UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ClinicSpace: MODELAGEM DE UMA FERRAMENTA-PILOTO PARA DEFINIÇÃO DE TAREFAS CLÍNICAS EM UM AMBIENTE DE COMPUTAÇÃO PERVASIVA BASEADO EM TAREFAS E DIRECIONADO AO USUÁRIO-FINAL

DISSERTAÇÃO DE MESTRADO

Fábio Lorenzi da Silva

Santa Maria, RS, Brasil

2009

ClinicSpace: MODELAGEM DE UMA FERRAMENTA-PILOTO PARA DEFINIÇÃO DE TAREFAS CLÍNICAS EM UM AMBIENTE DE COMPUTAÇÃO PERVASIVA BASEADO EM TAREFAS E DIRECIONADO AO USUÁRIO-FINAL

por

Fábio Lorenzi da Silva

Dissertação apresentada ao Curso de Mestrado em Computação do Programa de Pós-Graduação em Informática (PPGI), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação**

Orientador: Profa. Dra. Iara Augustin

Santa Maria, RS, Brasil

2009

Universidade Federal de Santa Maria

Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

ClinicSpace: MODELAGEM DE UMA FERRAMENTA-PILOTO PARA DEFINIÇÃO DE TAREFAS CLÍNICAS EM UM AMBIENTE DE COMPUTAÇÃO PERVASIVA BASEADO EM TAREFAS E DIRECIONADO AO USUÁRIO-FINAL

elaborada por
Fábio Lorenzi da Silva

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Iara Augustin, Dra.
(Presidente/Orientadora)

Daniela Bitencourt Rosa Leal, Dra. (UFSM)

Deise de Brum Saccol, Dra. (UNIPAMPA)

Santa Maria, 03 de Julho de 2009.

AGRADECIMENTOS

É com imensa alegria e satisfação que após finalizada a dissertação escrevo esta parte, para assim agradecer as pessoas que, direta ou indiretamente, participaram e ajudaram durante a realização do presente trabalho, e durante toda a caminhada trilhada até o seu término.

Em especial, agradeço imensamente a minha mãe, Fátima, por não medir esforços para eu chegar até aqui, e ainda pelo apoio incondicional e pela palavra amiga, sincera e de estímulo nos momentos que parecia não ter força e capacidade para continuar. Sei que muitas vezes deixou de lado seus planos e objetivos para propiciar e atender aos meus, e sou eternamente grato por isso. Além dela agradeço ao meu pai, Alberi, por ter me ensinado muitas coisas que guardarei para o resto da minha vida, e também por sempre ter me incentivado a estudar e a ser um Homem com caráter e dignidade. Mesmo não estando presente fisicamente comigo sei que está muito próximo, e que me acompanha constantemente na minha caminhada. A saudade é gigantesca, mas diminuída pela certeza que um dia nos reencontraremos. Espero poder educar e acompanhar meus filhos assim como vocês fizeram a mim, pois sem isso tudo não teria chegado até aqui.

A minha namorada, amiga e companheira, Daniane, pela compreensão em todos os momentos, principalmente naqueles não tão bons. Agradeço ainda pelo companheirismo, conselhos e pela força que sempre me deu para que eu pudesse transpor os obstáculos que foram surgindo. Certamente a sua ajuda foi muito importante para que tudo isso esteja acontecendo.

Agradeço ainda ao meu irmão, Fabrício, pela amizade, carinho, afeto e por ter me presenteado, junto com a Catiane, com o sobrinho e afilhado mais lindo e especial que eu podia ter, o Dérik. Este, faz com que eu o ame cada vez mais a cada momento que passamos juntos. Ainda, agradeço ao Marino pela amizade e companheirismo. Sempre disposto a me ajudar, e até mesmo se sobrecarregando nas atividades suprindo as minhas ausências que muitas vezes se deram por falta de tempo.

Em especial, agradeço a minha orientadora, Iara Augustin, pelo apoio e eterna disposição em auxiliar e a me orientar desde o primeiro momento em que começamos a trabalhar juntos. Sei que juntos conseguimos desenvolver um excelente trabalho, e muito graças ao seu comprometimento e disponibilidade em me auxiliar e orientar. Certamente você foi muito importante para a conclusão de mais essa etapa, e torço para que futuros trabalhos

sejam realizados por nós.

Aos amigos, pela amizade, companheirismo, e certeza que sempre posso contar a ajuda de vocês. Ainda, aos colegas e amigos do gMob, em especial ao Giuliano Lopes Ferreira e Tiago Antonio Rizzetti, que juntos participaram início do projeto ao qual essa dissertação está inserida. Destes ficam as lembranças de muitos acontecimentos, ajuda, e a certeza que nos encontraremos nas próximas etapas de nossas vidas. Desejo muito sucesso a todos.

Agradeço também a meus avós, tios, primos e familiares por todo o carinho e estímulo dispensado a mim nessa caminhada. Por último e não menos importante, agradeço a Deus por tudo que tem me propiciado na vida. Sou grato pelas experiências vividas, pelo amparo e pelos familiares e amigos que me rodeiam e que participam da minha vida.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

ClinicSpace: MODELAGEM DE UMA FERRAMENTA-PILOTO PARA DEFINIÇÃO DE TAREFAS CLÍNICAS EM UM AMBIENTE DE COMPUTAÇÃO BASEADA EM TAREFAS E DIRECIONADA AO USUÁRIO FINAL

Autor: Fábio Lorenzi da Silva
Orientadora: Iara Augustin
Data e Local da defesa: Santa Maria, 03 de Julho de 2009.

A Computação Ubíqua prevê o suporte às atividades humanas da forma mais integrada possível ao ambiente conhecido pelo profissional. Nessa perspectiva, uma das grandes áreas de aplicação é a Saúde, já que o Sistema de Saúde do futuro prevê o uso da Computação Ubíqua como forma de otimizar e automatizar as atividades clínicas. Focando o problema de rejeição dos sistemas computacionais na Saúde devido ao distanciamento destes da forma como os clínicos executam suas tarefas, o projeto ClinicSpace propõe a prototipação de uma ferramenta que permita aos clínicos a personalização e gerenciamento de suas tarefas diárias. Um dos grandes desafios enfrentados é como modelar atividades humanas em sistemas computacionais, respeitando a forma individualizada com que cada pessoa as realiza. Nesse sentido, a contribuição desse trabalho é a proposta de uma ferramenta-piloto que permite ao profissional programar e compor suas tarefas, a partir de uma modelagem das principais tarefas clínicas executadas nos ambientes hospitalares por profissionais clínicos, respeitando a forma individual que cada profissional as realiza. Uma vez criadas as tarefas, os profissionais clínicos podem reutilizá-las para a definição e reuso na criação de outras tarefas que julgarem necessário. Uma arquitetura gerencia a execução das tarefas da forma mais automática possível para não comprometer o controle que o profissional deve manter sobre o ambiente. Assim, espera-se reduzir o grau de rejeição encontrado nos sistemas informatizados dos hospitais e clínicas. A interface de programação de tarefas pelo profissional desenvolvida utiliza mecanismos providos pela Programação Orientada ao Usuário-Final com o objetivo de facilitar a utilização do sistema pelo profissional clínico e pela Computação Orientada a Atividades. Estudos de casos foram simulados para demonstrar a viabilidade da proposta. Testes de campo somente poderão ser realizados após a disponibilização do protótipo da arquitetura de gerenciamento e execução das tarefas, porém, esse trabalho está fora do escopo dessa dissertação.

Palavras-chave: computação ubíqua; computação orientada a atividades; tarefas clínicas; personalização de tarefas; programação orientada ao usuário-final.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Informatics
Federal University of Santa Maria

ClinicSpace: MODELING OF A PROTOTYPE TOOL TO DEFINITION OF CLINICAL TASKS IN A TASK-DRIVEN AND USER-CENTER COMPUTING ENVIRONMENT

Author: Fábio Lorenzi da Silva
Advisor: Iara Augustin
Santa Maria, July 03, 2009

Ubiquitous computing foresees the support to human activities in the most possible integrated environment known by the professional. On this perspective, a major area of its application is the Health System as the health of the future provides the use of ubiquitous computing as a way to automate and optimize the clinical activities. Addressing the problem of rejection of computer systems in health due to the remoteness of the way clinicians perform their tasks, the project "ClinicSpace" proposes a prototype of a tool that enables clinicians to customize and better manage their daily tasks. One of the big challenges is how to model human activities in computer systems, respecting the way that each individual performs them. This way, the contribution of this work is to propose a modeling of the main tasks performed in the clinical hospital settings by clinical professionals with the way that each individual performs the work. Once created the task, the medical professionals can reuse them for the definition and creation of other tasks they may judge necessary. An architecture manages the tasks in the most automatic way possible to undermine the control that the owner must maintain to the environment. Thus, it is expected to reduce the degree of rejection found in computerized systems of hospitals and clinics. The interface developed uses mechanisms provided by the End-user programming to facilitate the use of the clinical professional and Task-driven Computing. Case studies were simulated to demonstrate the feasibility of the proposal. Field tests may only be made after the release of the prototype of the architecture of the management and execution of tasks; however, this work is outside the scope of this dissertation.

Key-words: ubiquitous computing; Task-driven Computing; clinical activities; perform of tasks; End-user programming.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

EHS	*Electronic Health-care System*
pEHS	*pervasive EHS*
EXEHDA	*Execution Environment for Highly Distributed Applications*
ISAM	Infra-estrutura de Suporte às aplicações Móveis Distribuídas
ISAMpe	ISAM *Pervasive Environment* - Ambiente Pervasivo do ISAM
SAT	Serviço de Acesso a Tarefas
STA	Serviço de Tarefas Ativas
SCT	Serviço de Contexto de Tarefas
SGDT	Subsistema de Gerenciamento Distribuído de Tarefas
SGT	Serviço de Gerenciamento de Tarefas

LISTA DE APÊNDICES

SUMÁRIO

1 INTRODUÇÃO

Computação Pervasiva é uma nova área de pesquisa que está sendo considerada a terceira onda da computação (JANSEN et al, 2005). Originou-se da proposta de Mark Weiser, chamada Computação Ubíqua (*Ubiquitous Computing*) (WEISER, 1991). Em um espaço pervasivo, computadores e outros dispositivos digitais dos mais variados tipos estão totalmente integrados ao ambiente do usuário e objetivam auxiliá-lo em suas tarefas diárias. Este é um ambiente altamente dinâmico e heterogêneo. Os recursos, incluindo serviços, dispositivos e aplicações disponíveis podem alterar-se rapidamente. Diferentes espaços têm diferentes tipos de recursos disponíveis e diferentes políticas de uso dos recursos. Programas executando neste ambiente devem ser capazes de se adaptar à troca de contexto e à disponibilidade de recursos. Isto coloca um desafio para os desenvolvedores que devem especificar como o programa deve se comportar em diferentes contextos e quando diferentes tipos de recursos estão disponíveis. Além disso, diferentes espaços pervasivos podem ter diferentes modos de executar a mesma tarefa uma vez que eles têm variados serviços, aplicações e recursos. O desenvolvedor não pode esperar saber antecipadamente como as várias tarefas serão executadas nos diferentes espaços pervasivos. Assim, programadores necessitam de abstrações de alto nível para programar aplicações no espaço pervasivo sem ter que ter consciência dos recursos disponíveis, contexto, políticas e preferência dos usuários (AUGUSTIN, 2004).

Ambientes para Computação Pervasiva têm sido explorados através de uma série de protótipos de pesquisa na academia (ROMAN et al, 2002) (GARLAN; STEENKISTE; SCHMERL, 2002) (AUGUSTIN, 2004a) e indústria (MIT Oxigen project www.oxygen.lcs.mit.edu; IBM Pervasive Computing Unit www.research.ibm.com/thinkresearch/pervasive.shtml). Muitos enfatizam os requisitos de tecnologia e a previsibilidade da interação e comportamento do ambiente. Pela integração de sensores, computadores, dispositivos e redes foi possível desenhar a primeira geração de ambientes pervasivos, referenciados como “ambientes integrados”.

Agora os esforços concentram-se em deslocar o paradigma de “ambientes integrados” para “espaços programáveis”. Dessa forma, a Computação Pervasiva requer novas soluções,

uma vez que esta implica um novo modo de pensar sobre programas.

A área de Cuidados Clínicos *(Healthcare)* (CPH, 2007) é uma das principais áreas de atuação da Computação Pervasiva e muitas pesquisas estão sendo desenvolvidas tanto pela indústria quanto pela academia. Pesquisas nesta área objetivam o desenvolvimento de sistemas para ambientes hospitalares e também a extensão dos cuidados médicos aos pacientes em seus próprios lares (KOMNINOS; STAMOU, 2006).

Um dos grandes desafios que se apresenta é como modelar atividades humanas em sistemas computacionais, respeitando a forma individualizada com que cada pessoa as realiza. Nesse escopo, o projeto "ClinicSpace: auxílio às tarefas clínicas em um ambiente hospitalar do futuro baseado em tecnologias da Computação Ubíqua/Pervasiva", em desenvolvimento na Universidade Federal de Santa Maria (UFSM), grupo GMob (www.inf.ufsm.br/gmob), propõe o desenvolvimento de uma ferramenta-piloto que permita aos clínicos a programação personalizada de suas tarefas, as quais são gerenciadas e executadas em um ambiente pervasivo.

Argumenta-se que, para diminuir o impacto de interferência do sistema automatizado na atividade clínica, e a rejeição que este pode causar, o sistema deve equilibrar a pró-atividade (agir em nome do usuário) com a personalização (forma individual de cada um realizar sua atividade). Como inovação, a arquitetura de software projetada para tratar da modelagem e gerenciamento de tarefas clínicas integra conceitos oriundos das áreas: programação orientada ao usuário-final, computação orientada a atividades, ontologia e *middlewares* para gerenciamento do ambiente pervasivo/ubíquo. No caso dos ambientes hospitalares, usuários clínicos realizam muitas atividades clínicas no cuidado e tratamento de pacientes. Sistemas computacionais pervasivos direcionados a ambientes hospitalares devem auxiliar os usuários clínicos diariamente, facilitando tanto a execução das atividades de cuidados médicos como o auxílio em possíveis decisões que esses profissionais devam tomar em relação aos pacientes.

Esse trabalho iniciou a definição da arquitetura ClinicSpace e relata-se, neste texto, as principais questões relativas à modelagem das tarefas clínicas executadas nos ambientes hospitalares por profissionais clínicos e insere uma forma de individualizá-las, com o objetivo de projetar uma ferramenta-piloto que seja utilizada por esses profissionais na execução de suas atividades diárias. A modelagem resulta em tarefas que representam diferentes níveis de abstrações que encapsulam serviços e aplicações computacionais, as quais poderão ser utilizadas pelos profissionais clínicos para a definição das tarefas utilizadas no cuidado e tratamento dos pacientes. Uma vez definidas as tarefas, os usuários clínicos devem ser

capazes de reutilizá-las para a criação e definição de outras que julgar necessário. Assim, tanto as tarefas modeladas inicialmente e disponibilizadas no sistema quanto as outras tarefas definidas pelo usuário clínico devem estar disponíveis para serem reaproveitadas.

A ferramenta-piloto (interface) desenvolvida nesse trabalho permite que usuários utilizem as tarefas modeladas inicialmente e ainda programem outras que forem necessárias bastando: (i) reutilizar as tarefas já modeladas e comuns à tarefa desejada; (ii) associar tarefas para a definição de fluxos de execução. Para facilitar a interação do profissional clínico com o sistema, são utilizados mecanismos oriundos da computação orientada ao usuário final (derivada do termo *EUP – End- User Programming*) no desenvolvimento da ferramenta. Essa técnica provê abstrações de alto-nível para que usuários não especializados e não conhecedores de técnicas de programação possam criar, configurar e alterar sistemas computacionais inserindo seus conhecimentos, idéias e preferências (HAGUE; ROBINSON, 2006). Assim, usuários clínicos poderão personalizar as tarefas clínicas adequando-as a sua forma de fazer através da combinação, configuração, controle, definição e contextualização de suas tarefas de uma forma intuitiva e sem a necessidade do conhecimento de técnicas de programação.

O restante do texto divide-se da seguinte forma: no capítulo 2, abordam-se os aspectos relativos à Computação Ubíqua/Pervasiva com o foco no auxílio às atividades diárias dos usuários e uma análise dos trabalhos relacionados; no capítulo 3, apresentam-se a motivação, os conceitos adotados e a arquitetura proposta para o projeto ClinicSpace no qual o presente trabalho está inserido; no capítulo 4, explica-se a arquitetura de software criada para o suporte à criação e personalização de tarefas a serem realizadas pelos profissionais clínicos; no capítulo 5, apresentam-se uma discussão comparativa com trabalhos relacionados e os estudos de caso prototipados, com o intuito de avaliar a modelagem de tarefas e a Interface de Edição de Tarefas que foram desenvolvidas no presente trabalho. Por fim, são apresentadas as conclusões do presente trabalho bem como as referências bibliográficas utilizadas.

2 COMPUTAÇÃO PERVASIVA ORIENTADA AO USUÁRIO-FINAL E SUAS ATIVIDADES COTIDIANAS

Este capítulo aborda os aspectos relativos à Computação Ubíqua ou Pervasiva orientada ao usuário-final e nas suas atividades diárias de trabalho.

2.1 Conceitos da Computação Ubíqua

O constante avanço técnico em comunicação e computação torna o cenário visualizado pela Computação Pervasiva, onde a computação está totalmente inserida nas atividades das pessoas, uma realidade a ser alcançada em alguns anos. Além disso, previsões indicam que microprocessadores se tornarão pequenos e baratos o suficiente para serem embutidos tanto em dispositivos digitais, carros, eletroeletrônicos, brinquedos, ferramentas como também em objetos e roupas.

Este cenário está sendo considerado o novo paradigma do século 21 (SAHA; MUKHERJEE, 2003) (SATYANARAYANAN, 2001) ou a terceira onda da computação (JANSEN et al, 2005), pois permite o acoplamento do mundo físico ao mundo da informação (virtual) e fornece uma abundância de serviços e aplicações onipresentes visando que usuários, máquinas, dados, aplicações e diversos objetos do espaço físico interajam uns com os outros de forma transparente (RANGANATHAN et al, 2005a). Dessa forma, está-se movendo gradualmente em direção à visão de uma computação onipresente, ou ubíqua, e está-se incrementalmente acostumando-se a usar uma coleção de heterogêneos dispositivos, para suportar uma crescente faixa de atividades. A corrente geração de dispositivos interconectados é somente o ponto de partida em direção à computação ubíqua (CHALMERS, 2006).

Como o cenário pervasivo prevê uma mobilidade física (dos equipamentos e/ou dos usuários) e lógica (componentes da aplicação e serviços), potencialmente em escala global, este deve fornecer transparência ao usuário, de forma que o usuário possa acessar seu

ambiente computacional independente de localização, do meio de acesso e do tempo. O sistema de suporte para esse ambiente usa a metáfora de um ambiente virtual do usuário, onde as aplicações têm o estilo "siga-me" (follow-me applications) (AUGUSTIN, 2005).

Dois grandes exemplos de ambientes pervasivos que visam à automação das tarefas diárias são as Casas Inteligentes (RANGANATHAN; CAMPBELL, 2005) e os hospitais (BARDRAM; CHRISTENSEN; OLSEN, 2004). Considerando o caso hospitalar , algumas atividades clínicas são previsíveis e planejadas enquanto outras são aleatórias. As atividades variam de simples a complexas, algumas tem prioridade enquanto outras podem ser feitas quando houver tempo e, ainda, algumas atividades são ligadas a determinadas salas e presença de certos artefatos. O trabalho dos clínicos é extremamente móvel e estes não podem carregar equipamentos pesados. Logo, é interessante no ambiente o conceito de "computador público" que não armazena atividades computacionais, mas serve como um portal de acesso a elas. Este conceito requer uma infra-estrutura que gerencia, armazena e distribui atividades computacionais. Outra propriedade necessária é a inferência pró-ativa das atividades baseada na localização da pessoa e artefatos ao redor. O trabalho dos clínicos é também altamente colaborativo, pois o atendimento a um paciente, por exemplo, envolve várias especialidades. Colaboração representa interromper a tarefa em execução para atender a solicitação por demanda (chaveamento de atividades). O ambiente ainda requer acesso a diversificadas e atualizadas informações, que podem ser requisitadas por vários usuários clínicos no mesmo instante. Dessa forma, uma eficaz organização de dados e acesso pervasivo a ele é requerida. Dispositivos móveis devem se comunicar com a infra-estrutura disponível, numa organização de rede infra-estrutura, ou descobrir novos dispositivos, numa organização de redes *ad-hoc* ou *mesh* (AUGUSTIN; LIMA; YAMIN, 2006).

2.1.1 Primeira Geração: Foco na Infra-estrutura

Vários sistemas e protótipos de computação pervasiva têm sido desenvolvidos por pesquisadores a fim de demonstrar como este novo paradigma pode beneficiar domínios de aplicações específicos, como saúde e emergências (*smart hospital*), casa virtual (*smart home*), educação (*pervasive learning*), entretenimento e segurança de residências (*home security*) (AUGUSTIN; LIMA; YAMIN, 2006). As estratégias adotadas dividem-se em (i) projeto de aplicações experimentais, (ii) desenvolvimento de sistemas genéricos experimentais, (iii)

análise teórica, como o modelo *Mobile Ambients* (CARDELLI; GORDON, 1998) e a formalização do conceito de contexto (JANSEN et al, 2005).

Normalmente, os sistemas adotam a integração de sistemas povoados com diversos dispositivos computacionais com capacidades heterogêneas (*appliances*), tais como sensores, atuadores, computadores, micro-controladores, usando diversas redes e conectores. Esses sistemas objetivam demonstrar a habilidade de convergência das tecnologias emergentes e de compreender melhor os novos requisitos das aplicações. Observa-se que as questões de pesquisas em progresso concentraram-se em redes móveis, redes de sensores e *middlewares* (AUGUSTIN; LIMA; YAMIN, 2006).

As redes para ambientes pervasivos podem ser organizadas com duas estratégias:

- Infra-estruturada. Nela os pontos móveis conectam-se à infra-estrutura estática disponível, refletindo o modelo cliente-servidor de Sistemas Distribuídos;
- Redes *ad-hoc* ou *mesh*. Nessa abordagem as redes são altamente dinâmicas, refletindo o modelo *peer-to-peer* (P2P) de Sistemas Distribuídos.

Redes de sensores permitem monitorar o ambiente através da obtenção de informações sobre o estado corrente do ambiente (HAC, 2003). Para a disponibilização do estilo de aplicações 'siga-me' é necessário uma infra-estrutura de suporte ao seu projeto, implementação e execução. Nesse sentido, o projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis e Distribuídas – www.inf.ufrgs.br/~isam) (AUGUSTIN et al, 2004a) (YAMIN, 2004) defende que uma infra-estrutura de suporte à pervasividade em escala global pode ser construída através da integração de três áreas da computação que são a Computação Móvel, Computação em Grade e a Computação Consciente de Contexto (AUGUSTIN, 2004) (YAMIN, 2004).

2.1.2 Segunda Geração: Foco nas Atividades Humanas Cotidianas

A computação Ubíqua/Pervasiva objetiva auxiliar os usuários nas suas tarefas diárias (atividades humanas). Para tratar essa questão, são necessários um novo modelo baseado em tarefas e um modelo navegacional para estruturar programas. Tarefas podem ser definidas como um conjunto de ações executadas colaborativamente por humanos e pelo sistema pervasivo para alcançar um objetivo. Uma tarefa pode ser composta, estática ou dinamicamente, por um número de outras tarefas (subtarefas), que podem ser unidas usando

um conceito similar a *workflow*. Tarefas diferentes podem ter subtarefas comuns ou similares – desta forma, o reuso de tarefas já programadas é essencial. O desacoplamento temporal e espacial (códigos e dados) deve ser modelado (AUGUSTIN; LIMA; YAMIN, 2006). Por ter uma ligação direta com esse trabalho, o paradigma de orientação a tarefas é detalhado no item 2.3.

Os sistemas computacionais atuais suportam apenas o nível de aplicação (processos) e não o nível de atividades humanas, tal como os seres humanos as concebem. O desafio é definir um modelo computacional para tratar a atividade cotidiana, sendo este um equivalente digital mais próximo possível à atividade humana atualmente realizada e de acordo com a forma individual de cada um realizá-la. Um ponto de partida é iniciar o estudo a partir das atividades humanas de trabalho.

Para definir este modelo de atividades, considera-se de grande importância dar ao usuário o controle sobre a definição da atividade. A forma adotada de fazer isso é usar os conceitos da programação orientada ao usuário-final.

2.2 Programação Orientada ao Usuário-final

Mesmo com os grandes avanços atingidos pela computação, muitas são as barreiras e obstáculos enfrentados por usuários leigos na utilização de sistemas computacionais. Outro agravante dessa situação é que a programação dessas aplicações é função exclusiva de profissionais especializados em Informática. Dessa forma, usuários, muitas vezes, utilizam aplicações que não atendem às suas necessidades, não sendo adequadas à sua realidade. Para contornar esse cenário foi proposta a Programação orientada ao usuário-final (derivado do termo *End User Programming* - EUP) que visa criar abstrações para permitir que usuários não especializados possam criar, modificar, estender e adaptar as aplicações de acordo com as suas necessidades (HAGUE; ROBINSON, 2006).

A computação orientada ao usuário-final, que é um tópico da área de pesquisa Interação Humano-Computador (IHC), provê ao usuário-final ferramentas para programação de aplicações através de elementos mais intuitivos e naturais. Dessa forma, permite que os usuários possam inserir seus conhecimentos e idéias nestes softwares computacionais, utilizando para isso abstrações que fornecem uma importante e poderosa interface para o usuário no processo de desenvolvimento.

Na computação Ubíqua/Pervasiva, cujo objetivo é auxiliar o usuário nas suas tarefas diárias, essas técnicas também são de grande importância (BERTI; PATERNÓ; SANTORO, 2004) (TRUONG; HUANG; ABOWD, 2004). Através dessa solução, usuários que não possuem grande conhecimento de computação poderão habilitar e decidir de que forma os programas e dispositivos computacionais agirão no dia-a-dia no auxílio as suas tarefas. Assim, usuários poderão criar e configurar os sistemas e os inúmeros dispositivos móveis utilizados e presentes no ambiente caseiro e profissional, de acordo com suas preferências, necessidades, etc.

Atualmente, existem várias abordagens que tentam facilitar a vida de usuários leigos, permitindo que esses possam escrever, ou simplesmente modificar, as aplicações desejadas. As pesquisas desenvolvidas identificam três diferentes categorias de *End-User Programming* (EUP), e essas diferenciam-se pela forma de interação com o usuário-final para a programação das aplicações. As categorias são:

(i) linguagens textuais simplificadas;

(ii) programação visual;

(iii) programação por demonstração.

Essas categorias são descritas a seguir.

