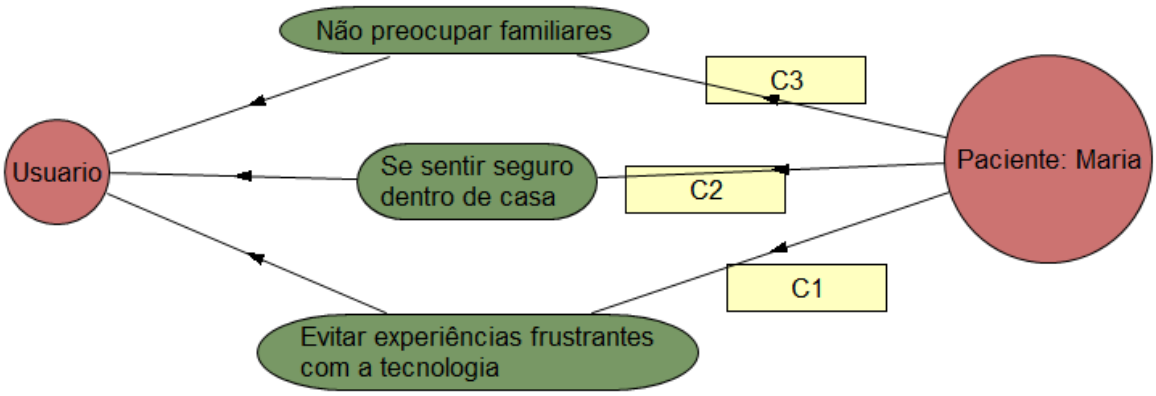

Figura 3.12: Diagrama Ator x Persona.

Depois de fazer isso precisa-se modelar os contextos que foram definidos para cada objetivo. Os contextos são modelados por meio de um grafo AND-OR que relaciona sentenças e fatos. Os paralelogramos representam quais os fatos do contexto e os retângulos representam as sentenças, além disso os triângulos preenchidos representam uma ligação AND, enquanto triângulos vazios representam uma ligação OR.

O primeiro contexto é referente a não se frustrar com o uso de tecnologia e para isso diz-se que caso o usuário caia, o sistema irá detectar sozinho e irá notificar a Central e caso ela aperte o botão o sistema notificará a Central de forma correta, ou seja, sem que aconteça algum erro entre o apertar o botão e o atendimento do paciente. Ele pode ser visto na Figura 3.13 usando a notação definida em [3].

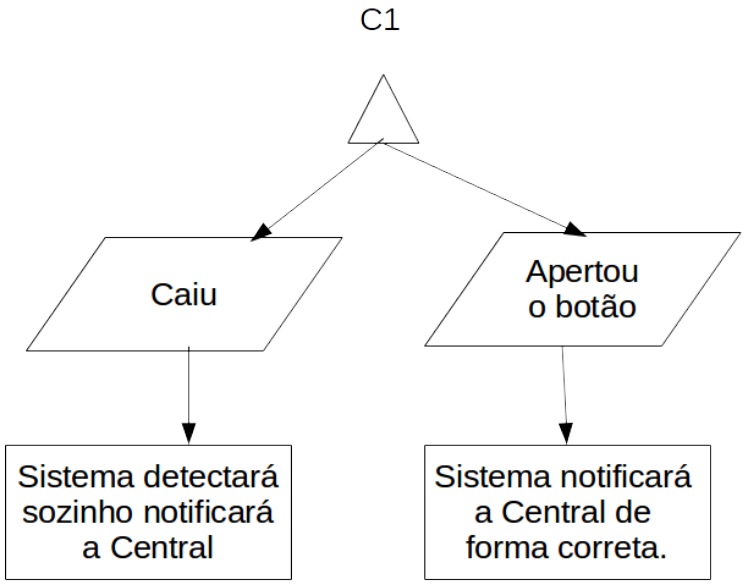

Figura 3.13: Contexto 1

O contexto 2 é referente a se sentir seguro em casa. Isso acontecerá enquanto todas as emergências que ocorrerem forem resolvidas com sucesso. Ele pode ser visto na Figura 3.14.

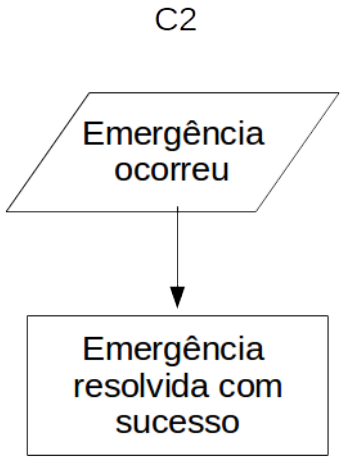

Figura 3.14: Contexto 2.

Por último, o contexto 3 é sobre não preocupar seus familiares, ele ocorrerá sempre que ocorrer uma emergência que o Usuário considerar leve e seus familiares não forem notificados. Ele pode ser visto na Figura 3.15.

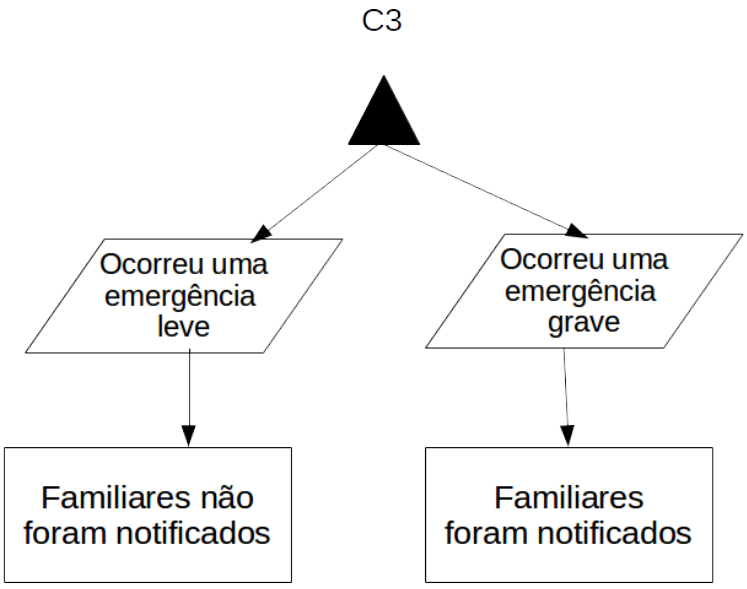

Figura 3.15: Contexto 3.

