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

ADICIONANDO AO MIDDLEWARE EXEHDA O SUPORTE A APLICAÇÕES ORIENTADAS A ATIVIDADES HUMANAS COTIDIANAS

DISSERTAÇÃO DE MESTRADO

Giuliano Geraldo Lopes Ferreira

Santa Maria, RS, Brasil
2009

ADICIONANDO AO MIDDLEWARE EXEHDA O SUPORTE A APLICAÇÕES ORIENTADAS A ATIVIDADES HUMANAS COTIDIANAS

por

Giuliano Geraldo Lopes Ferreira

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 Computação**

Orientadora: Prof.ª 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

ADICIONANDO AO MIDDLEWARE EXEHDA O SUPORTE A APLICAÇÕES ORIENTADAS A ATIVIDADES HUMANAS COTIDIANAS

elaborada por
Giuliano Geraldo Lopes Ferreira

como requisito parcial para obtenção do grau de
Mestre em Computação

COMISSÃO EXAMINADORA:

Iara Augustin, Dra.
(Presidente/Orientadora)

Taisy Silva Weber, Dra. (UFRGS)

Benhur de Oliveira Stein, Dr. (UFSM)

Santa Maria, 03 de julho de 2009.

RESUMO

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

ADICIONANDO AO MIDDLEWARE EXEHDA O SUPORTE A APLICAÇÕES ORIENTADAS A ATIVIDADES HUMANAS COTIDIANAS

Autor: Giuliano Geraldo Lopes Ferreira
Orientadora: Iara Augustin
Data e Local da Defesa: Santa Maria, 03 de julho de 2009

Atualmente, a Computação Pervasiva está direcionada ao desenvolvimento de ambientes programáveis e interativos, os quais auxiliarão o usuário em suas atividades diárias. O sistema de saúde do futuro prevê o uso da Computação Pervasiva como forma de otimizar e automatizar as atividades clínicas. Tendo em vista essa perspectiva, neste trabalho buscou-se adicionar, a um *middleware* de gerenciamento do ambiente pervasivo, o suporte ao gerenciamento e à execução de tarefas clínicas (aplicações pervasivas que auxiliam o clínico a realizar suas atividades), atendendo a alguns requisitos da computação orientada a atividades, e criando uma ferramenta para auxiliar os clínicos em suas atividades diárias. Na modelagem do sistema de gerenciamento de tarefas, o *middleware* EXEHDA – *Execution Environment for Highly Distributed Applications* – foi utilizado para gerenciar o ambiente pervasivo no qual as tarefas irão executar. Devido às características flexíveis do EXEHDA quanto à integração de novos serviços, o suporte do *middleware* ao gerenciamento de tarefas foi adicionado através de um novo subsistema, composto pelos serviços que implementam as novas funcionalidades. Portanto, a principal contribuição deste trabalho foi a modelagem da arquitetura do novo subsistema do EXEHDA, responsável pelo gerenciamento de tarefas no ambiente pervasivo. Como contribuição secundária, foi desenvolvido um protótipo do núcleo desse novo subsistema (um serviço para gerenciamento de tarefas), já prevendo certa interação com os demais serviços modelados para a arquitetura. Esse protótipo poderá ser usado, durante o andamento do projeto ClinicSpaces, como base para teste e avaliação dos outros serviços, bem como, para facilitar a participação de profissionais de outras áreas no prosseguimento do projeto.

Palavras-chave: computação pervasiva; computação ubíqua; middleware; computação orientada a atividades; tarefas clínicas; gerenciamento de tarefas.

ABSTRACT

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

ADDING TO EXEHDA MIDDLEWARE THE SUPPORT FOR APPLICATIONS ORIENTED TO DAILY HUMAN ACTIVITIES

Author: Giuliano Geraldo Lopes Ferreira
Advisor: Iara Augustin
Santa Maria, July 03, 2009

Currently, Pervasive Computing is focused on the development of programmable and interactive environments, which are intended to help the user in daily activities. The health system of the future envisages the use of Pervasive Computing as a way of optimizing and automating clinical activities. Under such perspective, the present study has tried to add, in a middleware for pervasive environment management, the supporting for management and accomplishment of clinical tasks (pervasive applications that help physicians perform their activities), fulfilling some requirements of activities-oriented computing, and creating a tool that will help physicians in their daily tasks. In modeling of the tasks management system, the middleware EXEHDA – Execution Environment for Highly Distributed Applications – has been used to manage the pervasive environment where tasks will be performed. Due to the flexible features of EXEHDA, as the integration of new services, the supporting of the middleware to the new tasks management was added as a new subsystem, composed by the services that implement the new features. So, the main contribution of present study was the modeling of the architecture for the new subsystem of EXEHDA, responsible for management of tasks in pervasive environment. As a secondary contribution, was developed a prototype of the core of this new subsystem (a service for managing tasks), already providing same integration within other services modeled for the architecture. This prototype can be used, during the course of the ClinicSpaces Project, as a basis for testing and evaluation of other services and to facilitate the participation of professionals from other areas in the project.

Key-words: pervasive computing; ubiquitous computing; middleware; task-driven computing; clinical activities; task management.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AVA	Ambiente Virtual da Aplicação
AVU	Ambiente Virtual do Usuário
BDA	Base da Dados pervasiva das Aplicações
CIB	*Cell Information Base* - Base de Informações da Célula
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
J2ME	Java Micro Edition
J2SE	Java Standard Edition
JVM	*Java Virtual Machine* - Máquina Virtual Java
OX	Objeto eXehda
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

Inspirado por cientistas sociais, filósofos e antropólogos, Mark Weiser e demais colaboradores dos laboratórios do Xerox PARC sintetizaram o que a computação deveria se tornar no futuro: algo invisível para o usuário-final. Esta idéia foi difundida em um artigo (WEISER, 1991). A partir daí nasce uma nova área de pesquisa, a Computação Ubíqua, ou simplesmente Ubicomp. Outros nomes são considerados sinônimos derivados dessas idéias: Computação Pervasiva, Computação Invisível e Inteligência Ambiente (*ambient intelligence)*. Nesse novo contexto, a computação é imersa no cotidiano da sociedade de tal forma a prover a colaboração transparente entre dispositivos do meio para o fornecimento de serviços a seus usuários e proporcionar um novo cenário de simbiose entre a vida social, cotidiana e tecnológica.

O campo da Computação Ubíqua ou Pervasiva está progredindo rapidamente. Apesar de a visão de Weiser parecer utopia na época e ainda estar longe de tornar-se realidade, a disponibilidade atual de comunicação sem fio e dispositivos portáteis tem oportunizado aos pesquisadores meios para criar soluções e protótipos para identificar as necessidades e requisitos tanto na visão de infra-estrutura, sistema e aplicação, como na visão do usuário.

O atual foco da Computação Pervasiva volta-se ao processamento das atividades humanas cotidianas. Projetos de pesquisa em Computação Pervasiva, como Aura (GARLAN et al, 2002) e Gaia (ROMAN et al, 2002), que introduziram os conceitos de Computação Orientada a Tarefas/Atividades (*Task-Oriented* ou *Activity-Driven Computing*), abordam os problemas relativos a um gerenciamento das tarefas do usuário de forma pró-ativa, os quais induzem o usuário a agir de acordo com a forma pré-definida e pré-programada do sistema pervasivo. Porém, considera-se que a pró-atividade do sistema não deve ser tão rigorosa, uma vez que este é projetado de forma genérica e não personalizada. Se o usuário desejar fazer de forma diferente, ele deve ter possibilidade de interagir, comandar e influenciar a execução das suas tarefas/atividades gerenciadas pelo sistema pervasivo. Assim, ambientes programáveis e interativos introduzem a segunda geração de sistemas ubíquos (2ªG). Novas aplicações requerem usabilidade, utilidade e ubiqüidade (AUGUSTIN; LIMA; YAMIN, 2006).

Dentro desse escopo, o grupo de pesquisa em Sistemas da Computação Móvel e Pervasiva (GMob) está desenvolvendo o projeto "ClinicSpaces: auxílio às tarefas clínicas em um ambiente hospitalar do futuro baseado em tecnologias da Computação Ubíqua/Pervasiva". O projeto propõe o desenvolvimento de uma ferramenta de software que permita aos médicos a personalização de suas tarefas clínicas, as quais são gerenciadas e executadas em um ambiente pervasivo/ubíquo. Esse ambiente será disponibilizado e gerenciado pelo *middleware* EXEHDA – *Execution Environment for Highly Distributed Applications* (EXEHDA, 2001), desenvolvido dentro do projeto ISAM – Infra-estrutura de Suporte às Aplicações Móveis (ISAM, 2001).

1.1 O Problema e os Objetivos

A introdução do conceito de orientação a tarefas (*activity-driven or task-oriented computing*) no ambiente pervasivo gerenciado pelo *middleware* EXEHDA (YAMIN et al, 2005) exige a inserção de um novo subsistema, chamado de Subsistema de Gerenciamento Distribuído de Tarefas, responsável por fazer a "tradução" (conversão) das tarefas definidas pelo usuário-final (clínico) para aplicações pervasivas, gerenciadas pelo *middleware* EXEHDA.

Assim, a proposta desta dissertação é modelar o novo Subsistema de Gerenciamento Distribuído de Tarefas para o *middleware* EXEHDA, que atua como uma camada intermediária entre a ferramenta orientada a tarefas, disponibilizada ao usuário-final, e os serviços atuais do *middleware*. Dessa forma, as tarefas são mapeadas em aplicações pervasivas, as quais fazem uso dos serviços do *middleware* para sua execução. Portanto, o novo subsistema faz o gerenciamento, em alto nível, de tarefas, deixando o gerenciamento das aplicações pervasivas para os outros subsistemas do EXEHDA.

Mais especificamente, os objetivos dessa dissertação são: (i) analisar a modelagem dos subsistemas atuais do EXEHDA, bem como o modo de interação entre eles, identificando os pontos que o novo subsistema terá que tratar; (ii) projetar e modelar o novo Subsistema de Gerenciamento Distribuído de Tarefas, de modo que esse atenda as necessidades introduzidas pelo conceito de orientação a tarefas; (iii) implementar e documentar um protótipo do núcleo[1] do novo subsistema, ou seja, o serviço que irá gerenciar a execução das tarefas, permitindo

[1] Outros serviços serão implementados por outros trabalhos.

que a arquitetura ClinicSpaces evolua sobre algo concreto, facilitando o entendimento e a participação de profissionais de outras áreas na execução do projeto.

1.2 Apresentação do texto

O restante do texto está organizado da seguinte maneira. No capítulo 2, são discutidas questões sobre a Computação Ubíqua na Saúde e é apresentada a infra-estrutura para gerenciar um ambiente ubíquo. Esse capítulo apresenta um estudo realizado sobre o projeto ISAM e o *middleware* EXEHDA, o qual será usado como base para o projeto ClinicSpaces. Em seguida, é feita uma breve apresentação desse projeto e seus objetivos.

O capítulo 3 apresenta como a orientação a tarefas foi aborda no projeto ClinicSpaces. Nesse capítulo, são discutidas questões que não são o foco desta dissertação, mas que estão estreitamente relacionadas com ela. Essas questões envolvem a modelagem de tarefas, as políticas adotadas no projeto e a interação do usuário com o sistema, entre outras. O capítulo também discute os outros trabalhos com essa temática, que estão sendo desenvolvidos paralelamente a este. Além disso, esse capítulo mostra uma visão geral da arquitetura projetada para o sistema ClinicSpaces e apresenta três casos de uso, nos quais o usuário interage com o sistema.

O capítulo 4 discute a modelagem do novo Subsistema de Gerenciamento Distribuído de Tarefas, apresentando os novos serviços do *middleware*, suas funcionalidades, e mostrando como esses novos serviços irão interagir entre si e com os outros subsistemas do EXEHDA. Além disso, também são discutidos detalhes da implementação do protótipo do núcleo do Subsistema de Gerenciamento Distribuído de Tarefas.

No capítulo 5, são discutidos trabalhos relacionados, fazendo-se um comparativo entre o que existe e o que foi desenvolvido neste trabalho. Neste capítulo também são apresentados os resultados obtidos com testes e avaliações realizados sobre um protótipo do Subsistema de Gerenciamento Distribuído de Tarefas.

Finalmente, no capítulo 6 são feitas as considerações finais, bem como, são sugeridas idéias para melhoramento do protótipo e para implementação de novas funcionalidades. Este capítulo resume o que foi realizado neste trabalho e quais os temas que ficaram em aberto e que estão sendo aprofundados em outros estudos.

2 COMPUTAÇÃO UBÍQUA/PERVASIVA NA SAÚDE

Computação Pervasiva (*Pervasive Computing*) é um novo paradigma computacional com tecnologia de comunicação e informação em qualquer lugar, acessível por qualquer pessoa, disponível todo o tempo (SAHA; MUKHERJEE, 2003). Os recursos computacionais estarão integrados ao ambiente físico da forma mais transparente possível (WEISER, 1991). Os dispositivos serão portais que permitirão ao usuário ter acesso ao seu ambiente computacional (aplicações e dados), dentro de seus limites de capacidade. Neste capítulo abordam-se as gerações da Computação Pervasiva/Ubíqua e o seu uso na área de Saúde.

2.1 Gerações da Computação Pervasiva/Ubíqua

A Computação Pervasiva pressupõe a integração do mundo físico ao mundo lógico da computação através da criação de Espaços Pervasivos. Esses espaços são locais impregnados de dispositivos e computadores totalmente integrados ao ambiente, de forma tão invisível quanto possível ao usuário final, para captar, interpretar e agir pró-ativamente, conforme as alterações observadas no ambiente (AUGUSTIN et al, 2006).

A primeira geração de ambientes para computação pervasiva focalizou *middlewares* para gerenciamento e disponibilização do ambiente com objetivo de entender os requisitos e necessidades destes. Os sistemas existentes permitiram levantar alguns dos conceitos inerentes à natureza do espaço pervasivo: comunicação, mobilidade, contexto e tarefas. Comunicação e mobilidade já são tratadas há muito tempo; contexto está sendo tratado atualmente; porém, tarefas exigem um esforço maior, pois estão no estágio de definição do que se deseja (AUGUSTIN; LIMA; YAMIN, 2006).

Recentemente, 2ª Geração, busca-se encontrar abstrações de alto nível que permitam projetar e programar aplicações que comporão um novo paradigma denominado Programação Orientada a Tarefas (*Task-Oriented Programming*) (AUGUSTIN; LIMA; YAMIN, 2006).

2.1.1 Características do Ambiente Ubíquo

Hoje, existem muitos dispositivos móveis que podem ser usados pelo usuário, porém, ainda é necessária sua intervenção para criar, manter e migrar seu ambiente de trabalho entre os dispositivos. O modelo de Computação Nômade (BAGRODIA et al, 1995), no qual o usuário utiliza um dispositivo portátil que armazena as aplicações e os dados, e que necessita de sincronizações periódicas com equipamento externo de maior porte, não atende mais à demanda de serviços requeridos pela dinamicidade da mobilidade física do usuário portando ou não seu dispositivo móvel.

Nessa perspectiva, o ambiente de execução deve gerenciar os componentes do meio computacional (código, dados, equipamentos, serviços) de forma a viabilizar o acesso a eles independentemente do equipamento utilizado, do local ou do momento do acesso, ou seja, o ambiente de execução deve viabilizar *acesso pervasivo* aos seus componentes (YAMIN, 2004).

Além disso, as aplicações no ambiente computacional devem atender à semântica *siga-me* (*follow-me*), a qual define que as aplicações são executadas no local onde o usuário se encontra, mesmo em deslocamento (AUGUSTIN; YAMIN; GEYER, 2005). Para isso, os usuários dispõem de um ambiente virtual dentro do ambiente computacional, que poderá ser acessado independentemente de localização ou de equipamento, sendo código e dados enviados sob demanda. Assim, o código da aplicação deve ser adaptável ao contexto corrente do usuário.

Portanto, pode-se dizer que o ambiente da Computação Pervasiva tem por premissas (*i*) a possibilidade de o usuário disparar aplicações a partir de qualquer nodo integrante do sistema e, após o disparo, (*ii*) a mobilidade parcial ou total de tais aplicações em resposta a modificações em seu contexto de execução, como a movimentação do usuário ou a alteração da disponibilidade dos dispositivos em uso pela aplicação (YAMIN, 2004) (AUGUSTIN et al, 2002).

Esse cenário permite o acoplamento do mundo físico ao mundo da informação e fornece uma abundância de serviços e aplicações ubíquos visando a que usuários, máquinas, dados, aplicações e objetos do espaço físico interajam uns com os outros de forma transparente (em *background*) (RANGANATHAN et al, 2005).

2.1.2 Computação Orientada a Atividades

Para se definir um sistema pervasivo é necessário que este esteja centrado no usuário final e forneça suporte às tarefas de seu trabalho diário - *Task-Oriented Computing* (CHRISTENSEN; BARDRAM, 2002). Também, a computação deve ser o mais invisível possível ao usuário final. Por um lado, esses sistemas pervasivos devem ser pró-ativos (agir em nome do usuário), por outro lado, um usuário, ao executar seu trabalho, deseja manter sua forma de executá-lo, sendo necessário que o sistema forneça mecanismos que lhes permitam interagir e personalizar o sistema, adequando-o melhor a sua forma de executar as tarefas.

Essa motivação já foi identificada há muito tempo na Computação, originando a área de *End-User Programming*, tendo vários resultados de estudos, mas permanece um desafio e é uma área ativa de pesquisa (GOODELL, 2005). A importância dessa área para o sucesso da Computação Pervasiva começa a ser reconhecida, e estudos recentes abordam o problema relacionado a usuários que desejam programar os sistemas eletrônicos e tarefas rotineiras em casas inteligentes (RANGANATHAN; CAMPBELL, 2005a) (KALOFONOS; REYNOLDS, 2006).

É necessário, portanto, um novo modelo baseado em tarefas e um modelo navegacional para estruturar os programas. Tarefas são definidas como um conjunto de ações (atividades) executadas colaborativamente por humanos e pelo sistema pervasivo para alcançar um objetivo. Uma tarefa pode ser composta, estática ou dinamicamente, por um número de subtarefas que podem ser unidas usando um conceito similar a *workflow* (AUGUSTIN; LIMA; YAMIN, 2006). Tarefas diferentes podem ter subtarefas comuns ou similares. Desta forma, o reuso de subtarefas já programadas é essencial, e o desacoplamento temporal e espacial (códigos e dados) deve ser modelado. Os conceitos de tarefas e subtarefas serão aprofundados no capítulo 3.