2.2.1 Linguagens textuais simplificadas

A abordagem de Linguagens Textuais simplificadas talvez seja a forma mais simples de habilitar usuários a escrever programas e a disponibilizar linguagens de programação fáceis de entender e de usar (TRUONG; HUANG; ABOWD, 2004). Provavelmente, a primeira linguagem que apresentou esse propósito foi BASIC, desenvolvido na década de 60. Posteriormente, surgiram também outras linguagens como HyperTalk da Apple HyperCard (WHEELER, 2004) e Action-Script da Adobe Flash (MOOCK, 2002). No entanto, essas linguagens não eram direcionadas aos usuários-finais já que esses não possuiam conhecimentos de computação e de programação, necessários para tal atividade.

Para permitir que usuários-finais possam programar suas próprias aplicações através de linguagens textuais, novas propostas de tais linguagens de programação foram propostas: *Chickenfoot* (BOLIN, 2005), CAMP: *Capture and Access Magnetic Poetry* (TRUONG; HUANG; ABOWD, 2004) e HANDS: *Human-Centered Advances for the Novice*

Development of Software (PANE, 2002). Essas linguagens não devem obrigar que o usuário aprenda sintaxes de linguagens, assim como o fazem os programadores especializados, e nem devem deter conhecimentos sobre conceitos de programação, como abstrações, variáveis e laços iterativos.

2.2.2 Programação Visual

Uma outra abordagem de computação direcionada ao usuário-final é apresentada através das linguagens, ou sistemas, de programação visual. Nesse estilo de EUP vários componentes visuais são utilizados nas interfaces gráficas utilizadas para interação com os usuários. Os componentes visuais podem ser utilizados com funcionalidades para abstração e visualização de códigos das aplicações, do fluxo de execução das instruções, etc.

Na categoria de Programação Visual, têm-se três diferentes formas de classificar as linguagens. São elas:

(i) linguagens baseadas em diagramas;

(ii) linguagens baseadas em formulários;

(iii) linguagens baseadas em ícones.

Essas diferentes formas de Programação Visual são descritas na sequência.

2.2.2.1 Linguagens baseadas em diagramas

As linguagens que se enquadram nessa forma de Programação Visual, usualmente, utilizam diagramas para representar os elementos presentes na aplicação, como também o fluxo de execução destes e das informações que são trafegadas entre os vários elementos possíveis. Muitas vezes, utilizam-se metáforas dos fluxos de dados, e representam ainda o envio e recebimento de dados através de componentes visuais, como linhas conectadas às "caixinhas" que abstraem elementos da aplicação. Assim, geralmente, representa-se a entrada na parte superior e a saída na parte inferior, sendo que as entradas servem para conectar uma linha e as saídas das "caixinhas" permitem conexões de múltiplas linhas.

Duas importantes linguagens nessa forma de classificação da Programação Visual são

LabView e a *Max/MSP*. O sistema *LabView* (*Laboratory Virtual Instrument Engineering Work-bench*) (MORIARTY et al, 2003) foi inicialmente desenvolvido para permitir aos cientistas executar e controlar testes, medir e avaliar resultados ou, apenas, vizualizá-los. Atualmente, tem suporte à programação orientada a objetos, incluindo herança e polimorfismo, e suporte a multithreads para a paralelização automática, possibilitando a construção de dois fluxos de dados independentes. Outra proposta é *Max/MSP* (WOLEK, 2002) que é similar a um ambiente de programação para aplicações multimídia. Nesse sistema são oferecidos uma variedade de componentes que podem ser plugados através de linhas para construir um fluxo de dados. Esses componentes também possuem as ligações de entrada e saída permitindo a visualização das funções dos dados e, também, a leitura dos resultados computados. Dessa forma, *Max/MSP* permite que músicos não profissionais possam desenvolver suas aplicações multimídia em tempo real, através de processamento de interfaces, áudio e vídeo.

2.2.2.2 Linguagens baseadas em formulários

Os ambientes de programação visual baseados em formulários provêem uma interessante interface para criação e manipulação de células em formulários, sendo que essas podem conter valores específicos ou fórmulas que referenciam valores contidos em outras células (ROTHERMEL; LI; BURNETT, 1997). Estes ambientes possuem a importante funcionalidade de recalcular todos os possíveis valores que podem ser influenciados no caso da alteração do conteúdo de uma célula, ou redefinição de uma fórmula. Os exemplos mais difundidos dessas linguagens são as aplicações de planilhas eletrônicas utilizadas por usuários comuns e especializados.

Diversas pesquisas (ROTHERMEL; LI; BURNETT, 1997) têm mostrado que o principal problema de linguagens de programação visual baseadas em formulários é que os erros são frequentemente efetivados e que os usuários tem pouca confiança na disponibilidade de seus programas. Para a correta e eficiente utilização desses ambientes, muitos esforços, pesquisas e aprimoramentos ainda devem ser feitos (ROTHERMEL; LI; BURNETT, 1997).

2.2.2.3 Linguagens baseadas em ícones

As linguagens de programação visuais baseadas em ícones são assim denominadas justamente por dispor desses elementos gráficos, ícones, para abstrairem aos usuários- finais aplicações, recursos, dispositivos, etc. Dessa forma, usuários manipulam elementos gráficos representados através de ícones para definirem as funcionalidades, formas com que devem operar, etc.

Duas importantes propostas de linguagens baseadas em ícones são a *jigsaw pieces* (HUMBLE, 2003) e o iCAP: *as icong-based programming environment* (DEY et al, 2006). O *jigsaw pieces* propõe uma interface para configuração de aplicações que se destinam a casas inteligentes e, assim, permite que as interconexões e ações sequenciais simples referentes aos dispositivos possam ser definidas facilmente. A linguagem iCAP é uma ferramenta para construção de aplicações pervasivas sensíveis ao contexto que propicia a captura de dispositivos e resultados, e informações contextuais fornecidos por kits de ferramentas *(toolkits)* direcionados a contextos. Além disso, o modelo iCAP foi modelado para habilitar os usuários a descrever as situações e comandos dos objetos da casa, e estes agirem de acordo com os estados possíveis, descrevendo comportamentos específicos e não preferências.

2.2.3 Programação por demonstração

Nas linguagens de programação por demonstração (*Programming by Demonstration* - PBD) o usuário interage com o sistema demonstrando como o programa deve se comportar. Uma aplicação é formada pela generalização de interações com o usuário na forma de regras "antes-depois" (MYERS et al, 2006).

Um importante exemplo da técnica PBD é o *Stagecast Creator*, formalmente conhecido por *KidSim* ou *Cocoa* (SMITH et al, 1994). Essa linguagem utiliza o estilo de planilhas eletrônicas para prover um jogo de tabuleiro com discretas células, e essas são ocupadas por objetos denominados **agentes,** que podem agir de acordo com as regras dadas no momento de cada passo. As regras consistem da parte "antes" e da "depois" e são criadas por demonstração. Essas regras são representadas graficamente e podem ser reescritas pelo usuário.

2.3 Paradigma Orientação a Tarefas

A noção de **Computação baseada em Tarefas** foi introduzida pelo Projeto Aura (GARLAN; STEENKISTE; SCHRMEL, 2002) como um meio da infra-estrutura configurar dispositivos de forma que o usuário possa manter a continuidade do que estava fazendo, conforme ele se deslocava de um lugar a outro. Nesse modelo, tarefas são modeladas como uma coleção de serviços, e a descrição do serviço (qualidades, atributos, preferências) é usada para encontrar os recursos necessários ou reconfigurar o sistema para executar a tarefa. O sistema mantém uma representação explícita da intenção do usuário e a natureza dos serviços requeridos. O sistema Aura é auto-gerenciável (pró-ativo), ou seja, não há programação de aplicações envolvidas.

2.3.1 Propostas e Projetos

2.3.1.1 Projeto *Activity-based Computing* (ABC)

Atualmente, paradigmas de computação são centralizados nas aplicações e nos documentos, mas não se destinam à espaços pervasivos. Assim, surgiu a Computação centralizada nas atividades sendo que uma das primeiras propostas é dada por Christensem e Bardram (2002). Estes sistemas computacionais devem suportar e manipular atividades de trabalhos humanos. As atividades são tarefas ou processos usados por pessoas para realizar uma tarefa (trabalho), utilizando computação como parte da atividade. Uma atividade pode envolver um conjunto de diferentes sistemas que possuem funcionalidades e dados distintos. Dessa forma, têm-se níveis de aplicações nos sistemas computacionais atuais; e, então, objetiva-se criar o suporte no nível de atividades promovendo a inserção de novas abstrações.

Segundo Christensem e Bardram (2002), para suportar usuários com suas atividades físicas de trabalho, os sistemas computacionais devem obrigatoriamente compreender o conceito de atividade e manipulá-la como objetos de primeira classe. Assim, sistemas computacionais, com serviços orientados a domínios (conjunto de serviços/aplicações

relacionados a um domínio particular), devem ser implementados e executados sob uma infra-estrutura de computação centralizada em atividades. O conceito de atividade computacional deve ser a equivalência digital de atividade física, e, assim, essas atividades podem ser classificadas da mesma maneira que as atividades humanas de trabalho. Por exemplo, no caso de ambientes hospitalares a atividade "prescrição de medicamento" deve embutir todo o estado relevante para a atividade: identificar o paciente, médico, horário, registros médicos, resultados laboratoriais, as aplicações usadas e o contexto corrente.

Com base nessas idéias iniciais, desenvolveu-se o projeto *Activity-based Computing* (ABC) (BARDRAM; CHRISTENSEN, 2004) que apresenta o suporte aos usuários-finais através de atividades computacionais. Essas atividades podem ser inicializadas, suspensas, armazenadas, retomadas em qualquer dispositivo computacional em qualquer instante de tempo, encaminhadas para outros usuários e compartilhadas entre diversos usuários. Além disso, a execução das atividades é adaptável de acordo com o contexto considerado dos usuários. Um dos principais objetivos do projeto é permitir que desenvolvedores de aplicações clínicas possam incorporar suporte à mobilidade, interrupções, atividades paralelas, cooperação e consciência de contexto para a concepção e implementação de seus programas que executarão em uma infra-estrutura hospitalar. Assim, o Framework ABC provê uma infra-estrutura de execução com serviços que suportam as características direcionadas a trabalhos clínicos. Além disso, é disponibilizado um modelo de programação para desenvolvimento de serviços e aplicações ABC (BARDRAM; CHRISTENSEN; OLSEN, 2004).

O projeto ABC tem seguido um processo de desenvolvimento interativo, tendo como alvo cinco importantes temas médicos que refletem as maiores áreas de trabalho em grandes hospitais: (i) administração médica realizada por enfermeiros; (ii) prescrição médica realizada por médicos; (iii) colaboração entre clínicos; (iv) conferências médicas; (v) cirurgias.

A infra-estrutura ABC (figura 2.1) é composta basicamente por quatro subsistemas:

- Subsistema consciente do contexto e da localização. Responsável pelo gerenciamento das informações contextuais e por monitorar a localização das entidades presentes no ambiente, fornecendo essas informações para outros subsistemas. Esse subsistema serve como um banco de dados de informações contextuais permitindo inclusive que outros subsistemas consultem essas informações e, ainda, pela notificação ao subsistema de gerenciamento de atividades sobre eventos que ocorreram no ambiente;
- Subsistema de gerenciamento de atividades. Responsável por armazenar e gerenciar as atividades, encaminhando-as aos *activity bars* que são interfaces de interação com o

usuário nos relevantes computadores públicos. As atividades são apresentadas ao usuário-final de uma forma não-intrusiva;

- Subsistema de descoberta de atividades. Responsável por inferir de uma forma pró-ativa as atividades que poderão ser executadas no ambiente, baseando-se nas informações contextuais e em heurísticas que definem processos de trabalho em ambientes de cuidados clínicos.
- Aplicações cliente. São aplicações do Sistema de Registro Eletrônico do Paciente que podem ser executadas em computadores públicos. Essas ainda são responsáveis pelo armazenamento das atividades e do seu estado, no subsistema de gerenciamento de atividades quando o usuário explicitamente solicitar essa operação ou, então, quando o usuário sair do computador público.

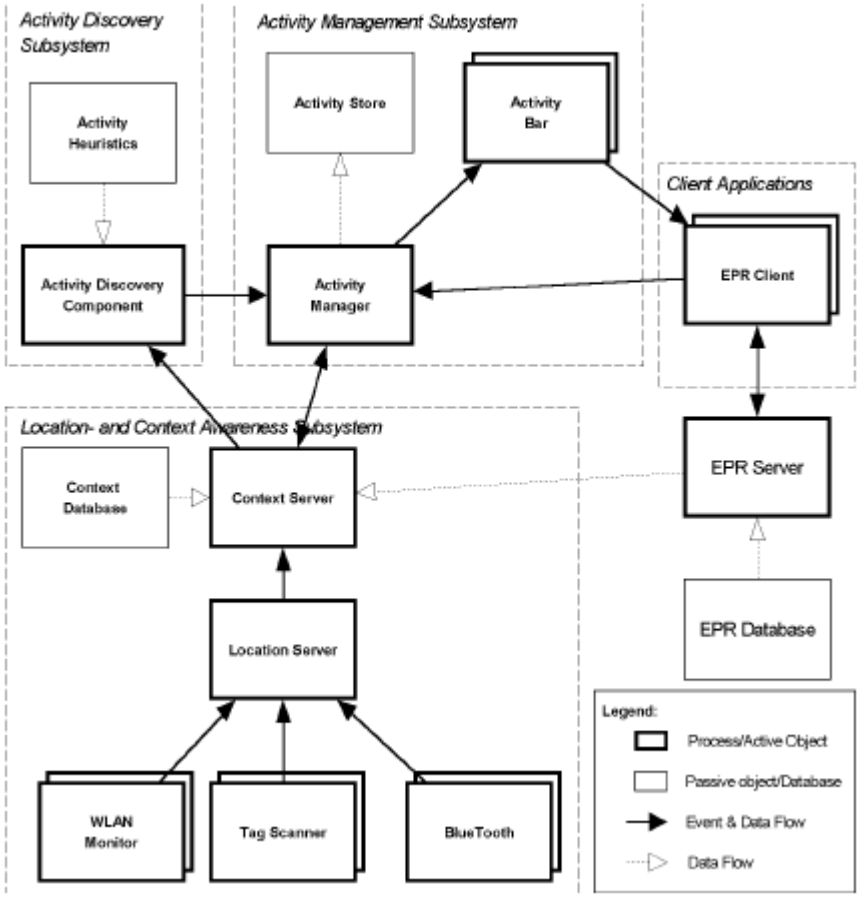

Figura 2.1 – Arquitetura *Activity-based Computing* (ABC). Fonte: (BARDRAM; CHRISTENSEN; OLSEN, 2004, p. 7).

O projeto ABC tem como foco principal prover uma infra-estrutura para o suporte à

execução de atividades clínicas. Nesse projeto ainda é disponibilizado um modelo de programação para o desenvolvimento de serviços e aplicações, mas a utilização desse modelo requer conhecimentos de programação que, geralmente, apenas profissionais de computação possuem. Assim, a programação de atividades não é uma tarefa simples e acessível aos profissionais clínicos. Além disso, não são providos mecanismos para permitir que os profissionais personalizem as atividades inserindo nelas a sua forma particular de executá-las. A disponibilização de mecanismos que facilitem a definição de atividades e também a personalização dessas por parte dos profissionais clínicos, são importantes aspectos tratados neste trabalho e que não são abordados no projeto ABC.

2.3.1.2 Projeto Gaia

O projeto Gaia (ROMAN et al, 2002) visualiza um futuro onde o espaço habitado pelas pessoas é interativo e programável, sendo denominado de espaços ativos *(Active Spaces)*. Considera que os usuários interagem com seus escritórios, casa, carros, etc. para requisitar informações, beneficiar-se dos recursos disponíveis e configurar o comportamento de seu habitat. Dados e tarefas estão sempre acessíveis e são mapeados dinamicamente para os recursos convenientes e presentes na localização corrente pelo sistema GaiaOS.

Espaços ativos são altamente dinâmicos já que o contexto e os recursos disponíveis nesses ambientes mudam rapidamente (ROMAN, 2002). O grande número de entidades presentes nesses espaços e o dinamismo associado a elas dificultam que os desenvolvedores programem esses ambientes. Dessa forma, como parte integrante do projeto Gaia foi criado o Modelo de Programação Olympus (RANGANATHAN; CAMPBELL, 2005) e, assim, facilitar a programação dos espaços ativos.

Considerando que diferentes ambientes podem ter diferentes formas de executar diferentes tipos de tarefas, desenvolvedores não podem esperar saber detalhes de como são as várias tarefas, para executá-las nos mais diversificados ambientes pervasivos (espaço físico composto por grande número de dispositivos digitais, como sensores e computadores). Dessa forma, o modelo de Programação Olympus propicia aos desenvolvedores referir-se as entidades reais (dispositivos digitais, recursos computacionais, etc.) como entidades virtuais em seus programas (RANGANATHAN et al, 2005) (RANGANATHAN; CAMPBELL, 2005).

O Olympus é ligado a um framework, que associa as entidades virtuais às entidades reais do espaço ativo. E isso é realizado baseado em constantes especificadas pelo desenvolvedor, em recursos disponíveis no espaço corrente, políticas de nível de espaço e no corrente contexto do espaço. Além disso, o framework utiliza hierarquias de ontologias e descrição de entidades para a descoberta semântica de entidades apropriadas. O principal conceito empregado no processo de descoberta de entidades é a separação de classes e descoberta de instâncias, sendo que primeiro se descobre todas as possíveis classes de entidades que satisfazem a todos os requisitos e, depois, utilizam-se funções de Utilidade para descobrir qual é a mais apropriada.

O modelo Olympus possui cinco importantes funcionalidades listadas a seguir.

- Utilizar ontologias para especificar hierarquias de diferentes tipos de entidades e para especificar propriedades dessas entidades.
- Algoritmos de combinação semântica que utilizam as hierarquias de ontologias para a descoberta apropriada de classes de entidades e execução de certas tarefas.
- Políticas e regras Sensíveis ao Contexto para escolha apropriadas de classes e instâncias de entidades.
- Função multi-dimensional para escolha das melhores instâncias de entidades para executar determinadas tarefas.
- Proposta de um conjunto básico de operadores de alto-nível para programar ambientes de computação pervasiva.

O desenvolvimento de aplicações para espaços ativos não é uma tarefa das mais simples já que estes espaços são diferentes uns dos outros, divergindo nos recursos que eles provêem, serviços suportados, etc... Dessa forma, o Olympus facilita essa tarefa permitindo ao usuário programar espaços ativos em termos de entidades virtuais, e essas são compostas essencialmente de variáveis que ainda não foram instanciadas. De acordo com as ontologias, as entidades virtuais são associadas às classes. O modelo define oito tipos básicos de entidades que são: `Application`, `ApplicationComponent`, `Device`, `Service`, `Person`, `PhysicalObject`, `Location`, `ActiveSpace`. Essas entidades são tratadas como objetos de primeira classe e podem ser armazenadas em variáveis, usadas em expressões e passadas como parâmetros para as funções.

O projeto GAIA destina-se a ambientes pervasivos como casas e escritórios, e não a ambientes hospitalares. A noção de tarefas apresentadas nesse projeto objetiva a adaptação de contexto visando suportar o que o usuário deseja fazer, encontrando os recursos disponíveis no ambiente e as configurações existentes (definidas pelo sistema ou pelo usuário) para

executar as tarefas da melhor forma possível nos determinados contextos. A programação das tarefas é possível através da Linguagem de Programação Olympus e, assim, o usuário poderá programar e personalizar as suas tarefas. A dificuldade é que apenas profissionais especializados sabedores de técnicas de programação conseguem utilizar a linguagem Olympus para a programação e definição de tarefas. Assim, a proposta defendida no presente trabalho de permitir aos usuários comuns programarem suas tarefas e personalizá-las adequando-as a sua forma particular de executar, não é possível no GAIA.

2.3.1.3 *Task Computing*

O framework orientado ao usuário *Task Computing* (MASUOKA; PARSIA; LABROU, 2003), pertencente à empresa Fujitsu, é uma proposta de um novo paradigma para usuários permitindo que esses realizem complexas tarefas em aplicações, dispositivos e serviços em ambientes pervasivos. Esse framework provê várias formas de interação do usuário com esses ambientes e, ainda, a interação entre diferentes ambientes pervasivos. Dessa forma, a proposta de *Task Computing* é possível pela disponibilidade de soluções para descoberta de serviços, publicação e gerenciamento dinâmico de serviços, além da criação de tarefas e execução de tarefas *on-the-fly*. A arquitetura (figura 2.2) é composta por quatro camadas:

- Camada de Realização. Camada que abrange o universo de dispositivos, aplicações, `e-services`;
- Camada de Serviço. Contém as várias fontes de funcionalidades disponibilizadas como serviços, e cada serviço tem associada uma descrição semântica dele;
- Camada de *Middleware*. Responsável por descoberta de serviços, execução e monitoramento de execução de serviços, gerenciamento das tarefas incluindo criação e publicação;
- Camada de Apresentação. Camada que provê a abstração da complexidade da tarefa ao usuário. Essa camada apresenta ao usuário um ambiente com funcionalidades que podem ser criadas dinamicamente pelos usuários na execução de tarefas.

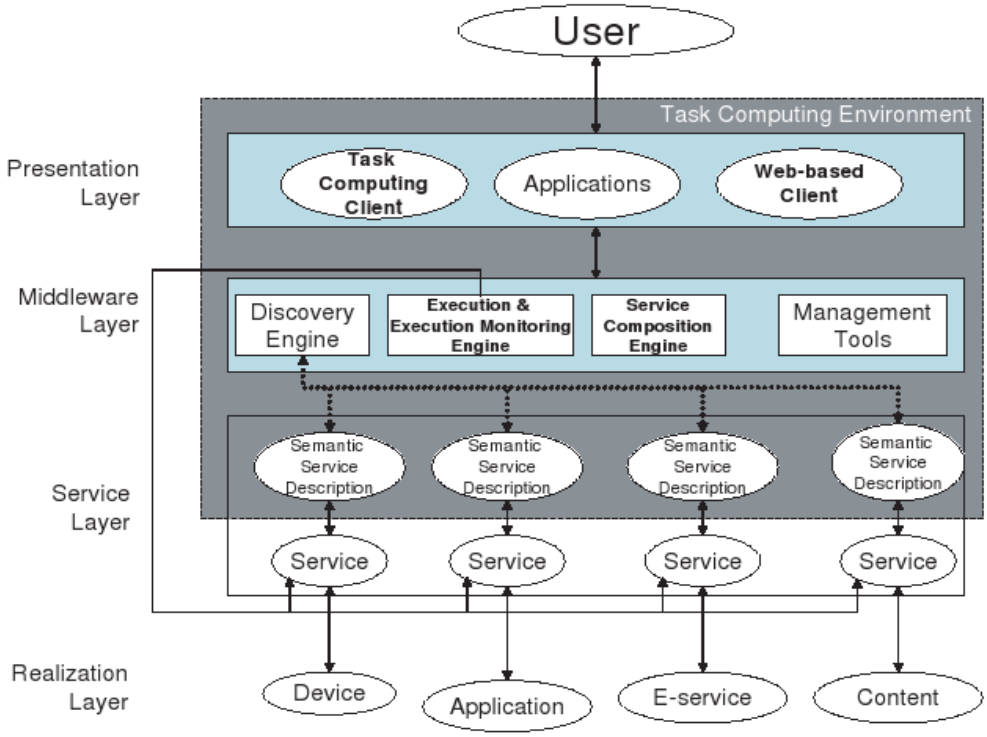

Figura 2.2 – Arquitetura do ambiente *Task Computing*. Fonte: (SONG; LABROU; MASUOKA, 2004, p. 2).

Considera-se que serviços são abstrações de funcionalidades que interessam ao usuário, sendo que essas funcionalidades são providas por diferentes tipos de fontes como dispositivos, aplicações e aplicações da web (*e-services*). No caso de dispositivos originando serviços, faz-se uma associação do dispositivo ao núcleo de funcionalidades que este provê, como, por exemplo, o telefone (dispositivo) que têm como principal funcionalidade realizar chamadas a outros telefones (serviço).

Como integrante da arquitetura têm-se o *Task Computing Client* (TCC) que se refere às interfaces via voz, texto e gráficos do usuário com o sistema e, assim, abstrai-se as complexidades de implementação do usuário. Dessa forma, a descoberta de serviços essencialmente é o processo de descoberta da Descrição Semântica do Serviço (SSD) de determinado serviço, já que esses contêm informações de como esses serviços devem ser manipulados pelos usuários. Para a descrição semântica dos SSD's foi utilizado OWL-S (*Semantic Markup for Web Services*) já que esse padrão suporta a descrição dos serviços na forma de metadados e propicia que seja feita a distinção entre a descrição do serviço e a sua implementação. A Descrição Semântica do Serviço (SSD) é dividida em: (i) ***Profile*** - *p*rovê informações descritivas dos serviços como nome, descrição textual, semântica de entrada e saída destes; (ii) **Processo** - descreve o serviço, em nível semântico do processo; (iii) ***Grounding*** - *d*escreve aonde a implementação do serviço pode ser encontrada e como os

objetos semânticos da parte do Processo, e os parâmetros da implementação do serviço, são mapeados para outros serviços.

A descoberta dos serviços, que é o processo de encontrar os serviços relacionados ao contexto do usuário, é reduzida à aquisição dos SSD'S dos serviços pelo TCC (SONG; LABROU; MASUOKA, 2004). A implementação da descoberta de serviços pode utilizar um ou mais mecanismos de descoberta e o usuário (ou os serviços e os provedores destes serviços) podem definir qual o mecanismo de descoberta é o mais apropriado, e que deverá ser empregado, para a descoberta de um serviço determinado.

O framework disponibilizado no projeto *Task Computing* visa capacitar o usuário a executar tarefas em aplicações, dispositivos e serviços em ambientes pervasivos. O projeto disponibiliza diversas formas para o usuário interagir com o framework, mas o foco principal está na infra-estrutra de suporte à execução e gerenciamento dos serviços. O principal aspecto abordado é a descoberta dinâmica de serviços que considera as informações de contexto para a definição dos serviços mais apropriados. O projeto não se destina a ambientes de saúde e não apresenta uma modelagem de tarefas capaz de suportar que os usuários possam programar e personalizar as suas tarefas, ao contrário do presente trabalho que visa disponibilizar essas funcionalidades aos usuários.

2.3.2 Teoria da Atividade

A teoria da atividade é um modelo que busca identificar um conjunto de componentes comuns constituintes da atividade e o relacionamento existente entre esses componentes. A teoria original foi proposta por pesquisadores russos na década de 20 e ao longo do tempo foi sendo aperfeiçoada e expandida (KAENAMPORNPAN; O'NEILL, 2005). A teoria identifica seis componentes na atividade: sujeito, objeto, ferramentas, regras, comunidade e colaboração. O sujeito realiza um atividade buscando atingir seu objetivo, utilizando para isso ferramentas baseadas em regras. Ainda, esse usuário está inserido em uma comunidade, onde se pode utilizar a colaboração entre indivíduos para que uma atividade seja realizada.