Uma vez modelados os contextos, a última subetapa dentro da Definição de Personas é a modelagem dos objetivos das personas. Nessa etapa procura-se modelar o sistema analisando as necessidades delas. O objetivo do sistema é prover a segurança ao paciente. Para isso ele precisa da localização dele, detectar uma emergência e notificá-la. Sua localização é uma informação conhecida quando existe sinal do comunicador, do GPS ou do Wi-Fi. Uma emergência é detectada quando os sinais da paciente são monitorados por meio dos sensores. Uma emergência é notificada quando a central é avisada. Podemos ver o modelo na Figura 3.16

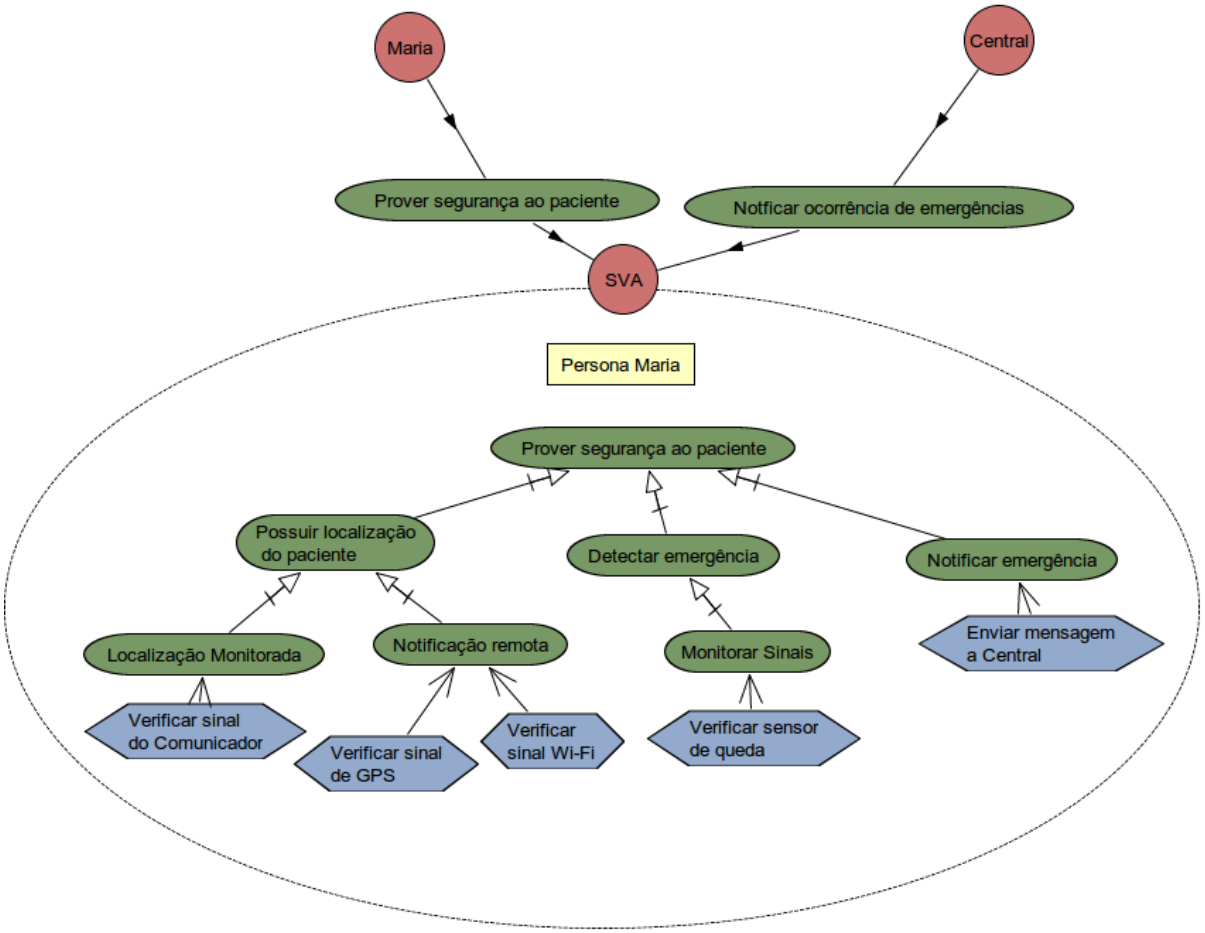

Figura 3.16: Modelo de Objetivos da Persona

Requisitos Finais

Nos requisitos finais termina-se de fazer a modelagem do sistema. Quando se tem mais de uma Persona envolvida, é preciso unir os modelos de seus objetivos em um único modelo do sistema. Como no nosso caso só existia uma Persona, o modelo de seus objetivos e o modelo do sistema acabam sendo o mesmo. Devemos então só terminar de montar o resto do modelo com os objetivos das outras entidades.

A modelagem final da Central ficou com o objetivo principal de atender emergências. Para isso ela deve receber uma localização, enviar ajuda ao local e confirmar depois que a assistência ocorreu. Além disso, ela tem como *softgoal* notificar familiares caso a emergência seja classificada como grave. A localização é recebida por meio de uma mensagem do SVA, a assistência se dá pelo envio de uma ambulância ao local e se confirma a assistência ligando para a Persona depois do ocorrido.