2.2 Uso da Computação Ubíqua na área da saúde

Uma das grandes áreas de aplicação de pesquisa e tecnologia da Computação Pervasiva é a Saúde, pela inerente natureza de suas atividades, onde o mundo físico é o gerador da informação que deve ser fornecida aos sistemas informatizados. A área de Saúde

possui muitos ambientes onde há necessidade de informações do mundo físico serem integradas pró-ativamente às soluções computacionais (mundo virtual). Essa é uma evolução da situação atual, na qual 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 (BARDRAM; BALDUS; FAVELA, 2006).

No ambiente hospitalar, as atividades variam de simples a complexas; algumas atividades são previsíveis e planejadas enquanto outras são aleatórias; algumas atividades têm prioridade enquanto outras podem ser feitas quando houver tempo; algumas atividades são ligadas a determinadas salas e presença de certos artefatos. O ambiente clínico também requer o acesso a um conjunto de variadas e atualizadas informações, que podem ser requeridas por vários profissionais ao mesmo tempo (AUGUSTIN; LIMA; YAMIN, 2006).

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 de ninguém, mas serve como um portal para acesso a elas (AUGUSTIN; LIMA; YAMIN, 2006). Esse conceito requer uma infra-estrutura que gerencia, armazena e distribui atividades computacionais. O procedimento de identificação e autenticação deve ser rápido, e o computador público recebe o ambiente virtual do usuário para que ele possa desenvolver suas atividades (YAMIN, 2004). 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 por natureza e o atendimento a um paciente envolve, em geral, várias especialidades (BARDRAM, 2003), (BARDRAM; CHRISTENSEN, 2007). Colaboração significa interromper a tarefa em execução para atender a solicitação por demanda.

A *Pervasive Healthcare* está sendo considerada a próxima etapa da *Web-based Healthcare Computing* que oferece vantagens competitivas aos provedores de serviços de saúde; em particular, aumenta a eficiência do serviço, a qualidade e melhora o gerenciamento da relação com o paciente (VARSHNEY, 2003).

O sistema de saúde do futuro prevê o uso de tecnologias da Computação Pervasiva formando um espaço inteligente, 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*) (VARSHNEY, 2003) (BARDRAM, 2003).

2.2.1 Situação Atual: uso da Computação Móvel em sistemas da Informática na Saúde

A Computação Móvel, que pode ser considerada a etapa anterior à Computação Pervasiva em termos de evolução dos sistemas de computação, já está presente em vários ambientes como hospitais e centros de saúde do país, particularmente no Instituto do Coração em São Paulo e no Hospital Albert Einstein. O uso de dispositivos móveis, como PDA's e tabletPCs, tem sido adotado para auxiliar as tarefas dos clínicos nesses locais, uma vez 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. Existe um vasto número de aplicações da Computação Móvel em ambientes hospitalares que ressaltam a melhora na qualidade das informações e satisfação dos pacientes, fortalecendo o processo de gestão e a tomada de decisão (ZENI et al, 2004). Algumas áreas de aplicação incluem: enfermagem, fármacos, educação médica, prescrição de medicamentos, informações clínicas sobre pacientes e funções administrativas.

Porém, médicos têm pouco tempo para se dedicar aos sistemas informatizados. Logo, demandam soluções de uso simplificado e interativo para se adequar às exigências da profissão.

2.2.2 O Hospital do Futuro

Uma tecnologia promissora é a proposta do ambiente computacional ubíquo de Mark Weiser (1991) que tenta inverter a situação atual da computação, a qual exige o conhecimento do usuário para manipular um computador, tornando a computação inserida no ambiente físico, em *background*, e centrada nas atividades rotineiras dos usuários, de tal forma que estes não precisam ter consciência de sua existência – a computação torna-se 'invisível', tão incorporada à rotina diária que desaparece do foco de atenção do usuário. Invisibilidade/Onipresença e centralização nas tarefas do usuário-final são os conceitos-pilares da Computação Ubíqua. Para atingir tal objetivo, muitos desafios ainda devem ser vencidos pela Computação.

Assim, o sistema computacional é visto como um assistente, o qual auxilia o médico a executar suas tarefas. As tarefas podem ser definidas com base em regras, fluxo de informação e execução disparada por eventos detectados por um sistema de reconhecimento de contexto.

Experiências nesse sentido estão sendo conduzidas por alguns projetos de pesquisa, como o do *Centre of Pervasive Healthcare* na Dinamarca que desenvolve o projeto *Hospital of the Future* (BARDRAM, 2005) (BARDRAM; BOSSEN, 2005) através do conceito de hospital interativo e o desenvolvimento de novos sistemas de computação clínica que introduzem uma coleção de serviços, como segurança, consciência do contexto e colaboração.

Como se vê, a Computação Pervasiva terá no futuro um enorme potencial de aplicabilidade na área da Saúde. Porém, é ainda uma realidade distante. Neste momento, em termos de pesquisa, a Computação Pervasiva na Saúde está em sua primeira geração, a qual procura entender as necessidades, características e tecnologias para projetar sistemas que criarão o hospital do futuro.

No Brasil, observa-se que o maior foco das pesquisas em Informática na Saúde é ainda em registros de pacientes e medicamentos, prontuário eletrônico e informação sobre cursos clínicos. São praticamente inexistentes projetos com o objeto de estudo definido na área de Computação Pervasiva; mas, internacionalmente, existem muitos projetos em andamento, conforme pode ser observado no UbiHealth 2006 (*4th International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*)[2], patrocinado pela IEEE. Alguns resultados disponíveis das pesquisas indicam que existe um alto potencial para otimizar e automatizar muitos fluxos de trabalho dentro dos hospitais (HOLZINGER; SCHWABERGER; WEITLANER, 2005), como, por exemplo, o tempo consumido para navegar entre registros dos pacientes.

2.3 Infra-estrutura para um ambiente ubíquo

Construir um ambiente pervasivo para o gerenciamento de tarefas do usuário pressupõe o desenvolvimento de softwares de alta complexidade. Por isso, optou-se por utilizar trabalhos já desenvolvidos como suporte ao desenvolvimento da arquitetura de serviços para gerenciamento de tarefas (AUGUSTIN et al, 2006), (YAMIN, 2004),

[2] UbiHealth 2006 – http://www.pervasivehealthcare.dk/UbiHealth2006

(AUGUSTIN et al, 2004b), (AUGUSTIN; YAMIN; GEYER, 2005), (FERREIRA, 2007). O *middleware* EXEHDA – *Execution Environment for Highly Distributed Applications* (EXEHDA, 2001), desenvolvido dentro do projeto ISAM – Infra-estrutura de Suporte às Aplicações Móveis (ISAM, 2001), é utilizado como base para o desenvolvimento da arquitetura de serviços necessários ao gerenciamento de tarefas, os quais serão integrados no EXEHDA, formando o novo Subsistema de Gerenciamento Distribuído de Tarefas.

2.3.1 Projeto ISAM e *Middleware* EXEHDA

O projeto ISAM – Infra-estrutura de Suporte às aplicações Móveis Distribuídas – (ISAM, 2001) (AUGUSTIN et al, 2002), vem criando uma infra-estrutura de suporte para projeto, implementação e execução de softwares pervasivos. A arquitetura ISAM provê um *ambiente virtual do usuário*, onde as aplicações têm o estilo *siga-me* (*follow-me*), permitindo ao usuário ter acesso ao seu ambiente computacional independentemente de localização e de tempo. Assim, ele atende às exigências desse novo cenário da Computação Pervasiva, onde co-existem a mobilidade física (equipamentos e/ou usuários) e a mobilidade do software (aplicações e serviços). No Apêndice A foram descritos mais detalhes sobre a arquitetura ISAM.

Integrando a arquitetura de software ISAM, foi implementado um *middleware* que visa criar e gerenciar um ambiente pervasivo, bem como promover a execução de aplicações que expressam a semântica *siga-me* sobre esse ambiente. Esse *middleware*, denominado EXEHDA – *Execution Environment for Highly Distributed Applications* – (YAMIN, 2004), é adaptável ao contexto e baseado em serviços, sendo chamado ISAMpe o ambiente por ele gerenciado.

2.3.2 Organização do Middleware EXEHDA

O *middleware* EXEHDA é estruturado em um núcleo mínimo com serviços carregados sob demanda, os quais estão organizados em subsistemas que gerenciam: (a) execução distribuída; (b) comunicação; (c) reconhecimento de contexto; (d) adaptação; (e) acesso pervasivo aos recursos e serviços; (f) descoberta de recursos; (g) gerenciamento de

recursos. O contexto é monitorado pró-ativamente, e o *middleware* permite que tanto a aplicação como ele próprio utilizem as informações de contexto para gerenciar a adaptação de seus aspectos funcionais e não-funcionais. Dessa forma, o mecanismo de adaptação do EXEHDA utiliza uma estratégia colaborativa entre a aplicação e o ambiente de execução para gerenciar o comportamento de cada componente da aplicação (YAMIN et al, 2002a). As políticas que irão reger os mecanismos de adaptação são especificadas no ambiente de desenvolvimento provido pelo ISAMadapt (AUGUSTIN, 2004).

As aplicações-alvo são distribuídas, adaptadas ao contexto em que executam e compreendem a mobilidade lógica e a física, sendo baseadas no modelo de programação ISAMadapt (AUGUSTIN, 2004). O ISAMadapt partiu das abstrações do Holoparadigma (BARBOSA et al, 2002) e adicionou novas abstrações e construções com uma semântica adequada ao ambiente pervasivo.

Na perspectiva do ISAM, entende-se por mobilidade lógica a reorganização dos componentes de software da aplicação, disparando ou não a migração de software. Por sua vez, a migração de software é quando algum componente da aplicação troca de equipamento hospedeiro. Já por mobilidade física, entende-se o deslocamento do usuário portando ou não um dispositivo móvel.

2.3.3 Suporte à semântica siga-me

No EXEHDA, o suporte à semântica *siga-me* é construído pela agregação de serviços relativos ao reconhecimento de contexto, ao acesso pervasivo e à comunicação (YAMIN, 2004). Dessa forma, o *middleware* possui dois mecanismos principais que dão suporte à semântica *siga-me*: (*i*) um mecanismo de monitoramento que permite inferir sobre o estado dos recursos e das aplicações; (*ii*) e outro que promove adaptações funcionais e não-funcionais de acordo com o contexto monitorado.

Entende-se por adaptação não-funcional a capacidade do *middleware* de modificar a localização física dos componentes das aplicações, seja pela migração do mesmo, ou pela instanciação remota dele. Por sua vez, a adaptação funcional é a capacidade do *middleware* de selecionar a implementação do componente, dentre as disponibilizadas no desenvolvimento, a ser utilizada em um determinado contexto de execução.

A instanciação remota do componente implica que o código a ser instanciado deve estar disponível através do acesso pervasivo a um repositório de código. Enquanto que o acesso ao ambiente computacional do usuário, independentemente de localização e de dispositivo empregado, implica o acesso pervasivo a seus dados e configurações pessoais.

2.3.4 O Ambiente Pervasivo ISAMpe

Do ponto de vista das aplicações pervasivas, o middleware EXEHDA é o provedor dos serviços que dão suporte às abstrações definidas no desenvolvimento de aplicações. A interação da aplicação com o ambiente computacional, através dos serviços disponibilizados pelo *middleware*, proporciona à aplicação a visão do ambiente pervasivo ISAMpe.

O ISAMpe é composto por três abstrações básicas: EXEHDAcel, EXEHDAbase e EXEHDAnodo. A EXEHDAcel denota a área de atuação de uma EXEHDAbase, sendo composta por essa e por EXEHDAnodos. A EXEHDAbase é responsável pelos serviços básicos do ISAMpe e, embora constitua uma única referência lógica, seus serviços podem estar distribuídos entre vários equipamentos. Por fim, os EXEHDAnodos são os equipamentos de processamento disponíveis no ISAMpe, sendo responsáveis pela execução das aplicações pervasivas. A Figura 1 representa o ambiente pervasivo disponibilizado pelo EXEHDA.

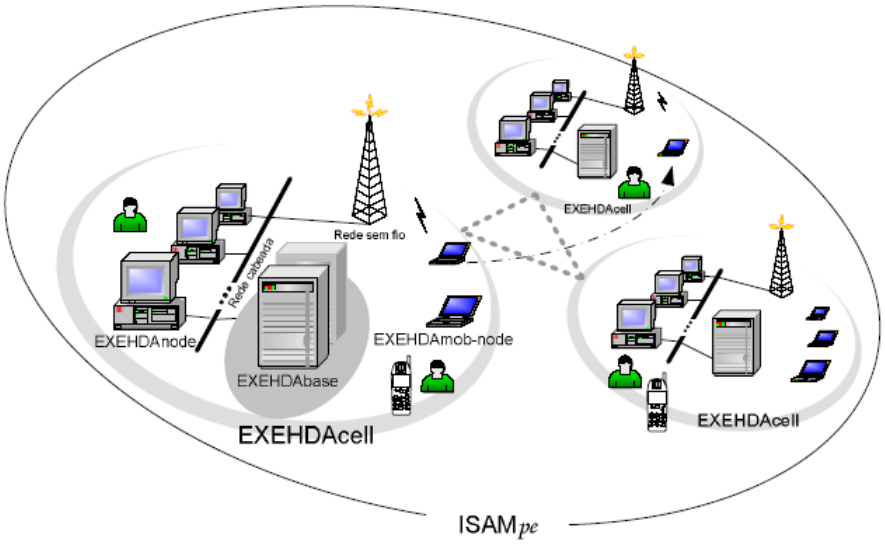

Figura 1 - ISAM Pervasive Environment [Fonte: (AUGUSTIN, 2004, p. 41)]

2.3.5 Núcleo Mínimo do EXEHDA

O núcleo mínimo do EXEHDA, o qual deve ser instalado em todo EXEHDAnodo que for integrado ao ISAMpe, é formado por dois componentes:

Profile Manager – interpreta as informações disponíveis no perfil de execução, que pode ser individualizado para cada EXEHDAnodo, e disponibiliza-as aos outros serviços do *middleware*.

Service Manager – realiza a ativação dos serviços no EXEHDAnodo a partir da informações disponibilizadas pelo *Profile Manager*, carregando sob demanda o código dos serviços a partir de um repositório de serviços, que pode ser local ou remoto.

Assim, as funcionalidades providas pelo EXEHDA são personalizáveis em nível de nodo, sendo determinada pelo perfil de execução[3]. O perfil de execução de cada EXEHDAnodo define o conjunto de serviços a ser ativado e os parâmetros para sua execução, associando a cada serviço uma implementação específica. Adicionalmente, o perfil de execução controla a política de carga de cada serviço, determinando se ele será ativado juntamente com a ativação do EXEHDAnodo (inicialização do *middleware*) ou sob demanda.

2.3.6 Subsistemas do Middleware EXEHDA

O Subsistema de Execução Distribuída é responsável pelo suporte ao processamento distribuído no EXEHDA. Esse subsistema interage com outros subsistemas, como o de reconhecimento do contexto e adaptação, para promover uma execução efetivamente pervasiva. Além disso, ele gerencia a execução e a migração das aplicações pervasivas.

O Subsistema de Reconhecimento de Contexto e Adaptação inclui serviços de aquisição de informações sobre o ISAMpe, de identificação em alto nível dos elementos de contexto e de disparo das ações de adaptação.

O Subsistema de Comunicação disponibiliza mecanismos para atender, principalmente, aspectos relacionados às desconexões, muito comuns em ambiente pervasivos devido tanto à existência de enlaces sem fio como às estratégias de economia de energia dos dispositivos móveis.

[3] Mais detalhes sobre o perfil de execução encontram-se no Apêndice B.

O Subsistema de Acesso Pervasivo tem por finalidade dar suporte à premissa da Computação Pervasiva de acesso em qualquer lugar e todo o tempo a dados e código. Dessa forma, esse subsistema gerencia o ambiente virtual do usuário, disponibilizando-o para acesso a partir de qualquer dispositivo integrante do ISAMpe.

No Apêndice C são descritos mais detalhes sobre os subsistemas do EXEHDA, bem como, sobre os serviços que compõem cada subsistema.

2.4 Projeto ClinicSpaces

Em termos científicos, o objetivo do projeto ClinicSpaces é estudar e pesquisar o potencial de uso de algumas técnicas da programação orientada a usuários-finais para projetar uma ferramenta que permita aos médicos a personalização de suas tarefas clínicas, as quais são gerenciadas e executadas em um ambiente pervasivo/ubíquo. Permitir que um médico intuitivamente "programe" (combine, configure, controle, defina, contextualize) as tarefas que serão executadas com auxílio do sistema ubíquo traz novos desafios para a pesquisa, grande parte desses relativos à definição de novos conceitos de aplicação para o hospital do futuro.

Como resultado tecnológico e inovação têm-se uma ferramenta-piloto de programação e gerenciamento de atividades clínicas, realizada por médicos de algumas especialidades. Visto que as conclusões do levantamento realizado sobre informatização de hospitais indicam que esta se refere, no país, principalmente, a atividades administrativas e registro eletrônico de saúde (prontuário eletrônico), esse projeto inova por propor um avanço nessa informatização, pois coloca sua atenção na atividade médica em si, e não somente no registro de procedimentos realizados, como atualmente.

Como esse projeto introduz um problema ainda não abordado, partiu-se do estudo e análise dos resultados de pesquisas relativas à programação de usuários-finais (*End-User Programming*). Esse estudo indicou o melhor caminho para se construir a ferramenta a ser usada pelo usuário (médico). Assim, optou-se pelo uso da Programação Visual devido a essa ser mais intuitiva para usuários leigos em programação, e por criar-se *widgets* (semelhantes aos ícones integrantes das interfaces gráficas) de tarefas–base que poderão ser manipulados pelos usuários.