Aplicando ao ambiente clínico o modelo genérico proposto pela teoria da atividade tem-se o usuário clínico atuando como o sujeito, que tem por objetivo diagnosticar e/ou tratar o paciente que atua como objeto da tarefa. Para realizar sua atividade, o clínico utiliza as ferramentas de mediação, as quais consistem em registros, procedimentos, equipamentos e

recursos, os quais propiciam a execução da tarefa. Essa mediação, geralmente, segue regras especificadas através de guias clínicos. A tarefa é executada no ambiente clínico, sendo está a sua comunidade, onde há uma diversidade de profissionais que, normalmente, realizam uma divisão do trabalho para a realização de uma tarefa.

Como exemplo, pode ser considerado o procedimento de diagnóstico clínico. Geralmente, após o paciente consultar-se com o seu médico, este, baseado nos sintomas informados pelo paciente e pelo seu histórico, poderá requisitar alguns exames complementares para auxiliar no diagnóstico. Esses exames são realizados por outros profissionais, um farmacêutico, por exemplo, e irão produzir um laudo que deverá ser analisado pelo médico. O médico se considerar ter as informações suficientes, produzirá um diagnóstico, podendo receitar tratamentos específicos ou, se julgar necessário, poderá requisitar outros exames para complementar as informações e produzir um diagnóstico mais preciso. No exemplo percebe-se a relação de colaboração entre os procedimentos realizados pelo médico e pelo farmacêutico, sendo partes complementares de uma mesma tarefa, porém com pontos-de-vista diferentes. O médico não precisa saber como é o processo de realização do exame, basta conhecer o resultado e o farmacêutico, por outro lado, não necessita conhecer o histórico do paciente ou mesmo os motivos que levaram à requisição do exame.

3 PROJETO CLINICSPACE

Este capítulo aborda o escopo de desenvolvimento deste trabalho: o projeto ClinicSpace.

3.1 Motivação e Problemas Abordados

A computação Ubíqua/Pervasiva possui inúmeras áreas de aplicação de pesquisa e tecnologia. A área da Saúde é uma das principais devido à inerente natureza de suas atividades, onde o mundo físico é o gerador da informação e deve se tornar também o fornecedor pró-ativo de informações aos sistemas informatizados.

A Computação Móvel que, em termos de evolução dos sistemas de computação, pode ser considerada como uma etapa anterior à Computação Ubíqua ou Pervasiva, já está presente em alguns ambientes como hospitais e centros de saúde do país. Nesses ambientes, os dispositivos móveis, como PDA's e tabletPCs, tem sido utilizados para auxiliar as tarefas dos clínicos já que essas são caracterizadas por cooperação, devido à multidisciplinaridade, alto grau de mobilidade e paralelismo das atividades. Sistemas de informações clínicas estão contribuindo para atender essas necessidades e influenciando as práticas e fluxos de trabalho de seus usuários.

Atualmente, os clínicos (pessoas) são os responsáveis por captar informações do ambiente e introduzi-las (digitá-las) nos sistemas informatizados, gerando uma série de problemas bem conhecidos, destacando-se o alto grau de rejeição dos sistemas computacionais pelos clínicos, principalmente em ambientes hospitalares.

A área de Saúde possui muitos ambientes onde há necessidade de informações do mundo físico serem integrados automaticamente às soluções computacionais (mundo virtual). A *Pervasive Healthcare Computing* oferece vantagens competitivas aos provedores de serviços de saúde; em particular, aumenta a eficiência do serviço, a qualidade e melhora do gerenciamento na relação com o paciente (VARSHNEY, 2003).

O sistema de saúde do futuro provê o uso de tecnologias da Computação Pervasiva formando um espaço inteligente *(ambient intelligence)*, reativo e pró-ativo, onde dispositivos móveis ou fixos estão inseridos no ambiente com objetivo de captar informações do meio físico e transmitir as alterações detectadas para os sistemas de gerenciamento de informações, os quais tomarão decisões e adaptar-se-ão às situações detectadas *(Context-Aware computing)*.

Em termos de pesquisa e inovação, a Computação Ubíqua na Saúde está em sua primeira geração, procurando atender as necessidades, características e tecnologias para projetar sistemas que criarão o hospital do futuro.

O Projeto ClinicSpace, em definição e desenvolvimento no grupo de pesquisa GMob do PPGI, visa a utilização de tecnologias providas pela Computação Ubíqua para o auxílio aos profissionais clínicos na execução de suas tarefas nos ambientes hospitalares. Ressalta-se que, mesmo o sistema ubíquo devendo ser pró-ativo (agir em nome do usuário), esse deve levar em conta que o usuário deseja manter a sua forma de executar o seu trabalho. Dessa forma, objetiva-se fornecer mecanismos que permitam aos usuários interagir e personalizar o sistema, adequando-o melhor a sua forma de executar as tarefas. Argumenta-se que a personalização do modo como cada um executa suas atividades, aliado a soluções da ubiqüidade, pode levar a diminuir a rejeição dos sistemas computacionais dentro do ambiente hospitalar.

O projeto propõe a definição de uma ferramenta-piloto, na forma de uma interface, para a personalização de tarefas, realizada pelo próprio usuário (médicos) com vistas ao gerenciamento de suas atividades clínicas, relativas a diagnóstico, tratamento, laboratorial e administrativas. O sistema proposto é visto como um assistente, auxiliando o médico a executar as tarefas que compõem suas atividades profissionais diárias. No escopo do projeto, esse trabalho iniciou a definição de atividades, tarefas e sua modelagem computacional, descritas a seguir, além de discutir a arquitetura de software necessária para seu gerenciamento.

3.2 Conceitos adotados

Os projetos analisados (capítulo 2) apresentam similaridades e diferenças entre si quanto à utilização dos termos Tarefa e Atividade. A utilização e definição dos termos Tarefa

(utilizado nos projetos Aura, Gaia e *Task Computing*) e Atividade (utilizado no projeto ABC) comprovam as diferenças de interpretação e de utilização desses termos nos projetos. Logo, a explicitação dos termos e de seus significados adotados no projeto ClinicSpace torna-se necessária, já que até o momento os projetos apresentam diversificadas formas de conceituar os termos.

No Projeto ClinicSpace, adota-se um modelo que se baseia no apresentado por Ranganathan and Campbell (2005) e na Teoria da Atividade. Define-se **Tarefa** como "o conjunto de ações executadas colaborativamente por humanos e sistemas de computação pervasivos para atingir os objetivos". As tarefas podem ser compostas por subtarefas e por outras tarefas que, juntas, compõem um fluxo de trabalho. As tarefas ainda são assistidas por aplicações computacionais. As subtarefas implementam serviços e pequenas aplicações disponibilizadas aos usuários-finais para que eles possam programar as suas próprias tarefas.

Já, o termo **Atividade** remete-se ao processo realizado por humanos de forma colaborativa, coordenada, distribuída, em um espaço determinado. No ClinicSpace, as atividades podem ser decompostas em tarefas, auxiliadas por aplicações computacionais, tais como a identificação de médico-paciente e contexto de uso, e seguem a forma particular de cada indivíduo de realizá-la. Esse aspecto é relevante no projeto, pois um dos objetivos é permitir ao profissional clínico a utilização de interfaces intuitivas para o usuário interagir com o sistema (utilizando mecanismos disponibilizados pela programação orientada ao usuário-final), diminuindo sua rejeição a este. Assim, o usuário poderá personalizar a realização das tarefas inserindo a sua própria forma e preferências particulares ao executá-las.

As tarefas clínicas adotadas são detalhadas no capítulo 4.

3.3 Propriedades e requisitos para a Modelagem das Tarefas

No modelo projetado para a arquitetura de software ClinicSpace, as tarefas são compostas por subtarefas ou tarefas reusáveis **(Decomposição).** Freqüentemente, diferentes tarefas possuem subtarefas em comum. Assim, as subtarefas e tarefas podem ser recombinadas de diferentes maneiras para originarem novas tarefas (**Recombinação**) e facilitar o processo de desenvolvimento de tarefas já que podem ser utilizadas subtarefas e tarefas já programadas (**Reuso**).

ClinicSpace destina-se a ambientes clínicos, os quais possuem características

peculiares, como as freqüentes interrupções que os profissionais clínicos sofrem ao executar as suas atividades diárias. Os clínicos, muitas vezes, acabam tendo que interromper seus procedimentos para atenderem a outras chamadas ou pacientes. Profissionais clínicos percorrem uma grande distância em um hospital, pois executam os procedimentos de cuidados clínicos aos pacientes em diversificados locais (BARDRAM; CHRISTENSEN; OLSEN, 2004). Dessa forma, o modelo deve considerar tais características do trabalho em ambientes hospitalares e permitir que as tarefas possam ser interrompidas e retomadas posteriormente, inserindo-se no modelo de co-rotina (**Interrupção**). As tarefas devem migrar e se adaptar as diferentes características do ambiente (**Móveis e Adaptativas**) de forma a acompanhar o usuário - semântica siga-me (AUGUSTIN, 2005). As tarefas também podem estar associadas a um contexto; caso não estejam, receberão essa associação dinamicamente **(Contextualizada).**

Para um correto e eficaz gerenciamento das tarefas é necessário o monitoramento das tarefas. As tarefas obrigatoriamente estão em um dos seis estados, que são:

- Inicializada. A tarefa foi criada e inicializada recentemente*;*
- Executando. A tarefa está ativa e executando no momento;
- Cancelada. A tarefa que estava executando foi cancelada;
- Pausada. A tarefa foi interrompida e pode ser ativada (retomada a sua execução) a qualquer momento;
- Finalizada. A tarefa foi finalizada por ter encerrado a sua execução.
- Encerrada. A tarefa foi encerrada após ter sido cancelada ou finalizada.

As transições entre os estados possíveis para as tarefas são representadas na Figura 3.1. Uma tarefa instanciada está no estado “Inicializada”. No momento que é iniciado a execução da tarefa, o estado dela passa a ser “Executando”. A transição entre esses estados pode ser automática e instantânea, ou então configurada através do agendamento de tarefas. Assim, uma tarefa no estado “Inicializada” também pode ir para o estado “Cancelada”, caso o usuário solicite.

Estando a tarefa no estado “Executando”, ela pode assumir três outros estados: (i) “Cancelada”, se o usuário cancelar a execução; (ii) “Finalizada”, se a sua execução for concluída; (iii) “Pausada”, quando sua execução é interrompida. Do estado “Pausada”, a tarefa pode retornar ao estado “Executando”, caso o usuário solicite. Ou pode ir para o estado “Cancelada”, se o usuário não desejar continuar a execução da tarefa. Já, dos estados “Finalizada” e “Cancelada”, uma tarefa só pode passar ao estado “Encerrada”, quando o sistema de gerenciamento liberar os recursos alocados a ela.

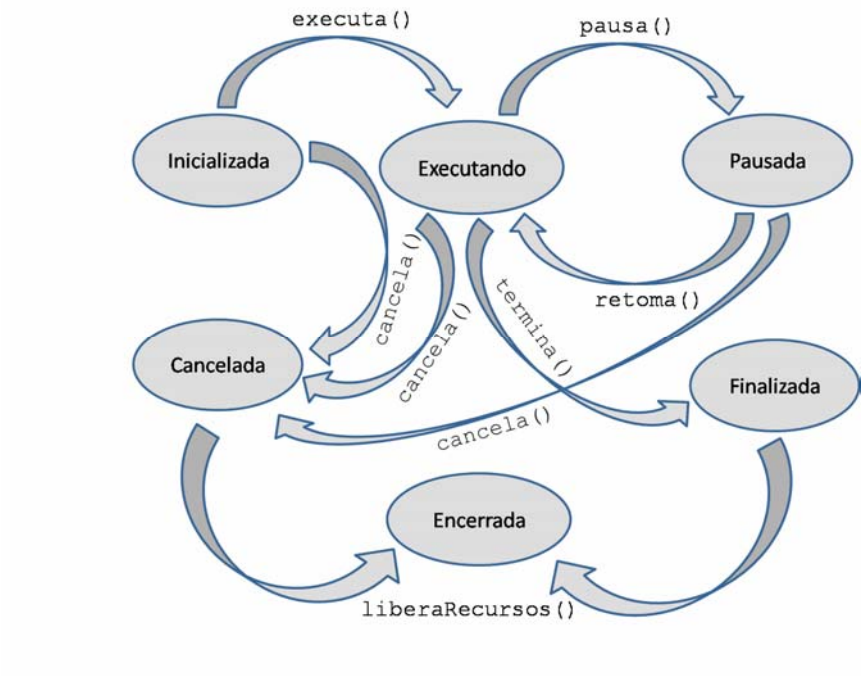

Figura 3.1 – Máquina de Estados para as Tarefas

3.4 Atributos Descritivos da Tarefa

As tarefas possuem um conjunto de informações que são importantes para a correta utilização e gerenciamento dessas. As informações de uma tarefa são:

- Criador. Identifica o criador da tarefa, sendo que cada tarefa possui apenas um único criador;
- Estado. Para o gerenciamento das tarefas, é necessário o conhecimento de em qual estado ela se encontra. Lembrando que uma tarefa obrigatoriamente encontra-se em um dos cinco estados citados.
- Descrição. Descrição textual das tarefas.
- Recursos. Responsável por informar pré-condições de execução da tarefa, aplicações assistentes a ela.
- Contexto. As tarefas podem estar associadas a contextos, e esses ainda podem ser opcionais ou obrigatórios para a execução da tarefa.
- Código. As tarefas possuem um código associado a elas, onde se encontram as instruções de código que serão executadas para a obtenção das funcionalidades particulares de cada tarefa.

3.5 Solução Adotada para a Programação de Tarefas pelo Usuário-Final

Decidiu-se na modelagem da arquitetura de software ClinicSpace utilizar os mecanismos providos pela programação orientada ao usuário-final para disponibilizar uma ferramenta-piloto cujo componente visível ao usuário-final é uma interface simples para que os profissionais clínicos possam criar tarefas, definir fluxo de execução a elas, de uma forma intuitiva e eficaz.

Para a modelagem dessa interface utilizam-se os resultados das análises realizadas nas propostas de programação orientada ao usuário-final, apresentadas no capítulo 2, as quais disponibilizam diversos mecanismos para a interação entre o usuário e o sistema. Algumas propostas sugerem o uso de sintaxes de linguagens simples e intuitivas (Linguagens Textuais Simplificadas) para atender as necessidades, pois essas linguagens de programação são fáceis de entender e de usar (TRUONG; HUANG; ABOWD, 2004). Outras abordagens baseiam-se na utilização de elementos gráficos como ícones, formulários e diagramas (Programação Visual) para proverem várias funcionalidades como abstração e visualização de códigos das aplicações e do fluxo de execução das instruções.

Usando as vantagens das duas abordagens, optou-se por implementar a interface baseada em uma solução híbrida que utiliza os preceitos do paradigma de Programação Visual e utiliza os elementos gráficos ícones e diagramas. Tarefas, subtarefas e contextos podem ser representados por ícones que remetam às funcionalidades específicas de cada elemento e ainda, servem para abstrair ao usuário-final aplicações, serviços, recursos, etc. Os diagramas são utilizados para representar o fluxo de execução de tarefas e subtarefas e o fluxo das informações que trafegaram entre os vários elementos. O intuito é fornecer uma interface intuitiva que possibilite ao usuário clínico, através da técnica de “clicar-e-arrastar”, definir a modelagem das suas tarefas diárias. Esse modelo é detalhado no capítulo 4.

3.6 Arquitetura ClinicSpace

A arquitetura para a programação e gerenciamento personalizado das tarefas, mostrada na 3.2, foi organizada em níveis que refletem as visões do sistema: (i) nível superior, é composto pelo usuário-final (médico) que interage com a ferramenta para (re) programar suas

tarefas que executarão num ambiente pervasivo; (ii) nível intermediário, é composto pelo mapeamento entre tarefas (definidas pelo usuário) e subtarefas (aplicações pervasivas) e pelo gerenciamento de ambas; (iii) nível inferior, é composto pelo conjunto de serviços do *middleware* de gerenciamento do ambiente pervasivo e de suporte à execução das aplicações pervasivas: EXEHDA (YAMIN et al, 2005).

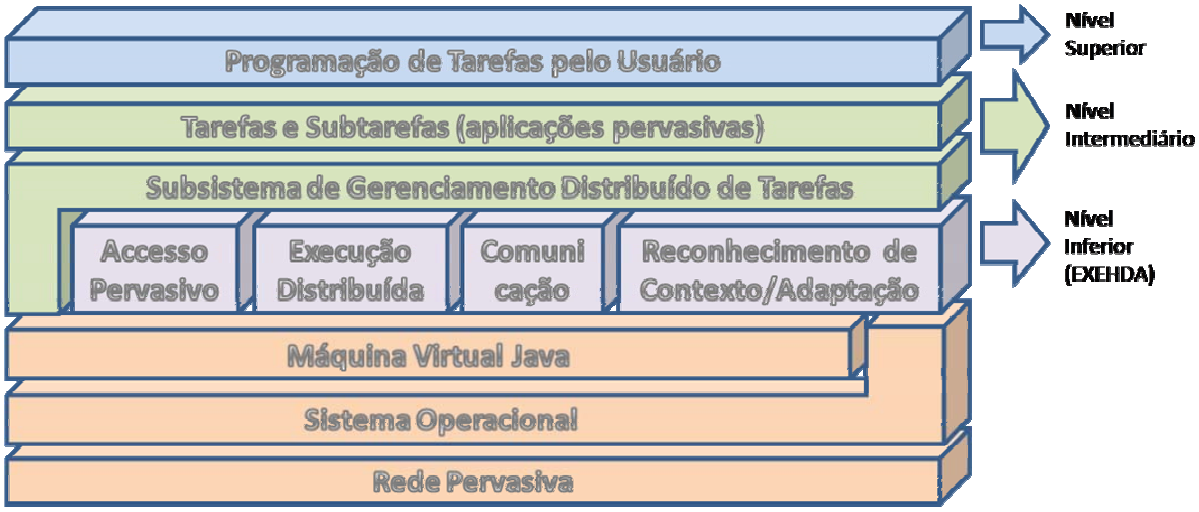

Figura 3.2 - Arquitetura para programação e gerenciamento de tarefas

A ferramenta-piloto proposta pela ClinicSpace consta de: (i) ferramenta gráfica de criação e personalização de tarefas, (ii) *middleware* para dar suporte à execução das tarefas denominado de novo Subsistema de Gerenciamento Distribuído de Tarefas - SGDT, e (iii) Sistema pEHS (pervasive Electronic Health-care System). A relação entre esses componentes é ilustrada na Figura 3.3.

Assim, o relacionamento entre os componentes é composto das seguintes etapas:

1. O usuário cria e personaliza as tarefas a partir de outras tarefas e subtarefas já modeladas;
2. A Ferramenta de Criação e Personalização de Tarefas interage com a APIOntologias (detalhado na seção 4.3.3) para a criação de uma descrição ontológica para a tarefa;
3. A descrição ontológica da tarefa é processada pelo SGDT e convertida em uma aplicação pervasiva;
4. O SGDT instancia e gerencia a tarefa, com o auxílio dos outros serviços do EXEHDA;
5. A tarefa gerencia a execução e a troca de informações de suas subtarefas;

6. As subtarefas invocam aplicações do pEHS, as quais são contextualizadas e adaptativas.

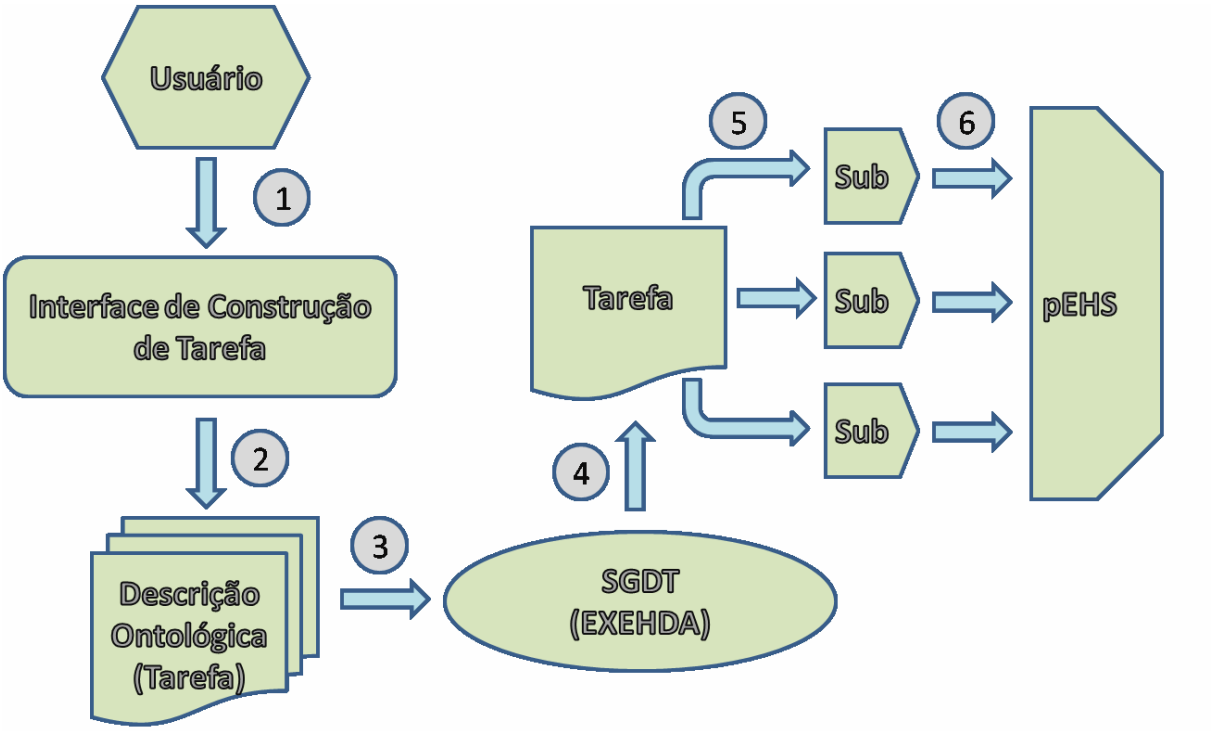

Figura 3.3 – Relacionamento entre os componentes que compõem o sistema proposto

A ferramenta de criação e personalização de tarefas é usada pelo usuário para: (i) criar tarefas simples, a partir de subtarefas (aplicações pervasivas do sistema); (ii) criar tarefas compostas (workflow), a partir de suas próprias tarefas; e (iii) personalizar suas tarefas de acordo com suas necessidades.

Neste primeiro momento, supõem-se a existência de um sistema pervasivo de informação de Saúde, denominado pEHS[1], que é um software de gerenciamento de informações de saúde com características de pervasividade, como migração da sessão do usuário e adaptação ao dispositivo. As características pervasivas do pEHS permitem que aplicações não terminadas possam ser retomadas. Assim, qualquer tarefa pode ser interrompida e retomada pela interface do usuário, sem que ele necessite conhecer o pEHS. Portanto, o pEHS deve implementar: (i) migração de suas aplicações; (ii) armazenamento temporário das informações que estão sendo utilizadas pela aplicação; (iii) adaptação da interface e das informações de suas aplicações.

[1] O pEHS está sendo modelado nos trabalhos de dissertação (em andamento) dos mestrandos Caroline Vicentini e Alencar Machado (PPGI - UFSM), também ligados ao projeto ClinicSpaces.

A execução das tarefas é gerenciada pelo *middleware* EXEHDA (YAMIN et al, 2005). Porém, como o EXEHDA não foi desenvolvido para ser orientado a tarefas, a introdução do conceito de orientação a tarefas (*activity-driven* ou *task-oriented computing*) exigiu a inserção de um novo subsistema no *middleware*, estendendo suas funcionalidades. O novo Subsistema de Gerenciamento Distribuído de Tarefas (SGDT)[2] tem a função de fazer a ponte entre tarefas e aplicações pervasivas, conforme definidas na arquitetura do *middleware*, auxiliando no processo de conversão de tarefas-aplicações. Portanto, o SGDT é responsável por gerenciar, em alto nível, as tarefas, delegando o gerenciamento das subtarefas (aplicações pervasivas) para os subsistemas atuais do EXEHDA.

3.6.1 Estrutura do Subsistema de Gerenciamento Distribuído de Tarefas (SGDT)

O novo Subsistema de Gerenciamento Distribuído de Tarefas (SGDT), mostrado na Figura 3.4, é responsável por gerenciar, em alto nível, as tarefas, delegando o gerenciamento das subtarefas (aplicações pervasivas) para os serviços atuais do EXEHDA.

O novo subsistema foi modelado de acordo com a especificação do *middleware* EXEHDA. Assim como os outros subsistemas, foi dividido em serviços: (i) Serviço de Acesso a Tarefas, (ii) Serviço de Contexto de Tarefas, (iii) Serviço de Inferência, (iv) Serviço de Gerenciamento de Tarefas, (v) Serviço de Tarefas Ativas; (vi) Serviço de Colaboração, e (vii) Serviço de Interceptação.

Serviço de Acesso às Tarefas

O Serviço de Acesso a Tarefas é responsável por armazenar e acessar informações das tarefas e da execução delas para cada usuário. Assim, esse serviço possui duas funcionalidades: (i) acesso à base de dados de tarefas codificadas, que é um repositório de tarefas disponíveis ao usuário; e (ii) acesso ao histórico de execução de tarefas, que armazena informações úteis ao Serviço de Inferência.

---

[2] O Subsitema de Gerenciamento Distribuído de Tarefas (SGDT) é abordado com maiores detalhes na dissertação do mestrando Giuliano Lopes Ferreira (PPGI-UFSM).