O ator usuário quer ser assistido. Além disso ele possui os objetivos da Persona Maria. Para ele ser assistido ele precisa ter sua emergência detectada e depois receber a visita de uma ambulância. O modelo pode ser visto na Figura 3.17.

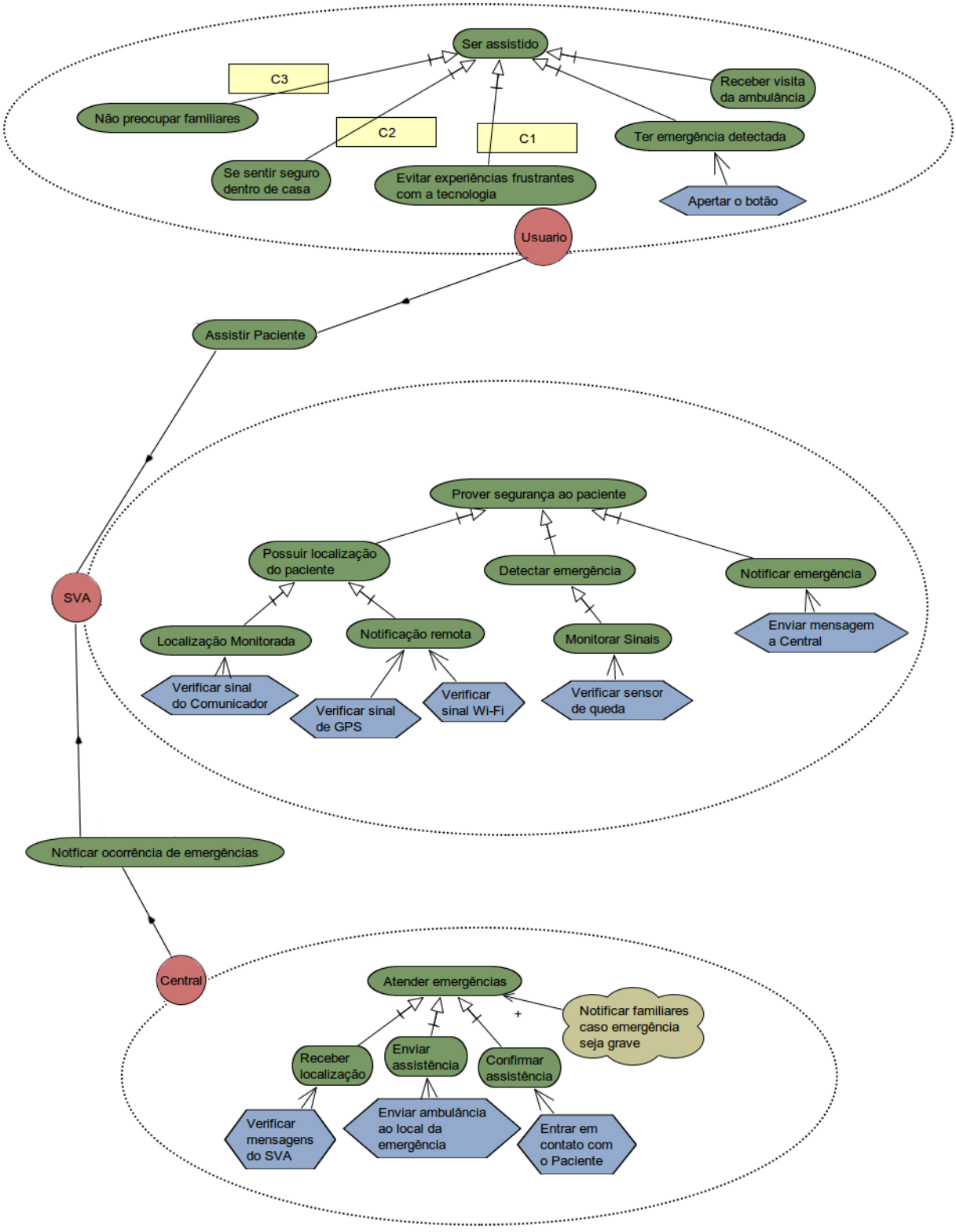

Figura 3.17: Modelo Completo

Design Arquitetural

A metodologia que é definida em [23] é feita somente para o levantamento de requisitos. Por isso não se entra em detalhes a partir dessa etapa de desenvolvimento. Desta forma, as próximas etapas foram realizadas de forma idêntica ao primeiro exemplo.

No caso, não houveram grandes diferenças de fato no modelo final do sistema. Por mais que existam alguns objetivos expressos de forma diferente eles de maneira geral retratam a mesma coisa. Por causa disso o modelo da arquitetura também ficou parecido. Tivemos dois agentes dentro do sistema. Um responsável por monitorar o Paciente, seja pelos sensores ou pelo botões e o outro responsável por notificar a Central caso aconteça alguma emergência.

O Monitorador deseja monitorar uma emergência por meio dos sensores de queda e do botão. O notificador espera receber uma mensagem do Monitorador para então Notificar a Central. Podemos verificar a arquitetura na Figura 3.18