O sistema proposto é visto como um assistente, o qual auxilia o médico a gerenciar, programar e executar suas tarefas diárias de forma intuitiva e pouco intrusiva. Tarefas podem

ser definidas pelo clínico (médico) com base em outras já programadas e regras previstas na ferramenta. O gerenciamento das atividades clínicas dá-se de forma pró-ativa, a partir da programação (personalizada pelo médico) e do fluxo de informação e execução disparado por eventos detectados pelo Sistema de Reconhecimento do Contexto e Adaptação do *middleware* EXEHDA.

3 ORIENTAÇÃO A TAREFAS CONTEXTUALIZADAS NO PROJETO CLINICSPACES

3.1 Visão geral

O Projeto ClinicSpaces visa prospectar soluções para a 2ª. Geração dos sistemas pervasivos – ambientes programáveis pelo usuário-final, através da utilização de tecnologias para o auxílio aos profissionais clínicos na execução de suas atividades nos ambientes hospitalares. Para diminuir o impacto de interferência do sistema automatizado no ambiente, e a rejeição que isso 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). Logo, o projeto engloba soluções de diversas áreas, tais como: Teoria da Atividade, Computação Orientada a Contexto, *End-User Programming*, Sistemas de Informação de Saúde.

Mesmo o sistema 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. Assim, objetiva-se fornecer mecanismos que permitam aos usuários interagir e personalizar o sistema, adequando-o melhor a sua forma de executar as atividades. O projeto propõe a definição de uma ferramenta-piloto que permita a personalização de tarefas, realizada pelo próprio usuário (médicos, por exemplo) visando ao gerenciamento de suas atividades clínicas, relativas a diagnóstico, tratamento, laboratorial e administrativa.

Devido à diversidade de interpretações semânticas dos termos atividade e tarefa, julgou-se necessário explicitar a definição em uso no projeto ClinicSpaces e nesta dissertação.

Atividades clínicas, como atendimento a pacientes, são processos realizados por humanos de forma colaborativa, coordenada e distribuída em um espaço determinado (RANGANATHAN; CAMPBELL, 2005b).

Segundo a Teoria da Atividade Humana (KAENAMPORNPAN; O'NEILL, 2005), atividades humanas podem ser modeladas com sujeito, objeto, ações (processos direcionados

ao objetivo) e operações (como a ação é executada). A Teoria da Atividade foi usada para a modelagem das Tarefas e definição de uma Ontologia[4] (LIBRELOTTO et al, 2008).

Dessa forma, adotou-se a seguinte definição: as atividades humanas podem ser decompostas em tarefas (ações), as quais são auxiliadas por aplicações computacionais, e seguem a forma particular de cada indivíduo de realizá-la (personalização). Tarefas simples são compostas por subtarefas (operações) e, quando agrupadas, formam uma tarefa composta que segue um fluxo de execução (workflow).

3.2 Modelagem das tarefas[5]

Para definir as tarefas–base e os mecanismos de composição foi necessário um estudo das atividades e procedimentos clínicos que podem ser auxiliados por sistemas computacionais. Foram usados, como subsídio, guias formais de procedimentos recomendados, que estão sendo desenvolvidos pela comunidade da Saúde, e o trabalho de Hallvard Laerum e Arild Faxvaag (LAERUM; FAXVAAG, 2004).

Segundo Augustin (2006), em termos de modelagem pervasiva, as atividades são executadas sob determinado contexto. Por isso, a modelagem de tarefas responde às questões que definem seu contexto: quem (paciente, acompanhante, clínico), onde (localização), quando (data e horário), com o quê (recursos lógicos e físicos), o que (composição de atividades).

> Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade (pessoa, serviço, aplicação, dispositivo, objeto, lugar, etc) considerada relevante para o comportamento da aplicação (AUGUSTIN; LIMA; YAMIN, 2006, p. 7).

Em uma análise inicial, pôde-se observar que a noção de contexto[6] para tarefas é formada por usuários, localização, tempo e recursos. E está relacionada com definições da Teoria da Atividade, tais como: ferramentas, regras, comunidade, divisão de trabalho.

Além disso, as tarefas possuem as seguintes características:

i. Decomposição, Recombinação e Reuso – são compostas por subtarefas reusáveis. Essas subtarefas podem ser recombinadas de diferentes maneiras

[4] A ontologia para atividades clínicas está em fase de desenvolvimento.
[5] A modelagem de tarefas é aprofundada no trabalho de dissertação (em andamento) do mestrando Fábio Lorenzi da Silva (PPGI - UFSM), também ligado ao projeto ClinicSpaces.
[6] A modelagem do contexto para atividades clínicas está sendo estudada no trabalho de dissertação (em andamento) do mestrando Tiago Antônio Rizzetti (PPGI - UFSM), também ligado ao projeto ClinicSpaces.

para formar diferentes tarefas; como diferentes tarefas possuem subtarefas em comum, estas podem ser reutilizadas para desenvolver novas tarefas e assim reaproveitar a programação das subtarefas já implementadas;

ii. Interrupção (preemptáveis) – as tarefas podem ser interrompidas e retomadas mais tarde (semelhante ao modelo de co-rotina);

iii. Mobilidade e adaptabilidade – podem migrar e adaptar-se para acompanhar o usuário (semântica siga-me);

iv. Contextualização – as tarefas são associadas a um contexto, o que pode ocorrer dinamicamente.

Tarefas são 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 atividades já programadas é essencial. As tarefas possuem um conjunto de informações que são de fundamental importância 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;
- 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.

Como exemplo de tarefa pode-se citar o "Atendimento a Pacientes". Essa tarefa representa, computacionalmente, as ações realizadas pelo médico durante uma consulta. A ela podem estar associados artefatos ou recursos que formam parte do seu contexto de execução.

As tarefas são gerenciadas considerando os possíveis estados das tarefas, os quais são: (i) inicializada, (ii) executando, (iii) pausada, (iv) finalizada, (v) cancelada, (vi) erro, (vii) encerrada. A implementação do gerenciamento dos estados será detalhada no capítulo 4.

Para simplificar a definição de tarefas por um usuário-final, uma tarefa possui uma descrição ontológica (LIBRELOTTO et al, 2008) com informações sobre seu criador, a descrição (funcionalidade e demais informações pertinentes), recursos utilizados, pré-condições para execução, especialidade médica, contexto utilizado e implementação. A representação das tarefas através de ontologias é objeto de estudo em outra frente de atuação do projeto ClinicSpaces, e até o momento não está totalmente terminada. Por isso, ela é descrita superficialmente neste trabalho.

As subtarefas, além de terem a descrição ontológica (LIBRELOTTO et al, 2008), são implementadas diretamente como objetos do EXEHDA (OX - Objeto eXehda), ou seja, aplicações pervasivas, que dizem respeito às funcionalidades do sistema EHS utilizado ou de funcionalidades do EXEHDA. Na descrição ontológica de cada subtarefa devem ser especificados, além de outras informações, os contextos necessários à sua execução e os contextos opcionais, ou seja, os que influenciam a execução da subtarefa, mas não são essenciais para ela. Os contextos opcionais de cada subtarefa são especificados e disponibilizados na Ferramenta de Construção de Tarefas[7], no nível do usuário, à medida que elas são adicionadas à tarefa que está sendo construída. Assim, o usuário pode optar pela adição do contexto opcional se julgar necessário. No Apêndice E são apresentados alguns exemplos de subtarefas.

Retomando o exemplo de tarefa citado anteriormente, a tarefa de "Atendimento a Pacientes" poderia ser formada pelas seguintes subtarefas, as quais representam algumas das ações realizadas pelo médico ao atender o paciente: "buscar informações específicas no prontuário do paciente", "inserir informações sobre o atendimento no prontuário", e "prescrever um tratamento".

Além disso, julgou-se necessário criar níveis de classificação[8] para as subtarefas, de acordo com as informações que elas precisam para realizar seu processamento e com os resultados produzidos por elas. Dessa forma, pode-se aplicar regras de composição de tarefa baseadas nesses níveis, evitando que se construam tarefas inconsistentes, cujas subtarefas não tenham seus pré-requisitos satisfeitos, o que impossibilitaria sua execução.

[7] A Ferramenta de Construção de Tarefa é abordada e prototipada no trabalho de dissertação (em andamento) do mestrando Fábio Lorenzi da Silva (PPGI - UFSM), também ligado ao projeto ClinicSpace.
[8] A classificação em níveis também é abordada na dissertação do Fábio Lorenzi da Silva (PPGI-UFSM).

A Figura 2 exemplifica a relação existente entre as tarefas, as subtarefas, a Ferramenta de Construção de Tarefas (citada anteriormente), o novo Subsistema de Gerenciamento Distribuído de Tarefas – SGDT (tema central deste trabalho), e o sistema pEHS (que disponibiliza as aplicações clínicas).

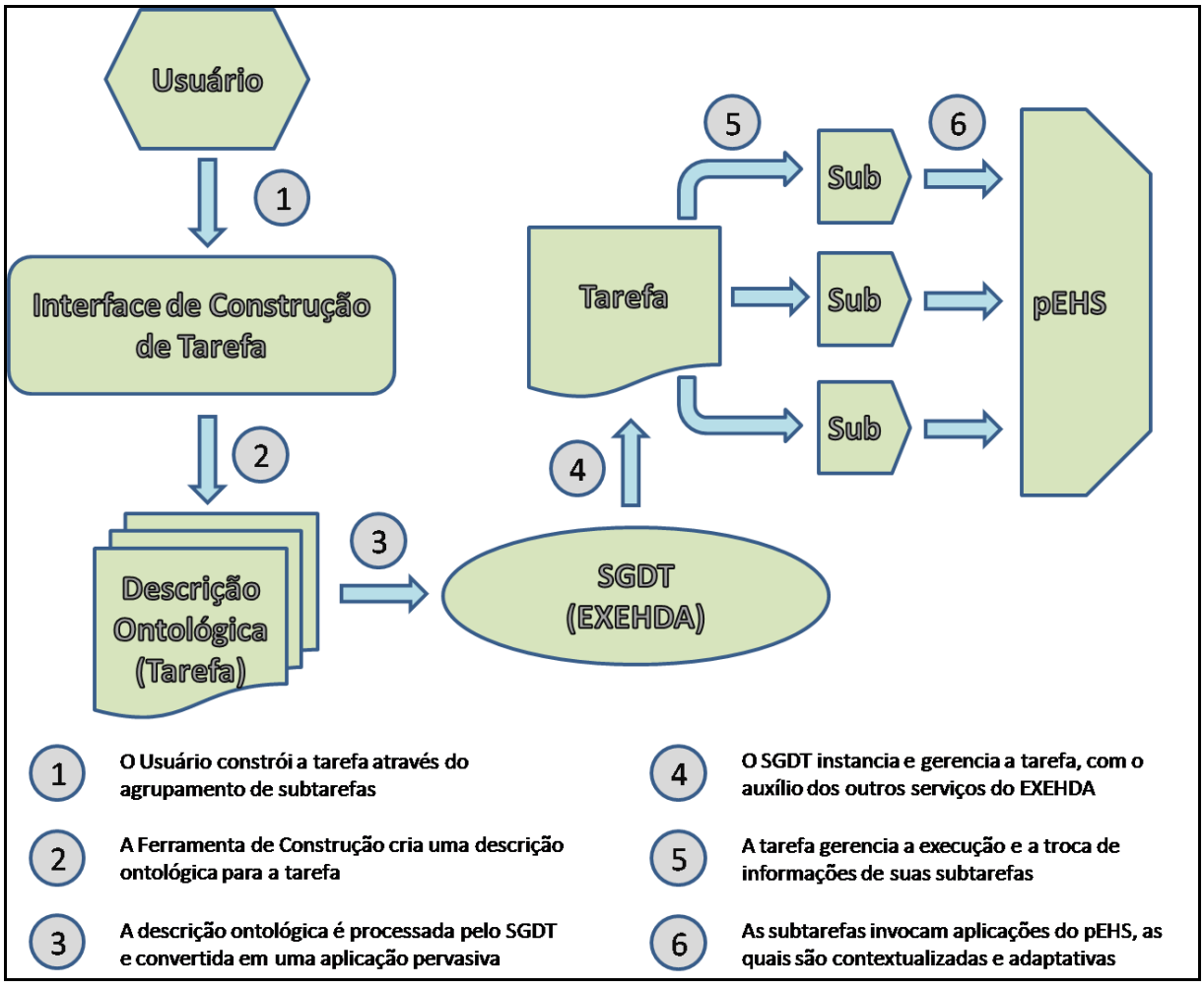

Figura 2 - Relacionamento entre Tarefas, Subtarefas, Ferramenta de Construção de Tarefas, SGDT, e pEHS

3.3 Políticas de criação e gerenciamento de tarefas

As tarefas são criadas pelos usuários-finais, a partir da interface de programação de tarefas, a qual disponibiliza as subtarefas e/ou tarefas já criadas pelos usuários. Assim, uma tarefa tem somente um proprietário. No momento da criação, o proprietário pode definir o modo de compartilhamento da tarefa, que pode ser: (i) não compartilhada - esse é o modo

padrão; (ii) compartilhada com profissionais da mesma especialidade; ou (iii) compartilhada com todos os profissionais.

Quando um usuário se interessa por uma tarefa compartilhada, o sistema faz uma cópia da tarefa, a qual não será influenciada por alterações na original, a menos que o usuário deseje atualizá-la. Assim, tanto o criador como o usuário que está usando a tarefa podem fazer alterações (personalizações) sem que essas sejam refletidas nas outras cópias.

No sistema, ainda foi previsto que quando o criador da tarefa fizer modificações nela e optar por compartilhar a nova tarefa, os usuários que já haviam adicionado a antiga tarefa receberão um aviso de atualização, podendo atualizar a tarefa ou manter a antiga.

Quanto ao disparo das tarefas, como visto anteriormente, os sistemas pervasivos devem ser pró-ativos (agir em nome do usuário). Em contra-partida, um usuário, ao executar seu trabalho, deseja manter sua forma de executá-lo, sendo necessário que o sistema forneça mecanismos que lhes permitam interagir e personalizá-lo, adequando-o melhor a sua forma de executar as tarefas.

Por isso, as tarefas são sugeridas para o usuário, de acordo com o contexto no qual ele está inserido. O usuário pode, então, disparar a execução da tarefa, descartar a sugestão, ou ainda, simplesmente ignorá-la.

A reação do usuário, em relação à tarefa sugerida, é armazenada para ser posteriormente processada, influenciando futuras inferências do sistema, quando houver uma situação semelhante.

3.4 Conjunto mínimo de tarefas

O conjunto mínimo de tarefas, definido para este trabalho, é baseado no estudo de Hallvard Laerum e Arild Faxvaag (LAERUM; FAXVAAG, 2004). No Apêndice D é apresentado o conjunto mínimo de tarefas, bem como, suas respectivas definições e classificações. A classificação das tarefas está associada às atividades clínicas realizadas, enquadrando-se nas seguintes categorias:

Diagnóstico: Essa categoria é composta por atividades relacionadas a métodos, procedimentos ou técnicas para determinar a natureza, ou identidade, da doença ou enfermidade, inclusive requisição de exames. Não se inclui nessa categoria os procedimentos

realizados para obtenção dos resultados de exames. Ex: procurar informações específicas em registros do paciente (em um sistema de informação de saúde);

Tratamento: Essa categoria é composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para melhorar o estado de saúde dos pacientes e, ainda, outras atividades para tratar doenças, enfermidades e ferimentos. Ex: prescrever receitas (receitar medicamentos ou outros tipos de tratamentos auto-administráveis);

Laboratorial: Essa categoria é composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para realizar exames laboratoriais e exames de diagnóstico por imagem (como raioX e ultrasom). A separação em dois tipos de exames é interessante do ponto de vista da adaptação sensível ao contexto. Exames laboratoriais têm como resultado informações textuais (laudos, tabelas, valores numéricos). Já exames de outro tipo resultam em imagens que necessitam de maior capacidade de armazenamento, de processamento e de exibição. Essa é uma característica importante para a adaptação da aplicação ao contexto, que, nesse caso, é o dispositivo. Essa categoria inclui ainda atividades que se relacionam com o sistema laboratorial para a obtenção de informações de exames. Ex: obter resultados de exames clínicos (análises laboratoriais);

Administrativa: Essa categoria é 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, não compreendidas nas categorias anteriores. Ex: registrar códigos para diagnósticos ou procedimentos executados.

3.5 Arquitetura para a programação e gerenciamento das tarefas

A arquitetura para a programação e gerenciamento personalizado das tarefas, mostrada na Figura 3, 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)definir 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.

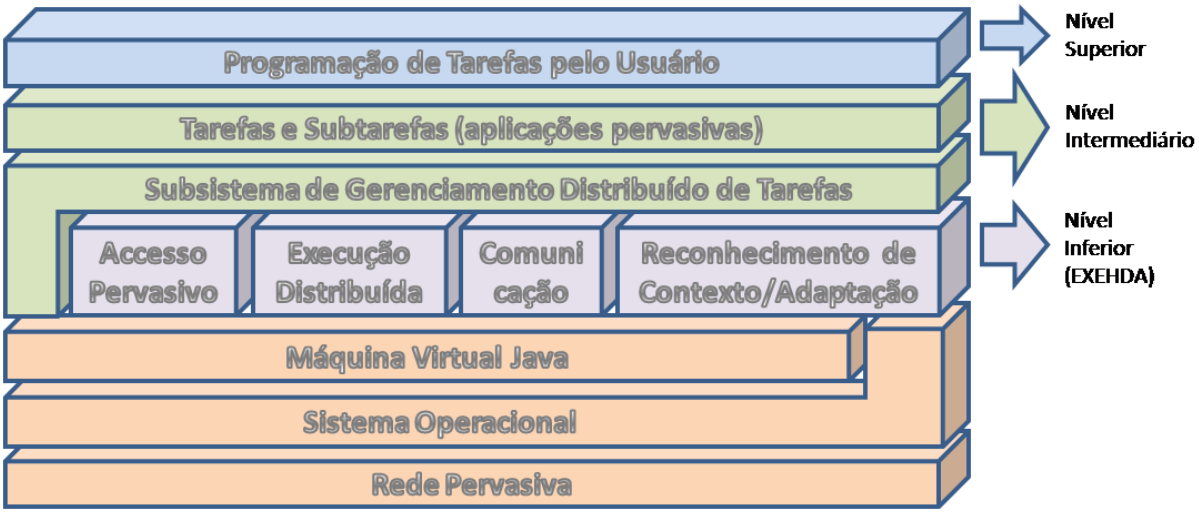

Figura 3 - Arquitetura para programação e gerenciamento de tarefas

A ferramenta-piloto consta de: (i) ferramenta gráfica de criação e personalização de tarefas, (ii) *middleware* para dar suporte à execução das tarefas, e (iii) Sistema pEHS (pervasive Electronic Health-care System).