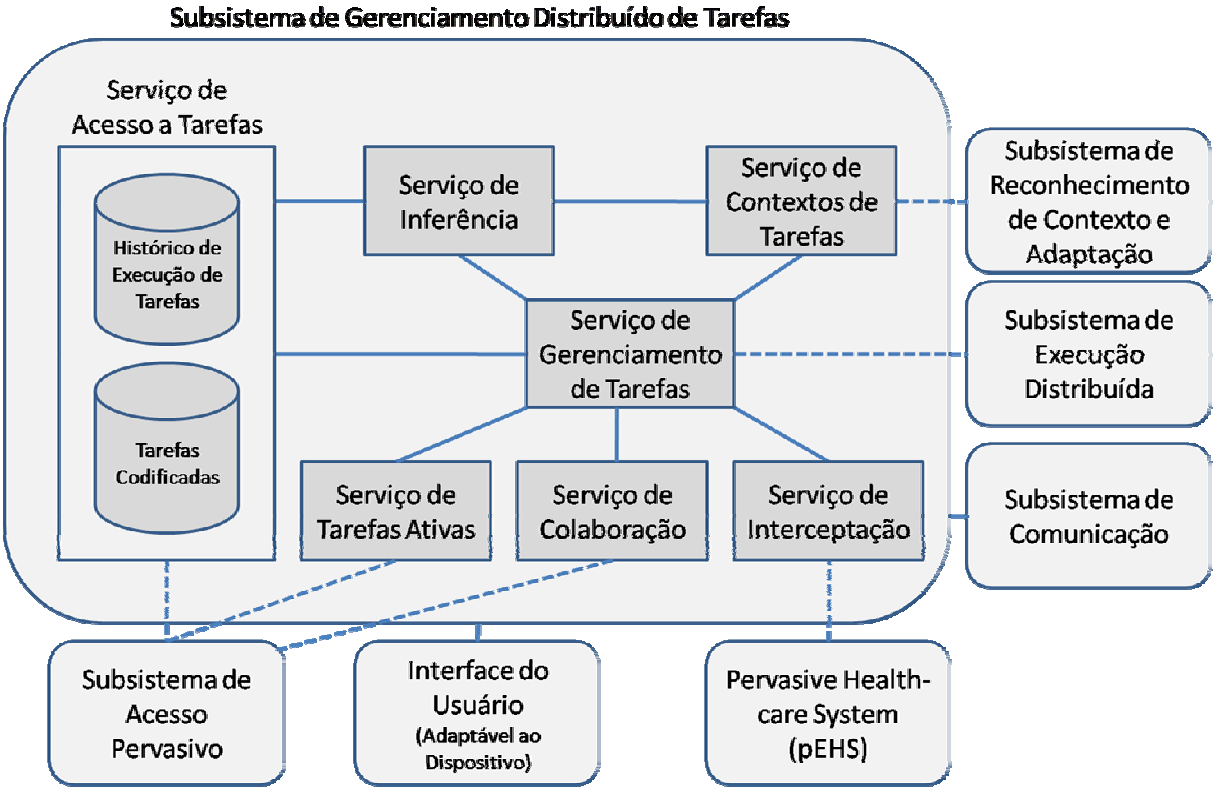

Figura 3.4 - Subsistema de Gerenciamento Distribuído de Tarefas

A Base de Dados de Tarefas Codificadas armazena as tarefas e subtarefas disponibilizadas aos usuários. A representação das tarefas e subtarefas foi modelada através de ontologias (LIBRELOTTO et al, 2008). Dessa forma, será implementado um banco de dados ontológico para armazenar as tarefas. Esse banco de dados será acessado através da APIOntologias. Assim, disponibiliza-se uma forma para que os módulos consultem informações na base de tarefas sem que para isso tenham de conhecer a estrutura de como as informações estão armazenadas.

A associação entre os usuários e as tarefas disponíveis a eles também é responsabilidade do bando de dados que armazena as tarefas. Para isso, o banco deve manter a relação de tarefas disponíveis para cada usuário na forma de cópias da tarefa original. Assim, quando um profissional personaliza (modifica) sua tarefa, as outras cópias não sofrem alterações.

A Base de Dados do Histórico de Execução das Tarefas tem a função de armazenar as informações de execução das tarefas para cada usuário, relativas à forma particular com que eles realizam suas atividades. Essas informações de execução das tarefas são utilizadas pelo Serviço de Inferência para processamento da preferência do usuário, por exemplo. Assim, o Subsistema de Gerenciamento Distribuído de Tarefas pode utilizar essas informações para

otimizar a rotina do profissional, dentro do escopo das atividades diárias auxiliadas pela computação pervasiva.

Além disso, essa base de dados mantém informações quantitativas de utilização das tarefas, como número de vezes que a tarefa foi executada, data da última execução e data da última modificação da tarefa. Com base nessas informações, pode-se gerenciar o ambiente (principalmente a interface gráfica) do usuário para evoluir a usabilidade do sistema.

Serviço de Contexto de Tarefas

A utilização do contexto nas tarefas é fundamental pelo seguinte motivo:

> Não é viável projetar uma aplicação móvel para cada ambiente possível e encarregar o usuário de selecionar e ativar a aplicação adequada ao ambiente corrente. É necessário um comportamento adaptativo – consciente do contexto - da própria aplicação. Codificar uma aplicação pervasiva é especificar qual código deve executar no ambiente em que a aplicação está no momento. Deste objetivo deriva a necessidade de adaptação dinâmica ao contexto (AUGUSTIN; LIMA; YAMIN, 2006, p. 7).

O Serviço de Contexto de Tarefas disponibiliza, através de uma interface de consulta e de um serviço de subscrição (*publish/subscribe)*, as informações de contexto necessárias ou úteis para o disparo e execução das tarefas. Portanto, o Serviço de Contexto de Tarefas é responsável pela integração do SGDT com o Subsistema de Reconhecimento de Contexto do EXEHDA.

Em uma primeira análise, o contexto é formado por usuários, localização, tempo e recursos. Além disso, podem integrar a noção de contexto informações sobre outras pessoas (pacientes e profissionais) próximas ao usuário, e outras informações do ambiente clínico. Devido a essa complexidade de definição do que é contexto para tarefas clínicas, a modelagem do contexto, bem como a do Serviço de Contexto de Tarefas está sendo desenvolvida em outro trabalho de dissertação[3], também ligado ao projeto ClinicSpace.

Serviço de Inferência

O Serviço de Inferência tem a função de processar informações de contexto, disponibilizadas pelo Serviço de Contexto de Tarefas, juntamente com dados históricos de tarefas executadas, objetivando tornar o Subsistema de Gerenciamento Distribuído de Tarefas (SGDT) pró-ativo, inferindo as tarefas que o usuário costuma utilizar em determinado

[3] Trabalho de dissertação do mestrando Tiago Antônio Rizzetti (PPGI - UFSM).

contexto.

A inferência de tarefas que o usuário costuma utilizar em determinado contexto possibilita tanto a sugestão de tarefas ao usuário, melhorando a usabilidade do sistema e reduzindo o impacto do sistema computacional na rotina dele, como o disparo automatizado das tarefas associadas, pelo usuário, a determinados contextos.

O maior problema do disparo automático de tarefas está relacionado à taxa de erro na inferência, o que prejudica a usabilidade do sistema. Por isso, esse serviço exige um estudo[4] mais aprofundado sobre técnicas de inferência comportamental, do qual serão retiradas as regras de inferência usadas pelo SGDT. Além disso, como o Serviço de Inferência é dependente do Serviço de Contexto de Tarefas e da modelagem do contexto, ele não foi implementado no protótipo do SGDT.

Serviço de Colaboração

O Serviço de Colaboração disponibiliza a comunicação e a transferência de tarefas em execução entre os usuários. Esse serviço foi modelado para atender a uma necessidade do ambiente clínico. Segundo Bardram e Christensen (2007), a colaboração é um aspecto fundamental no trabalho de médicos, enfermeiros, e outros profissionais da saúde. Dessa forma, o suporte à colaboração foi projetado com duas funcionalidades: Comunicação e Transferência de Tarefas em Execução (delegação).

Uma forma muito freqüente de colaboração entre os profissionais da saúde é, segundo Bardram e Christensen (2007), a consulta à opinião de outro profissional. Por isso, foi previsto, para o Serviço de Colaboração[5], o suporte à comunicação dos usuários, inicialmente por troca de mensagem, podendo ser expandido para formas mais robustas, como vídeo conferência. Através dessa funcionalidade, os profissionais podem compartilhar informações sobre o paciente, discutir problemas e soluções, solicitar e dar opiniões sobre determinado caso.

Uma possível implementação para essa funcionalidade é a criação de uma tarefa dedicada à comunicação entre os usuários. Essa tarefa não seria exatamente uma tarefa clínica. Ela executaria em *background,* funcionando como um cliente do Serviço de Colaboração, notificando o usuário sobre novas mensagens, e disponibilizando uma interface

[4] O Serviço de Inferência é abordado no trabalho de dissertação do mestrando Marcos Vinícius B. de Souza (PPGI-UFSM).
[5] O Serviço de Colaboração será modelado pelo mestrando Marcelo Lopes Kroth (PPGI-UFSM).

para leitura e escrita de mensagens.

Outra forma de colaboração é a transferência de tarefas em execução entre os profissionais, realizada principalmente na troca de plantão. Para isso, o sistema deve permitir aos usuários transferir suas tarefas não concluídas para outro profissional. Dessa forma, essa funcionalidade foi modelada da seguinte maneira: quando um usuário recebe a tarefa que outro transferiu para ele, o sistema permite ao profissional que recebeu a tarefa escolher entre aceitá-la ou rejeitá-la. Ao rejeitá-la, a tarefa retorna ao emissor. Já aceitando a tarefa, ela será incorporada a sua lista de tarefas interrompidas, podendo ser reiniciada ou retomada do ponto onde foi parada.

Serviço de Interceptação

O Serviço de Interceptação é responsável por fazer o tratamento de eventos gerados pelo sistema pervasivo de informação de saúde (pEHS). Os eventos podem ser críticos, quando exigem a atenção imediata do usuário, ou normais, quando o usuário deve ser apenas notificado.

Eventos críticos correspondem a algum tipo de urgência ou emergência monitorada pelo sistema pEHS, como sinais vitais dos pacientes. Já eventos normais são disparados em casos nos quais o pEHS somente precisa notificar o usuário.

Esse serviço é bastante dependente do pEHS, pois somente após as funcionalidades desse estarem bem definidas é que se poderá classificar seus eventos e fazer a integração com o Serviço de Interceptação. Essa integração envolve, além do tratamento de eventos, a troca de informações sobre o usuário, para que o sistema pEHS tenha conhecimento se ele está usando o sistema computacional ou não.

Para tratar um evento, o Serviço de Interceptação deve: (i) interagir com o Serviço de Gerenciamento de Tarefas (SGT)[6] para interromper as tarefas em execução, no caso de o evento ser crítico; ou (ii) notificar o usuário, quando o evento é normal.

Sempre que um evento normal é interceptado, o Serviço de Interceptação dispara uma rotina que mostra, na interface do usuário, a notificação do evento, com informações suficientes para que o usuário possa decidir se deve dar atenção ao evento, ou se devem continuar as tarefas que estava realizando. Assim, se o usuário optar por dar atenção ao evento, o Serviço de Interceptação solicita ao SGT a interrupção das tarefas que estavam

[6] Trabalho de dissertação do mestrando Giuliano Lopes Ferreira (PPGI-UFSM).

sendo executadas e o disparo da tarefa associada ao evento.

Entretanto, se o evento interceptado for crítico, o Serviço de Interceptação, ao mesmo tempo em que gera o alerta na interface do usuário, solicita ao SGT a interrupção das tarefas que estão em execução e o disparo de uma tarefa que auxiliará o usuário a tratar o evento. Dessa forma, os dois tipos de eventos são tratados, pelo Serviço de Interceptação, de maneira semelhante, com a diferença que o tratamento de um evento normal pode ser postergado pelo usuário, caso ele esteja ocupado no momento em que o evento é interceptado.

Em ambos os casos, quando o tratamento do evento termina, o Serviço de Interceptação notifica o SGT para que seja retomada a execução das tarefas que haviam sido interrompidas.

Para aperfeiçoamento desse serviço, sugere-se a modelagem de uma hierarquia de prioridades de eventos, categorizando-os em níveis, de modo que um evento de menor prioridade não possa interromper o tratamento de um de maior prioridade. Essa categorização depende de um maior aprofundamento no estudo sobre urgências e emergências clínicas e hospitalares.

Serviço de Gerenciamento de Tarefas

O Serviço de Gerenciamento de Tarefas (SGT) é o núcleo do Subsistema de Gerenciamento Distribuído de Tarefas (SGDT), e é responsável por gerenciar a execução das tarefas dos usuários. Esse serviço mapeia as tarefas (na forma ontológica, definida pelo usuário) em aplicações pervasivas (objetos) do EXEHDA. Para isso, o SGT: utiliza o Serviço de Acesso a Tarefas para buscar e instanciar a tarefa que será executada; obtém os serviços do *middleware* necessários à execução; configura e dispara a execução da tarefa. Cada subtarefa da tarefa é executada seqüencialmente, utilizando dados produzidos pelas anteriores, dados pré-configurados na personalização da tarefa e informações obtidas do Serviço de Contexto de Tarefas.

O SGT também é responsável pela migração das tarefas. Quando o usuário troca de dispositivo, sua sessão é fechada no dispositivo que ele estava usando e restaurada no novo dispositivo. Dessa forma, antes da sessão ser fechada, o SGT interrompe as tarefas que estavam em execução e executa a migração delas para o dispositivo onde se encontra o Serviço de Tarefas Ativas. No momento em que a sessão é restaurada, o SGT informa a interface do usuário sobre as tarefas interrompidas. Se ele solicitar a retomada de uma tarefa, o SGT dispara a migração dela para o hospedeiro atual do usuário e inicia a execução do

ponto onde ela havia sido interrompida. Porém, antes da retomada, o Serviço de Contexto é consultado para verificar se houve mudanças no contexto utilizado pela tarefa, o que pode ocasionar adaptações na execução da tarefa.

Durante uma adaptação, provocada por uma migração, a subtarefa acessa o Serviço de Contexto de Tarefas para obter informações sobre o dispositivo. Então, o Serviço de Acesso a Tarefas é usado para obter a melhor interface gráfica para esse dispositivo, caso ela possua uma. Ou a informação do dispositivo é usada para instanciar a interface gráfica da aplicação do sistema pervasivo de informação de saúde (pEHS).

Serviço de Tarefas Ativas

O Serviço de Tarefas Ativas é responsável por manter informações sobre as tarefas ativas de cada usuário. Uma tarefa é considerada ativa quando ela foi inicializada pelo Serviço de Gerenciamento de Tarefas (SGT) e ainda não foi concluída nem cancelada. Assim, esse serviço disponibiliza uma API (através de requisições HTTP) para consultar as tarefas ativas de determinado usuário, bem como, adicioná-las a sua lista, removê-las e atualizá-las.

Sistema pEHS

Para iniciar a modelagem da arquitetura ClinicSpace, supõe-se a existência de um sistema pervasivo de informação de saúde denominado pEHS (*pervasive Electronic Health-care System*)[7], que é um software de gerenciamento de informações de saúde com características de pervasividade, como migração da sessão do usuário e adaptação ao dispositivo usado. Determinadas subtarefas interagem com o pEHS para ativar aplicações específicas que trabalham como os dados do sistema clínico. Para isso, o sistema pEHS deve disponibilizar aplicações que possam ser invocadas com passagem de parâmetros, o que permite às subtarefas e o reconhecimento automático de contexto pré-configurar as aplicações.

As características pervasivas do sistema pEHS permitem que aplicações não terminadas possam ser retomadas pela subtarefa, quando uma tarefa interrompida estiver sendo retomada. Assim, qualquer tarefa pode ser interrompida e reiniciada ou retomada

[7] Esse tema está sendo estudado em outra frente de do projeto ClinicSpace, pelos mestrandos Caroline Vicentini e Alencar Machado (PPGI-UFSM).

diretamente pela interface do usuário, sem que ele necessite conhecer o pEHS.

Portanto, o Sistema pEHS deve implementar, entre outras funcionalidades: (i) migração de suas aplicações, permitindo que uma aplicação seja interrompida e retomada pela subtarefa; (ii) armazenamento temporário das informações que estão sendo utilizadas pela aplicação, para que a subtarefa, ao ser interrompida, possa armazenar as informações parciais da aplicação que está sendo usada e recuperá-las quando a tarefa for retomada; (iii) adaptação da interface e das informações de suas aplicações, para que as aplicações possam acompanhar o usuário, quando ele trocar de dispositivo (adaptação sensível ao contexto).

Interface gráfica do usuário

A interface gráfica para o usuário será uma extensão da interface de criação e personalização de tarefas apresentada nesse trabalho. Além do foco atual, a interface gráfica disponibilizada ao usuário servirá para o gerenciamento das tarefas e também irá utilizar as técnicas providas pela Computação orientada ao usário-final.

O Subsistema de Gerenciamento Distribuído de Tarefas (SGDT), através do Serviço de Gerenciamento de Tarefas, disponibiliza métodos para que o usuário possa gerenciar a execução de suas tarefas. Dessa forma, a interface gráfica de gerenciamento das tarefas deve fazer uso desses métodos para informar o usuário sobre tarefas disponíveis e tarefas ativas, e para disponibilizar o controle sobre elas (disparar, interromper, retomar, cancelar).

Além disso, a interface gráfica deve interagir com outros serviços do SGDT. O Serviço de Interceptação disponibiliza informações do pEHS e a interface deve monitorar essas informações e mostrá-las ao usuário. O Serviço de Colaboração disponibiliza informações sobre comunicação e transferência de tarefas, as quais devem ser repassadas aos usuários pela interface gráfica.

Já a interação da interface gráfica com o Serviço de Contexto de Tarefas, diferentemente das anteriores, não resulta em novas informações para o usuário. Essa interação tem o objetivo de obter informações do contexto para adaptar a própria interface.

Portanto, a interface gráfica deve ser tratada como uma aplicação pervasiva que utiliza tanto serviços do Subsistema de Gerenciamento Distribuído de Tarefas, como dos outros subsistemas do EXEHDA. Os primeiros sendo usados para permitir que o usuário gerencie suas tarefas; os últimos, para gerenciar sua própria execução no ambiente pervasivo. Sendo assim, essa aplicação deve ser implementada de modo semelhante a um agente móvel, com a função de “seguir” o usuário, carregando as informações necessárias (sessão), e instanciando

a “tela gráfica” adequada ao dispositivo em uso.

4 ARQUITETURA DE SOFTWARE PARA EDIÇÃO DE TAREFAS CLÍNICAS PELO USUÁRIO-FINAL

Este capítulo aborda os passos seguidos para a modelagem das tarefas clínicas e a definição da ferramenta-piloto de edição dessas tarefas pelo usuário-final.

4.1 Interface de interação com o usuário

Para que o profissional clínico possa criar e editar tarefas e fluxos de execução, implementou-se a Interface de Edição de Tarefas. Essa interface foi modelada considerando mecanismos providos pela programação Orientada ao Usuário-final (*End-User Programming*) com o intuito de disponibilizar uma interface simples para a utilização dos profissionais clínicos. Assim, facilita-se aos profissionais a utilização do sistema, já que os mecanismos utilizados abstraem detalhes de implementação de sistemas, dos quais o usuário não possui conhecimento.

A arquitetura ClinicSpace inova ao suportar um importante aspecto relacionado ao trabalho de usuários clínicos que se refere ao fato de que os profissionais da área possuem a sua própria forma de realizar as atividades referentes aos cuidados clínicos de seus pacientes (individualização, personalização). Profissionais de uma mesma especialidade médica apresentam diferenças na forma com que executam as suas atividades clínicas. Com a disponibilização de uma interface amigável, os profissionais clínicos podem definir a sua própria forma de executar as tarefas clínicas e, assim, aproximar o sistema ao ambiente profissional em que atuam e que tanto conhecem.

Na interface de interação com o usuário, as tarefas (subtarefas, tarefas e fluxos de execução) são representadas através de ícones que remetem à funcionalidade da tarefa. Caso o profissional tenha dúvida sobre qual elemento o ícone representa, basta a ele posicionar o cursor do mouse sobre o ícone que irá ser visualizado o nome da subtarefa, tarefa ou fluxo. Essas tarefas podem ser arrastadas sobre a janela principal de edição de tarefas para serem

dispostas da forma que o usuário julgar necessário. O primeiro esboço da Interface de Edição de Tarefas é ilustrado na Figura 4.1. Vale ressaltar que essa interface foi implementada para teste e comprovação da viabilidade da proposta. Novas melhorias na Interface de Edição de Tarefas estão previstas para serem implementadas no decorrer do projeto ClinicSpace a fim de tornar a interface realmente intuitiva.

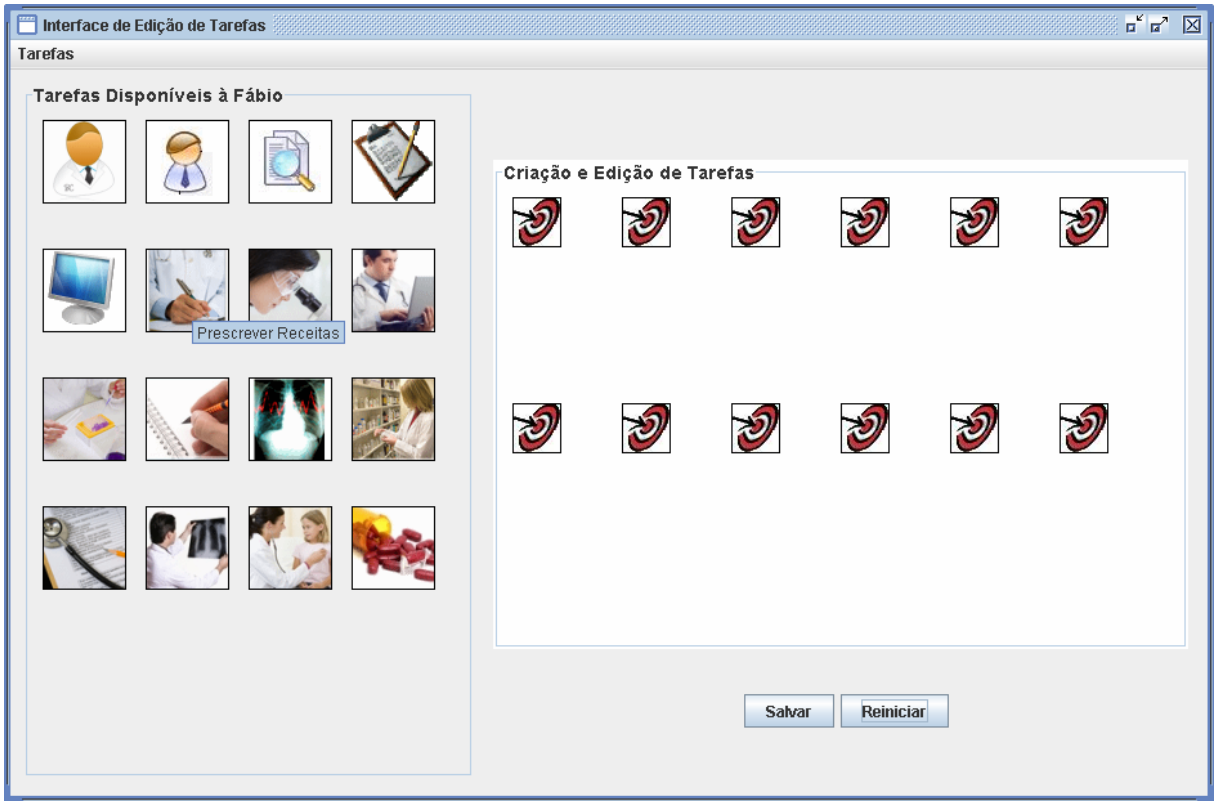

Figura 4.1 – Primeiro esboço da Interface de Edição de Tarefas

Na parte esquerda da interface (grupo identificado por "Tarefas Disponíveis ao usuário") estão disponíveis os ícones que representam as subtarefas, tarefas e fluxos de execução que determinado usuário tem acesso. Vale ressaltar que, em alguns casos, não é possível representar todos os elementos na interface, mas, caso o usuário deseje utilizar algum outro elemento, bastará acessar o menu "Tarefas" para buscar o elemento desejado. A partir dos elementos (subtarefas, tarefas e fluxos) disponibilizados na interface, o profissional clínico poderá editá-los e/ou criar outras tarefas e fluxos que julgar necessário. Em ambos os casos, o profissional clínico utilizará o local na interface identificado como "Criação e Edição de Tarefas" para (re) organizar os ícones conforme desejado.

Como mencionado anteriormente, o usuário pode utilizar a Interface de Edição de Tarefas

para definir fluxos de execução de tarefas. Para tal composição de fluxo, basta ao usuário clicar-e-arrastar os ícones referentes aos elementos desejados ao local de Criação e Edição de Tarefas. Ao arrastar esses elementos, a Interface de Edição de Tarefas altera o título do local de Criação e Edição de Tarefas para "Criando uma nova tarefa" para, assim, sinalizar ao usuário que uma nova tarefa (tarefa ou fluxo) esta sendo formada. Os elementos são inseridos no fluxo na mesma ordem que o usuário insere os ícones, a não ser que o usuário arraste um novo elemento e posicione esse entre dois elementos já existentes. Nesse caso, os elementos serão reorganizados desfazendo-se o acoplamento anterior e formando os novos acoplamentos. Caso o profissional tenha adicionado os elementos em uma determinada ordem, ele poderá reorganizá-las para definir a nova ordem desejada.

Supondo o caso em que o profissional clínico deseja compor um novo fluxo de execução. Para tal ele arrastou, para o local apropriado, o ícone que representa a tarefa "Revisar os problemas do paciente", a tarefa "Adicionar nota diária sobre as condições do paciente" e, por fim, a tarefa "Prescrever receita". O fluxo formado até esse momento está ilustrado na Figura 4.2. Caso o profissional deseja reorganizar os elementos invertendo a ordem das tarefas "Adicionar nota diária sobre as condições do paciente" e "prescrever receita", a Interface de Edição de Tarefas realiza as verificações necessárias para a troca (verificação das regras de associação descritas na seqüência do trabalho) e realiza a reordenação desejada pelo profissional clínico. A ferramenta emite um alerta ao profissional sinalizando que a reorganização foi realizada (Figura 4.3). Vale ressaltar que se não for possível tal reorganização, a ferramenta avisa ao profissional clínico e desfaz a operação.

Em termos de implementação, um fluxo de execução é considerado uma nova tarefa, mas isso é transparente ao usuário. Os fluxos também podem ser editados como tarefas e são armazenados no ambiente do usuário, até que esses sejam desmembrados ou excluídos, deixando de existir. As tarefas são duplicadas, ao se compor um fluxo de execução, passando a existir uma cópia da tarefa no fluxo e a original isolada, a qual poderá ser utilizada pelo usuário para ser executada independente de outras, ou usada para compor algum outro fluxo.

O usuário pode compor novas tarefas, ou fluxos de execução, utilizando outras tarefas e fluxos já existentes. A partir das tarefas existentes, o usuário pode ir editando-as, e as alterações se refletem apenas na nova tarefa que está sendo criada. Ao final desse processo de criação, uma nova tarefa é gerada, sendo essa composta pelas subtarefas, tarefas e fluxos de execução que o usuário definiu para ela. Vale ressaltar que todas as operações de edição e criação de novas tarefas são validadas pelo sistema para atestar se determinada tarefa é correta. Para isso são testadas e validadas as regras de associação de tarefas e subtarefas (as

regras serão detalhadas no item 4.3.1).