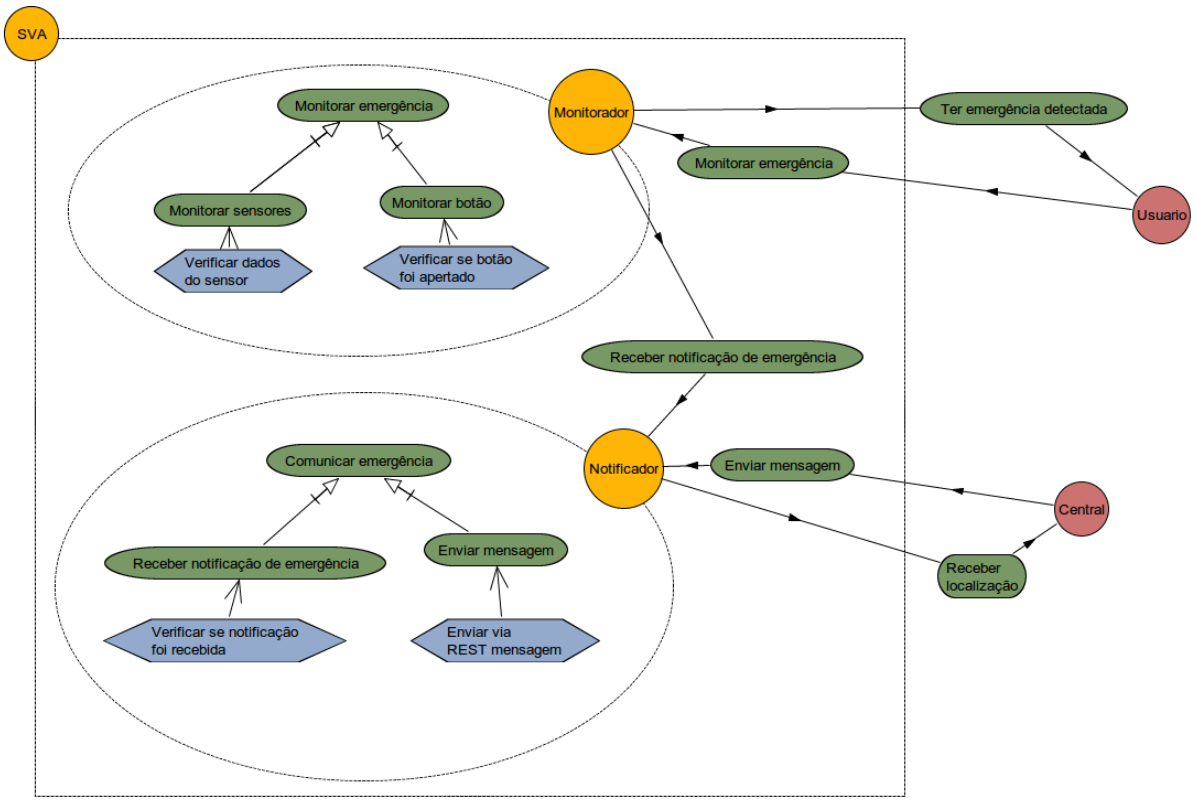

Figura 3.18: Arquitetura do Sistema utilizando a abordagem de personas

Design Detalhado

Mostra-se aqui os mesmos diagramas utilizados no processos anterior para depois compará-los.

No diagrama de atividades de Maria nota-se que seu estado de vida está diferente. Agora, sempre que ela estiver fazendo alguma atividade de risco ressaltamos que ela lembra se quando ocorreu alguma queda tudo ocorreu de forma esperada e se ela não teve complicações com o sistema. Fez-se isso pois foi preciso deixar claro o contexto de que ela deseja se sentir segura em casa. Além disso no final de uma emergência ela também verifica se algum familiar dela foi notificado de forma errônea. Podemos ver o diagrama na Figura 3.19.