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, supõe-se a existência de um sistema pervasivo de informação de Saúde, denominado pEHS[9], 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.

[9] 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 or task-oriented computing*) exigiu a inserção de um novo subsistema no *middleware*, estendendo suas funcionalidades. O novo Subsistema de Gerenciamento Distribuído de Tarefas (SGDT) 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 Interação do usuário com o ambiente pervasivo

Na interação do usuário com o ambiente pervasivo (hospital), mais especificamente na sua movimentação pelo ambiente, existem três casos a serem considerados.

- O usuário pode mover-se dentro do ambiente pervasivo portando um dispositivo móvel. Nesse caso, não há desconexão, e a sessão do usuário permanece ativa durante o deslocamento.
- O usuário pode mover-se pelo ambiente sem carregar um dispositivo. Nesse caso, ao sair do dispositivo onde estava trabalhando, o sistema encerra a sessão aberta, interrompendo e armazenando as tarefas que estavam em execução. Ao aproximar-se de outro dispositivo, seu *login* é habilitado nesse dispositivo, bastando o usuário *logar-se* para que sua sessão seja restaurada, e suas tarefas sejam mostradas na tela do dispositivo.
- O usuário pode trocar de dispositivo sem mover-se. Nesse caso, ele pode *deslogar-se* do dispositivo que está usando e *logar-se* no que irá usar. Ou, simplesmente, *logar-se* no dispositivo que irá utilizar, fazendo com que o sistema encerre sua sessão no dispositivo que estava sendo usado. Em ambos os casos, as tarefas em execução são interrompidas, armazenadas e disponibilizadas para continuação na tela do dispositivo no qual o usuário se *logou*.

Toda vez que a sessão do usuário é encerrada, o novo subsistema do EXEHDA faz a migração da sessão e das tarefas para um servidor, com o auxílio dos outros serviços do *middleware*. Da mesma forma, o SGDT usa esses serviços para restaurar a sessão e as tarefas

do usuário, quando esse *logar* em um dispositivo. Além disso, como será visto na sessão 4.9, o sistema pEHS utilizado como suporte às atividades clínicas, permite que as informações em uso sejam armazenadas e restauradas posteriormente.

3.7 Cenários de Utilização do Sistema ClinicSpaces

Para exemplificar a utilização da ferramenta-piloto, a seguir descrevem-se três cenários que usam diferentes momentos do gerenciamento e definição das tarefas.

3.7.1 Cenário 1: Criação de uma Tarefa[10]

Um médico tem como uma de suas atividades o atendimento a pacientes. Para tal, ele deseja construir tarefas que o auxiliarão nessa atividade, e serão executadas conforme a sua forma individual de trabalhar. Por exemplo, se o médico costuma primeiramente verificar as últimas informações do prontuário do paciente, ele inicia a construção da tarefa com uma subtarefa de "busca de informações do paciente". Nessa subtarefa, o usuário pode configurar filtros como especialidade médica, período das informações e número de registros a serem buscados.

A ferramenta de construção adiciona, automaticamente, uma subtarefa de "identificação do paciente" (contexto), visto que é necessário identificar qual prontuário deve ser acessado. Em seguida, o médico adiciona uma subtarefa de "visualização da informação", que pode ser gráfica, texto, ou impressa dependendo do contexto onde ela executar.

Com isso, em nível conceitual, uma tarefa já está formada e pronta para ser utilizada. Mas, a atividade que o usuário está personalizando vai além disso. Contudo, é importante salientar que, se o médico já tivesse uma tarefa específica para busca de informações de pacientes, ela poderia ser adicionada no lugar dessas três subtarefas, otimizando o processo de criação. Nesse caso, a tarefa criada seria tratada, em nível de gerenciamento, como uma tarefa composta.

[10] As questões relativas a este cenário são tratadas no trabalho de dissertação do mestrando Fábio Lorenzi da Silva (PPGI-UFSM).

Continuando a composição da tarefa, o médico poderia adicionar uma subtarefa de anotação para registrar no prontuário do paciente as informações de atendimento. Essa subtarefa faria uso da subtarefa de identificação do paciente, adicionada anteriormente, para identificar o prontuário a ser usado. Em seguida, poderia ser adicionada uma tarefa para solicitação de exames, que anotaria a solicitação no prontuário e encaminharia a solicitação para o laboratório (se fosse o caso).

Finalmente, o médico poderia adicionar uma tarefa de "prescrição de medicamento" ou "prescrição de tratamento", que anotaria a prescrição no prontuário e a imprimiria. Encerrada a construção/personalização da tarefa, essa é armazenada no Ambiente Virtual do Usuário gerenciado pelo EXEHDA. Nesse momento, o médico pode optar por compartilhar a tarefa com outros profissionais.

3.7.2 Cenário 2: Execução de uma Tarefa

Vamos supor que o médico irá atender a um paciente (realizar a atividade programada no cenário 1). A interface do sistema está acessível a ele e estão disponibilizadas suas tarefas. O sistema detecta a entrada do paciente (contexto) e dispara a execução da tarefa criada no cenário 1.

O Subsistema de Gerenciamento Distribuído de Tarefas (SGDT) obtém do Sistema de Reconhecimento do Contexto a identificação do profissional, a configuração do dispositivo e outras informações necessárias à instanciação da tarefa. Processando a descrição ontológica da tarefa, o SGDT encontra as subtarefas que a compõem e busca o código delas.

A primeira subtarefa a ser executada é a de identificação do paciente. Essa utiliza os serviços do SGDT, o qual usa sensores gerenciados pelo EXEHDA, para identificar o paciente que está sendo atendido. A informação retornada por essa subtarefa é repassada para as outras subtarefas automaticamente.

Em seguida, é disparada a subtarefa de busca de informações do paciente, a qual invoca a aplicação responsável por isso no pEHS. A informação retornada é usada como parâmetro para a próxima subtarefa, que exibe os registros do prontuário no formato escolhido pelo usuário ou adequado ao dispositivo em uso. É a partir desse momento que começa a interação do usuário com a tarefa, pois a execução das subtarefas anteriores é transparente.

Após o médico revisar as informações do paciente, ele encerra a aplicação (subtarefa), e o SGDT dispara a próxima subtarefa, que irá armazenar as anotações do médico no prontuário. Essa subtarefa invoca uma aplicação específica do pEHS, parametrizando-a com a identificação do profissional e do paciente. Ao término dessa subtarefa, a tarefa de solicitação de exames é disparada. Aqui vale salientar que se trata de uma tarefa simples, criada anteriormente pelo usuário. Essa tarefa é independente, podendo ser executada isoladamente.

A tarefa de solicitação de exames armazena a solicitação do médico no prontuário do paciente e encaminha-a para o laboratório, desde que o sistema do laboratório esteja integrado ao pEHS. Ao término dessa tarefa, o SGDT dispara a tarefa de prescrição de medicamento ou tratamento, que registra a prescrição do médico no prontuário e envia-a para a impressora mais próxima, obtida através do EXEHDA.

Este cenário, conversão de tarefas em aplicações pervasivas e o gerenciamento de tarefas, é o foco dessa dissertação. Portanto, será aprofundado nos próximos capítulos

3.7.3 Cenário 3: Colaboração

Suponhamos que, durante o atendimento a um paciente (cenário 2), o médico precise perguntar algo a outro profissional. Para isso, o médico acessa o módulo de colaboração do SGDT, o qual verifica que o outro profissional da especialidade requerida está on-line e envia uma mensagem para ele solicitando a informação desejada. Enquanto isso, a tarefa de atendimento fica pausada, aguardando o término da comunicação.

Quando a mensagem chega ao destino, o SGDT apenas gera uma notificação na interface do usuário. Assim, o profissional pode abrir a notificação, caso esteja livre ou realizando atividades que podem ser interrompidas. No caso de tarefas que não podem ser interrompidas, a notificação fica aguardando o término das tarefas. E o SGDT responde ao solicitante indicando que o profissional requisitado está ocupado. Ao abrir a notificação, o SGDT interrompe, temporariamente, as tarefas que estavam em execução. Assim, o usuário pode responder à solicitação e voltar às suas tarefas.

O médico solicitante recebe, então, a resposta e tem a opção de armazenar a conversa no prontuário, se julgar relevante ou se inseriu essa subtarefa no momento da programação de tarefas. Após o encerramento da comunicação, a tarefa que havia sido pausada é retomada, e o médico segue com o atendimento.

4 PROPOSTA DE NOVO SUBSISTEMA PARA O EXEHDA: GERENCIAMENTO DISTRIBUÍDO DE TAREFAS

Como visto no capítulo 3, a execução das tarefas é gerenciada pelo *middleware* EXEHDA (EXEHDA, 2001). Porém, como o EXEHDA não foi desenvolvido para ser orientado a tarefas, tornou-se necessária a inserção de um novo subsistema no *middleware*, estendendo suas funcionalidades.

O novo Subsistema de Gerenciamento Distribuído de Tarefas (SGDT), mostrado na Figura 4, é responsável por gerenciar, em alto nível, as tarefas, delegando o gerenciamento das subtarefas (aplicações pervasivas) para os serviços atuais do EXEHDA.

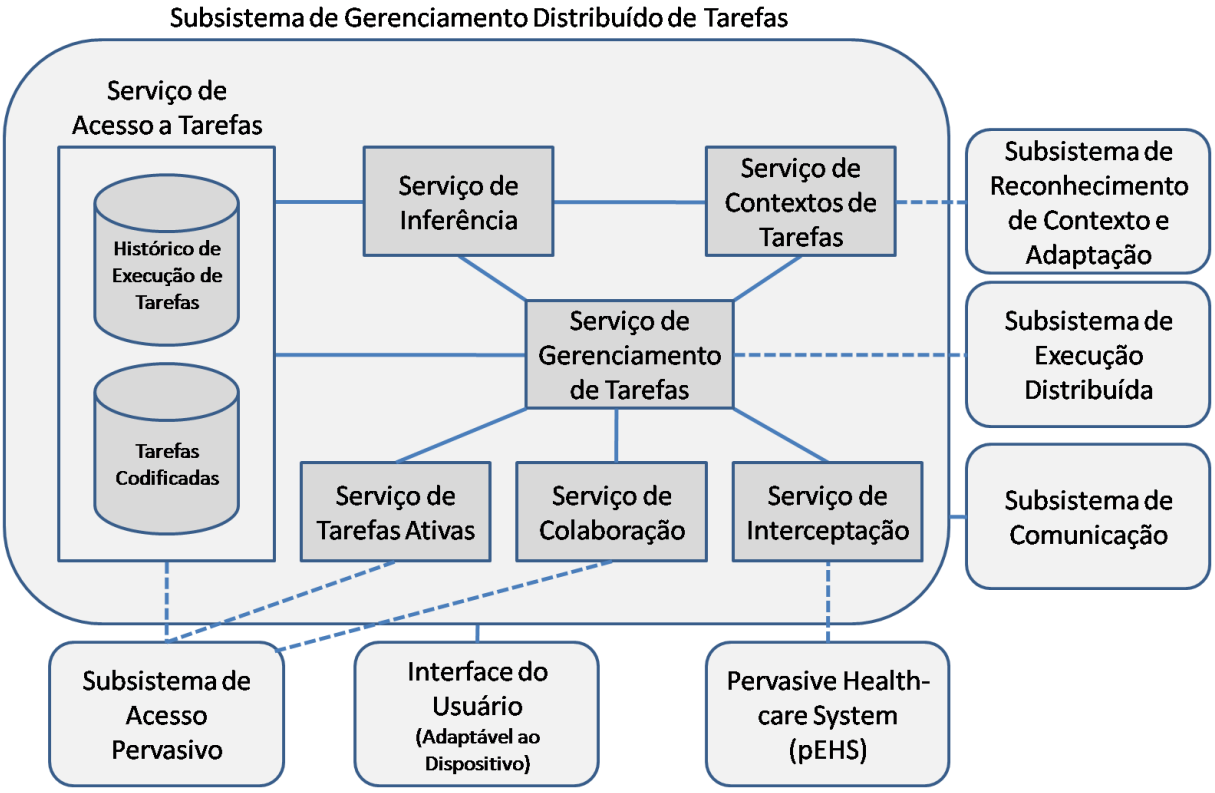

Figura 4 - Subsistema de Gerenciamento Distribuído de Tarefas

O novo subsistema foi modelado de acordo com a especificação do *middleware*. E por isso, assim como os outros subsistemas do EXEHDA, 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.

Nas sessões seguintes são apresentados os serviços modelados para o novo Subsistema de Gerenciamento Distribuído de Tarefas do EXEHDA. Por último, na sessão 4.8, será discutido como esses novos serviços interagem com os serviços atuais do *middleware,* e na sessão 4.10, será abordada a questão da interface gráfica, que permite a definição de tarefas pelo usuário-final, e sua interação com o SGDT e com o EXEHDA.

4.1 Serviço de Acesso a 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.

A Base de Dados de Tarefas Codificadas armazena as tarefas e subtarefas disponibilizadas aos usuários. Como visto no capítulo 3, 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 API Jena (Dickinson, 2008).

Porém, o banco de dados ontológico será abordado em outra frente de atuação do projeto ClinicSpaces. E por isso, a prototipação do Serviço de Acesso a Tarefas foi feita com um parser XML, que processa a descrição ontológica das tarefas. Fica como trabalho futuro atualizar esse serviço para fazer uso do banco de dados, através da API Jena.

A associação entre os usuários e as tarefas disponíveis a eles também é responsabilidade do banco de dados ontológico. 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.

A implementação dessa funcionalidade poderia ser feita através da utilização do serviço Logger do EXEHDA (ver Apêndice C), ou poderia ser modelada uma ontologia específica para esse fim. Porém, como essa é uma funcionalidade secundária (não essencial para a prototipação do SGDT) e dependente do Serviço de Inferência (o qual será modelado em outro trabalho[11]), ela não foi implementada no protótipo inicial do SGDT, ficando como um trabalho a ser realizado numa segunda etapa do desenvolvimento do projeto ClinicSpaces, após o conceito de contexto de tarefas clínicas estar mais amadurecido, o que permitirá avaliar quais as informações de execução são realmente importantes para a inferência das atividades dos usuários.

4.1.1 Funcionamento do Serviço de Acesso a Tarefas

Essa subsessão discute questões de implementação do Serviço de Acesso a Tarefas (SAT), dando uma visão geral de como o acesso a Base de Dados de Tarefas Codificadas é feito no protótipo implementado como parte desta dissertação. Salienta-se que devido ao banco de dados ontológico estar sendo desenvolvido em outro trabalho paralelo, a prototipação do SAT foi feita com um parser XML, baseado numa descrição ontológica de tarefas, a qual também está sendo evoluída paralelamente a este trabalho.

[11] O Serviço de Inferência está sendo modelado no trabalho de dissertação do mestrando Marcos Vinícius B. de Souza (PPGI-UFSM).

O Serviço de Acesso a Tarefas, atualmente, disponibiliza dois métodos para que o Serviço de Gerenciamento de Tarefas tenha acesso às tarefas dos usuários. Um deles busca a lista de tarefas disponíveis a determinado usuário (ver Quadro 1) e retorna os identificadores delas. Essa busca é feita em um arquivo XML contido no Ambiente Virtual do Usuário (AVU) mantido pelo *middleware* EXEHDA. Esse arquivo segue o modelo de um arquivo que informa as aplicações pervasivas disponíveis a um usuário do EXEHDA. Porém, futuramente, esse método usará o banco de dados ontológico para buscar as tarefas disponíveis a um usuário.

```
public List<String> findAvailableTasks(String login)
        throws TaskLoadException {
    final List<String> availableTasks = new LinkedList<String>();
    //obtém o elemento raiz e a lista de nodos "Task"
    final Element root;
        root = parserXml.makeParser(login,
                "available_tasks.xml",
                ParserXml.AVAILABLE_TASKS);
    final NodeList taskNodes = root.getElementsByTagName("Task");

    for (int i = 0; i<taskNodes.getLength(); i++) {
        final Element taskNode = (Element) taskNodes.item(i);
        //obtém o identificador da tarefa e insere na lista
        String task = taskNode.getTextContent();
        availableTasks.add(task);
    }
    return availableTasks;
}
```

Quadro 1 - Método findAvailableTasks() do SAT. O tratamento de exceções foi removido do código, pois não é essencial para seu entendimento.

Outro método, *findTask*, é responsável por buscar uma tarefa que será executada pelo Serviço de Gerenciamento de Tarefas. Esse método busca o descritor XML da tarefa através de seu identificador. Futuramente, essa busca também será feita no banco de dados ontológico.

Após a busca (ver fragmento de código no Quadro 2), o descritor é processado por um *parser* criado para essa finalidade. O *parser* faz a validação do descritor através de uma estrutura padrão definida com XML Schema (W3C, 2000). Tanto tarefas como subtarefas possuem seu *schema*, mostrados, respectivamente, na Figura 5 e na Figura 6, as quais representam a ontologia definida para tarefas simples e subtarefas. Feita a validação e o

processamento do descritor, a tarefa é instanciada de acordo com seu tipo (simples ou composta).

```
public Task findTask(String login, String idTask)
        throws TaskLoadException {
    final String taskFileName = new String(idTask + ".xml");
    //obtém o elemento raiz do descritor
    final Element taskNode;
    taskNode = parserXml.makeParser(login,
                taskFileName, ParserXml.TASK);
    //instancia de acordo com o tipo (Simples ou Composta)
    final String taskType = parserXml.getTaskType(taskNode);
    final Task task;
    task = getTaskInstance(taskType);
    parserXml.parseTask(task, taskNode);
    loadSubTasks(task, taskNode);
    return task;
}
```

Quadro 2 - Método findTask() do SAT. O tratamento de exceções foi removido do código, pois não é essencial para seu entendimento.