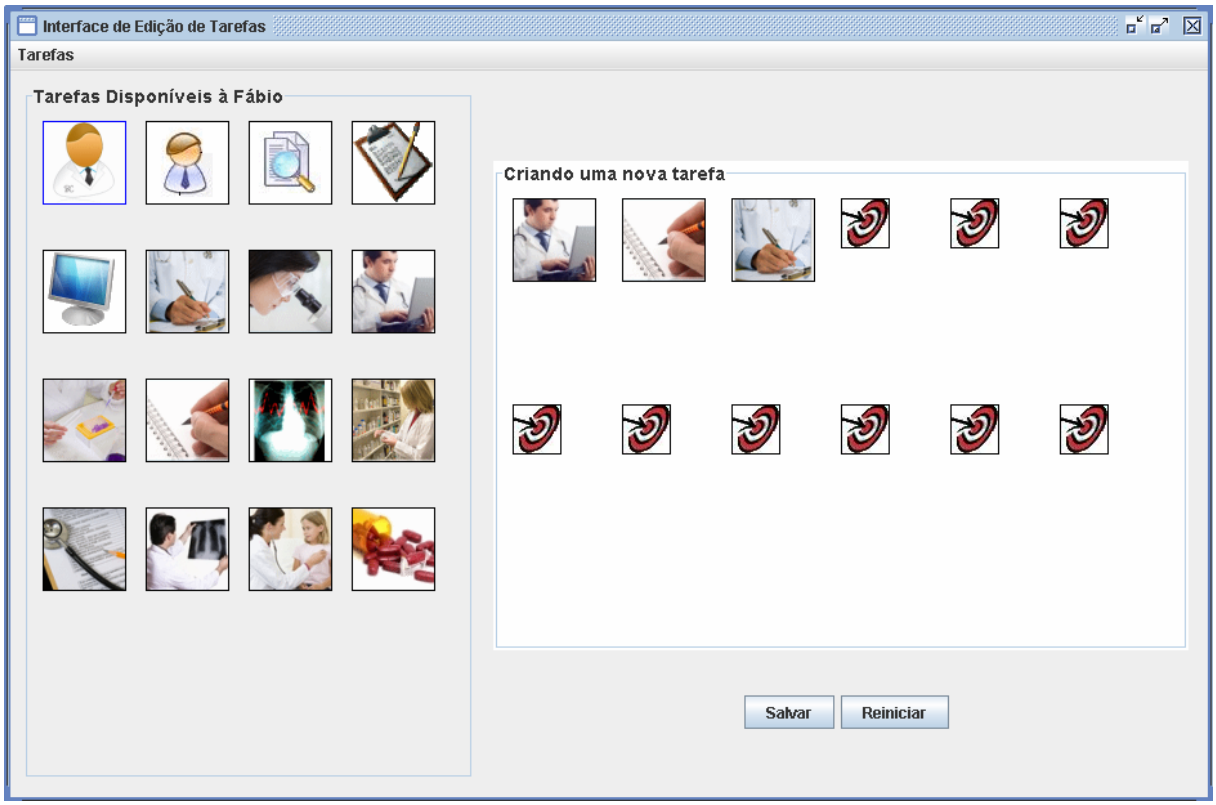

Figura 4.2 – Criando um novo fluxo de execução

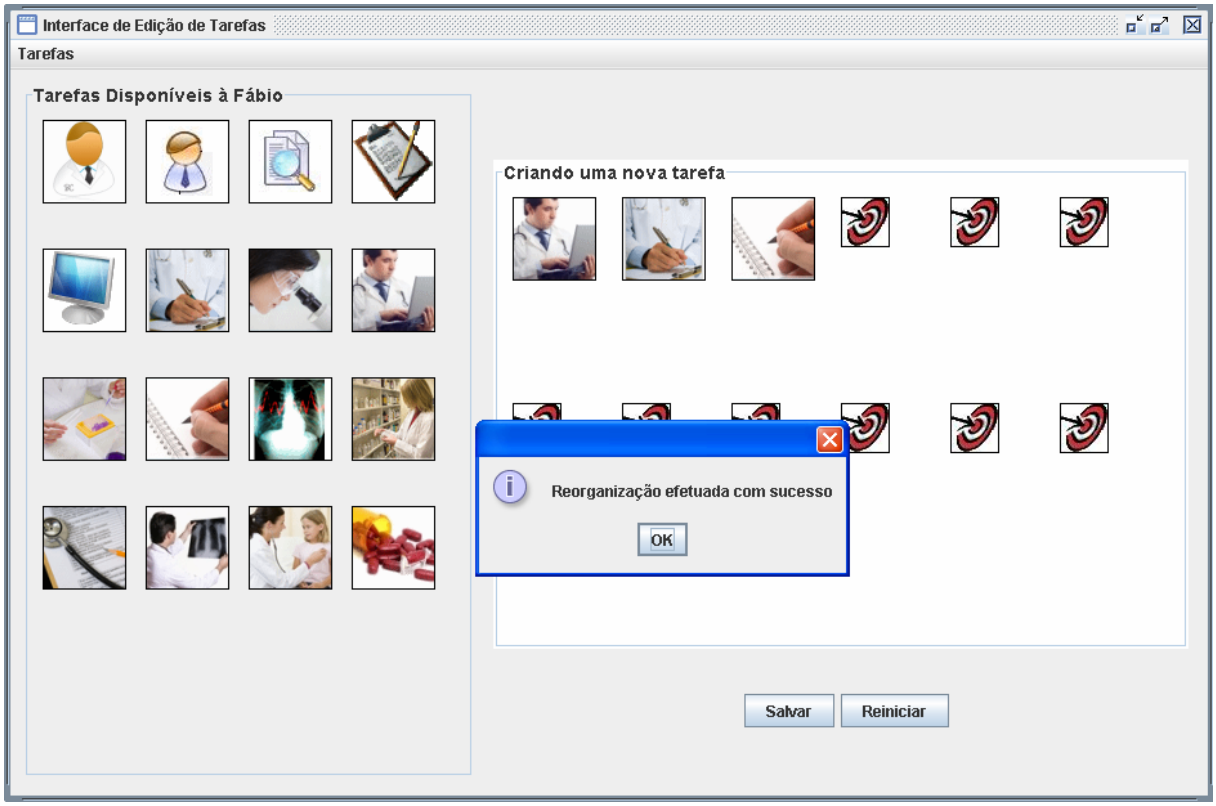

Figura 4.3 – Reorganizando os elementos do fluxo

4.2 Modelagem de tarefas clínicas

Nos ambientes hospitalares, os profissionais clínicos executam inúmeras e diversificadas atividades diariamente. Objetivando modelar as tarefas clínicas e tratar a diversidade de atividades executadas pelos profissionais, encontrou-se uma pesquisa realizada por Laerum e Faxvaag (2004) a qual definiu atividades clínicas realizadas em períodos de freqüência diária, semanal, mensal ou esporádica, e adotou-se a classificação de tarefas baseada na proposta de Kumar (2003). As tarefas classificam-se em:

- Tarefas Clínicas de Diagnóstico. Categoria composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para determinar a natureza, ou identidade, da doença ou enfermidade. Não são inclusos nessa categoria procedimentos realizados com amostras em laboratórios;
- Tarefas Clínicas de Tratamento. Categoria composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para melhorar as condições dos pacientes, e também outras quaisquer atividades executadas para tratar doenças, enfermidades e ferimentos;
- Tarefas Laboratoriais. Categoria composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para determinar a composição, qualidade ou concentração de espécime. Nessa categoria estão inclusas atividades que medem o tempo e taxa das reações e, também, as atividades que se relacionam com o sistema laboratorial para a obtenção de informações de exames como análises laboratoriais, raios-x, etc;
- Tarefas Administrativas. Categoria composta por atividades relacionadas à manutenção de informações clínicas dos pacientes nos sistemas. Entende-se por manutenção quaisquer atividades relacionadas a operações de inclusão, modificação, atualização e exclusão de informações clínicas a serem mantidas e organizadas nos sistemas computacionais.

Como o projeto prevê a implementação de uma ferramenta-protótipo para a definição e gerenciamento das tarefas clínicas a serem executadas no ambiente hospitalar, identificou-se um conjunto mínimo de tarefas clínicas para serem disponibilizadas para os usuários. Os usuários clínicos poderão construir novas tarefas e fluxos de execução conforme a sua preferência e necessidade a partir das tarefas, e das subtarefas que as compõem, do conjunto mínimo.

Para identificação do conjunto mínimo de tarefas clínicas, analisaram-se os resultados publicados da pesquisa de Laerum e Faxvaag (2004), realizada com profissionais clínicos (especialmente médicos) para a identificação e definição das principais tarefas clínicas executadas pelos profissionais dessa área. O estudo propõe um conjunto de vinte e quatro (24) atividades realizadas em ambientes hospitalares, sendo que essas foram validadas por profissionais clínicos. O estudo ainda considerou alguns aspectos como relevância da atividade e periodicidade com que ela é executada.

Para a implementação do protótipo da interface e edição de tarefas, utilizou-se uma consolidação das atividades executadas com maior freqüência ao dia e, assim, chegou-se ao número das onze (11) tarefas clínicas para compor o conjunto mínimo de tarefas. Vale ressaltar que o conjunto mínimo de tarefas foca unicamente o trabalho realizado por médicos sendo esse justamente o público-alvo ao qual se destina o protótipo inicial a ser desenvolvido no projeto em que esse trabalho está inserido. As tarefas clínicas pertencentes ao conjunto mínimo, juntamente com sua definição e categoria, são apresentadas no Quadro 4.1. Nota-se que o foco das tarefas é na parte clínica, diferentemente dos atuais registros eletrônicos de gestão hospitalar (EHR) que visam à parte administrativa e financeira (VICENTINI, 2009).

As tarefas clínicas pertencentes ao conjunto mínimo de tarefas foram modeladas através da definição de um conjunto de subtarefas, mapeadas para serviços e aplicações computacionais relativas ao sistema eletrônico de saúde adotado e ao gerenciamento do ambiente pervasivo. A partir dessas tarefas, e nas subtarefas que as compõem, o usuário pode usá-las e reutilizá-las para a criação de outras tarefas e fluxos de execução.

Tendo em mente que as tarefas e fluxos são compostos pelos usuários, e as subtarefas são definidas via programação (serão mapeadas a serviços computacionais (FERREIRA et al, 2009)), buscou-se identificar quais são as subtarefas que compõem cada uma das tarefas do conjunto mínimo de tarefas. Como a arquitetura ClinicSpace pode ser comparada a um agente de gerenciamento que facilita a entrada de dados no sistema de saúde utilizado, cada subtarefa é associada à pelo menos uma aplicação do sistema eletrônico de saúde adaptado a essa arquitetura - pEHS (VICENTINI, 2009) - ou a serviços do *middleware* de gerenciamento do ambiente (FERREIRA et al, 2009) que deverá ser invocado para a interação com o usuário. Além disso, o sistema, através da passagem de parâmetros, poderá informar previamente alguns dados para a aplicação, minimizando a entrada de dados por parte do usuário (RIZZETTI, 2009).

Para exemplificar a definição das subtarefas que compõe uma tarefa clínica, utiliza-se a tarefa “Requisitar Análises Laboratorias” (ver figura 4.4). Para o suporte a essa tarefa é

necessária a execução de um conjunto de aplicações e serviços (subtarefas). Nesse caso, o conjunto de subtarefas clínicas que compõe a tarefa é:

- **Identificar Profissional.** Subtarefa utilizada para a identificação do profissional clínico que, nesse caso, irá fazer a requisição da análise laboratorial. Note que o sistema de gerenciamento das tarefas (FERREIRA et al, 2009) fará a identificação automaticamente, usando o reconhecimento de contexto (RIZZETI, 2009);
- **Identificar Paciente.** Subtarefa utilizada para a identificação do paciente sobre o qual será efetuado a análise;
- **Preencher Requisição Laboratorial.** Subtarefa que propicia o preenchimento das informações relativas à requisição, o encaminhamento da requisição e o registro dessas no sistema pervasivo de informações de Saúde (pEHS).

Figura 4.4 – Exemplo de tarefa com as subtarefas que a compõem

De acordo com esse exemplo, percebe-se os elementos necessários para o suporte à tarefa "Requisitar Análises Laboratoriais". Com a utilização de elementos em alto-nível, é abstraído para o profissional clínico os serviços e as aplicações necessárias para a composição das tarefas que eles utilizam para o cuidado e tratamento de seus pacientes. O profissional clínico pode utilizar essa abstração para a definição de suas tarefas e, ainda, adequá-la a sua maneira de realizar determinada tarefa. Ressalta-se que, para isso, o profissional não precisa ter conhecimentos sobre informática, sobre programação e sobre as aplicações computacionais a serem utilizadas. O sistema desenvolvido no projeto ClinicSpace, e no qual o presente trabalho é inserido, é quem tem o completo entendimento da forma com que deve associar as abstrações disponíveis ao usuário (subtarefas, tarefas e fluxos de execução) com as aplicações e serviços que representam cada abstração (FERREIRA et al, 2009). Além disso, o profissional poderá utilizar as abstrações já definidas

.	Tarefa	Definição	Classificação
1	Revisar os problemas do paciente	Obter informações suficientes para formular os principais problemas dos pacientes para, então, poder realizar ou requisitar novas investigações ou tomar decisões clínicas.	Diagnóstico
2	Procurar informações específicas em registros do paciente	Procurar específicas e limitadas quantidades de informações sobre o paciente nos registros do paciente (EHR).	Diagnóstico
3	Obter os resultados de novos testes ou investigações	Identificar e obter resultados de investigações já executadas e analisadas e que ainda não tenham sido acessadas por um clínico.	Diagnóstico
4	Adicionar notas diárias sobre as condições do paciente	Escritas sobre avaliações das condições do paciente e, assim, formular notas de progresso do estado do paciente.	Tratamento Diagnóstico
5	Requisitar análises clínicas em laboratórios de análises clínicas	Requisitar uma ou diversas análises clínicas em laboratórios bioquímicos. A reserva do teste pode ser feita pelo médico ou outro profissional.	Diagnóstico
6	Obter resultados de exames clínicos	Obter resultados novos, ou velhos, de exames clínicos executados em laboratórios bioquímicos.	Laboratorial
7	Requisitar raios-X, ultra-som e tomografias computadorizadas	Requisitar raios-X, ultra-sons ou tomografias computadorizadas, e ainda prover um sumário clínico do paciente. A reserva da investigação pode ser feita pelo médico ou outro profissional.	Diagnóstico
8	Obter resultados de raios-X, ultra-som e tomografias computadorizadas	Obter resultados novos, e velhos, de raios-X, ultra-sons ou tomografias computadorizadas.	Laboratorial
9	Requisitar tratamentos	Requisitar procedimentos de tratamentos como medicamentos, cirurgias, etc. a serem executadas no hospital e que, geralmente, não sejam administradas pelos pacientes.	Tratamento
10	Prescrever receitas	Receitar medicamentos (ou outros tipos de tratamentos auto-administráveis) para o paciente comprar, coletar ou administrar. A ordem (receita) deve incluir instruções para o paciente sobre como e quando o tratamento deve ser aplicado.	Tratamento
11	Registrar códigos para diagnósticos ou procedimentos executados	Executar seleções em vários sistemas de classificação para os procedimentos clínicos executados ou para diagnósticos, e documentar a seleção.	Administrativa

Quadro 4.1 – Conjunto Mínimo de Tarefas

pelo sistema e as que ele definiu para a criação e definição de outras que julgar necessário. Com isso, permite-se o reaproveitamento de subtarefas, tarefas e fluxos de execução já definidos anteriormente. Um exemplo de reaproveitamento de tarefas é o fluxo de execução ilustrado na Figura 4.5 que é formado por três tarefas pertencentes ao conjunto mínimo de tarefas ("**Requisitar Análises Laboratoriais**", "**Obter resultados de exames laboratoriais**", "**Adicionar notas diárias**").

A composição das outras tarefas clínicas pertencentes ao conjunto mínimo de tarefas encontra-se disponível no Apêndice A.

Figura 4.5 – Exemplo de fluxo de execução

4.3 Detalhes de Implementação da Interface de Edição de Tarefas pelo Usuário-Final

Os aspectos relativos à implementação da Interface de Edição de Tarefas são abordados na seqüência.

4.3.1 Regras de Associação

Para a criação de tarefas e fluxos de execução, foi necessária a definição de regras para a verificação da associação dos elementos (validação). Essas regras foram criadas para a certificação de que a tarefa, ou fluxo, realmente represente um fluxo válido (correto) de execução de serviços e aplicações (abstraídas para o usuário através das subtarefas). Assim,

pode-se garantir que os serviços e aplicações poderão ser executados no sistema ClinicSpace, na sequência definida na tarefa ou no fluxo de execução de tarefas. Além disso, garante-se que as informações oriundas de cada elemento são propagadas corretamente ao longo da tarefa ou fluxo.

No modelo proposto, tanto tarefas quanto fluxos de execução são diferentes níveis de abstração das subtarefas, já que esses elementos de mais alto nível são formadas por subtarefas que, posteriormente, serão traduzidas para serviços do sistema (FERREIRA et al, 2009). Dessa forma, a verificação da associação de diferentes elementos deve considerar se as subtarefas podem, ou não, ser associadas a elementos tratáveis pelo sistema ClinicSpace.

Para isso, foram identificadas quatro diferentes categorias para a classificação das subtarefas. Uma subtarefa obrigatoriamente pertence a uma categoria e, a partir dessas categorias, as regras de associação são implementadas. As categorias são:

- Subtarefas de **Identificação**. Nessa categoria são incluídas as subtarefas que permitem a identificação de profissionais e pacientes;
- Subtarefas de **Busca**. Nessa categoria são incluídas todas as subtarefas responsáveis por busca de informações de profissionais e pacientes. As informações a serem buscadas geralmente estão armazenadas no Sistema EHR, onde são armazenadas as informações clínicas dos pacientes;
- Subtarefas de **Preenchimento.** Nessa categoria são incluídas as subtarefas responsáveis pelo registro de informações dos pacientes no Sistema EHR. Engloba ainda as subtarefas que permitem o preenchimento de requisições de exames, análises, encaminhamentos, etc.;
- Subtarefas de **Visualização.** Nessa categoria são incluídas as subtarefas que permitem a visualização de informações dos pacientes. Essas subtarefas servem para a visualização de informações armazenadas no Sistema de Informações dos Pacientes e também podem ser informações providas por subtarefas de outras categorias que são executadas anteriormente.

A partir da identificação das categorias foram definidos níveis para cada uma delas (Quadro 4.2), com o intuito de facilitar o processo de verificação das regras de associação.

Categoria	Identificação	Busca	Preenchimento	Visualização
Nível	1	2	2	3

Quadro 4.2 – Níveis das categorias de subtarefas.

Os níveis das subtarefas são considerados para a composição de tarefas e fluxos de

execução, já que dois elementos só poderão ser associados caso a regra definida seja satisfeita. Para que seja possível a associação de uma **subtarefa A** a uma outra **subTarefa B**, o nível da categoria que a **subtarefa B** pertence deve ser maior ou igual ao nível da categoria da **subtarefa A**. A **subtarefa B** não precisa ter nível maior que a precedente, caso essa primeira seja da categoria de Visualização. Caso a subtarefa seja dessa categoria (visualização), será possível ter após ela (na sequência de subtarefas que compõe a tarefa ou fluxo de execução) uma outra subtarefa de nível inferior.

As regras citadas para o tratamento da associação de elementos são verificadas no momento de criação e edição de uma tarefa ou fluxo. Nos casos de criação ou edição de uma tarefa, a cada subtarefa adicionada são verificadas se as regras são satisfeitas. No momento de adição de uma nova subtarefa algumas verificações devem ser feitas. São elas:

- (i) Caso exista uma subtarefa anterior ao local onde a nova subtarefa será adicionada, deve ser verificada se essa permite a associação com a subtarefa que se deseja adicionar no fluxo;
- (ii) Caso existam subtarefas posteriores ao local onde está sendo adicionado a subtarefa, deve ser verificada a associação da subtarefa que está sendo adicionada com a subtarefa posterior ao local da adição.

Exemplificando as duas regras acima: considere que uma determinada tarefa possua as subtarefas A e B. Caso o usuário deseja adicionar uma outra subtarefa C entre as duas anteriores, deverá ser verificado se a subtarefa A permite a associação com a subtarefa C (i). Além disso, a outra associação a ser verificada é a associação da subtarefa C com a subtarefa B (ii).

As tarefas do conjunto mínimo de tarefas foram modeladas levando em conta as Regras de Associação citadas acima. A partir dessas tarefas e subtarefas, o usuário clínico pode criar outras tarefas e, também, definir fluxos de execução para elas. Vale ressaltar que um fluxo de execução é na verdade uma outra abstração de tarefa para o usuário. Isso é válido já que tanto uma tarefa quanto um fluxo é uma sequência de execução de subtarefas, onde as informações devem ser propagadas entre elas, para a realização de um determinado trabalho.

Como dito anteriormente, o usuário pode utilizar as tarefas já modeladas ou, então, as que ele criou para criar fluxos de execução para elas. Assim como no caso das tarefas, deve ser criado um mecanismo para a verificação se o fluxo é válido. Um fluxo pode conter subtarefas, tarefas e outros fluxos. Esses diferentes elementos são considerados diferentes níveis de abstração dos serviços e aplicações do sistema ClinicSpace (subtarefas). Assim, a política para a associação de tarefas e fluxos é baseada nas regras de associação de subtarefas

citadas anteriormente.

Para que uma `tarefa A` seja associada a uma `tarefa B` para a composição de um fluxo, as subtarefas que compõem essas tarefas devem ser compatíveis para que as informações possam ser propagadas e, assim, ter a composição de um fluxo válido. No momento, a política definida é que a ultima subtarefa da `tarefa A` deve ser compatível com a primeira subtarefa da `tarefa B`. A mesma lógica vale para o caso de fluxos de execução dentro de outros fluxos. A verificação da associação sempre irá verificar se o último elemento da primeira tarefa, ou fluxo, é compatível com o primeiro elemento da segunda tarefa ou fluxo. A Figura 4.6 ilustra um determinado fluxo de execução formado por três tarefas (Tarefa A, Tarefa B, Tarefa C). Para que esse fluxo de execução seja considerado válido, a SubTarefa3 que é a última subtarefa a compor a `Tarefa A` deve ser compatível com a `SubTarefa1`. Ainda, a `subTarefa4` que é a última subtarefa a compor a `Tarefa B`, deve ser compatível com a `SubTarefa2` que é a primeira subtarefa a compor a `Tarefa C`. Assim, é garantido que a propagação das informações será possível do início ao fim do fluxo de execução.

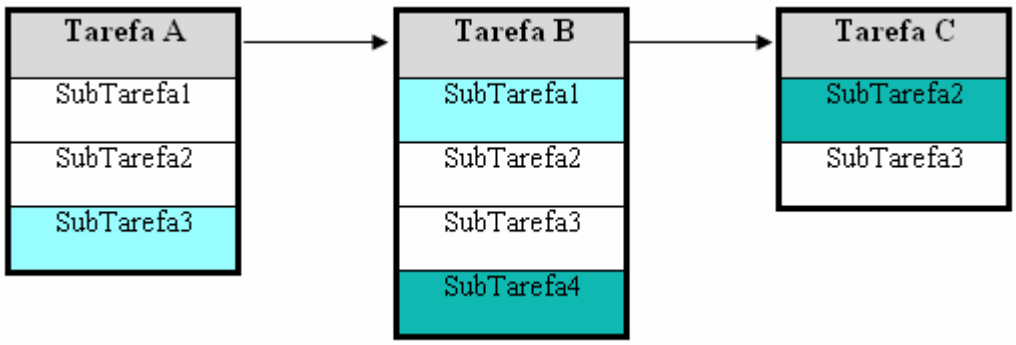

Figura 4.6 – Fluxo de execução e a compatibilidade de seus elementos

Essas verificações também são realizadas no momento em que o profissional clínico deseja reorganizar uma tarefa ou fluxo que esteja criando. A reorganização dos elementos só deve ser permitida após a verificação das políticas e regras de associação citadas anteriormente. O processo de reorganização é explicado com maiores detalhes no capítulo 5, Estudo de Caso.

4.3.2 Estrutura do Módulo Edição de Tarefas

O módulo de Edição de Tarefas é composto por diversas classes que interagem entre si para juntas dispor as funcionalidades esperadas do módulo. No conjunto de classes que compõem o módulo de Edição de Tarefas existem classes que dizem respeito ao tratamento e gerenciamento dos elementos gráficos que são dispostos na tela para serem arrastados pelo usuário. Além dessas, outras classes são responsáveis pela modelagem das tarefas clínicas e das políticas que são utilizadas para permitir, ou não, a edição e composição de novas tarefas. As classes do módulo, representadas na figura 4.7, são explicadas na sequência.

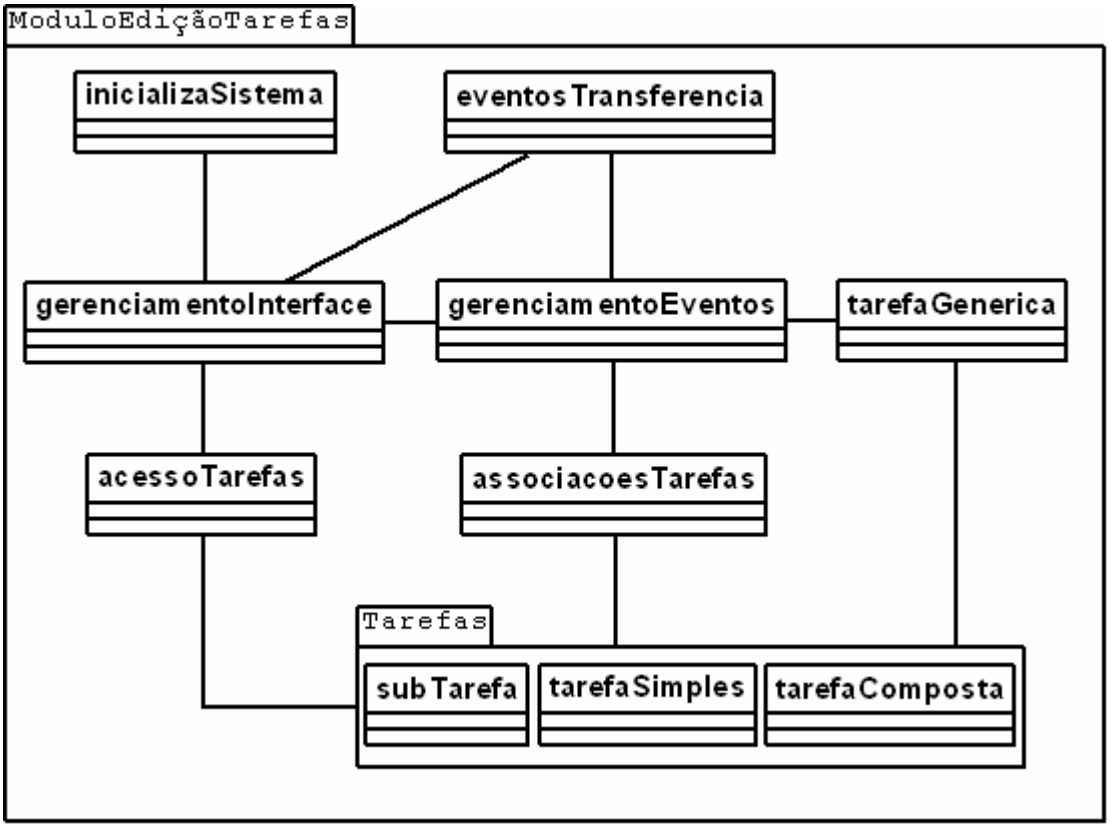

Figura 4.7 – Estruturas de classes do Módulo de Edição de Tarefas

- **inicializaSistema**: Classe responsável por inicializar da Interface de Edição de Tarefas. Para isso, a classe **inicializaSistema** autentica o usuário que está acessando o sistema, e instancia a classe responsável por gerenciar a interface de edição de tarefas (**gerenciamentoInterface).**
- **gerenciamentoInterface:** Classe responsável por desenhar e gerenciar todos os

elementos gráficos dispostos na tela. A **gerenciamentoInterface** monta todo ambiente gráfico de Edição de Tarefas e ainda implementa as rotinas responsáveis por desenhar os componentes gráficos que representam as subtarefas, tarefas e fluxos de execução.