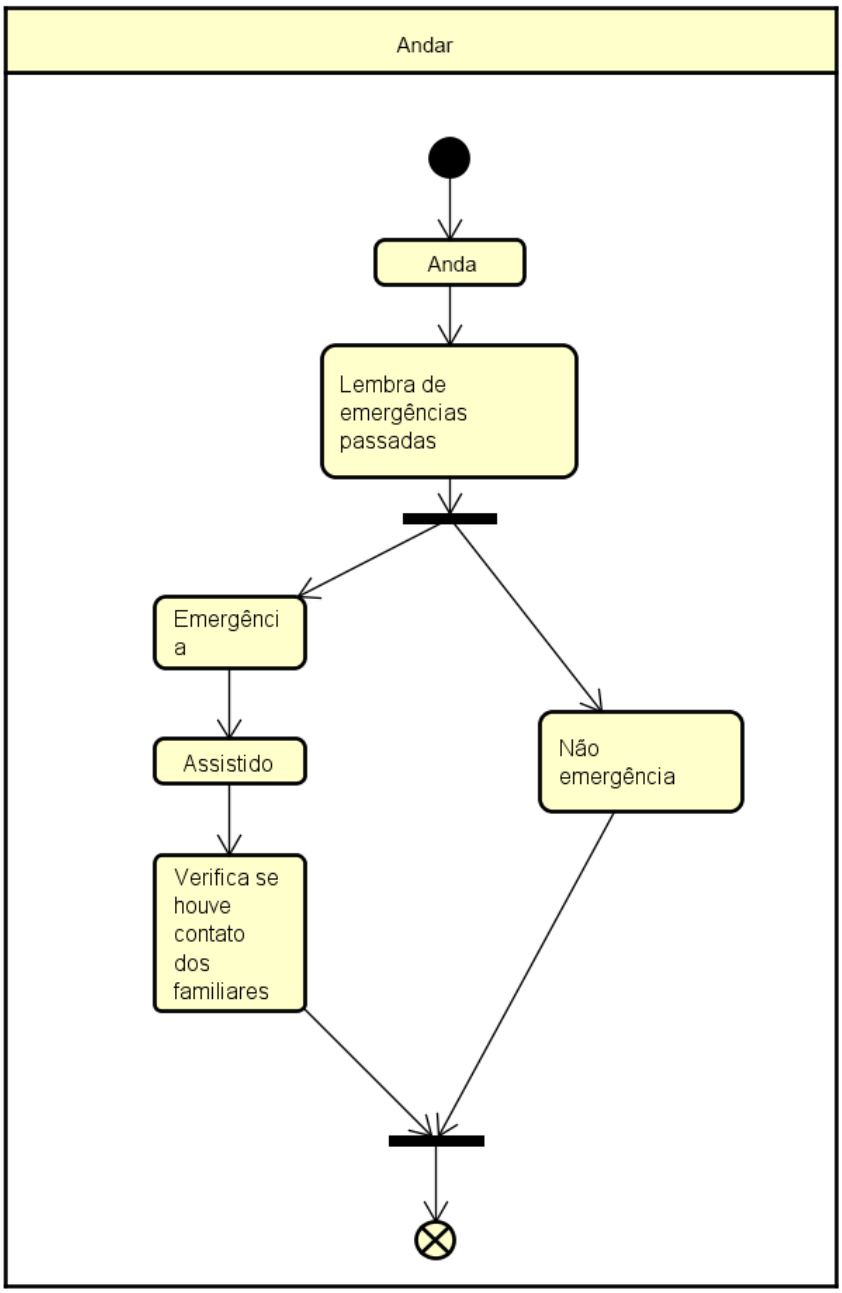

Figura 3.19: Diagrama de atividades de Maria.

No diagrama de sequência do ato de levantar Maria colocou-se mais algumas etapas. Agora quando a ambulância chega ao local ela notifica a central da gravidade da emergência, que então entre em contato com os familiares caso ela seja grave. E no final ela entra em contato de novo informando que a assistência ocorreu com sucesso para que a Central então entre em contato com o Paciente para verificar se está tudo certo de fato. Podemos ver o diagrama na Figura 3.20

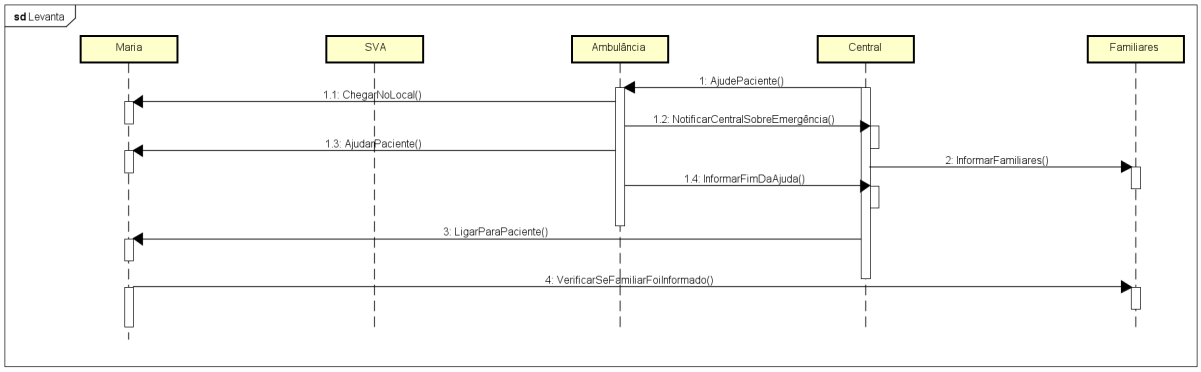

Figura 3.20: Diagrama de sequências do ambiente de vida assistida para persona Maria.

Implementação

Como as diferenças do sistema estavam mais relacionadas aos processos relacionados a seu funcionamento, não houve necessidade de alterar muitas coisas do sistema produzido em 3.2.1. As únicas alterações que aconteceram foram para prover um suporte para testar o serviço via Web utilizando um *Add-On* do JADE chamado WSIG [18]. Por meio dele foi possível fazer uma integração web que permitiria ao usuário testar o funcionamento do sistema de maneira simples, tanto para o paciente simular que ele apertou o botão, quanto para a central ver como seria o recebimento de uma mensagem.

Pela interface web o paciente iria interagir com o sistema para indicar possíveis emergências. Podemos ver a tela onde o usuário vê quais são os sinais que o sistema possui para determinar a localização na Figura 3.21.

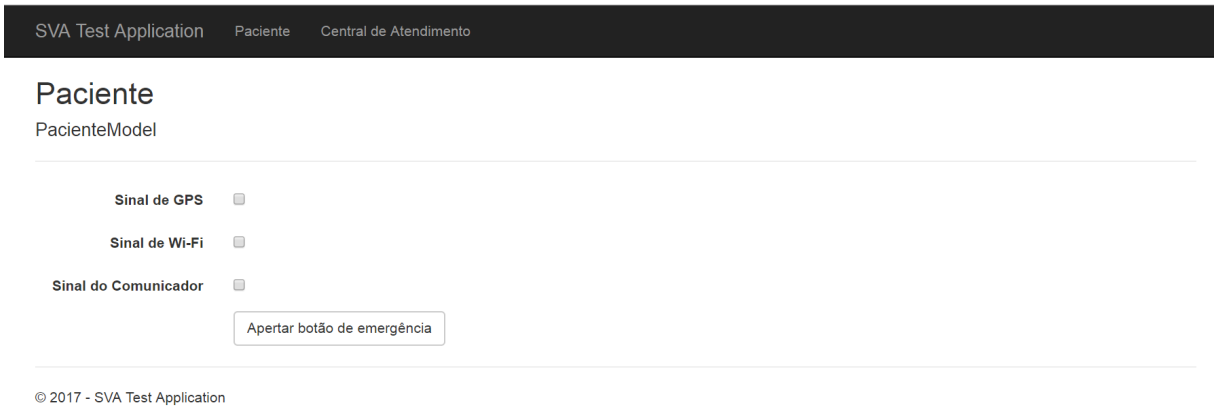

Figura 3.21: Interface Web de simulação de apertar o botão de emergência.

Capítulo 4

Análise do estudo

Esse Capítulo está dividido em três partes. A primeira será uma comparação entre os dois exemplos que foram desenvolvidos, apontando suas semelhanças e diferenças. Depois disso comenta-se quais dessas diferenças foram graças a abordagem de personas. Então, responderemos às questões definidas utilizando o método GQM.