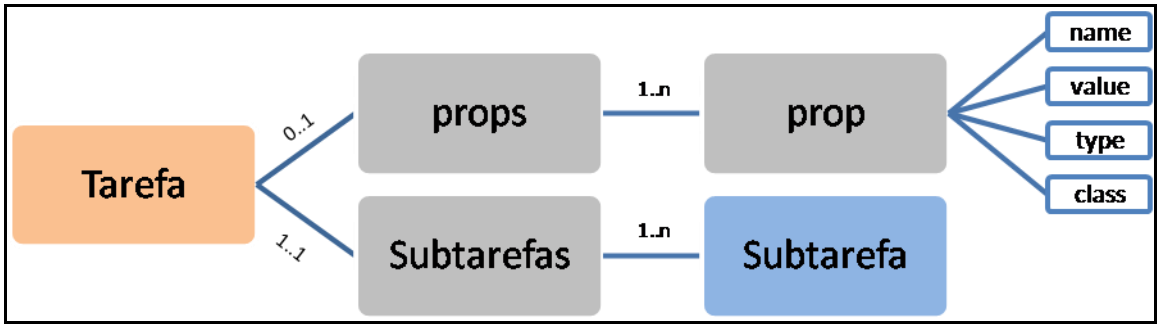

Figura 5 - XML Schema para tarefas

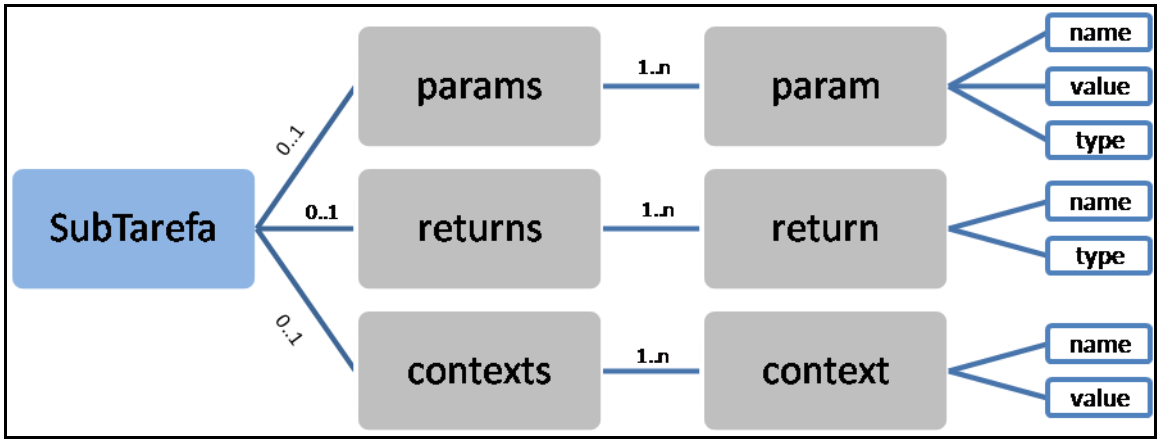

Figura 6 - XML Schema para subtarefas

Em seguida, todos os descritores (XML) das subtarefas que compõem a tarefa são buscados e cada subtarefa é instanciada e adicionada à tarefa (ver Quadro 3). Finalmente, a tarefa está carregada e é enviada ao Serviço de Gerenciamento de Tarefas para execução.

```
private void loadSubTasks(Task task, Element taskNode) {
    if(task.getType().equals("Simple")){
        final List<String> subTaskList = parserXml
                .getsubTaskList(taskNode);
        for (String idSubTask : subTaskList) {
                SubTask subTask = findSubTask(idSubTask);
                subTask.setOwner(task);
                task.addSubTask(subTask);
        }
    } else if(task.getType().equals("Composed")) {
        throw new UnsupportedOperationException(
                "Composed Task: Not yet implemented!");
    }
}
private SubTask findSubTask(String idSubTask) {
    String fileName = new String(idSubTask + ".xml");
    Element subTaskNode = parserXml.makeParser(null,
            fileName, ParserXml.SUB_TASK);
    String subTaskClass = parserXml
            .getSubTaskClass(subTaskNode);
    final SubTask subTask = getSubTaskInstance(subTaskClass);
    parserXml.parseSubTask(subTask, subTaskNode);
    return subTask;
}
private SubTask getSubTaskInstance(String className) {
    SubTask subTask = null;
    Class clazz = Class.forName(className);
    subTask = (SubTask) clazz.newInstance();
    return subTask;
}
```

Quadro 3 - Busca e instancia subtarefas. O tratamento de exceções foi removido do código, pois não é essencial para seu entendimento.

4.2 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.

Como visto no capítulo 3, 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[12], também ligado ao projeto ClinicSpaces.

4.3 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 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[13] 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.

[12] Trabalho de dissertação do mestrando Tiago Antônio Rizzetti (PPGI - UFSM).
[13] O Serviço de Inferência é abordado no trabalho de dissertação do mestrando Marcos Vinícius B. de Souza (PPGI-UFSM).

4.4 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 (BARDRAM; 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 (BARDRAM; CHRISTENSEN, 2007), a consulta à opinião de outro profissional. Por isso, foi previsto, para o Serviço de Colaboração[14], 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 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.

Como o EXEHDA não incorpora os conceitos de tarefas e colaboração entre os usuários, não foi previsto, no seu desenvolvimento, nenhuma forma de troca de objetos (OX) entre usuários. Isso impõe um problema para a implementação da transferência de tarefas em execução.

[14] O Serviço de Colaboração será modelado pelo mestrando Marcelo Lopes Kroth (PPGI-UFSM)

Para evitar os problemas causados por essa limitação, o controle de usuários e suas tarefas ativas foi implementado no Subsistema de Gerenciamento Distribuído de Tarefas (SGDT). Assim, o EXEHDA controla o Ambiente Virtual do Usuário, e o SGDT controla a lista de tarefas ativas dos usuários. Com isso, pode-se trocar tarefas entre os usuários, sem que haja necessidade de envolver diretamente os outros serviços do *middleware*. Essa implementação será abordada na sessão 4.7.

4.5 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) 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 deve 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 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. A classificação das tarefas na hierarquia de prioridades poderia ser personalizada pelo usuário. Porém, essa categorização depende de um maior aprofundamento no estudo sobre urgências e emergências clínicas e hospitalares.

4.6 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).

4.6.1 Funcionamento do Serviço de Gerenciamento de Tarefas

Essa subsessão discute questões de implementação do SGT, dando uma visão geral de como o gerenciamento das tarefas é feito no protótipo implementado como parte desta dissertação.

O Serviço de Gerenciamento de Tarefa utiliza o Serviço de Tarefas Ativas para manter, para cada usuário, uma lista de tarefas ativas, ou seja, tarefas instanciadas e que ainda não foram concluídas nem canceladas. Além disso, o SGT disponibiliza métodos, listados no Quadro 4, para que o usuário, através de uma interface gráfica, possa gerenciar suas tarefas (iniciar, executar, interromper, retomar, cancelar).

```
public Task initializeTask(String login, String idTask);
public void executeTask(Task task);
public void pauseTask(Task task);
public void cancelTask(Task task);
public void resumeTask(Task task);
public List<Task> getActiveTasks(String login);
public List<String> getAvailableTasks(String login);
```

Quadro 4 - Métodos para gerenciamento das tarefas

O gerenciamento das tarefas está baseado em transições de estado. Para diminuir a sobrecarga do SGT e para permitir a migração das tarefas sem a necessidade de todo o SGT executar remotamente, o estado das tarefas e o gerenciamento da execução de suas subtarefas são controlados pela própria tarefa. Para isso, e para atender outras necessidades de

implementação, o conceito (abstrato) de tarefa, que, basicamente, é um conjunto de subtarefas (associadas a aplicações concretas) e/ou outras tarefas organizadas de acordo com a preferência do usuário, foi materializado para um objeto (aplicação pervasiva), o qual, além de manter informações sobre a tarefa, implementa o gerenciamento de suas subtarefas e a troca de informações entre elas.

Como visto no capítulo 3, existem dois tipos de tarefas: tarefas simples (formadas por subtarefas) e tarefas compostas (formada por outras tarefas). Apesar de elas terem características comportamentais diferentes, suas características funcionais são idênticas. Em outras palavras, para o SGT, elas possuem a mesma função e são tratadas da mesma forma, apesar de executarem de maneira diferente. Por isso, foi criada uma Interface para tarefas (listada no Quadro 5), e o SGT gerencia os dois tipos através dessa interface. Assim, tanto a tarefa simples como a composta implementam seu comportamento sob essa interface padrão.

```
public interface Task extends Serializable {
    public TaskState getState();
    public void setState(TaskState state);
    public List<SubTask> getSubTasks();
    public void setSubTasks(List<SubTask> subTasks);
    public void addSubTask(SubTask subTask);
    public void removeSubTask(SubTask subTask);
    public void setParameter(String key, Object value);
    public Object getParameter(String key);
    public boolean containsParameter(String key);
    public void execute();
    public void pause();
    public void continue();
    public void cancel();
    public void finalize();
    public void activate();
    public void deactivate();
    public boolean isActive();
    public boolean isFinalized();
}
```

Quadro 5 - Interface de Tarefa

A mesma técnica foi utilizada no gerenciamento e implementação das subtarefas. Uma interface (listada no Quadro 6) define as funcionalidades necessárias a uma subtarefa, cada uma delas implementa seu comportamento sob essa interface, e a tarefa trata todas da mesma forma.

Além disso, para facilitar a programação de subtarefas, foi criada uma subtarefa abstrata (genérica), que implementa as funcionalidades comuns às subtarefas. Essas

funcionalidades, em sua maioria, são relativas ao gerenciamento. Dessa forma, uma subtarefa pode estender (herança de classes) essa subtarefa abstrata e implementar somente suas funcionalidades comportamentais específicas.

```
public interface SubTask extends Serializable {
    public Task getOwner();
    public Method getTaskCallBack();
    public void setOwner(Task owner);
    public void setTaskCallBack(String callBack, Class... paramsType);
    public Set<String> getParamKeys();
    public Set<String> getReturnKeys();
    public Set<String> getContextKeys();
    public Object getReturnValue(String key);
    public void setParamValue(String key, Object value);
    public boolean hasParameterSet(String key);
    public void setContextValue(String key, Object value);
    public boolean hasContextSet(String key);
    public void addParameter(String key, Object value);
    public void addReturn(String key, Object value);
    public void addContext(String key, Object value);
    public SubTaskState getState();
    public void setState(SubTaskState state);
    public void execute();
    public void terminate();
    public void pause();
    public void resume();
    public void cancel();
    public void destroy();
}
```

Quadro 6 - Interface de Subtarefa

Estado	Situação ocorrida
INICIALIZADA	Estado após a tarefa ser instanciada pelo Serviço de Acesso a Tarefas
EXECUTANDO	Estado setado quando o gerenciador inicia a execução da tarefa
CANCELADA	Estado setado pelo usuário ao cancelar a execução da tarefa
PAUSADA	Estado setado pelo usuário ao pausar a execução da tarefa
FINALIZADA	Estado setado pelo gerenciador após o término da execução de todas as subtarefas que compõem a tarefa
ENCERRADA	Estado setado pelo gerenciador após encerrar a tarefa
ERRO	Estado setado se ocorrer um erro que impeça a execução da tarefa

Quadro 7 - Estados das Tarefas

Como visto anteriormente, o gerenciamento da execução das tarefas é feito através de transições de estado na tarefa e nas subtarefas que a compõem. Os estados de uma tarefa (enumerados no Quadro 7) representam sua situação atual. Já os estados das subtarefas (enumerados no Quadro 8) indicam etapas de sua execução que foram concluídas e não precisam mais ser executadas, no caso de uma retomada da execução da tarefa.

Estado	Situação ocorrida
CARREGADA	Estado após ser instanciada pelo Serviço de Acesso a Tarefas
INICIALIZADA	Estado após o gerenciador inicializar parâmetros de controle
CONFIGURADA	Estado após os parâmetros de execução serem configurados pelo gerenciador
EXECUTADA	Estado após a subtarefa concluir sua execução (setado pela própria subtarefa)
FINALIZADA	Estado após os dados de retorno serem obtidos pelo gerenciador
ERRO	Estado setado pelo gerenciador em caso de exceção durante a configuração

Quadro 8 - Estados das SubTarefas

A Figura 7 representa a máquina de estados para tarefas. Uma tarefa instanciada pelo Serviço de Acesso a Tarefas e configurada pelo SGT está no estado "Inicializada". No momento em que o Serviço de Gerenciamento de Tarefas inicia a execução da tarefa, o estado dela passa a ser "Executando". Essa transição é automática e instantânea, mas futuramente pretende-se adicionar ao SGT a funcionalidade de agendamento de tarefas[15], o que tornará, possivelmente, essa transição mais útil. Prevendo isso, uma tarefa no estado "Inicializada" também pode ir para o estado "Cancelada", se o usuário fizer essa solicitação ao SGT.

Estando, a tarefa, no estado "Executando", ela pode assumir três outros estados: (i) "Cancelada", se o usuário cancelar a execução; (ii) "Finalizada", se todas as suas subtarefas forem concluídas; (iii) "Pausada", quando sua execução é interrompida. A interrupção pode ser solicitada tanto pelo usuário, quanto por outros serviços do SGDT, como Serviço de Colaboração (para iniciar uma comunicação), Serviço de Interceptação (para tratar um evento do pEHS), o Serviço de Gerenciamento de Tarefas (para migração da tarefa), etc.

[15] Este tema está sendo estudado pelo mestrando Tiago Antônio Rizzetti (PPGI-UFSM)

Do estado “Pausada”, a tarefa pode retornar ao estado “Executando”, caso o usuário solicite ao SGT sua retomada. 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 SGT libera os recursos alocados a ela, tornando-a indisponível ao SGDT.

Finalmente, o estado de “Erro” é usado quando ocorre alguma exceção que impossibilita a continuação da execução da tarefa. A exceção geralmente ocorre durante a instanciação, mas esse estado, conceitualmente, pode ser atingido a partir de qualquer um dos outros, bastando que a exceção seja prevista e tratada.

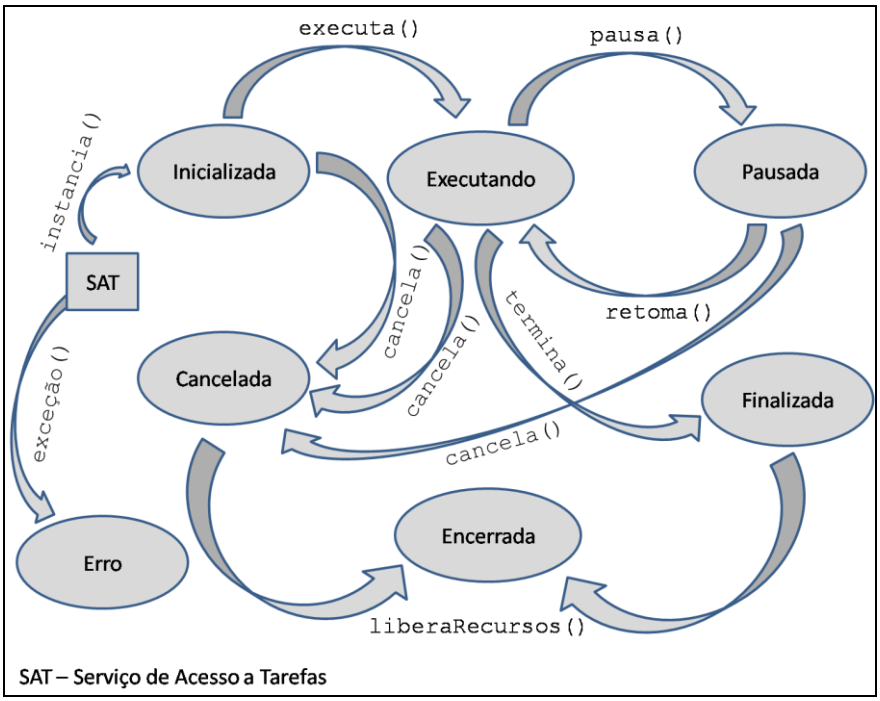

Figura 7 - Máquina de estados para tarefas

A Figura 8 representa a máquina de estados para subtarefas. O estado das subtarefas denota etapas de seu ciclo de vida que já foram executadas e não deverão ser processadas novamente durante uma retomada da execução da tarefa. Assim, o estado “Carregada” indica que a subtarefa foi instanciada pelo Serviço de Acesso a Tarefas. Já o estado “Inicializada” indica que seus parâmetros de controle foram inicializados pela tarefa. Esses parâmetros são utilizados para transferência do controle de execução entre a subtarefa e a tarefa.

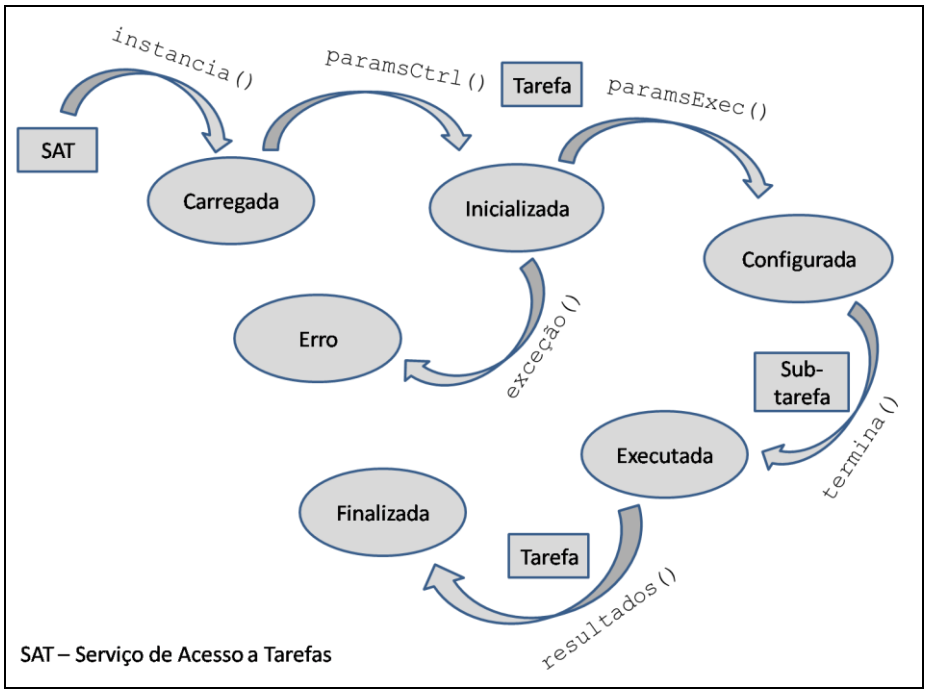

Figura 8 - Máquina de estados para subtarefas

Uma subtarefa no estado “Configurada” já teve seus parâmetros de execução configurados pela tarefa, e está pronta para ser executada. Esses parâmetros são dados e informações de contexto que a subtarefa necessita para processar sua função. Quando ela termina seu processamento, a própria subtarefa deve trocar seu estado para “Executada”, indicando que os dados resultantes estão prontos para serem copiados. A tarefa, então, transfere esses dados para seu mapa de parâmetros (disponibilizando-os para as próximas subtarefas) e muda o estado da subtarefa para “Finalizada”.