- **gerenciamentoEventos:** Classe que implementa as rotinas de obtenção e tratamento dos eventos gerados pelos componentes gráficos dispostos na tela e que são movidos pelo usuário. Vale ressaltar que os eventos gerados são do padrão Java de suporte a *Drag and Drop,* sendo necessário à implementação das rotinas-padrões que capturam e tratam esses eventos. Permite, assim, ao desenvolvedor a implementação das funcionalidades que são desejadas de acordo com os diversos eventos. Alguns exemplos desses eventos gerados pelos objetos são: `mouseDragged`, `mouseMoved`, `mouseReleased`, etc...
- **eventosTransferencia:** Classe utilizada para o tratamento dos eventos de transferência de informações dos objetos da classe **gerenciamentoEventos**. A cada movimento de objetos gráficos (que são objetos da classe **gerenciamentoEventos**), várias operações devem ser executadas. É necessária a verificação de que o movimento é válido se o componente pode ser movido para aquele determinado local, etc. Além disso, implementa rotinas-padrões para o tratamento de transferência de informações dos objetos *Drag and Drop* em Java. Caso as verificações e validações sejam satisfeitas, deve-se realizar o tratamento de que o movimento resulta (cópia, transferência, deslocamento dos objetos).
- **tarefaGenerica:** Classe que representa uma tarefa genérica do modelo de tarefas da arquitetura ClinicSpace. Classe que implementa o objeto Java que é desenhado na tela, sendo uma extensão da classe **gerenciamentoEventos.** Conforme os eventos vão sendo gerados devido à movimentação dos componentes, os eventos são capturados e tratados pelas classes **gerenciamentoEventos** e **eventosTransferencia.** Essas classes conseguem a obtenção dos componentes envolvidos nos eventos (que são objetos da classe **tarefaGenerica**) e a partir desses componentes obtém-se qual elemento do modelo de tarefas (subtarefa, tarefa ou fluxo de execução) o componente representa.
- **associacoesTarefas:** Classe onde são tratadas as associações de tarefas do módulo. Essa classe gerencia as associações dos elementos, verificando se as regras das subtarefas, tarefas e fluxos são satisfeitas. Esse gerenciamento é necessário tanto no momento em que o usuário está arrastando os ícones para compor uma nova tarefa ou fluxo, quanto no momento em que ele está reorganizando os elementos já dispostos em um elemento. Em ambos os casos, é essa classe que realiza as verificações e as

validações que são necessárias para permitir uma associação dos elementos.

- **acessoTarefas:** Classe responsável por gerenciar o acesso as tarefas. Essa classe dispõe à **gerenciamentoInterface** o conjunto de tarefas que deve ser disponibilizado ao usuário, e também recebe dessa as novas tarefas que foram criadas e que devem ser salvas no banco de armazenamento de tarefas. Para prover essas funcionalidades, a classe **acessoTarefas** interage com a **ApiOntologias** (abordada na seção 4.3.3) que é a responsável pela interação com a base de dados que armazena as tarefas.

As classes restantes fazem parte do pacote de classes utilizadas para a modelagem das tarefas clínicas. O pacote de classes **Tarefas** representa todos os níveis de abstração disponibilizados para o usuário, englobando assim as subtarefas (classe **subTarefa**), tarefas (classe **tarefaSimples**) e os fluxos de execução **(tarefaComposta)**.

- **subTarefa**: representa os serviços e aplicações do *middleware* de gerenciamento de tarefas e do ambiente pervasivo (EXEHDA expandido (FERREIRA et al, 2009)) e que permitem a abstração das tarefas clínicas. Nessa classe são definidas informações sobre a aplicação que cada objeto representa, imagem ou ícone utilizado para representar determinada subtarefa, informações de identificação do objeto, etc. Cada objeto dessa classe possui um tipo, que é utilizado para a verificação e validação das associações.

Por questões relativas à representação das tarefas clínicas através de Ontologias, foi necessário o desmembramento de tarefas em duas classes, a classe **tarefaSimples** e a classe **tarefaComposta**.

- **tarefaSimples**: Classe que representa as tarefas clínicas do sistema, sendo que os objetos dessa classe são compostos apenas por objetos da classe **subTarefa.** Nessa classe são armazenadas informações como a qual classificação a tarefa pertence (Laboratorial, Diagnóstico, Tratamento ou Administrativa), ícone que representa determinada tarefa, conjunto de subtarefas que são necessárias para o suporte da tarefa, etc.
- **tarefaComposta**: Classe que representa os fluxos de execução de tarefas e subTarefas. Nessa classe são armazenadas informações referentes à identificação dos objetos da classe, imagem utilizada para representar o tal fluxo, o proprietário do fluxo, etc. Objetos da classe **tarefaComposta** podem ser compostos por subtarefas (classe **subTarefa**), tarefas (classe **tarefaSimples**) e por outros fluxos de execução (classe **tarefaComposta**).

4.3.3 Representação das Tarefas Clínicas através de Ontologias

Na arquitetura ClinicSpace as subtarefas, tarefas e fluxos de execução são representados através de Ontologias. Ontologias podem ser vistas como um conjunto coerente de coleções estruturadas de informação. Uma ontologia é uma teoria lógica para descrever o significado pretendido de um vocabulário formal, isto é, seu compromisso com uma conceitualização particular do mundo. Estas incluem estruturas que permitem manipular termos de uma forma muito eficiente e útil para o usuário e mecanismos de validação para comunicação entre programas. A importância de seu uso é devida à capacidade de representar hierarquias de classes de objetos (taxonomias) e seus relacionamentos (LIBRELOTTO et al, 2008).

As tarefas do conjunto mínimo de tarefas foram utilizadas para a especificação da ontologia, em um trabalho conjunto com Jonas Gassen - mestrando em Nanociências, do Centro Universitário Franciscano. A figura 4.8 apresenta a visão da rede semântica das tarefas e subtarefas encontradas na ontologia. Percebe-se a relação específica entre as tarefas e subtarefas que as compõem através de cada ligação. A maioria das subtarefas compõe mais de uma tarefa, o que indica uma nova ligação, gerando tal emaranhado de relações.

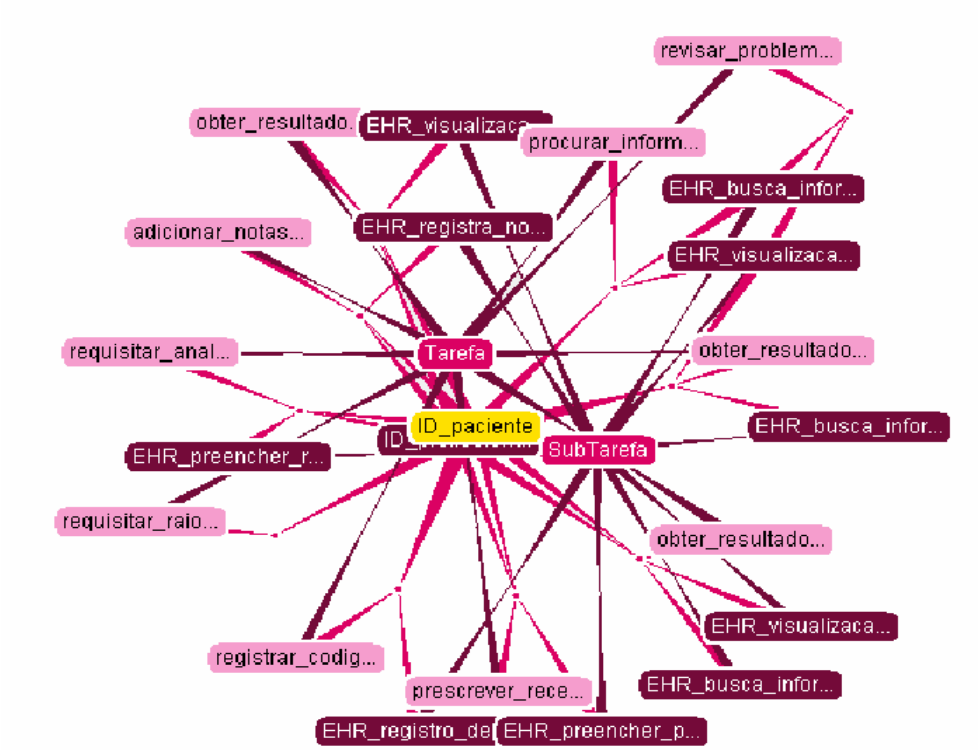

Figura 4.8 – Visão da rede semântica representada na ontologia. Fonte: (LIBRELOTTO et al, 2008, p. 4)

Para facilitar a visualização da ontologia, a Figura 4.9 representa somente as relações envolvendo a tarefa "**Revisar problemas do paciente**". Essa é uma Tarefa composta por quatro subtarefas responsáveis pelas funcionalidades de identificar o profissional que vai realizar a tarefa, identificar o paciente a ser consultado, buscar as informações do paciente e visualizar as informações buscadas.

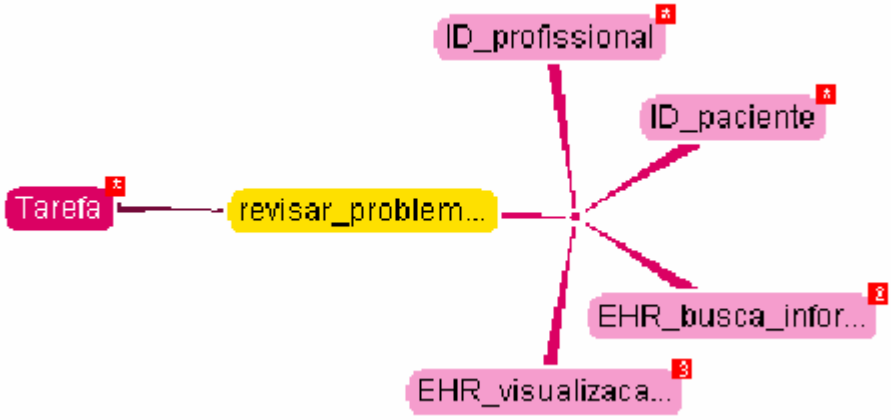

Figura 4.9 – Rede semântica da tarefa "Revisar problemas do paciente". Fonte: (LIBRELOTTO et al, 2008, p. 5)

A Ontologia é representada em OWL (*Web Ontology Language*) (MCGUINENESS; HARMELEN, 2004) que é uma linguagem para definir e instanciar ontologias, fornecendo uma especificação que permite representar conhecimento conceitual com o qual se distingue recurso de informações semanticamente. As informações de contexto futuramente também serão representadas através de Ontologias. Muitas vantagens advêm dessas representações, destacando-se a possibilidade da utilização dessas informações na Inferência de Tarefas, prevista no projeto ClinicSpace.

Diversos módulos integrantes da Arquitetura ClinicSpace utilizam as informações das subtarefas, tarefas e fluxos representadas através de ontologias. Para isso, programou-se uma classe denominada **ApiOntologias** para a intermediação da comunicação entre os módulos do ClinicSpace com as ontologias de representação de tarefas. Assim, abstrai-se, para os módulos, o entendimento da forma como são representados e armazenados os elementos. Basta a invocação dos métodos implementados na **ApiOntologias** para a obtenção das informações desejadas. A classe foi implementada em Java, facilitando com isso a integração com os módulos da Arquitetura ClinicSpace.

No caso específico do Módulo de Edição de Tarefas, essa API é utilizada para a

obtenção das subtarefas, tarefas e fluxos de execução que estarão sendo criadas ou editadas (figura 4.10). Assim, esses elementos são carregados para o módulo de Edição ficando a disposição do profissional clínico. De acordo com as operações que o profissional clínico realiza na interface gráfica, o módulo invoca os métodos disponibilizados na **ApiOntologias** para carregar tarefas, salvar tarefas e fluxos criados pelo usuário, etc. Objetos das classes **subTarefa, tarefaSimples** e **tarefaComposta** são enviados e recebidos da **ApiOntologias** e esses representam respectivamente as subtarefas, as tarefas e os fluxos de execução.

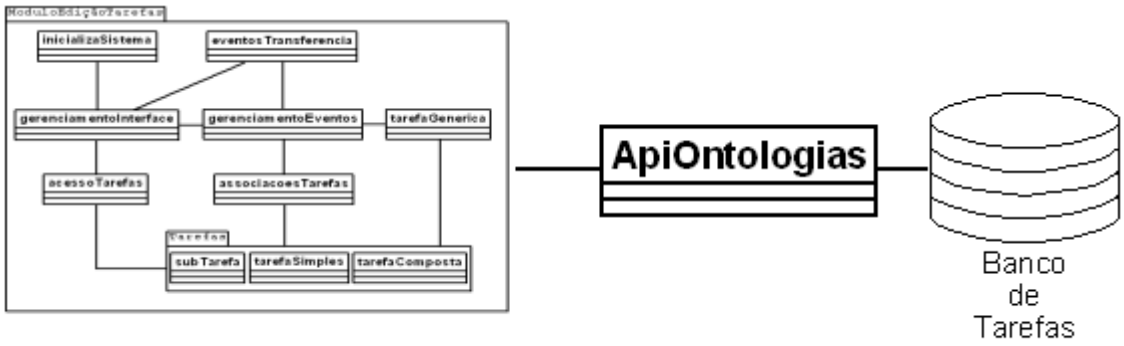

Figura 4.10 – Módulo de Edição de Tarefas e as Ontologias

4.4 Sequência de Ações no Processo de Definição de Tarefas e Fluxos

Em termos de implementação, um conjunto de ações precisou ser implementado para permitir as funcionalidades demonstradas nas seções 5.1 e 5.2. A descrição dessas etapas são abordadas na seqüência.

Nessa subseção, descrevem-se as etapas mais importantes para o processo de Criação e Edição de Tarefas e Fluxos de Execução. Dessa forma, o processo inicia-se com o Módulo de Edição de Tarefas obtendo as subtarefas, tarefas e fluxos de execução. O processo é descrito na sequência.

4.4.1 Obtenção das tarefas clínicas e inicialização da Interface de Edição de Tarefas.

Conforme já mencionado, no modelo proposto na arquitetura ClinicSpace as tarefas (subtarefas, tarefas e fluxos de execução) são modeladas utilizando a representação via Ontologias. Os módulos que necessitam desses objetos utilizam a **ApiOntologias** para a interação com a base de dados que contém as tarefas. Assim, o usuário não precisa conhecer as subtarefas, tarefas e fluxos de execução nem como são armazenadas e representadas na base computacional.

Para a preparação do Módulo de Edição de Tarefas e das abstrações providas por essa interface, diversas etapas são executadas. O conjunto de etapas, figura 4.11, é:

- (i) o processo inicia-se com o usuário se autenticando no Módulo de Edição de Tarefas. Assim, a classe **inicializaSistema** identifica o usuário e instancia a classe responsável pela criação e gerenciamento da interface (**gerenciamentoInterface);**
- (ii) a classe **gerenciamentoInterface**, responsável pela preparação do ambiente, utiliza a **acessoTarefas** para obter as tarefas e fluxos de execução pertencentes ao usuário.
- (iii) a classe **acessoTarefas** interage com a **ApiOntologias** para obter as tarefas e fluxos de execução pertencentes ao usuário e armazenadas na base de dados;
- (iii) a partir da obtenção dos objetos, a classe criadora da Interface de Edição de Tarefas (**gerenciamentoInterface**) pode, então, criar os mecanismos necessários para o tratamento da técnica *Drag and Drop* em Java. Para isso são criados e inicializados os componentes gráficos que irão representar as Tarefas e, também, os objetos responsáveis pelo gerenciamento dos eventos gerados pela movimentação desses objetos na tela. Assim, são criados e inicializados os objetos das classes **gerenciamentoEventos**, **tarefaGenerica** e **eventosTransferencia;**
- (iv) a preparação da interface de Edição de Tarefas é finalizada com o desenho dos componentes gráficos que representam as tarefas (subtarefas, tarefas e fluxos) do modelo.

Finalizadas as etapas acima, o módulo de Edição de Tarefas está pronto para a interação com o usuário. O profissional clínico pode, então, utilizar as tarefas já definidas para a criação de outras que julgar necessário. As etapas que compreendem a definição de outras tarefas e fluxos são descritas a seguir.

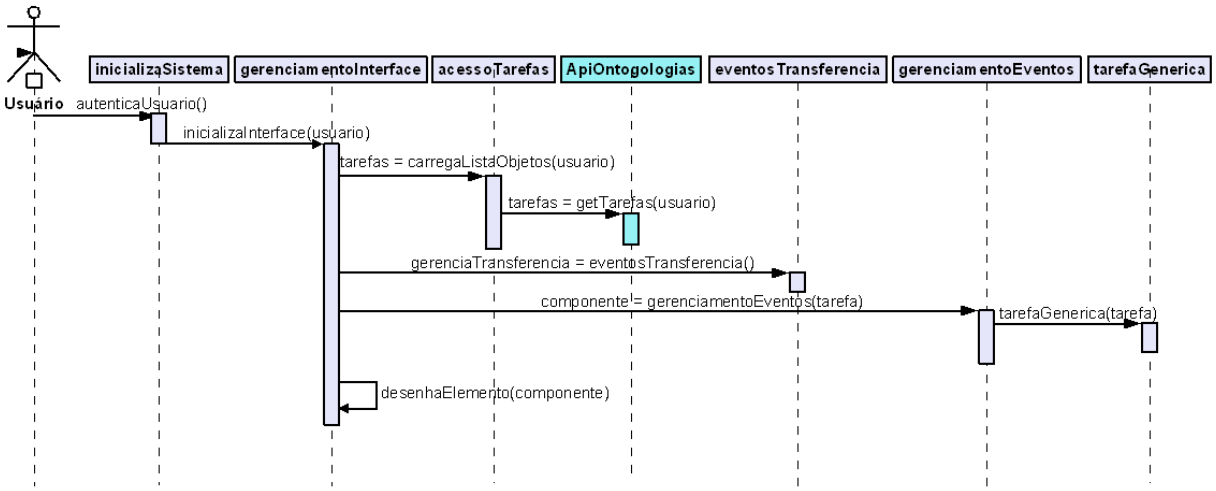

Figura 4.11 – Diagrama de Seqüência para a inicialização da Interface de Edição de Tarefas.

4.4.2 Criação e edição de Tarefas

Após as tarefas e fluxos terem sido carregadas da base de dados e também disponibilizadas na interface como componentes gráficos Java passíveis de serem movidos, permite-se, ao usuário, a criação de novas tarefas e fluxos. Para esse processo são utilizadas as subtarefas, tarefas e fluxos de execução já codificadas, bastando ao profissional clínico a utilização desses elementos conforme a sua necessidade ou preferência. Assim, a sequência para o processo, figura 4.12, é:

- (i) o usuário, ao mover determinado componente no módulo de Edição de Tarefas, dispara a classe responsável pela captura e tratamento de tal evento;
- (ii) a classe **gerenciamentoEventos,** após capturar o evento gerado, utiliza a classe **eventosTransferencia** para atestar se é válido o movimento e se é possível a transferência de informações dos componentes *Drag and Drop.* Sendo possível a transferência de informações, finalizam-se as verificações dos componentes gráficos Java responsáveis pelo tratamento de *Drag and Drop;*
- (iii) a partir dessa etapa, inicia-se o processo de tratamento das tarefas (subtarefas, tarefas e fluxos de execução) que estão representadas através dos ícones. Assim, a classe **gerenciamentoEventos** obtêm as tarefas envolvidas na associação;
- (iv) invoca-se, então, a classe responsável pelo gerenciamento das políticas utilizadas para verificação da associação de Tarefas (classe **associacoesTarefas**). Essa classe

aplica as políticas necessárias para verificar se a associação é válida. Como todas as verificações baseiam-se nas subtarefas que compõem as tarefas e fluxos, nessa etapa são obtidas as subtarefas sobre as quais serão aplicadas as regras de associação;

- (v) a classe **associacoesTarefas** verifica as regras específicas contidas nas classes do pacote de Tarefas e, então, retorna a informação sobre a permissão ou não de associação;
- (vi) caso a associação seja válida, a associação dos elementos é realizada, sendo, então, desenhados os elementos com a nova associação já representada. Caso a associação não seja permitida, o usuário é informado de que a associação desejada por ele não é possível e o movimento dos ícones é desfeito.

Essas etapas são executadas a cada movimento de componente gráfico realizado pelo profissional clínico. No momento que o profissional deseja salvar a nova tarefa ou fluxo criado, solicita-se a ele as informações necessárias (nome, classificação, ícone que será utilizado para a representação, etc.). Posteriormente, o módulo de Edição de Tarefas interage novamente com a API **ApiOntologias** para o armazenamento da nova Tarefa ou fluxo criado pelo profissional clínico. A nova tarefa, ou fluxo, é armazenada na base de dados e passa a integrar o conjunto de Tarefas pertencente ao profissional clínico. Esse poderá utilizar essa nova tarefa ou fluxo para a criação de outros, quando julgar necessário. Para isso, repetem-se as etapas descritas.

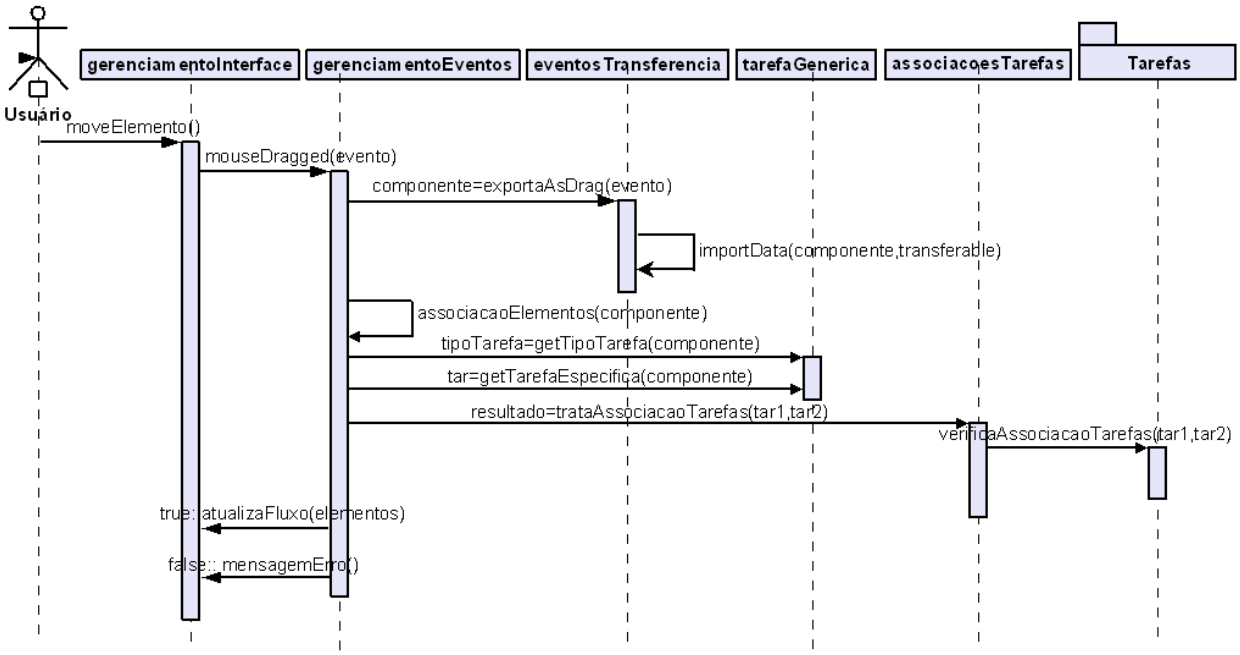

Figura 4.12 – Diagrama de Seqüência da composição de novas tarefas e fluxos.

5 DISCUSSÃO DOS RESULTADOS

Este capítulo apresenta uma avaliação por comparação com trabalhos relacionados e os estudos de casos que procuram demonstrar a viabilidade e inovação da solução proposta.

5.1 Avaliação Comparativa

A noção de Computação baseada em Tarefas foi introduzida pelo Projeto Aura (GARLAN; STEENKISTE; SCHMERL, 2002) como um meio de a infra-estrutura pró-ativamente (sem interferência do usuário) configurar dispositivos de forma que o usuário possa manter a continuidade do que estava fazendo, conforme ele se deslocava de um lugar a outro. Nesse modelo, tarefas são modeladas como uma coleção de serviços. A descrição do serviço (qualidades, atributos, preferências) é usada para encontrar os recursos necessários ou reconfigurar o sistema para executar a tarefa. O sistema mantém uma representação explícita da intenção do usuário e a natureza dos serviços requeridos. O Sistema Aura é autogerenciável, ou seja, não há a programação de aplicações envolvidas por parte do usuário.

A partir dessa primeira proposta outras idéias surgiram, destacando-se a apresentada pelo Projeto GAIA (ROMAN et al, 2002) que inseriu no *Middleware* Gaia o modelo de Programação Olympus (RANGANATHAN; CAMPBELL, 2005). Neste projeto, visualiza-se um futuro onde o espaço habitado pelas pessoas é interativo e programável. Assim, os usuários podem interagir com seus escritórios, casas, carros, etc. para requisitar informações, beneficiarem-se dos recursos disponíveis e configurar o comportamento de seu habitat. Porém, esse projeto não focaliza a programação pelo usuário-final.