4.1 Comparação

Para realizar a comparação serão analisados os produtos finais de cada etapa em comum entre os processos e analisá-los.

4.1.1 Requisitos Iniciais

Depois do primeiro processo tinha-se um modelo que relaciona os principais atores envolvidos no contexto do nosso sistema. No caso do segundo processo possuía-se a relação entre os *stakeholders* e os atores e um modelo que relaciona esses atores por meio de objetivos.

Como no segundo processo ainda tem-se uma etapa antes dos requisitos finais, não foi preciso entrar em tantos detalhes na modelagem do objetivos dos atores. Se por um lado tem-se que isso torna a primeira etapa do processo mais rápida, ela também contém menos informações para serem analisadas e confirmadas.

Além disso, no primeiro exemplo existia somente a relação entre os atores sem o sistema, já no segundo o sistema estava inserido.

Percebe-se que o intuito do primeiro processo está mais relacionado em modelar o ambiente e contexto no qual o sistema está inserido para tentar analisar depois o que ele pode resolver, já o segundo procura já desde o começo deixar claro qual é o papel do sistema no ambiente.

4.1.2 Requisitos Finais

Como o primeiro processo não possui a etapa de Definição de personas analisou-se direto os Requisitos finais. Porém, percebe-se que por causa dessa etapa, ao modelar o diagrama final de objetivos do sistema notamos que existiam vários objetivos novos que deviam ser levados em consideração por causa da Persona Maria. Porém esses novos objetivos não mudaram muito a modelagem do sistema em si. Eles fizeram mais diferença na modelagem dos outros atores.

Considerando que no exemplo utilizado em [23] essa abordagem trouxe novos objetivos ao próprio sistema, nota-se que diferentes personas trazem diferentes contribuições a modelagem. Iremos discutir mais a frente que as principais contribuições da Persona foi prover insumos para verificar se o sistema será útil ao usuário e não na modificação de sua modelagem.

Outra diferença importante entre os modelos é que, como o processo orientado a objetivos também focou no desenvolvimento dos outros atores como agentes, seus modelos acabaram ficando um pouco mais complexos, enquanto com a abordagem de personas focou mais em objetivos gerais e não entrou muito no mérito de subobjetivos e tarefas. E mesmo assim, a modelagem do sistema, que seria nosso produto principal, ficou bem parecida.

4.1.3 Design Arquitetural

Pode-se perceber que as arquiteturas finais são basicamente as mesmas, o que era de se esperar considerando que a principal diferença entre as modelagens finais não aconteceu no sistema.

4.1.4 Design Detalhado

Voltou-se a notar mudanças a partir dessa etapa. Agora, os diagramas da abordagem das personas ficaram bem mais completos do que os da outra abordagem. Isso ocorreu porque como os outros atores tiveram novos objetivos a serem cumpridos, a interação deles com os sistemas precisou ser mais especificada graças aos contextos que foram definidos. Para garantir que eles aconteçam novas responsabilidades que não aparecem em forma de objetivos no sistema apareceram como especificações de funcionamento do sistema.

O que antes era só um troca de mensagem simples entre dois atores, virou um diagrama de sequência bem mais completo e complexo. Em quase todos os diagramas de interação entre os atores aconteceram essas mudanças.

Além disso, no próprio comportamento que era esperado de cada ator existiram mudanças significativas, não ao funcionamento em si do sistema, mas do que é esperado dele,

como notamos no que seria o ciclo de vida de um usuário. Por mais que isso não mude a implementação do sistema em si, isso serve de insumo para os desenvolvedores ficarem mais atentos ao desenvolverem certas funcionalidades do sistema.

4.1.5 Implementação

A etapa de implementação foi bem diferente. Primeiro havia objetivos diferentes, o processo orientado em objetivos tinha foco num ambiente que envolvia os atores que usariam o sistema como agentes e no processo orientado a personas o foco era só o sistema que proveria uma interação web para usuários reais. Porém, a implementação do sistema foi bem parecida tirando detalhes para permitir a interação web.

4.2 Discussão

Percebe-se que o uso de personas trouxe várias diferenças no final dos processos. O objetivo do segundo processo era tentar fazer um sistema que provesse um auxilio melhor para o usuário. E o que se nota foi que isso foi o resultado. Mesmo não tendo grandes alterações no sistema por causa de um número limitado de personas definidas, [23] já discutiu que elas trazem sim mudanças significativas a modelagem dos objetivos e requisitos do sistema. Neste trabalho apresentamos que, mesmo com exemplos simples, a utilização de personas nos trouxe também benefícios para o desenvolvimento de SMAs.

4.3 GQM

Após desenvolver os exemplos ilustrativos pode-se notar as diferenças entre os dois processos. Para tentar exemplificar essas diferenças e demonstrar se houve ou não alguma melhoria no sistema final propõe-se o seguinte GQM.

4.3.1 Objetivos

- G1: Mostrar que personas ajudam a melhorar a qualidade do entendimento do usuário em relação ao que o SMA deve fazer para melhor atendê-lo no processo de desenvolvimento de um SMA.

4.3.2 Perguntas

- Q1.1: O processos foi mais fácil de se fazer?
- Q1.2: O modelo ficou mais completo?

- Q1.3: O processo atendeu mais as necessidades de implementação de um SMA?

4.3.3 Métricas e Indicadores

- M1.1.1: Tempo levado para concluir os processos.
- M1.1.2: Quantidade de artefatos a serem feitos.
- M1.2.1: Quantidade de Objetivos.
- M1.2.2: Quantidade de Tarefas.
- I1.3.1: Mais especificação dos agentes.
- I1.3.2: Mais artefatos de insumo de testes.