Por fim, da mesma forma que acontece com as tarefas, o estado “Erro” é usado quando ocorre alguma exceção que impede a execução da subtarefa. Essa exceção, geralmente, está associada à configuração dos parâmetros de execução. Mas, conceitualmente, esse estado é atingível a partir de qualquer outro.

Portanto, como se pode observar, o ciclo de vida de uma tarefa é gerenciado através de seus estados e dos estados de suas subtarefas. Assim, o ciclo de vida de uma tarefa inicia quando o usuário (e futuramente o Serviço de Inferência) solicita ao SGT a sua execução. O SGT utiliza o Serviço de Acesso a Tarefas (SAT) para solicitar a instanciação da tarefa. Como visto na sessão 4.1, o SAT processa a descrição ontológica dela, instancia uma nova tarefa de acordo com seu tipo (Simples ou Composta) e instancia as subtarefas que a compõem. O

Serviço de Gerenciamento de Tarefas, então, inicia a execução da tarefa carregada pelo SAT, esta executa suas subtarefas seqüencialmente (ver fragmento de código do Quadro 9).

```
private void executeSubTasks() {
    boolean breakFor = false;
    for(SubTask subTask : getSubTasks()) {
        switch(subTask.getState()) {
            case LOADED: //subtarefa instanciada
                subTask.setTaskCallBack("continue");
                subTask.setState(SubTaskState.INITIALIZED);
            case INITIALIZED: //configurada com parâmetros de controle
                    setupParameters(subTask);
                    setupContexts(subTask);
                    subTask.setState(SubTaskState.CONFIGURED);
            case CONFIGURED: //configurada com parâmetros de execução
                subTask.execute();
                //aqui não pode continuar; controle fica com a subtarefa
                breakFor = true;
                break;
            case PROCESSING: //estado interno
                subTask.resume();
                //aqui não pode continuar; controle fica com a subtarefa
                breakFor = true;
                break;
            case EXECUTED: //a subtarefas terminou sua execução
                getReturns(subTask);
                subTask.setState(SubTaskState.FINALIZED);
                break;
            case FINALIZED: //próxima subtarefa
                break;
            case ERROR: //repassa o erro pra tarefa
                setState(TaskState.ERROR);
                break;
        } //END-SWITCH
        sgt.updateTask(this); //atualiza a tarefa no servidor
        //interrompe o laço FOR se o controle ficar com a subtarefa
        if(breakFor){
            break;
        }
    } //END-FOR
    //se não foi interrompido, todas as tarefas finalizaram
    if(!breakFor){
        setState(TaskState.FINALIZED);
        sgt.updateTask(this); //atualiza a tarefa no servidor
    }
}
```

Quadro 9 - Executa subtarefas. O código foi resumido, substituindo-se partes com muitos detalhes por comentários que tornam o código mais claro. Após processar cada “case”, o estado da tarefa é testado e o processamento é interrompido se o estado não for *RUNNING*.

Cada subtarefa é configurada com parâmetros de controle e de execução, sendo os de controle necessários para que ela possa devolver o controle da execução para a tarefa, e os de execução para passar resultados das subtarefas anteriores para as próximas. A passagem de resultados de uma subtarefa para as próximas é feito através de um mapa de parâmetros, indexado pelo nome do parâmetro. O nome dos parâmetros e retornos de uma subtarefa são identificados na sua descrição ontológica.

A passagem de parâmetro funciona da seguinte maneira: quando uma subtarefa termina, a tarefa armazena em seu mapa de parâmetros os objetos produzidos pela subtarefa, indexando-os pelo nome contido na descrição ontológica dela; quando outra subtarefa vai ser executada, a tarefa verifica os parâmetros necessários (na descrição ontológica), busca-os no mapa e configura-os na subtarefa; se a tarefa não encontra o parâmetro no mapa, ela verifica se há um valor padrão para o parâmetro no descritor da subtarefa; caso não haja um valor padrão, a execução é interrompida, e a subtarefa vai para o estado "Erro", indicando que ela é dependente de outra subtarefa que deveria produzir esse parâmetro e por algum motivo não produziu ou não foi executada. Isso pode denotar um erro na construção da tarefa.

Após a configuração, a tarefa passa o controle da execução para a subtarefa. Quando a subtarefa termina sua execução, ela faz uma requisição à tarefa (usando os parâmetros de controle) indicando que esta pode continuar sua execução. A tarefa copia os resultados da subtarefa para seu mapa de parâmetros, para posteriormente configurar as próximas subtarefas. Esse ciclo é repetido até que todas as subtarefas tenham sido terminadas, o que conclui a tarefa.

Além disso, durante o ciclo de vida da tarefa, toda vez que há transição de estado em uma de suas subtarefas, a tarefa verifica seu próprio estado. Se o estado da tarefa foi modificado pelo SGDT, a execução da subtarefa é interrompida e o estado da tarefa é processado (ver fragmento de código no Quadro 10).

Portanto, os métodos de interação (para gerenciamento das tarefas) do SGT somente alteram o estado da tarefa, e esse é processado quando ela atinge um ponto estável (troca de estado da subtarefa). Porém, como o controle de execução passa para a subtarefa, enquanto ela está realizando seu processamento, a tarefa não pode interrompê-la nesse período. Mas, como toda subtarefa possui uma referência para a instância da tarefa que a contém (parâmetro de controle), ela pode monitorar o estado da tarefa e interromper sua própria execução, enquanto o controle da execução estiver com ela. Esse procedimento foi projetado dessa maneira porque cada subtarefa tem uma função específica, que, muitas vezes, não pode ser interrompida. Assim, como a tarefa não tem como controlar se uma subtarefa pode ou não ser

interrompida, e em que ponto ela pode ser interrompida, esse controle foi delegado para a própria implementação da subtarefa.

```
private void processState() {
    final String login = getUser();
    switch (getState()) {
        case FINALIZED:
            sgt.removeTask(login, this);
            setState(TaskState.CLOSED);
            finalize();
            break;
        case PAUSED:
            break;
        case CANCELED:
            sgt.removeTask(login, this);
            setState(TaskState.CLOSED);
            finalize();
            break;
        case ERROR:
            sgt.removeTask(login, this);
            finalize();
            break;
        case RUNNING:
            break;
    }
}
```

Quadro 10 - Processa estado da tarefa

4.7 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 (ver Quadro 11).

Essa funcionalidade, que a princípio poderia estar incluída no SGT, foi implementada como um novo serviço do *middleware*, por questões de desempenho e integração com a arquitetura do EXEHDA. Se essa funcionalidade fosse integrada ao SGT, as informações sobre as tarefas ativas estariam distribuídas pelas instâncias do serviço nos diversos EXEHDAnodes pelos quais o usuário passou durante seu trabalho. Isso tornaria o

gerenciamento da consistência dessa informação muito complexo. Uma alternativa seria implementar dois tipos de SGT: um cliente, que somente gerenciaria as tarefas; e um servidor, que, além de gerenciar as tarefas, manteria as informações sobre as tarefas ativas. Porém, essa solução duplicaria todo o código de gerenciamento das tarefas em duas classes distintas, o que ocasiona problemas de manutenção de código.

```
protected void handleRequest(ServiceRequest req) throws IOException {
    switch (getSgtOperation(req)) {
        case SGT_OP_GET:
            handleSgtGet(req);
            break;
        case SGT_OP_ADD:
            handleSgtAdd(req);
            break;
        case SGT_OP_REM:
            handleSgtRem(req);
            break;
        case SGT_OP_UPD:
            handleSgtUpd(req);
            break;
    }
}
```

Quadro 11 - API para controle das tarefas ativas

Por isso, optou-se por separar o gerenciamento das tarefas e o controle de tarefas ativas em dois serviços distintos. Dessa forma, o SGT executa localmente em cada EXEHDAnode, e somente o Serviço de Tarefas Ativas funciona como um serviço cliente/servidor. Esse tipo de arquitetura também é usado para outros serviços do *middleware*.

4.8 Integração com os serviços atuais do EXEHDA

O Subsistema de Gerenciamento Distribuído de Tarefas (SGDT) foi modelado como um conjunto de serviços do *middleware*, de forma que cada serviço do SGDT pode ser adicionado e configurado no perfil de execução do *middleware* (ver Apêndice B). Mantendo a organização do EXEHDA, os serviços do SGDT são “plugáveis”, o que permitiu que fosse implementado um protótipo com apenas os serviços básicos: Serviço de Gerenciamento de

Tarefas (núcleo do gerenciamento de tarefas), Serviço de Tarefas Ativas e parte do Serviço de Acesso a Tarefas.

O gerenciamento dos serviços no EXEHDA é feito por um serviço pertencente ao núcleo básico do *middleware*, chamado *ServiceManager*. O *ServiceManager* obtém o perfil de execução através de um *ProfileManager*, e ativa os serviços de acordo com suas configurações no perfil de execução.

Todo serviço do EXEHDA, para ser gerenciado pelo *ServiceManager*, deve implementar a interface *Service*, a qual define os métodos `start()` e `stop()`, utilizados pelo *ServiceManager*. Além disso, os serviços podem estender um serviço abstrato chamado *AbstractService*, que implementa algumas facilidades para gerenciar o *log* do serviço.

Para ser localizado pelo *ServiceManager*, todo serviço deve definir e registrar um nome, através do qual ele será identificado no *middleware*. Esse nome é definido como um atributo público da interface do serviço, para que outros serviços e OXs tenham acesso a ele. O registro do nome do serviço no *middleware* é feito durante sua inicialização, através do nome definido no perfil de execução.

O Serviço de Acesso a Tarefas interage diretamente com o Subsistema de Acesso Pervasivo do EXEHDA para buscar, no AVU, os descritores das tarefas e subtarefas e a lista de tarefas disponíveis ao usuário. Dessa forma, o protótipo implementado exige que cada usuário possua, no seu AVU, uma cópia dos descritores de suas tarefas (cópia personalizada), bem como, um arquivo contendo a lista de tarefas disponíveis a ele.

Como visto na sessão 4.1, futuramente as informações sobre as tarefas e subtarefas serão buscadas no banco de dados ontológico. Nesse caso, o Serviço de Acesso a Tarefas usará o Subsistema de Acesso Pervasivo somente para conseguir uma referência (endereço) para o banco de dados, que é considerado, pelo *middleware*, um recurso.

Além disso, o Subsistema de Acesso Pervasivo, possivelmente, será utilizado pelo Serviço de Colaboração, quando esse for implementado.

O Serviço de Contexto de Tarefas interliga o SGDT ao Subsistema de Reconhecimento de Contexto do EXEHDA. Basicamente, o Serviço de Contexto de Tarefas obtém as informações de contexto do *middleware* e disponibiliza-as, em alto nível, para as tarefas e subtarefas, e para os outros serviços do SGDT.

O Serviço de Gerenciamento de Tarefas (SGT) não interage diretamente com nenhum outro serviço do EXEHDA. Porém, apesar de ele não invocar nenhum serviço do *middleware*, a execução das tarefas e subtarefas (aplicações pervasivas) é gerenciada pelo Subsistema de

Execução Distribuída do EXEHDA. Por isso, pode-se dizer que o SGT interliga o Subsistema de Gerenciamento Distribuído de Tarefas ao Subsistema de Execução Distribuída.

Na interligação desses dois subsistemas é que ocorre a "conversão" de tarefas em aplicações pervasivas. O SGT obtém a tarefa na sua forma abstrata (descrição ontológica programada pelo usuário) e a instancia em sua forma concreta, ou seja, uma aplicação pervasiva que gerencia a execução e a troca de informações das subtarefas que estão relacionadas na descrição ontológica. Após ser instanciada, a tarefa executa como qualquer outra aplicação pervasiva, utilizando os recursos do EXEHDA, e sendo gerenciada por ele.

Porém, o SGT continua tendo o controle sobre a tarefa. Quando o usuário ou uma mudança no contexto provoca a interrupção ou o cancelamento da tarefa, o SGT altera o estado dessa tarefa, fazendo com que ela processe essa nova situação e interrompa ou cancele a execução de suas subtarefas.

No processo de migração é que a tarefa se diferencia de uma aplicação pervasiva comum gerenciada pelo Subsistema de Execução Distribuída do EXEHDA. Enquanto uma aplicação comum é transferida de forma independente, por esse subsistema, entre os dispositivos, uma tarefa está sempre associada à sessão do usuário (médico). Por isso, ela não migra de forma independente, através do Subsistema de Execução Distribuída. Ao invés disso, uma tarefa só migra acompanhando a sessão do usuário.

A sessão do usuário é representada pela interface de interação dele com o sistema ClinicSpaces, a qual atua como um agente móvel, seguindo o usuário de acordo com as mudanças de contexto. Assim, a migração desse "agente" dispara a migração das tarefas. O funcionamento do agente móvel será abordado na sessão 4.10.

Como conseqüência da migração das tarefas não ser diretamente (por invocação do serviço) gerenciada pelo Subsistema de Execução Distribuída do *middleware*, cabe ao SGT gerenciar a ativação e desativação da tarefa, cada vez que ela é migrada. Durante a ativação, a tarefa se reconfigura para o contexto atual, obtendo as informações necessárias através do Serviço de Contexto de Tarefas.

Já as subtarefas, devem implementar métodos específicos para ativação e desativação, pois, uma vez que as subtarefas geralmente fazem parte do pEHS e possuem suas próprias interfaces gráficas, a adaptação ao contexto (ativação) é delegada a elas, visto que a tarefa não possui informações suficientes para fazer tal adaptação. Assim, como as subtarefas são gerenciadas pela tarefa, esta apenas executa os métodos para ativação e desativação de suas subtarefas.

O Serviço de Tarefas Ativas (STA) foi implementado como uma extensão do *HttpService* do EXEHDA, o qual é usado como base para todos os serviços do tipo cliente/servidor do *middleware*. Para isso, foi definida uma URL para esse serviço. Essa URL é interceptada por um tratador (*Handler*) específico do STA, que a converte em uma requisição HTTP. Dessa forma, os outros serviços do *middleware* podem utilizar o STA sem terem uma instância dele, apenas construindo a URL de acordo com a especificação e solicitando uma conexão para essa URL.

Se a requisição é feita num host o qual está executando o cliente do STA, ela é encaminhada ao servidor, que executa a operação solicitada e devolve o resultado ao cliente, que repassa a informação ao serviço requisitante. No entanto, se a requisição é feita no host servidor, ela é tratada localmente, sendo processada diretamente pelo servidor STA.

O formato especificado para a URL foi: *sgt://usuario@cell/op*. Onde:

- sgt – indica o protocolo usado;
- usuario – indica para qual usuário a operação está sendo solicitada;
- cell – indica a EXEHDAcell à qual pertence o usuário;
- op – indica a operação que está sendo solicitada.

As operações atualmente implementadas são:

- add – adiciona uma tarefa à lista do usuário;
- rem – remove uma tarefa da lista do usuário;
- upd – atualiza as informações de uma tarefa da lista do usuário;
- get – obtém a lista de tarefas ativas do usuário.

Outras operações, como troca de tarefas entre usuários, podem ser adicionadas facilmente, sem grandes alterações no serviço, apenas adicionando sua funcionalidade na classe servidora do STA.

4.9 Integração com o Sistema pEHS

Para iniciar a modelagem da arquitetura ClinicSpaces, supõe-se a existência de um sistema pervasivo de informação de saúde denominado pEHS (pervasive *Electronic Health-care System*)[16], que é um software de gerenciamento de informações de saúde com

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

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 ao 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 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).

4.10 Interação com a interface gráfica do usuário

A interface gráfica para o usuário está sendo estudada em outra frente de pesquisa do projeto ClinicSpaces[17]. Apesar de o foco desse estudo ser a interface de construção e personalização de tarefas, a interface para gerenciamento das tarefas é uma extensão daquela, pois as mesmas questões de *End-User Programming* devem ser aplicadas em ambas.

Como visto no na sessão 4.6, 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).

[17] Trabalho de dissertação (em andamento) do mestrando Fábio Lorenzi da Silva (PPGI - UFSM)

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.

Segundo a especificação do EXEHDA, essa aplicação (ou agente) deve possuir um ativador (uma classe que implementa a interface *Activator* definida pelo *middleware*). Esse ativador define dois métodos: *deactivate*, invocado pelo Subsistema de Execução Distribuida antes da migração; e *activate*, invocado quando a aplicação pervasiva é instanciada e quando ela chega ao destino de uma migração. Assim, o primeiro deve salvar o estado interno da aplicação, salvando tudo que o usuário estava fazendo na interface do SGDT. Já o segundo deve remontar o estado da aplicação e ativar a interface adequada ao contexto.

5 DISCUSSÃO E RESULTADOS

5.1 Discussão de trabalhos relacionados

O Projeto Gaia (ROMAN et al, 2002) deu origem ao *middleware* Gaia. Esse *middleware* estende as funcionalidades de um sistema operacional tradicional, provendo gerenciamento do contexto, suporte a computação móvel e gerenciamento de atuadores. Dessa forma, as aplicações podem ser construídas de forma genérica (sem suposições sobre hardware ou ambiente) e o *middleware* encarrega-se de adaptá-las de acordo com os recursos de cada ambiente.