O projeto *Activity-based Computing* (BARDRAM; CHRISTENSEN, 2007) apresenta uma proposta de utilização de Computação baseada em Tarefas destinados a Ambientes de Saúde. Nesse projeto, desenvolveu-se o Framework ABC que provê uma infra-estrutura de execução de serviços que suporta características inerentes ao trabalho dos profissionais

clínicos. Assim, os serviços podem ser inicializados, suspensos, armazenados, retomados em qualquer dispositivo computacional, em qualquer instante de tempo, encaminhado para outros usuários e compartilhado entre diversos usuários. Além disso, a execução dos serviços é adaptável de acordo com o contexto dos usuários. O projeto visava permitir que desenvolvedores de aplicações clínicas pudessem incorporar suporte à mobilidade, interrupções, atividades paralelas, cooperação e consciência de contexto para a concepção e implementação de seus programas.

O projeto *Task Computing* (MASUOKA; PARSIA; LABROU, 2003) disponibiliza um framework que visa capacitar o usuário a executar tarefas em aplicações, dispositivos e serviços em ambientes pervasivos. O projeto disponibiliza diversas formas para o usuário interagir com o framework, mas o foco principal está na infra-estrutra de suporte à execução e gerenciamento dos serviços. O principal aspecto abordado é a descoberta dinâmica de serviços que considera as informações de contexto para a definição dos serviços mais apropriados.

Muitas das idéias apresentadas por esses projetos influenciaram a proposta do projeto ClinicSpace. Destaca-se o projeto *Activity-based Computing*, que norteou a definição de conceitos relativos a tarefas para a área da Saúde. Porém, como se pode observar no Quadro 5.1, nenhum dos projetos apresenta a visão centralizada no usuário defendida pelo Projeto ClinicSpace. Um dos diferencias do projeto é a possibilidade do balanceamento entre a execução automática e o controle sobre a programação e execução das tarefas por parte do usuário, já que é disponibilizado aos profissionais clínicos meios para que ele possa agendar suas tarefas ou relacioná-las a determinados contextos (automação). Além disso, é permitido ao profissional controlar manualmente as tarefas já que esse pode executar, interromper, continuar e cancelar as tarefas.

O presente trabalho acrescenta a contribuição de permitir a ação ativa dos profissionais clínicos na programação de sistemas que gerenciam a execução de suas tarefas clínicas. Assim, os profissionais podem utilizar a ferramenta de Programação de Tarefas para personalizar as tarefas e torná-las o mais próximo possível da sua forma particular de executá-las, reduzindo o grau de rejeição e insegurança que eles podem ter ao usar sistemas totalmente pró-ativos, como tem sido o foco de sistemas como Aura e Gaia. Além disso, outro importante diferencial do presente trabalho é a utilização de mecanismos providos pela programação orientada ao usuário-final com o objetivo de facilitar a utilização do sistema pelo profissional clínico. Espera-se, com essas funcionalidades, contribuir para uma redução na rejeição dos sistemas computacionais em atividades clínicas hospitalares.

	Computação baseada em Tarefas					
	Contexto	Automatização	Controle	Ambientes Hospitalares	Personalização	Interface Orientada ao usuário final
Aura	X	X				
Gaia	X					
ABC	X	X	X	X		
Taks Computing	X	X	X			X
ClinicSpace	X	X	X	X	X	X

Quadro 5.1 – Comparativo dos projetos relacionados a este trabalho

5.2 Estudos de Casos

Para exemplificar o uso da interface, foram realizados alguns estudos de casos.

5.2.1 Criação de novas Tarefas

Os profissionais clínicos executam diariamente as suas atividades de cuidados clínicos de seus pacientes e, para isso, esse profissional pode utilizar as tarefas já disponibilizadas pelo sistema (pré-projetadas) ou, então, construir as tarefas que o auxiliarão nessa atividade. Caso já exista a tarefa que o usuário necessita, basta a ele selecioná-la para utilizá-la. Caso contrário, o profissional clínico, ao construir a tarefa, pode definir a sua própria forma de trabalhar e, assim, definir as etapas que costuma realizar na ordem desejada por ele. Por exemplo, se o médico costuma primeiramente verificar as últimas informações do prontuário do paciente, ele inicia a construção da tarefa arrastando uma subtarefa de busca de informações do paciente ao local de criação e edição de tarefas na Interface (Figura 5.1). O profissional poderá parametrizar essa tarefa informando alguns filtros, como a quantidade de registros a serem buscados, o período das informações, etc.

A Interface de Edição de Tarefas, ao identificar que o profissional clínico deseja buscar as informações do paciente, adiciona automaticamente uma subtarefa de identificação

desses, para a partir dessa informação saber qual prontuário deve ser acessado (Figura 5.2). Após isso, o profissional pode adicionar uma subtarefa que permita a visualização das informações buscadas. Com isso, em nível conceitual, uma nova tarefa já está formada e pronta para ser utilizada. Contudo, é importante salientar que, se o profissional clínico já disponibiliza de uma tarefa específica para a busca de informações (essa é uma tarefa pertencente ao conjunto mínimo de tarefas modeladas e já disponibilizadas aos clínicos) ele poderia utilizar essa tarefa ao invés de adicionar essas três subtarefas, otimizando o processo de criação. Nesse caso, a tarefa criada seria tratada, em nível de gerenciamento, como uma tarefa composta, já que seria formada por outra tarefa e não mais por apenas subtarefas.

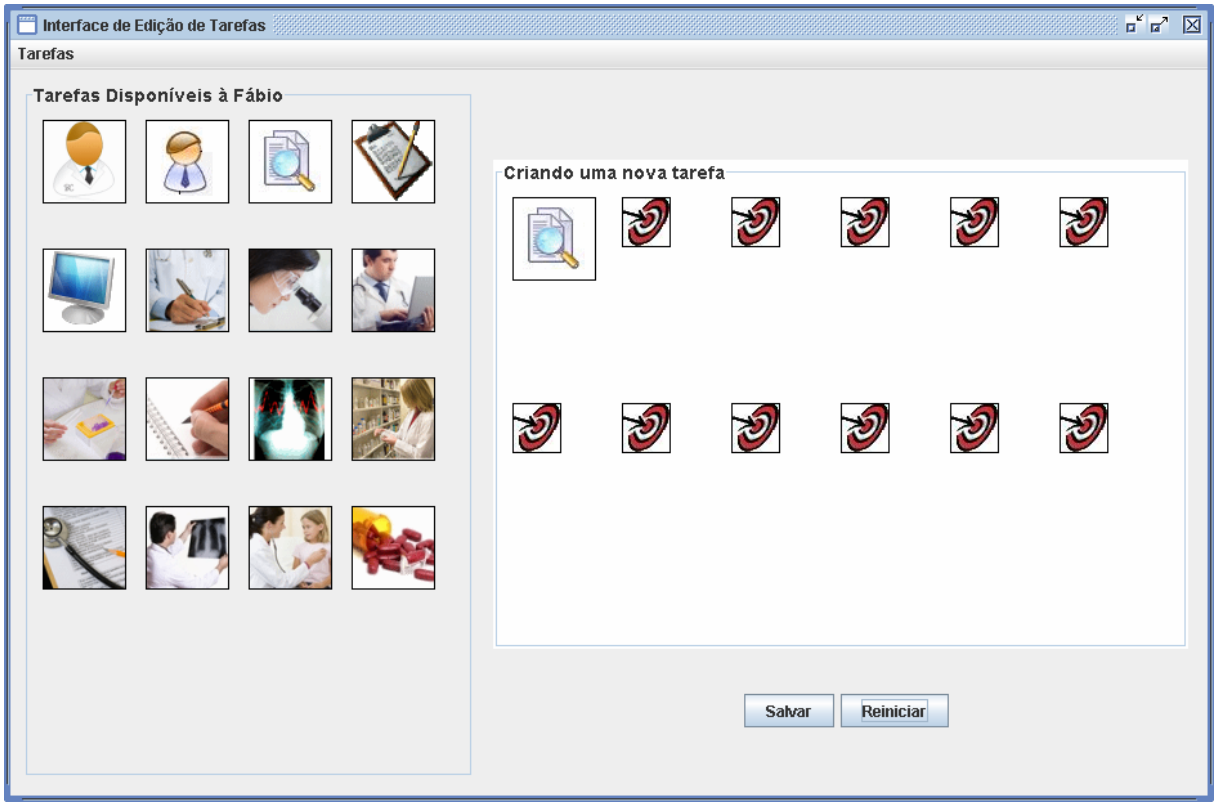

Figura 5.1 – Criação de uma nova tarefa

Continuando a composição da tarefa, o médico poderia adicionar uma subtarefa de registro de informações para armazenar no registro de informações do paciente as informações de atendimento. Essa subtarefa de registro de informações faz uso da subtarefa de identificação do paciente, adicionada automaticamente pela Interface de Edição de Tarefas, para identificar a qual o registro de informações de paciente deve ser utilizado.

Considerando que o profissional clínico deseja finalizar esse processo, basta a ele clicar em “Salvar” e informar dos dados respectivos à nova tarefa criada. Essa será

armazenada na base que mantém as tarefas desse profissional clínico que poderá reutilizar essa tarefa futuramente para a criação e definição de outras tarefas e fluxos de execução.

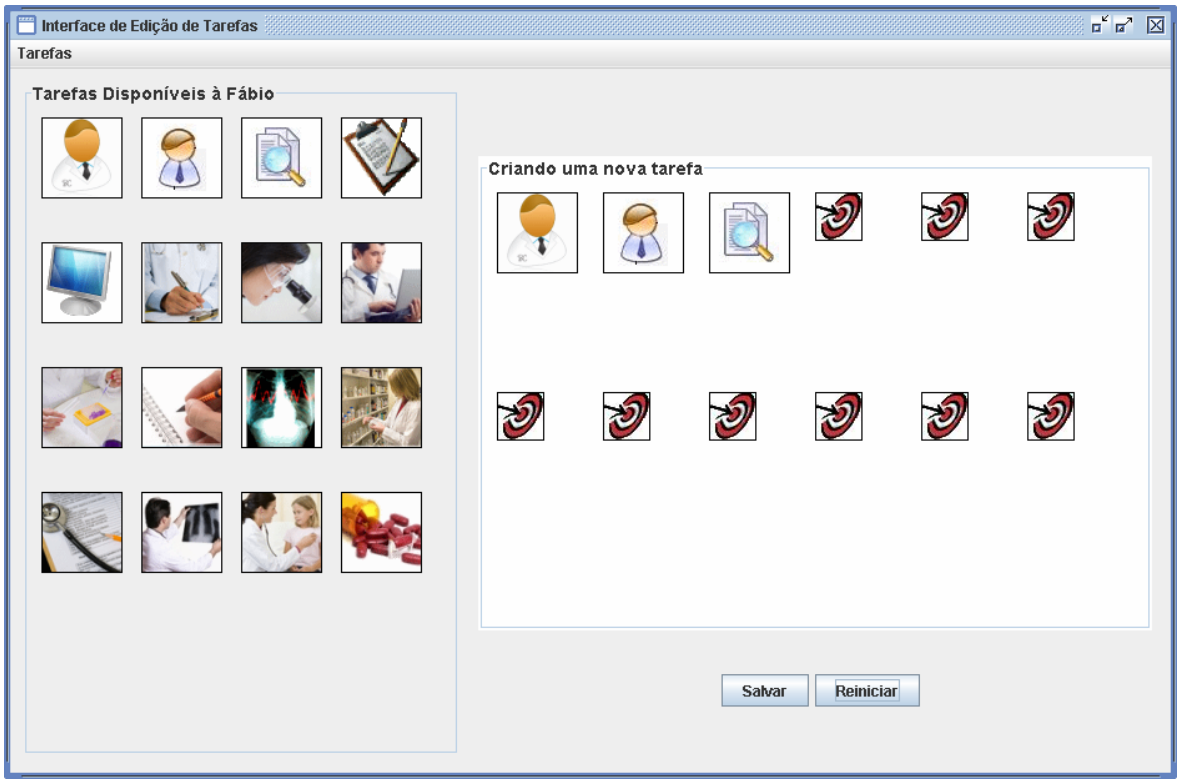

Figura 5.2 – Interface de Edição de Tarefas adicionando as subtarefas necessárias

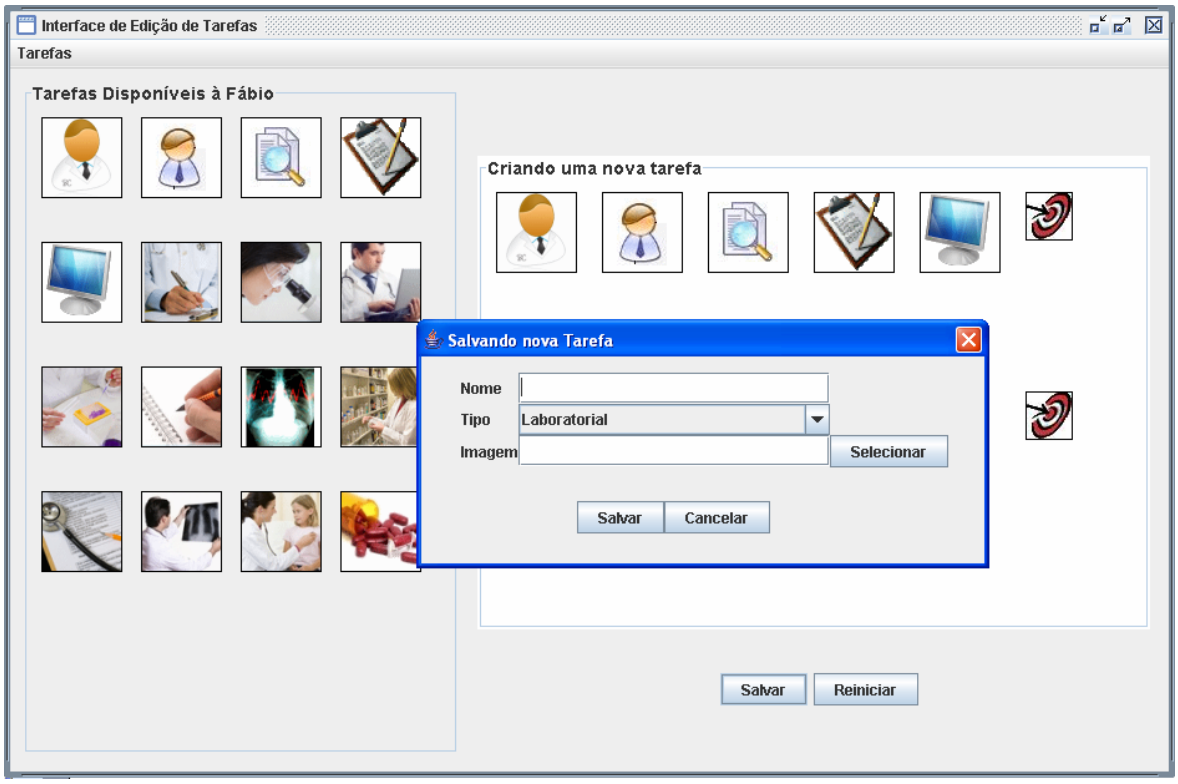

Figura 5.3 – Salvando a nova tarefa criada

Lembrando que as regras de associação (seção 4.3.1) são verificadas e validadas pela Interface de Edição de Tarefas no momento que o profissional clínico está adicionando novas subtarefas à tarefa que está sendo criada. As regras também serão verificadas pela interface, caso o profissional decida alterar a ordem das subtarefas, reorganizando os elementos que compõem a nova tarefa. Em ambos os casos, se as regras de associação não forem satisfeitas, a interface de Edição de Tarefas emite um sinal ao profissional clínico informando o erro e desfazendo a operação (ver Figura 5.4).

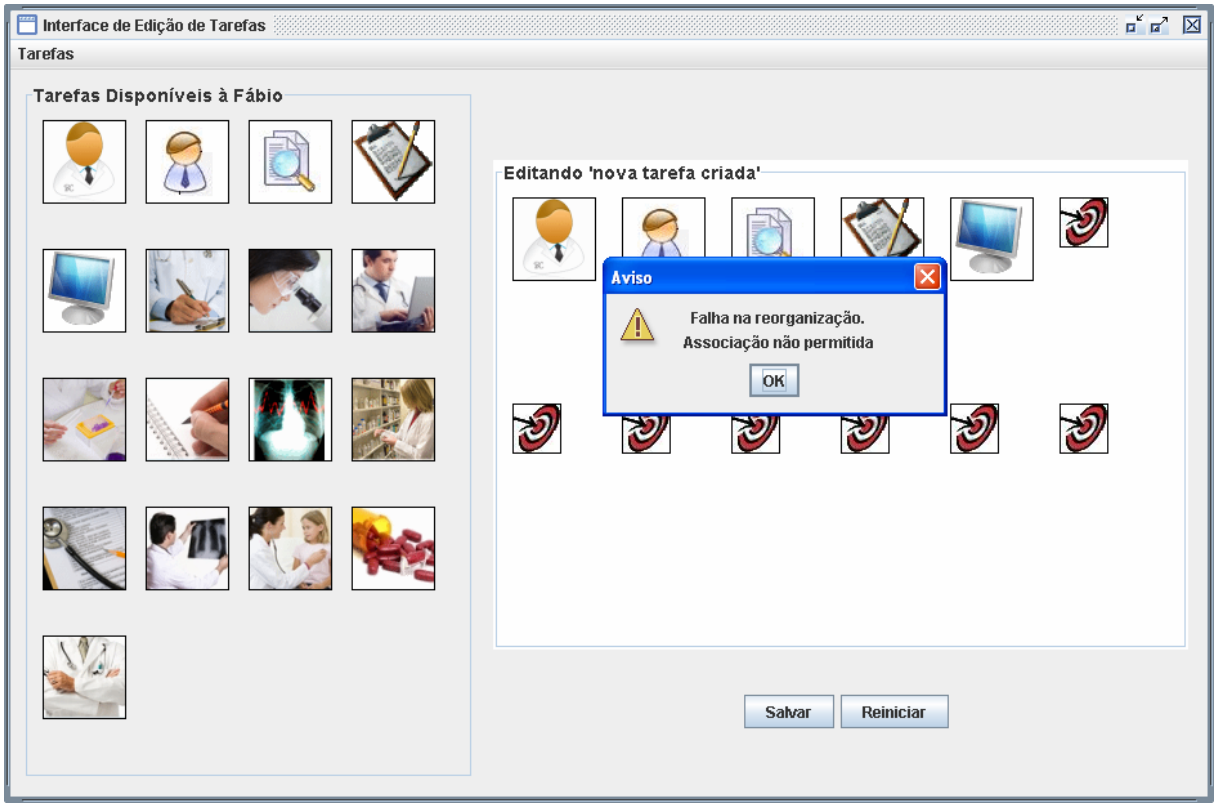

Figura 5.4 – Erro na validação das Regras de Associação de Tarefas

5.2 Edição de Tarefas já modeladas

Os profissionais clínicos, ao utilizar a Interface de Edição de Tarefas, têm a sua disposição as tarefas já modeladas inicialmente, pertencentes ao conjunto mínimo de tarefas, e as outras tarefas e fluxos criados por eles. A partir dessas tarefas e fluxos, o profissional pode editá-las para adequar à necessidade desejada, no momento. Dessa forma, facilita-se a criação

e edição de novas tarefas e fluxos já que podem ser (re)utilizadas outras tarefas já modeladas.

Supondo que o profissional clínico tenha construído, em um outro momento, um fluxo (tarefa composta em termos de gerenciamento) que represente o conjunto de tarefas que ele execute ao chegar pela manhã no hospital e revisar o estado de seus pacientes. Por exemplo, caso o profissional clínico tenha criado uma tarefa "Rotina diária de verificação dos Pacientes" composta pelas seguintes tarefas: (i) Obter resultados de exames laboratoriais; (ii) Revisar os problemas do paciente; (iii) Adicionar notas diárias sobre as condições do paciente. No momento que o profissional selecionar a tarefa "Rotina diária de verificação dos Pacientes", a Interface de Edição de Tarefas disponibiliza o conjunto de tarefas que a compõe (Figura 5.5).

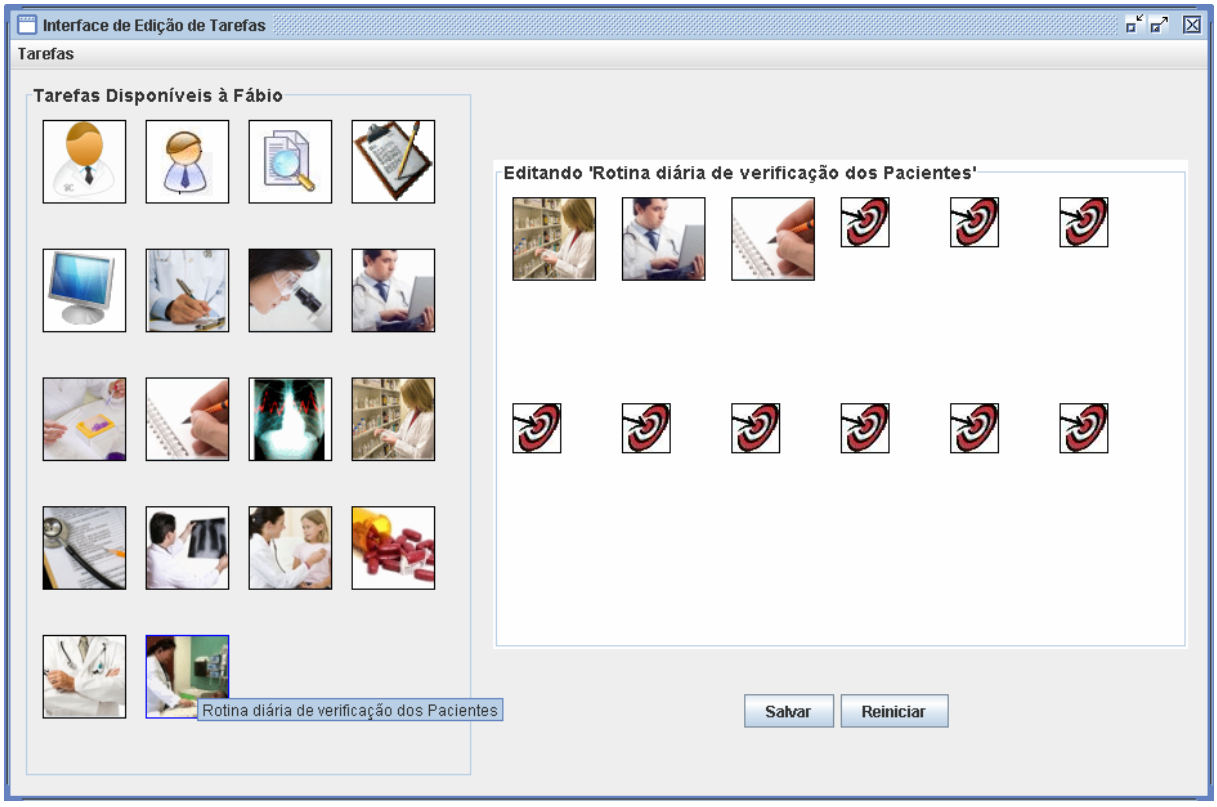

Figura 5.5 – Visualizando os elementos que compõem o fluxo

Ao visualizar as tarefas que compõem o fluxo, o profissional pode realizar diversas operações. Entre elas, destacam-se:

- Alterar a ordem das tarefas que compõem o fluxo "Rotina diária de verificação dos Pacientes". Basta reorganizar os elementos que compõe a tarefa, utilizando a técnica de clicar e arrastar os ícones que as representam;

- Agregar novas funcionalidades à “Rotina diária de verificação dos Pacientes”. Por exemplo, o profissional deseja adicionar a tarefa “Obter resultados de raios-x”. Para isso, basta a ele arrastar o ícone que representa essa tarefa em direção ao conjunto de tarefas que compõe a “Rotina diária de verificação dos Pacientes”. Caso o profissional não tenha arrastado a tarefa no local apropriado para formar a ordem desejada (Figura 5.6), ele poderá reorganizar os elementos para, assim, redefinir a ordem das tarefas que compõem o fluxo (Figura 5.7).

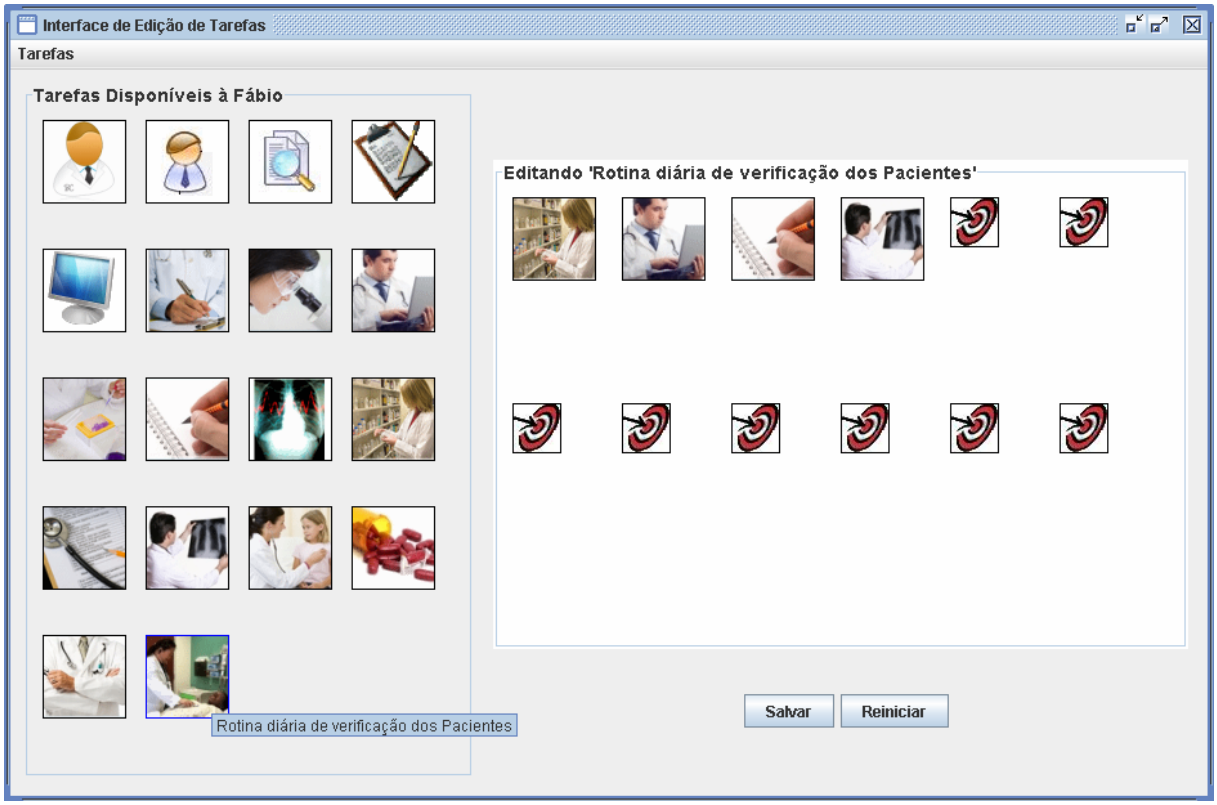

Figura 5.6 – Adicionando novas tarefas ao fluxo

Ressalta-se que a Interface de Edição de Tarefas utiliza as Regras de Associação para verificar se a operação desejada pelo usuário de reorganizações dos elementos ou adição de novos é permitida. Em caso negativo, a Interface sinaliza o problema para o profissional e desfaz a operação. No momento que o profissional finaliza o processo de editar a tarefa, basta salvar as alterações e informar os dados relativos à identificação da nova tarefa. Assim, uma nova tarefa será criada contendo todas as alterações realizadas pelo profissional clínico.