O foco deste trabalho é o desenvolvimento do SMA, por isso o GQM é mais voltado para questões mais técnicas de desenvolvimento. Porém ainda é possível fazer uma análise de se houve ou não uma aceitação melhor dos usuários ao sistema utilizando a abordagem de personas. Para isso podemos definir outro GQM que pode ser estudado em trabalhos futuros com o foco em se o Usuário se sentiu mais contemplado com o sistema, se seus objetivos estão sendo alcançados de forma mais eficiente, se é perceptível algum tipo de diferenciação na experiência entre o uso de diferentes usuários, etc.

Diferente de como o GQM foi abordado primeiro serão analisadas as métricas, as respostas às perguntas, para então verificar se o objetivo foi alcançado ou não.

M1.1.1: Tempo levado para concluir os processos

Foi utilizado uma ferramenta web chamada Toggl para a contagem do tempo que foi gasto para cada etapa durante o processo. Seu funcionamento é simples, você pode definir projetos nos quais você está trabalhando e depois pode acionar um contador de tempo enquanto você trabalha para contabilizar as horas trabalhadas em cada projeto.

Tanto para o primeiro quanto para o segundo, gastou-se um media 30h ao todo desde a modelagem até a implementação. Como os primeiros modelos do primeiro exemplo eram mais completos, gastou-se mais tempo para fazê-los. Já no segundo exemplo gastou-se mais tempo para fazer os modelos finais de Design Detalhado. O tempo de implementação dos dois foi parecido. Porém precisamos enfatizar que somente uma Persona foi modelada em nosso exemplo.

M1.1.2: Quantidade de artefatos feitos

Como simplificação, foi considerado que todos os diagramas feitos na etapa de Design detalhado são um único artefato. Além disso, foi considerado também que todos os

cartões personas que foram criados, e todos os seus respectivos modelos de objetivos são dois artefatos. Assim no primeiro exemplo tivemos um artefato no final de cada etapa, tendo 5 artefatos no final. Já no segundo exemplo tivemos 9 artefatos. A diferença se encontra na tabela de relação entre *stakeholders* e atores e nos cartões e modelos de objetivos de cada Persona.

M1.2.1: Quantidade de objetivos

O primeiro exemplo teve no modelo final 13 objetivos (Figura 3.2) e o segundo 19 objetivos (Figura 3.17).

M1.2.2: Quantidade de Tarefas

O primeiro exemplo teve no modelo final 18 tarefas (Figura 3.2) e o segundo 9 tarefas (Figura 3.17).

I1.3.1: Mais especificação dos agentes

Conforme notamos nos diagramas UML (Figuras 3.5 e 3.20), percebe-se que o comportamento dos agentes e consequentemente do sistema está bem mais detalhado.

I1.3.2: Mais artefatos de insumo de testes

O modelo possui mais objetivos (Figura 3.17) e contextos (Figuras 3.13, 3.14 e 3.15) atrelados a eles que servem como insumo para verificar se o sistema atenderá ao usuário que não existiam antes.

Tendo analisado as métricas e os indicadores pode-se então analisar as perguntas feitas.

Q1.1: O processo foi mais fácil de se fazer?

Não. Os processos tiveram um nível de dificuldade parecidos.

Q1.2: O modelo ficou mais completo?

Sim. Por mais que a modelagem dos objetivos finais tenha ficado equilibrada, considerando os contextos e os diagramas UML o modelo utilizando personas ficou mais completo no quesito de prover mais informações sobre a interação do usuário com o sistema.

Q1.3: O processo atendeu mais as necessidades de implementação de um SMA?

Sim. Como já discutido, existiu muito mais insumos sobre como o sistema deveria funcionar para se implementar o SMA para garantir seu bom funcionamento.

Após responder as perguntas tem-se que verificar se o objetivo foi ou não alcançado.

G1: Mostrar que personas auxiliam no processo de desenvolvimento de um SMA

Pode-se dizer que as personas auxiliam sim no processo pois conseguiu-se um modelo mais completo e a abordagem orientada a personas permitiu uma maior quantidade de informações sobre o sistema para a implementação do SMA.

Capítulo 5

Conclusão

Neste capítulo vamos então relacionar nossa análise feita no Capítulo 4 com os objetivos definidos no Capítulo 1. Depois serão feitas considerações finais e serão descritos os trabalhos futuros.

Tinham-se como objetivos secundários os de desenvolver dois protótipos, um utilizando a abordagem orientada a objetivos e a outra a abordagem orientada a personas. O desenvolvimento desses protótipos foi descrito no Capítulo 3, logo os objetivos foram alcançados.

O objetivo principal era investigar se o uso de um método de levantamento de requisitos orientado a personas traz indícios de que existem vantagens ao processo de desenvolvimento de sistemas orientados a agentes. Conforme foi analisado por meio do GQM proposto no final do Capítulo 4, chegou-se a conclusão de que existiram vantagens em utilizar a abordagem de personas. Portanto o objetivo principal dessa monografia foi alcançado.

5.1 Considerações

Podemos observar que a utilização de personas permitiu que as modelagens e as preocupações ao se desenvolver o sistema fossem mais focadas nas necessidades do usuário. Além disso, no final do processo, conseguimos mais detalhamento do sistema sem precisar de grandes esforços pensando em possíveis cenários, só de criar uma Persona com necessidades já nos vimos em um situação de ter que definir novos objetivos e cenários foram definidos conforme ia-se criando as personas.

Porém, não cabe a esse trabalho dizer qual abordagem é melhor. A discussão entre as possíveis abordagens de se desenvolver um sistema é delicada e complicada. De maneira geral chega-se a conclusão de que as abordagens são melhores ou piores dependendo do seu objetivo, dos recursos e tempo disponível para gastar em cada etapa de seu processo.

Pode-se afirmar que utilizar a abordagem de personas permite uma visão mais direcionada ao usuário e sua experiência provendo insumos para tal.