O projeto ISAM (ISAM, 2001) (AUGUSTIN et al, 2002), assim como o Gaia, deu origem a um *middleware* de gerenciamento do ambiente pervasivo, que integra as premissas da computação em grade, da computação móvel, e da computação consciente do contexto. O *middleware* EXEHDA (EXEHDA, 2001) (YAMIN, 2004) cria e gerencia um ambiente virtual para o usuário, no qual as aplicações executam de forma distribuída e adaptável ao contexto. Assim, o EXEHDA permite ao usuário ter acesso ao seu ambiente computacional independentemente de localização e de tempo.

O conceito de Computação Baseada em Tarefas foi introduzido pelo Projeto Aura (GARLAN et al, 2002) como um meio de o *middleware* gerenciar o ambiente pervasivo de forma que o usuário possa manter a continuidade de suas atividades, ao mesmo tempo em que ele desloca-se de um lugar a outro. Nesse projeto, tarefas são modeladas como coleções de serviços; a descrição do serviço é usada para encontrar os recursos necessários ou reconfigurar o *midleware* para executar a tarefa. Porém, o sistema Aura é automatizado e pró-ativo, e não permite a personalização das tarefas (serviços), o que leva ao aumento da interferência do sistema no ambiente.

O projeto Activity-based Computing (BARDRAM; CHRISTENSEN, 2007) apresenta uma proposta de utilização da Computação Baseada em Tarefas em ambientes da saúde. Nesse projeto, foi desenvolvido um framework que provê a infra-estrutura necessária para execução de serviços que dão suporte às características inerentes ao trabalho dos profissionais

da saúde. Dessa forma, os serviços podem ser inicializados, suspendidos, armazenados, retomados em qualquer dispositivo e a qualquer tempo, encaminhados para outros usuários ou compartilhados entre diversos usuários. O projeto visa, ainda, permitir que desenvolvedores de aplicações clínicas possam incorporar, aos seus programas, suporte a mobilidade, interrupções, atividades paralelas e cooperação.

Muitas das idéias apresentadas por esses projetos influenciaram este trabalho. Destacando-se o EXEHDA, que foi utilizado como plataforma-base para este trabalho, ou seja, *middleware* para gerenciar o ambiente pervasivo; e 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 12, nenhum dos projetos apresenta a visão centrada no usuário final defendida pelo Projeto ClinicSpaces. Este projeto apresenta o diferencial de permitir aos clínicos a programação personalizada das tarefas, e de possibilitar o balanceamento entre a execução automatizada e o controle sobre a execução das tarefas. O primeiro conseguido através do agrupamento de subtarefas representando os passos que o médico costuma realizar para desenvolver sua atividade. O segundo, através da disponibilização de meios para o médico agendar suas tarefas ou relacioná-las a determinados contextos (automatização), bem como, para controlar manualmente as tarefas (executar, interromper, continuar, cancelar).

Apesar de existirem outros projetos sendo desenvolvidos na área de Computação Orientada a Atividades, a maioria deles está focado em *SmartHomes*, o que dificulta uma comparação mais efetiva, devido aos requisitos dessa área não serem os mesmos da *UbiHealth*.

	Computação Pervasiva		**Computação Baseada em Tarefas**		
	Mobilidade	Contexto	Automatização	Personalização	Controle
Gaia	X	X			
ISAM	X	X			
Aura	X	X	X		
Activity-based Computing	X	X	X		X
ClinicSpaces	X	X	X	X	X

Quadro 12 - Comparativo entre os projetos relacionados a este trabalho

5.2 Avaliação dos resultados

5.2.1 Protótipo

Como forma de avaliar a proposta desenvolvida neste trabalho, bem como o impacto do gerenciamento de tarefas no *middleware* e na execução das aplicações pervasivas (tarefas), foi implementado um protótipo do Subsistema de Gerenciamento Distribuído de Tarefas, contendo o Serviço de Gerenciamento de Tarefas (SGT), o Serviço de Acesso a Tarefas (SAT) e o Serviço de Tarefas Ativas (STA). Esses serviços foram integrados ao EXEHDA e instanciados em dois nodos: base e cliente. No nodo base são mantidas as descrições ontológicas das tarefas (acessadas pelo SAT) e as tarefas ativas (quando o usuário não está usando nenhum outro nodo do sistema). O nodo cliente é usado pelo usuário para executar suas tarefas.

Para testar e avaliar o funcionamento dos serviços foram implementadas quatro subtarefas, as quais representam: identificação do paciente, busca do prontuário eletrônico, exibição das informações do prontuário e adição de informações ao prontuário. Usando-se essas quatro subtarefas foi criada uma tarefa que simula o atendimento a um paciente.

Já para simular a interação do usuário com os serviços foi implementada uma interface gráfica simples (Figura 9), a qual disponibiliza botões para controle das tarefas. Assim, a interface mostra uma lista de tarefas disponíveis e uma de tarefas ativas, com seus respectivos estados (executando ou interrompida). Como forma de interação, a interface gráfica permite iniciar as tarefas disponíveis, e pausar, retomar ou cancelar as tarefas ativas. Além disso, a interface gráfica disponibiliza um comando para disparar a migração da sessão do usuário para determinado host, simulando a mobilidade do usuário.

5.2.2 Metodologia

Como visto anteriormente, para avaliar o funcionamento do Subsistema de Gerenciamento Distribuído de Tarefas (SGDT), o EXEHDA (com os novos serviços integrados) foi configurado e inicializado em dois nodos: um nodo base, responsável pelo

gerenciamento da célula; e um nodo comum (cliente), representando um dispositivo de usuário, o qual faz parte da célula.

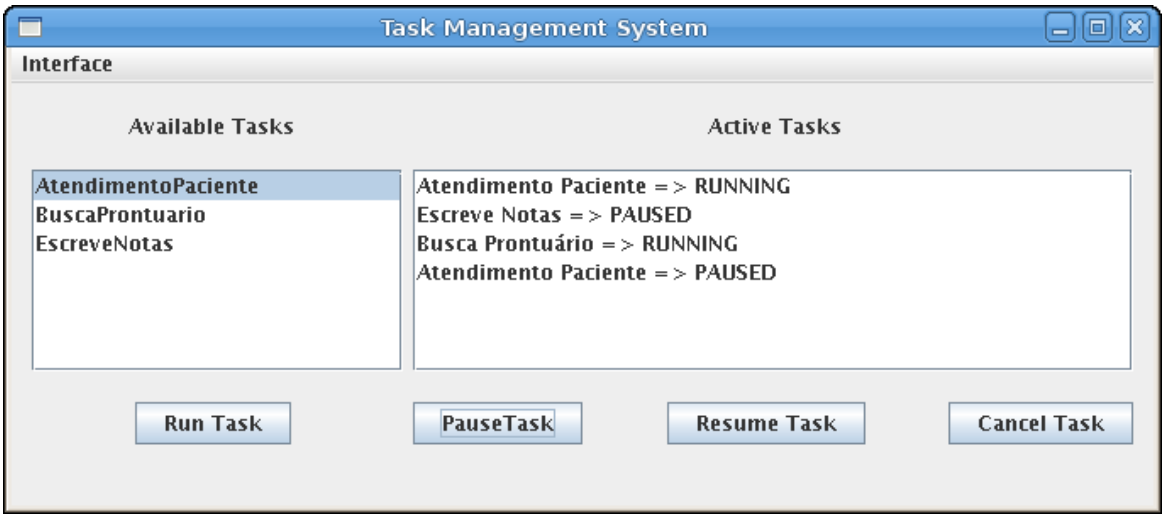

Figura 9 - Interface de interação com o SGDT para testes e avaliação do protótipo

Dessa forma, a aplicação do usuário (interface gráfica que permite a interação dele com suas tarefas) é iniciada e, através da API do SGDT, busca as tarefas disponíveis ao usuário, bem como, as tarefas que foram iniciadas e não concluídas em outra sessão (tarefas ativas). Como essas informações estão localizadas no nodo-base, as instâncias locais dos serviços SAT e STA geram requisições às respectivas instâncias remotas. Por isso, as operações desses dois serviços foram monitoradas nos testes para determinar o impacto deles no sistema. Assim, foram analisados o número de requisições feitas por esses serviços e o tempo para inicialização das tarefas.

O gerenciamento (controle) da execução das tarefas é feito, localmente, pela instância do Serviço de Gerenciamento de Tarefas. Desse modo, esse serviço foi monitorado em termos de tempo de processamento para controle das tarefas, ou seja, quanto tempo é gasto com gerenciamento durante a execução das tarefas. Esse tempo foi avaliado de forma absoluta, uma vez que relacioná-lo com o tempo total de execução da tarefa se torna impreciso, já que cada usuário realiza suas tarefas de forma diferente e em tempos variados.

Os equipamentos usados nos testes foram:

- EXEHDAbase: Pentium Core 2 Duo, 3.0 GHz, 2GB RAM;
- EXEHDAnodo: AMD Turion, 2.4 GHz, 2GB RAM;
- Rede Ethernet 10/100 Mbps.

5.2.3 Resultados

A avaliação do Serviço de Acesso a Tarefas mostrou que para carregar uma tarefa é feita uma requisição para obter o descritor da tarefa e mais uma para cada subtarefa que a compõe, já que o *parser* processa o descritor seqüencialmente. Esse comportamento deve-se ao fato de o protótipo ainda não estar usando o banco de dados ontológico que está sendo desenvolvido para o projeto ClinicSpaces, o que permitiria buscar toda a informação necessária com apenas uma requisição.

Quanto ao tempo necessário para inicialização das tarefas, os testes de avaliação apresentaram valores entre 110ms e 190ms, com média em torno de 135ms. Além disso, observou-se certo aumento nessa média quando se considera apenas a execução da 1ª tarefa (logo após o *middleware* ser inicializado), como se pode observar na Figura 10. Esse aumento é causado pela inicialização dos serviços do *middleware*. Outro ponto observado foi um leve aumento no tempo de inicialização da tarefa em conseqüência de um aumento no número de subtarefas que a compõe. Porém esse aumento está relacionado com a complexidade das tarefas e subtarefas (gerando objetos mais "pesados" para serem carregados) e não diretamente com os serviços do *middleware*.

Já a avaliação do Serviço de Tarefas Ativas, apontou alguns pontos a serem melhorados no protótipo desse serviço. Para manter consistentes as informações sobre a execução das tarefas, o serviço centraliza esses dados na base da célula. Dessa forma, é feita uma requisição para adicionar a tarefa à lista do usuário (inicialização), uma para removê-la da lista (conclusão da tarefa), e várias para atualizar o estado de execução da tarefa. É nesse ponto que o serviço pode ser melhorado. Apesar de não ser um problema para a execução normal do sistema, o número de requisições (em torno de 15) para atualização do estado, durante o ciclo de vida da tarefa, pode transformar esse serviço em um gargalo, quando se pensa em escalabilidade.

Dessa forma, poder-se-ia armazenar o estado das tarefas na instância local do serviço e enviar à instância remota (centralizadora) somente em determinadas situações, como migração para outro dispositivo ou encerramento da aplicação (interface). Porém, deve-se levar em consideração que, no caso de ocorrer uma falha no dispositivo (como falta de bateria), o usuário perderia parte de sua tarefa, e teria que refazer essa parte em outro dispositivo. Como sugestão para melhoramento deste serviço, seria interessante ter uma forma de configurá-lo de acordo com o dispositivo usado, considerando suas características.

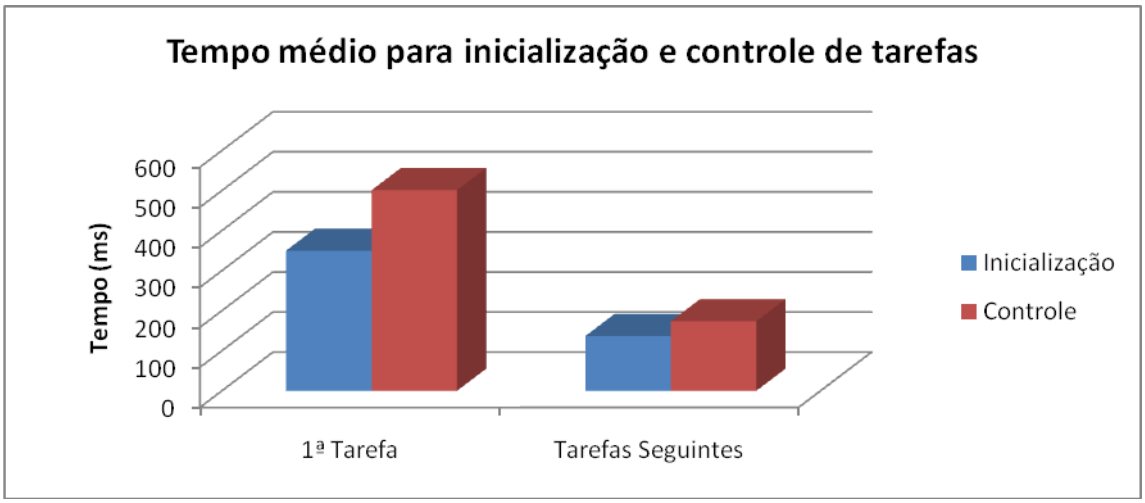

Figura 10 - Gráfico do tempo médio para inicialização e controle da 1ª tarefa e das tarefas seguintes

O Serviço de Gerenciamento de Tarefas foi avaliado em termos de tempo utilizado para gerenciamento das tarefas. Nos testes, esse tempo ficou entre 130ms e 200ms, como média em torno de 174ms. Assim como o tempo para inicialização da tarefa, o tempo para controle é maior na execução da primeira tarefa, como pode ser observado na Figura 10, também devido à inicialização do serviço.

Durante os testes, também foram avaliadas questões relativas à escalabilidade, como gerenciamento de tarefas concorrentes. A Figura 11 mostra uma comparação entre os tempos médios para inicialização e controle de uma tarefa e os tempos médios para inicialização e controle de várias tarefas (de 5 a 10) simultaneamente. Como se pode observar no gráfico, os tempos aumentaram, mas ainda estão dentro de níveis aceitáveis, ficando em torno de 180ms para inicialização e 290ms para gerenciamento.

Ainda foram comparados os tempos para inicialização e gerenciamento de várias tarefas localmente (inicializando e executando no mesmo nodo) com os tempos para inicialização e gerenciamento de tarefas com migração (iniciando em um nodo e continuando a execução em outro). A Figura 12 apresenta essa comparação. Como se pode observar, o tempo para inicialização praticamente não mudou, uma vez que esta não é influenciada pela migração. Já o tempo para gerenciamento teve um pequeno aumento, indicando que a migração das tarefas tem certo impacto sobre o gerenciamento. Na prática, o processamento extra deve-se à necessidade de interrupção de todas as tarefas antes da migração e de retomada das mesmas após chegarem ao destino. Ressalta-se que o tempo para migração (transferência dos objetos), apesar de muito baixo, não foi contabilizado nos gráficos, uma

vez que trata-se de um serviço já disponibilizado pelo *middleware*, e por isso, não faz parte do escopo dessa avaliação.

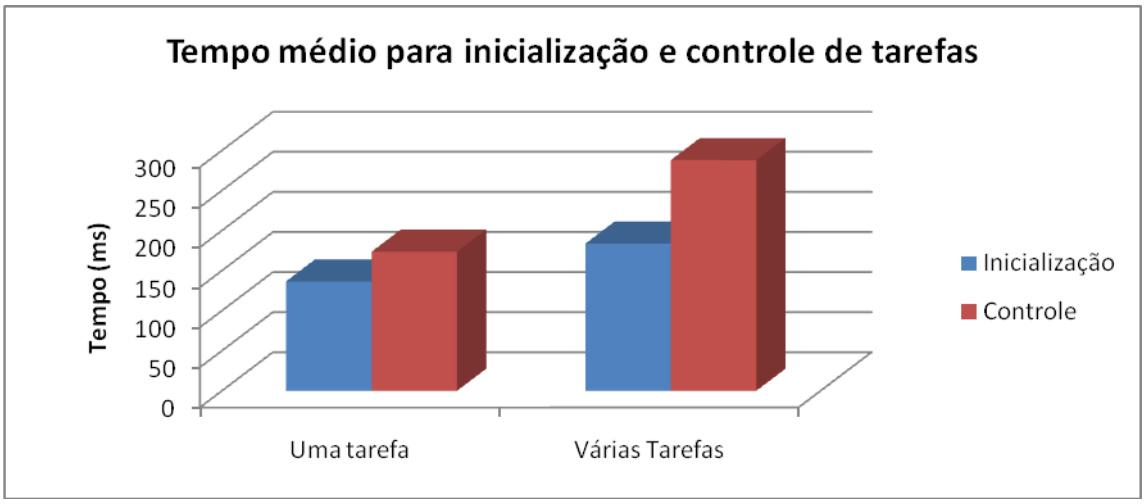

Figura 11 - Gráfico do tempo médio para inicialização e controle de uma tarefa e várias tarefas simultâneas

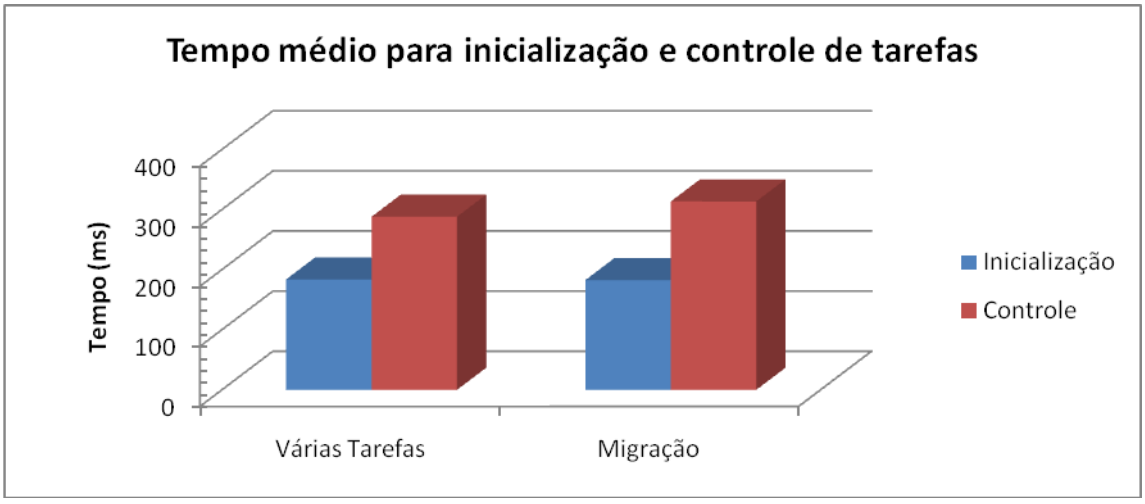

Figura 12 - Gráfico do tempo médio para inicialização e controle de tarefa com e sem migração

5.2.4 Conclusão

A avaliação apresentada nesta sessão mostrou que a proposta de adicionar, a um *middleware* da computação pervasiva, suporte ao gerenciamento de tarefas cotidianas de profissionais da saúde é promissora e praticável. O impacto dos novos serviços, necessários

ao gerenciamento das tarefas, foi mínimo, tanto do ponto de vista do *middeware* como do usuário que está executando as aplicações (tarefas).