Após definidas as tarefas, elas serão armazenadas e executadas sob o gerenciamento do sistema SGT (FERREIRA et al, 2009).

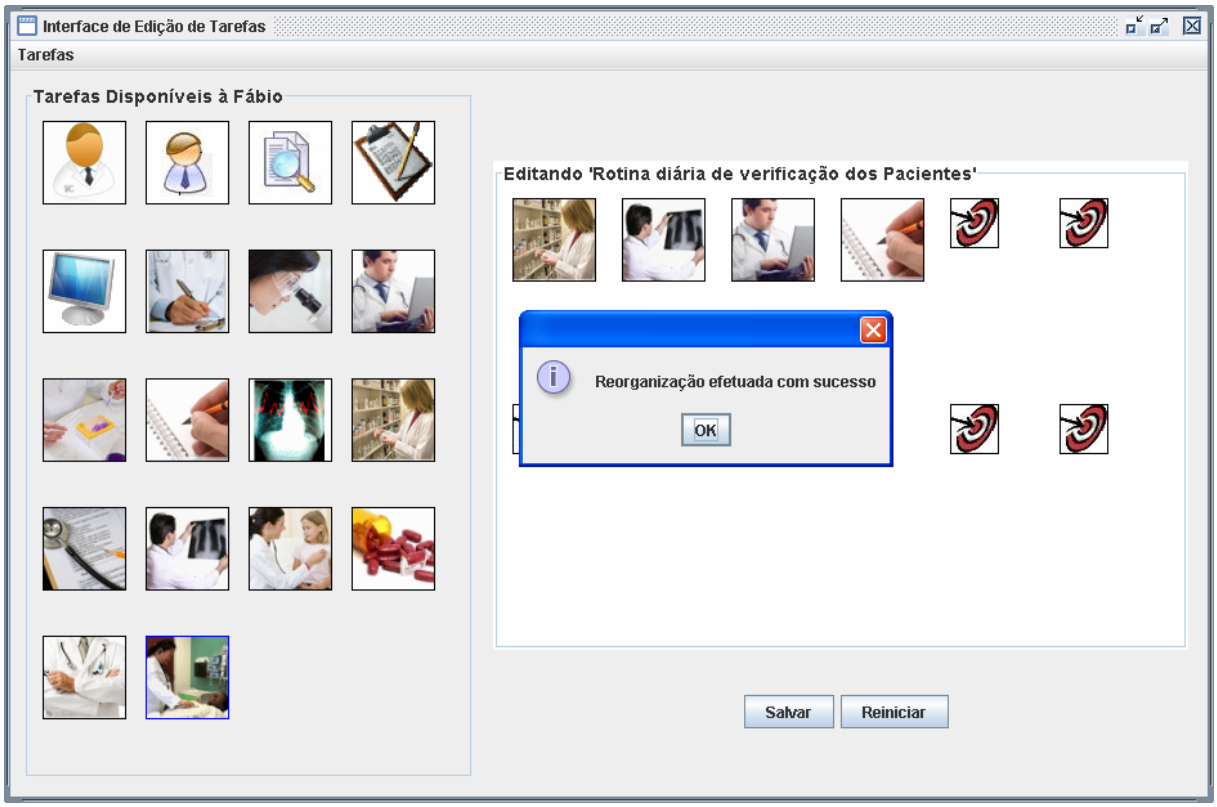

Figura 5.7 – Reorganizando as tarefas do fluxo

5.3 Análise da modelagem de tarefas e da ferramenta implementada

Os casos exemplificados nas seções 5.1 e 5.2 demonstram a simplicidade no uso da interface de edição de tarefas médicas e a viabilidade da ferramenta implementada, a qual permite a criação e edição de tarefas e fluxos personalizados. Os clínicos podem ainda reutilizar as tarefas e fluxos, tanto as definidas na modelagem de tarefas quanto as criadas por eles mesmos, para a criação de outras que julgarem necessárias. Através da modelagem de tarefas médicas realizada, e da Interface de Edição de Tarefas, os profissionais clínicos poderão personalizar as tarefas e fluxos adequando-as a sua forma particular de executá-las.

A fim de facilitar a interação dos profissionais clínicos com a Interface de Edição de Tarefas, foram utilizadas abstrações (ícones) para a representação das subtarefas, tarefas e fluxos de execução. Esses mecanismos, oriundos da técnica de Programação orientada ao usuário-final (seção 2.2) tornam mais fácil aos profissionais clínicos a personalização de suas tarefas diárias.

A Interface de Edição de Tarefas desenvolvida no presente trabalho, é uma primeira versão da ferramenta-piloto a ser desenvolvida no projeto ClinicSpace. Essa ferramenta sofrerá melhorias no decorrer do desenvolvimento do projeto com o intuito de facilitar a utilização real da ferramenta por parte do usuário final. O acréscimo de novas funcionalidades e abstrações é necessário para tornar a interface mais intuitiva para o profissional clínico. Após realizada essas melhorias, será realizado um estudo sobre a usabilidade da Interface de Edição de Tarefas pelos profissionais da área médica a fim de avaliar a ferramenta-piloto prevista no projeto ClinicSpace.

6 CONCLUSÃO

Ambientes pervasivos do futuro próximo envolvem interação, coordenação, cooperação de numerosos, acessíveis e, frequentemente, invisíveis dispositivos e serviços. Nesses ambientes, computadores e outros dispositivos digitais dos mais variados tipos estão totalmente integrados ao ambiente do usuário e objetivam auxiliá-lo em suas atividades diárias. Esses ambientes são altamente dinâmicos e heterogêneos, já que os recursos, incluindo serviços, dispositivos e aplicações, podem alterar-se rapidamente. Diferentes espaços têm diferentes tipos de recursos disponíveis e diferentes políticas de uso dos recursos. Dessa forma, programadores de aplicações para esses espaços necessitam de abstrações de alto nível para programar aplicações no espaço pervasivo sem ter que ter consciência dos recursos disponíveis, contexto, políticas e preferências dos usuários. Com os esforços para o deslocamento do paradigma de "ambientes integrados" (primeira geração da Computação Pervasiva) para "espaços programáveis" (segunda geração da Computação Pervasiva), novas solução para a programação estão sendo propostas. Dessa forma, as linguagens de programação baseadas em tarefas visam criar abstrações de alto nível para a modelagem e programação de aplicações para esses espaços.

Muitas são as áreas de aplicação da computação pervasiva, sendo a área de Cuidados Clínicos uma das principais, devido a sua natureza. Para que possam ser aplicadas as idéias de Programação Orientada a Tarefas nesses ambientes, é necessária a modelagem das tarefas clínicas executadas por profissionais clínicos. A diversidade e flexibilidade destas tarefas, assim como também os requisitos, contextos e recursos necessários devem ser considerados na modelagem. Além disso, ferramentas simples e eficazes devem ser implementadas visando a criação de abstrações que permitam a sua utilização, sem rejeição por parte dos profissionais clínicos. Sabe-se que esses profissionais, muitas vezes, não detêm grande conhecimento sobre computação, mas reconhecem a necessidade da utilização de recursos computacionais para o auxílio de suas atividades diárias de cuidados clínicos de seus pacientes.

Este trabalho apresenta uma proposta de modelagem de Tarefas Clínicas baseadas nas atividades de cuidados médicos realizadas diariamente pelos profissionais dessa área no cuidado de seus pacientes. As tarefas modeladas e disponibilizadas, inicialmente, baseiam-se nos resultados do estudo realizado por Laerum e Faxvaag (2004). Nesse estudo buscou-se a

identificação de um conjunto de atividades realizadas pelos profissionais clínicos nos ambientes hospitalares, utilizando para isso questionários e entrevistas que foram aplicados aos profissionais dessa área.

Nessa dissertação, apresentam-se as abstrações para a representação das Tarefas Clínicas modeladas anteriormente. As abstrações disponibilizadas visam facilitar a interação do profissional clínico com o sistema. Assim, as tarefas já modeladas são representadas e abstraídas através de componentes gráficos que remetem à sua funcionalidade. Permite-se, ao profissional, a criação e definição de tarefas que julgar necessário, seguindo sua própria forma de realizar as atividades profissionais diárias (personalização). Para a edição e criação de novas tarefas implementou-se o Módulo de Edição de Tarefas que utiliza as abstrações e representações gráficas para facilitar a interação com o usuário. Essa interface gráfica foi modelada com o intuito de facilitar o processo de interação com o profissional clínico e, assim, possibilitar que esse não precise ter conhecimentos de computação e de programação para a modelagem das tarefas.

Os mecanismos utilizados visam permitir que profissionais clínicos utilizem a ferramenta para personalizar as tarefas clínicas executadas no cuidado de seus pacientes. Assim, tem-se uma visão centralizada no usuário (User-Center Computing) do sistema já que esse poderá utilizar o seu próprio conhecimento para a criação e definição de outras tarefas. Dessa forma, permite-se que cada profissional programe as suas próprias tarefas e, com isso, defina a sua forma particular de executar as tarefas de cuidados clínicos.

O Módulo de Edição de Tarefas foi desenvolvido utilizando-se a tecnologia *Java Standard Edition* (J2SE) e também a API *Drag and Drop* da linguagem Java. Além disso, foram utilizados mecanismos providos pela Programação Orientada ao Usuário-Final visando facilitar o processo de interação do profissional clínico com a interface gráfica de Edição de Tarefas.

O trabalho realizado é o primeiro passo para a criação da Arquitetura ClinicSpace e deixa margem para melhoramentos e trabalhos futuros. Entre eles, destacam-se:

- Aperfeiçoamento Visual da Interface. Estudar a possibilidade e viabilidade de aperfeiçoar o processo de definição das imagens que representam os elementos do modelo (subtarefas, tarefas e fluxos), melhorando o aspecto visual da interface. Como já foi detectado que é possível a representação das Tarefas através de ícones gráficos que remetem à funcionalidade que determinada tarefa apresenta, o processo de definição das imagens pode ser mais explorado visando facilitar ainda mais a interação com o profissional clínico.

- Interface Intuitiva, usando técnicas de Interface Humano-Computador (IHC). A interface gráfica de interação com o profissional clínico foi modelada visando a disponibilização de um protótipo simples, que permitisse demonstrar a utilidade da ferramenta sendo proposta. A partir disso, novas melhorias e mecanismos de IHC podem ser implementados, objetivando tornar a Interface mais intuitiva para os clínicos. Para isso, estudos aprofundados sobre como modelar interfaces gráficas intuitivas devem ser realizados. Técnicas e mecanismos apresentados pela Programação orientada ao usuário-final devem ser utilizados novamente;
- Avaliação da Interface pelos Usuários-Finais. Previsto no projeto ClinicSpace, a ferramenta-piloto deve ser avaliada em campo, pelos médicos, após a implementação da Arquitetura de Gerenciamento das Tarefas e a inserção de um sistema pervasivo de informação de saúde - pEHS (VICENTINI, 2009) nessa arquitetura. O objetivo desse teste de campo será apresentar aos usuários a arquitetura ClinicSpace, suas funcionalidades, e avaliar a utilidade e impacto da ferramenta proposta sob o ponto de vista dos profissionais clínicos da área médica, aos quais a ferramenta se destina;
- Aperfeiçoamento das Regras de Associação de Tarefas. As Regras de Associação (descritas na subseção 4.3.1) devem ser melhoradas com o intuito de refinar o processo de tratamento da associação de tarefas. As melhorias podem ser na forma como são verificadas a associação de duas determinadas tarefas ou fluxos. Um exemplo disso é que, atualmente, verifica-se se a última subtarefa da primeira tarefa ou fluxo é compatível com primeira subtarefa da segunda tarefa ou fluxo. Além dessa política utilizada atualmente, outras podem ser acrescentadas e, com isso, ter-se uma verificação de associação mais otimizada;
- Compartilhamento de Tarefas. Permitir aos profissionais clínicos compartilhar com outros profissionais as tarefas e os fluxos criados por ele. O sistema poderá prover mecanismos para que o usuário, no momento da criação de alguma tarefa ou fluxo, informe se deseja ou não compartilhá-la com outros usuários clínicos. Além disso, prover mecanismos para acessar tarefas criadas e compartilhadas por outros profissionais e, com isso, utilizá-las como base de conhecimento para a criação das suas próprias.

O trabalho foi concluído atingindo satisfatoriamente os objetivos propostos. Suas contribuições ocorrem na definição da arquitetura do projeto ClinicSpace, e no primeiro passo para a disponibilização da ferramenta-piloto prevista no projeto. Como o projeto continua em desenvolvimento e, à medida que questões são aprofundadas, novas revisões dos conceitos

adotados poderão ser necessárias.

6.1 Publicações

Os temas desenvolvidos nesse trabalho, modelagem de tarefas clínicas e a prototipação de uma ferramenta-piloto para a Personalização de Tarefas, geraram e influenciaram diversas publicações. São elas:

WPUC 2008 – "Introduzindo a Orientação a Tarefas Clínicas em um Middleware de Gerenciamento do Espaço Pervasivo". In: II Workshop on Pervasive and Ubiquitous Computing, 20th International Symposium on Computer Architecture and High Performance Computing.

CAPSI 2008 – "Uma Ontologia aplicada a um Ambiente Pervasivo Hospitalar". In: 8ª Conferência da Associação Portuguesa de Sistemas de Informação.

WIM 2009 – "Introduzindo o Gerenciamento de Tarefas Clínicas em um Middleware da Computação Pervasiva". In: IX Workshop de Informática Médica, XXIX Congresso da Sociedade Brasileira de Computação. (*aceito para publicação*)

SBSI 2009 – "A definição de uma API para o Processamento de Ontologias em Hospitais Pervasivos". In: V Simpósio Brasileiro de Sistemas de Informação. (*aceito para publicação*)

WMUPS 2009 – "Middleware for Management of End-user Programming of Clinical Activities in a Pervasive Environment". In: 2009 Workshop on Middleware for Ubiquitous and Pervasive Systems, Fourth International Conference on Communication System Software and Middleware. (*aceito para publicação*)

CLEI 2009 – "Ferramenta para a Programação pelo Usuário-Final de Tarefas Clínicas em um Ambiente de Saúde Ubíquo". In: XXXV Conferência Latino-Americana de Informática. (*aceito para publicação*)

REFERÊNCIAS BIBLIOGRÁFICAS

AUGUSTIN, I. **Abstrações para uma Linguagem de Programação visando Aplicações Móveis em um Ambiente de Pervasive Computing.** 2004. Tese (Doutorado em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

AUGUSTIN, I. et al. ISAM, Joing Context-awareness and Mobility to Building Pervasive Applications. **In: Móbile Computing Handbook**, 2004.

AUGUSTIN, I.; LIMA, J. C. D.; YAMIN, A. C. Computação Pervasiva: como Programar Aplicações. In: X SIMPOSIO BRASILEIRO DE LINGUAGENS DE PROGRAMAÇÃO (SBLP), 2006, Itatiaia, RJ. Anais... [S.l.]: SBLP, 2006.

BANAVAR, G.; BERNSTEIN, A. Software Infrastructure and Design Challenges for Ubiquitous Computing. **Communications of the ACM**, vol. 45, issue 12, p. 92-96, 2002.

BARDRAM, J. E.; CHRISTENSEN, H. B. Real-time collaboration in Activity Based Architectures. **Proceedings of the Fourth Working IEEE**, p. 325, 2004.

BARDRAM, J. E.; CHRISTENSEN, H. B.; OLSEN, K. Activity-Driven Computing Infrastructure-Pervasive Computing in Healthcare. Techinal Report CfPC-2004-PB-65, Center for Pervasive Computing, 2004.

BARDRAM, J. E.; CHRISTENSEN, H. B. Pervasive Computing Support for Hospitals: An overview of the Activity-Based Computing Project. **IEEE Pervasive Computing**, vol. 6, issue 1, p. 44-51, 2007.

BERTI, S.; PATERNÒ, F.; SANTORO, C. Natural development of ubiquitous interfaces. **Communications of the ACM**, vol. 47, issue 9, p. 63-64, 2004.

BOLIN, M. **End-user programming for the web.** Tese de Mestrado. 2004. Massachusetts Institute of Technology, 2005.

CARDELLI, L.; GORDON, A. D. Mobile Ambients. First International Conference on Foundations of Software Science and Computation Structure, p. 140-155, 1998.

CHALMERS, D. et al. Ubiquitous Computing Experience, Design and Science. 2007. Disponível em <http://www-dse-doc.ic.ac.uk/Projects/UbiNet/GC/index.html>. Acesso em: 10 set. 2007.

CHRISTENSEN, H. B.; BARDRAM, J. Supporting Human Activities – Exploring Activity-Centered Computing. **Proceedings of the 4th International Conference on Ubiquitous Computing**, p. 107-116, 2002.

CPH. Center for Pervasive Healthcare. 2007. Disponível em <http://www.cfph.dk>. Acesso em 14 jun 2007

FERREIRA, G. L. et al. Middleware for Management of End-user Programming of Clinical Activities in a Pervasive Environment. **In: 2009 Workshop on Middleware for Ubiquitous and Pervasive Systems**, Fourth International Conference on Communication System Software and Middleware, 2009.

GARLAN, D.; STEENKISTE, P.; SCHRMEL, B. Project Aura: Toward Distraction-free Pervasive Computing. **IEEE Pervasive Computing,** vol. 1, issue 2, p. 22-31, 2002.

HAC, A. Wireless Sensor Network Designs. 2007. Disponível em < http://www.cs.sysu.edu.cn/selab/references/WSN/A.Hac(2003)Wireless%20Sensor%20Network%20Design.pdf >. Acesso em 12 set 2007.

HAGUE, R.; ROBINSON, P.r; End-User Programming of Reconfigurable Systems: Experiences with Auto-Adaptative and Reconfigurable Systems. **Software-Pratice and Experience**, vol. 3, issue 11-12, p. 1-2, 2006.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Infrastructure for Pervasive Computing: Challenges. **Workshop on Pervasive Computing,** INFORMATIK 01, p. 214-222, 2001.

HUMBLE, J. Playing whith your bits: user-composition of ubiquitous domestic environments. **Proceedings of the 5th Annual Conference on Ubiquitous Computing,** UbiComp, 2003.

JANSEN, E. et al. A Programming Model for Pervasive Spaces. **International Conference on Service-Oriented Computing**, p. 86, 2005.

KAENAMPORNPAN, M.; O'NEILL, E. Integrating History and Activity Theory in Context Aware System Design. In: 1ST INTERNATIONAL WORKSHOP ON EXPLOITING CONTEXT HISTORIES IN SMART ENVIRONMENTS, 11 may 2005, Munich. **Anais eletrônicos...** [S.l.:s.n], 2005. Disponível em: <http://www.ipsi.fraunhofer.de/ambiente/echise2005/downloads/echise2005-proceedings.pdf>. Acesso em: 17 jun. 2008.

KOMNINOS, A.; STAMOU, S. HealthPal: Na Intelligent Personal Medical Assistant for Supporting the Self-Monitoring of Healthcare in the Ageing Society. In **Proceedings of UbiHealth 2006: The 4th International Workshop on Ubiquitous Computing for Pervasie Healthcare Applications**, UbiHealth 2006**,** 2006**.**

KUMAR, A. et al. Context-based task ontologies for clinical guidelines. **Ontologies in Medicine: Proceedings of the Workshop on Medical Ontologies,** IOS Press, 2003.

LAERUM, H.; FAXVAAG, A. Task-oriented evalution of electronic medical record systems: development and validation of a questionnaire for physicians. **BMC Medical Informatics and Decision 2004**, 4:1, 2004.

LIBRELOTTO, G. R. et al. OntoHealth - Um framework para o gerenciamento de ontologias em ambientes hospitalares pervasivos. In: II WORKSHOP ON PERVASIVE AND UBIQUITOUS COMPUTING, 2008, Campo Grande. **Anais...** [S.l.:s.n], 2008.

MASUOKA, R.; PARSIA, B.; LABROU, Y. Task Computing – the semantic web meets pervasive computing. **2nd International Semantic Web Conference**, ISWC, Filadélfia, vol. LNCS 2870, p. 866-881, 2003.

MCGUINENESS, D. L.; HARMELEN, F. V. OWL – Web Ontology Language overview. Disponível em < http://www.w3.org/TR/owl-features >. Acesso em: 24 ago. 2008.

MYERS, B. et al. Invited research overview: end user programming. **CHI'06 extended abstracts on Human factors in computing systems**, ACM Press, p. 75-80, 2006.

MOOCK, C. **Action Script for Flash MX**: the Definitive Guide 2e. O'Reilly & Associates, 2002. 1088p.

MORIARTY, P. et al. Graphical computing in the undergraduate laboratory: Teaching and interfacing with LabVIEW. **American Journal of Physics**, 71 (10), p. 1062 – 1074, 2003.

PANE, J. F. **A programming system for children that is designed for usability.** 2002. Tese de Phd - Carnegie Mellon University, Pittsburg, 2002.

RANGANATHAN, A. et al. Olympus: a High-Level Programming Model for Pervasive Computing Environments. **3rd IEEE International Conference on Pervasive Computing and Communications**, Percom, p. 7-16, 2005.

RANGANATHAN, A. et al. Towards a Pervasive Computing Benchmark. **3rd International Conference on Pervasive Computing and Communications Workshops**, PerCom, p. 194-198, 2005a.

RANGANATHAN, A.; CAMPBELL, R. Supporting Tasks in a Programmable Smart Home. **3rd International Conference on Smart Homes and Health Telematic**, 2005.

RIZZETTI, T. Framework para gerenciamento e personalização de contexto orientado à Tarefas. 2009 (Trabalho em andamento).

ROMAN, M. et al. Gaia: a Middleware Infrastructure to Enable Active Spaces. **IEEE Pervasive Computing**, v. 1, issue. 4, p. 65-67, 2002.

ROTHERMEL, G.; LI, L.; BURNETT, M. Testing strategies for form-based visual programs. **Proceedings of the Eighth International Symposium on Software Reliability Engineering**, ISSRE'97, IEEE Computer Society, p. 96-107, 1997.

SAHA, D.; MUKHERJEE, A. Pervasive Computing: a Paradigm for the 21st Century. **IEEE Computer**, vol. 36, issue 12, p. 25-31, 2003.

SATYNARAYANAN, M. Pervasive Computing: Vision and Challenges. **IEEE Personal Communications,** vol. 8, issue 4, p. 10-17, 2001.

SMITH, D. et al. Kidsim: programming agents without a programming language. **Communications of the ACM,** vol. 37, issue 7, p. 54-67, 1994.

SONG, Z.; LABROU, Y.; MASUOKA, R. Dynamic Service Discovery and Management in Task Computing. **First Annual International Conference Mobile and Ubiquitous Systems**: Networking and Services, MobiQuitous'04, p. 310-318, 2004.

TRUONG, K. N.; HUANG, E. M.; ABOWD, G. D. Camp: A magnetic poetry interface for end-user programming of capture applications for the home. **UbiComp 2004**: **Ubiquitous Computing**, p. 143-170, 2004

VARSHNY, U. Pervasive Healthcare. **IEEE Computer**, vol 36, issue 12, p. 138-140, 2003.

VICENTINI, C. Proposta de Arquitetura para Sistemas de Registro de Saúde Ubíquo. 2009 (Trabalho em andamento).

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional direcionado às Aplicações Distribuídas.** 2004. Tese (Doutorado em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

YAMIN, A. et al. EXEHDA: adaptative middleware for building a pervasive grid environment. **Frontiers in Artificial intelligence and Applications** – Self – organization and Automatic Informatics, IOS Press, vol. 135, p. 203-219, 2005.

WEISER, M. The Computer of the 21st Century. **Scientific American,** vol. 3, issue 3, p. 3-11, 1991.

WHEELER, K. HyperTalk: The Language for the Rest of Us. Disponível em < http://www.memoryhole.net/~kyle/papers/hypertalk.pdf >. Acesso em: 05 dez. 2007.

WOLEK, N. A Granular Toolkit for Cycling74's Max/MSP. **In: SEAMUS 2002 National Conference**, vol. XVI:2, p. 34-46, 2002.

APÊNDICE A – Conjunto mínimo de tarefas e as subtarefas que as compõem.

A seguir são listadas as subtarefas que compõem as tarefas pertencentes ao conjunto mínimo de Tarefas. As subtarefas de Identificação são implementadas como solicitações ao módulo Contexto de Tarefas para identificação do profissional e/ou do paciente. As subtarefas "pEHS" são chamadas às aplicações do sistema pEHS, que podem ser feitas em *background* (quando não necessitam a interação do usuário) ou através de interfaces gráficas (quando é necessário a interação com o usuário).

1. Revisar os Problemas do Paciente.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Buscar Informação do Paciente;
    4. pEHS - Visualizar Informações.

2. Procurar Informações Específicas no Registro do Paciente.
    1. Identificar Profissional clínico;
    2. Identificar Paciente;
    3. pEHS - Buscar Informação do Paciente (Parâmetro: Informação requisitada);
    4. pEHS - Visualizar Informações.

3. Obter os resultados de novos Testes e Investigações.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Buscar Informação do Paciente (Parâmetro: Informações novas);
    4. pEHS - Visualizar Informações.

4. Adicionar Notas Diárias sobre Condições do Paciente.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Registrar Nota diária.

5. Requisitar Análises Laboratoriais.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Preencher requisição Laboratorial*.

6. Requisitar Exames de Vídeo/Imagem (raio-x, tomografia, etc.).
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Preencher requisição de exames de vídeo/imagem*.

7. Obter Resultados Laboratoriais.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Buscar Informações do Paciente (Parâmetro: Requisições Laboratoriais);
    4. pEHS - Visualizar Informações.

8. Obter Resultados de Exames de Vídeo/Imagem.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Buscar Informações do Paciente (Parâmetro: Requisições de vídeo/imagem);
    4. pEHS - Visualizar Informações.

9. Requisitar Tratamentos
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Preencher requisição de tratamento*;

10. Escrever Prescrições médicas
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Preencher prescrição**.

11. Registrar códigos para diagnóstico e procedimentos executados.
    1. Identificar Profissional;
    2. Identificar Paciente;
    3. pEHS - Registrar diagnósticos e procedimentos executados.

* Engloba as etapas de preenchimento da requisição, encaminhamento para os setores responsáveis e registro no Prontuário Eletrônico do Paciente armazenada no sistema eletrônico de saúde (EHR pervasivo).

** Engloba as etapas de preenchimento da prescrição, registro no Prontuário Eletrônico do Paciente e Impressão da prescrição.