5.2 Trabalhos Futuros

Por último serão comentados possíveis trabalhos futuros e relacionados. Procurou-se nesse trabalho iniciar um estudo sobre a relação entre personas e SMA. Para isso, queria-se primeiro demonstrar que isso traria algum benefício de fato a área para gastar mais tempo e aprofundar o que pode ser feito utilizando essa abordagem. Como notou-se que de fato existe uma vantagem de se usar a abordagem, pretende-se em trabalhos futuros definir ainda mais como elas podem ser utilizadas, criando uma metodologia mais clara nas etapas de Design Arquitetural, Design Detalhado e Implementação por exemplo. Com uma metodologia mais definida e com a relação entre personas e agentes mais clara, pode-se realizar outros estudos e desenvolvimentos para mostrar exatamente o que se pode alcançar utilizando-as.

Porém, o uso de personas não se limita somente ao desenvolvimento de SMAs. Durante o estudo de ER, necessário para o estudo realizado nessa monografia, foi feito outro tipo de estudo sobre como personas podem ser utilizadas.

Tinha-se como uma das hipóteses que um levantamento de requisitos mal conduzido é uma das maiores causas de problemas nos sistemas para a ES. Esse fato levou os engenheiros de software a se preocuparem também com o grau de confiabilidade que se pode atribuir aos softwares, a dependabilidade do sistema, isso é, o tanto que se poderia depender do sistema para que ele funcione corretamente. É descrito em [11] o GODA, um *framework* capaz de verificar a dependabilidade de um sistema que possua um contexto dinâmico. Ele foi pensado para garantir que o sistema terá um bom funcionamento e que seus requisitos estão sendo cumpridos conforme o esperado. Uma vez que ele utilize modelagem orientada a objetivos com contextos por meio do Tropos para fazer parte dessa análise, percebe-se tanto pelo trabalho nessa monografia quanto no realizado em [23] que pode existir a possibilidade de utilizar personas como abordagem para realizar as modelagens utilizadas por ele. Pode-se também estudar a possibilidade de se otimizar a ferramenta para que ela possua suporte para lidar com essa nova abordagem.

Referências

[1] Systems and software engineering – vocabulary. *ISO/IEC/IEEE 24765:2010(E)*, pages 1–418, Dec 2010. 5

[2] Taom4e - tool for agent oriented modeling, June 2016. 14

[3] Raian Ali, Fabiano Dalpiaz, e Paolo Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requir. Eng.*, 15(4):439–458, 2010. 18, 39

[4] Victor R. Basili, Gianluigi Caldiera, e H. Dieter Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994. 4

[5] Fabio Luigi Bellifemine, Giovanni Caire, e Dominic Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007. 11

[6] Grady Booch, James Rumbaugh, e Ivar Jacobson. *Unified Modeling Language User Guide, The (2Nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005. 14

[7] Michael Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987. 9

[8] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, e John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004. 14

[9] Virginia Dignum e Frank Dignum. Modelling agent societies: Co-ordination frameworks and institutions. In *Proceedings of the10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving*, EPIA '01, pages 191–204, London, UK, UK, 2001. Springer-Verlag. 11

[10] Lin Liu e Eric Yu. Designing information systems in social context: A goal and scenario modelling approach. *Inf. Syst.*, 29(2):187–203, April 2004. 14

[11] Danilo Filgueira Mendonça, Genaína Nunes Rodrigues, Raian Ali, Vander Alves, e Luciano Baresi. Goda: A goal-oriented requirements engineering framework for runtime dependability analysis. *Information and Software Technology*, 80:245 – 264, 2016. 55

[12] John Mylopoulos, Lawrence Chung, e Eric Yu. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1):31–37, January 1999. 14

[13] V. T. Nunes. *Dynamic Process Adaptation: Planning in a Context-Aware Approach.* Tese de doutorado do programa de pós-graduação em engenharia de sistemas e computação, COPPE, Universidade Federal do Rio de Janeiro, Agosto 2014. 1

[14] Stefan Poslad. Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4), November 2007. 13

[15] Roger Pressman. *Software Engineering: A Practitioner's Approach.* McGraw-Hill, Inc., New York, NY, USA, 7 edition, 2010. 5, 6

[16] Anand S. Rao e Michael P. Georgeff. BDI agents: From theory to practice. In *IN PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON MULTI-AGENT SYSTEMS (ICMAS-95*, pages 312–319, 1995. 9

[17] Stuart J. Russell e Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition, 2003. 10, 13

[18] Dominik Ryżko e Weronika Radziszewska. *Integration between Web Services and Multi-Agent Systems with Applications for Multi-commodity Markets*, pages 65–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 46

[19] Onn Shehory e Arnon Sturm. *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks.* Springer Publishing Company, Incorporated, 2014. 3, 10

[20] Ian Sommerville. *Engenharia de Software.* Pearson, 2007. 1, 6, 10

[21] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Wiley Publishing, 1st edition, 2009. 6, 7, 8, 9, 13

[22] Karel Vredenberg, Scott Isensee, e Carol Righi. *User-Centered Design: An Integrated Approach with Cdrom.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. 2, 16

[23] Nayara. Watanabe. Uma Proposta de Modelagem Orientada a Personas para o Modelo de Objetivo Orientado a Contexto, 2005. Monografia (Bacharel em Engenharia da Computação), UnB (Universidade da Brasília), Brasília, Brasil. x, 2, 17, 18, 22, 38, 44, 49, 50, 55

[24] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, e Anders Wesslén. *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000. 19

[25] Michael Wooldridge. *An Introduction to MultiAgent Systems.* Wiley Publishing, 2nd edition, 2009. 10, 13

[26] Eric S. Yu. Conceptual modeling: Foundations and applications. chapter Social Modeling and I*, pages 99–121. Springer-Verlag, Berlin, Heidelberg, 2009. 14