Além disso, a avaliação dos serviços indicou pontos que podem ser melhorados no protótipo, como foi o caso do Serviço de Tarefas Ativas. Esses melhoramentos podem ser feitos sem grande impacto nos outros serviços, já que, tendo-se a interface fixa, a implementação pode ser modificada de nodo para nodo.

Portanto, de modo geral, o resultado foi bastante satisfatório. O protótipo correspondeu ao esperado e funciona juntamente com EXEHDA sem gerar sobrecarga ao *middleware*. Dessa forma, possivelmente servirá para testar e avaliar outras partes do sistema ClinicSpaces.

6 CONCLUSÕES

Em sua primeira geração, a Computação Pervasiva focalizou o desenvolvimento de *middlewares* para gerenciamento e disponibilização do ambiente pervasivo, com o objetivo de entender os requisitos e as necessidades desse ambiente. Atualmente, inicia-se a preocupação com as questões de atividades humanas cotidianas e centralização no usuário-final, que podem ser consideradas de sua segunda geração; assim, uma frente da Computação Pervasiva está se direcionando ao desenvolvimento de ambientes programáveis pelo usuário-final e interativos, os quais auxiliarão o usuário em suas atividades cotidianas.

Tendo em vista o atual enfoque da Computação Pervasiva em atividades humanas e o potencial de sua aplicabilidade na área da saúde, esse trabalho, dentro do escopo do projeto ClinicSpaces, propôs pesquisar uma forma de adicionar, a um *middleware* de gerenciamento do ambiente pervasivo, suporte à computação orientada a tarefas.

Até o momento, o desenvolvimento da arquitetura projetada no projeto ClinicSpaces foi dividido em várias frentes de atuação, dentre elas: modelagem de tarefas (Fabio Lorenzi da Silva); ontologias para tarefas clínicas (Jonas Gassen); modelagem do contexto para tarefas clínicas (Tiago Antonio Rizzetti); modelagem de um sistema para gerenciamento das tarefas (foco dessa dissertação), pEHS (Caroline Vicentini e Alencar Machado), sistema de Inferência e Disparo Automático de Tarefas (Marcos Vinícius B. de Souza), Colaboração e Comunicação (Marcelo Kroth), Análise da Interface mais Intuitiva para o Usuário-Final (Mikael de Souza Fernandes). Dessa forma, este trabalho descreve uma primeira abordagem para a modelagem do gerenciamento de tarefas e a arquitetura de software correspondente.

Na modelagem do sistema de gerenciamento de tarefas, o *middleware* EXEHDA foi escolhido para gerenciar o ambiente pervasivo. E, a ele, foram adicionados novos serviços necessários ao gerenciamento de tarefas. Alguns desses serviços dependem do resultado das outras frentes de atuação do projeto ClinicSpaces. Por isso, esses serviços foram modelados conceitualmente, em termos de funcionalidades que eles devem prover e possíveis formas de implementá-los, deixando-se a definição dos detalhes comportamentais para serem aprofundadas em outros trabalhos, adicionados incrementalmente na seqüência de desenvolvimento do projeto.

Portanto, o primeiro resultado desse trabalho foi o modelo de arquitetura para o novo Subsistema de Gerenciamento Distribuído de Tarefas (SGDT) do EXEHDA, que é responsável pelo gerenciamento de tarefas no ambiente pervasivo. Esse novo subsistema segue a mesma especificação dos outros subsistemas do *middleware*. Por isso, seus serviços também podem ser "plugados" independentemente uns dos outros.

Cada serviço projetado do SGDT possui funcionalidades distintas e refletem as frentes de atuação do projeto ClinicSpaces. Assim, o Serviço de Contexto de Tarefas é resultado do estudo sobre contexto para tarefas clínicas; o Serviço de Acesso a Tarefas será produto do estudo sobre ontologias para tarefas clínicas e banco de dados ontológico; o Serviço de Interceptação sofrerá influência do estudo sobre sistemas pervasivos para gerenciamento de informações de saúde (pEHS); o Serviço de Colaboração e o Serviço de Inferência ainda estão em fase de levantamento de requisitos; finalmente, o Serviço de Gerenciamento de Tarefas, núcleo do SGDT, é produto direto do estudo desenvolvido neste trabalho.

Conseqüentemente, o resultado secundário deste trabalho foi um protótipo[18] do Serviço de Gerenciamento de Tarefas, já prevendo certa interação com os demais serviços. Esse protótipo poderá ser usado, durante o andamento do projeto ClinicSpaces, como base para teste e avaliação dos outros serviços, bem como, para facilitar a participação de profissionais de outras áreas (saúde, principalmente) no prosseguimento do projeto. Essa forma de participação de outros profissionais foi prevista desde o início do projeto. Nesse ponto, julga-se que, tendo um protótipo em cima do qual são feitas as discussões, avaliações e sugestões, pode-se obter maior retorno por parte dos profissionais de outras áreas (fora da computação).

Concluindo, a modelagem completa de uma arquitetura para gerenciamento de tarefas (as quais representam atividades humanas) em um ambiente pervasivo é um trabalho que envolve estudos aprofundados em diversas áreas. Por isso, neste trabalho foi modelada a arquitetura para o gerenciamento de tarefas, definindo-se suas funcionalidades (serviços do *middleware*) e requisitos, e deixou-se o detalhamento e a implementação das funcionalidades para trabalhos específicos.

[18] Maiores detalhes sobre o protótipo podem ser encontrados no Apêndice F (diagrama de classes do protótipo).

6.1 Trabalhos em andamento

A Figura 13 mostra um resumo da arquitetura projetada para o ClinicSpaces, indicando também os trabalhos que estão sendo desenvolvidos nas várias frentes de atuação do projeto.

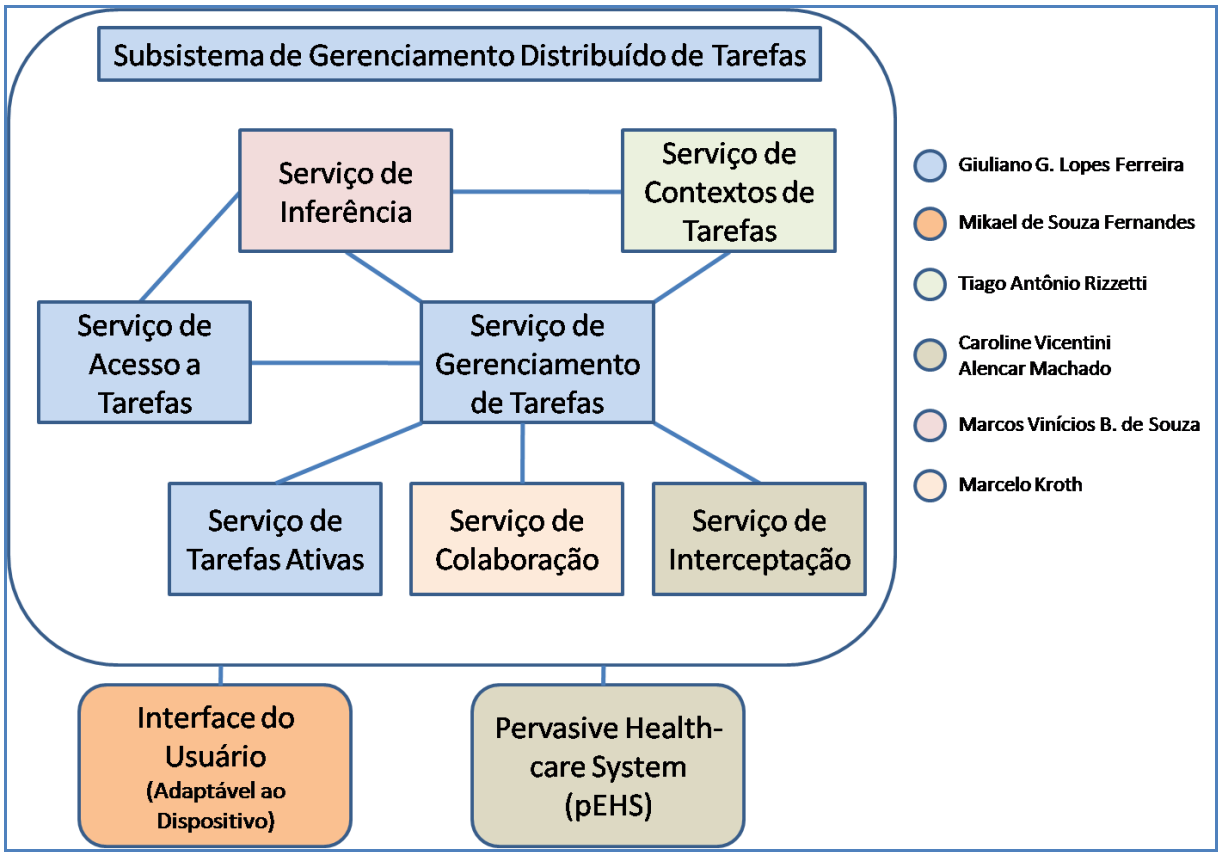

Figura 13 - Arquitetura ClinicSpaces e suas frentes de atuação

6.2 Publicações

As publicações relativas ao trabalho desenvolvido são:

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.

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)

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 – “Adaptando o Middleware EXEHDA para o Tratamento de Atividades Clínicas”. In: XXXV Conferência Latino-Americana de Informática. (submetido à publicação)

REFERÊNCIAS

AUGUSTIN, Iara et al. ISAM: a Software Architecture for Adaptive and Distributed Mobile Applications. **Proceedings of the Seventh International Symposium on Computers and Communications**, 2002.

AUGUSTIN, Iara et al. ISAMadapt - um Ambiente de Desenvolvimento de Aplicações para a Computação Pervasiva. In: 8TH BRAZILIAN SYMPOSIUM ON PROGRAMMING LANGUAGES, 26-28 maio 2004, Niterói. **Anais eletrônicos...** Universidade Federal Fluminense: Niterói, 2004a. Disponível em: <http://sblp2004.ic.uff.br/papers/augustin-yamin-silva-real-geyer.pdf>. Acesso em: 16 jun. 2008.

AUGUSTIN, Iara et al. ISAM, joing context-awareness and mobility to building pervasive applications. In: **Mobile Computing Handbook**. I. New York: CRC Press, 2004b.

AUGUSTIN, Iara et al. ISAMadapt: abstractions and tools for designing general-purpose pervasive applications. **Software-Practice & Experience**, Volume 36, Issue 11-12, p. 1231-1256, 2006.

AUGUSTIN, Iara. **Abstrações para uma Linguagem de Programação visando Aplicações Móveis em um Ambiente de Pervasive Computing**. 2004. 194f. Tese (Doutorado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

AUGUSTIN, Iara; LIMA, João Carlos Damasceno; YAMIN, Adenauer Correa. Computação Pervasiva: como Programar Aplicações. In: X SIMPOSIO BRASILEIRO DE LINGUAGENS DE PROGRAMAÇÃO (SBLP), 2006, Itatiaia, RJ. **Anais...** [S.l.]: SBLP, 2006.

AUGUSTIN, Iara; YAMIN, Adenauer; GEYER, Cláudio F. Resin. Managing the follow-me semantics to build large-scale pervasive applications. **Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing**, p. 1-8, 2005.

BAGRODIA, R. et al. Vision, issues, and architecture for nomadic computing. **IEEE Personal Communications**, Volume 2, Issue 6, p. 14-27, 1995.

BARBOSA, Jorge Luis Victoria et al. Holoparadigm: a Multiparadigm Model Oriented to Development of Distributed Systems. **Proceedings of the Ninth International Conference on Parallel and Distributed Systems**, p. 165-170, 2002.

BARDRAM, Jakob E. Hospitals of the Future: Ubiquitous Computing support for Medical Work in Hospitals. **fifth International Conference on Ubiquitous Computing (Ubicomp 2003)**, p. , 2003.

BARDRAM, Jakob E. Activity-based computing: support for mobility and collaboration in ubiquitous computing. **Personal and Ubiquitous Computing**, vol. 9, issue 5, p. 312-322, 2005.

BARDRAM, Jakob E.; BALDUS, Heribert; FAVELA, Jesus. Pervasive Computing in Hospitals. In: **Pervasive Healthcare**: Research and Applications of Pervasive Computing in Healthcare. [S.l.]: CRC Press, 2006. p. 49-78.

BARDRAM, Jakob E.; BOSSEN, Claus. Mobility Work: The Spatial Dimension of Collaboration at a Hospital. **Computer Supported Cooperative Work**, vol. 14, issue 2, p. 131-160, 2005.

BARDRAM, Jakob E.; CHRISTENSEN, Henrik 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.

CHRISTENSEN, Henrik Bærbak; BARDRAM, Jakob. Supporting Human Activities - Exploring Activity-Centered Computing. **Proceedings of the 4th international conference on Ubiquitous Computing**, p. 107-116, 2002.

DICKINSON, Ian. Jena Ontology API. 2008. Disponível em: <http://jena.sourceforge.net/ontology/index.html>. Acesso em: 12 jan. 2008.

EXEHDA. Ambiente de Execução direcionado à Computação Pervasiva. 2001. Disponível em: <http://www.inf.ufrgs.br/~exehda>. Acesso em: 14 jun. 2008.

FERREIRA, Vinícius Pereira. **GRADEp-MC**: Desenvolvimento Sobre o GRADEp de Aplicações Baseadas em Mobilidade de Código. 2007. 105f. Monografia (Curso de Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007.

GARLAN, D. et al. Project Aura: toward distraction-free pervasive computing. **IEEE Pervasive Computing**, vol. 1, issue 2, p. 22-31, 2002.

GOODELL, Howie. End-User Programming. 2005. Disponível em: <http://www.cs.uml.edu/~hgoodell/EndUser/>. Acesso em: 14 jun. 2008.

HOLZINGER, Andreas; SCHWABERGER, Klaus; WEITLANER, Matthias. Ubiquitous Computing for Hospital Applications: RFID-Applications to Enable Research in Real-Life Environments. **Proceedings of the 29th Annual International Computer Software and Applications Conference**, volume 2, p. 19-20, 2005.

ISAM. Infra-estrutura de Suporte às Aplicações Móveis. 2001. Disponível em: <http://www.inf.ufrgs.br/~isam>. Acesso em: 14 jun. 2008.

KAENAMPORNPAN, Manasawee; O'NEILL, Eamonn. 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.

KALOFONOS, Dimitris N.; REYNOLDS, Franklin D.. Task-Driven End-User Programming of Smart Spaces Using Mobile Devices. In: Technical Report NCR-TR-2006-001 of Nokia Research Center. Cambridge: Nokia Research Center, 2006. Disponível em: <http://research.nokia.com/files/NRC-TR-2006-001.pdf>. Acesso em: 14 jun. 2008.

LAERUM, Hallvard; FAXVAAG, Arild. Task-oriented evaluation of electronic medical records systems: development and validation of a questionnaire for physicians. **BMC Medical Informatics and Decision Making 2004**, vol. 4, n. 1, p. 1-16, 2004.

LIBRELOTTO, Giovani Rubert 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.

RANGANATHAN, A. et al. Towards a pervasive computing benchmark. **Proceedings of Third IEEE International Conference on Pervasive Computing and Communications Workshops**, p. 194-198, 2005.

RANGANATHAN, Anand; CAMPBELL, Roy H. Self-Optimization of Task Execution in Pervasive Computing Environments. **Proceedings of the Second International Conference on Automatic Computing**, p. 333-334, 2005a.

RANGANATHAN, Anand; CAMPBELL, Roy H. Supporting Tasks in a Programmable Smart Home. In: **From Smart Homes to Smart Care, vol. 15**. Amsterdam: IOS Press, 2005b. p. 3-10.

ROMAN, Manuel et al. A Middleware Infrastructure for Active Spaces. **IEEE Pervasive Computing**, vol. 1, issue 4, p. 74-83, 2002.

SAHA, Debashis; MUKHERJEE, Amitava. Pervasive Computing: a Paradigm for the 21st Century. **IEEE Computer**, vol. 36, issue 3, p. 25-31, 2003.

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

W3C. XML Schema. 2000. Disponível em: <http://www.w3.org/XML/Schema>. Acesso em: 12 nov. 2008.

WEISER, Mark. The Computer of the 21st Century. **Scientific American**, volume 265, número 9, 1991.

YAMIN, Adenauer et al. Collaborative Multilevel Adaptation in Distributed Mobile Applications. **Proceedings of the XII International Conference of the Chilean Computer Science Society**, p. 82, 2002a.

YAMIN, A. et al. A Framework for Exploiting Adaptation in Highly Heterogeneous Distributed Processing. **Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing**, p. 125-132, 2002b.

YAMIN, Adenauer Corrêa et al. Towards Merging Context-Aware, Mobile and Grid Computing. **International Journal of High Performance Computing Applications**, vol. 17, n. 2, p. 191-203, 2003.

YAMIN, Adenauer et al. EXEHDA: adaptive middleware for building a pervasive grid environment. In: **Frontiers in Artificial Intelligence and Applications**: Self-Organization and Autonomic Informatics (I), vol. 135. Amsterdam: IOS Press, 2005. p. 203-219.

YAMIN, Adenauer Corrêa. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. 194f. Tese (Doutorado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

ZENI, Cristine et al. Panorama do uso de computação móvel com conexão Wireless. In: IX CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, 2004, Ribeirão Preto. **Anais eletrônicos...** [S.l.:s.n], 2004. Disponível em: <http://www.sbis.org.br/cbis9/anais.htm>. Acesso em: 14 jun. 2008.

GLOSSÁRIO

Aplicação Pervasiva – Objeto do EXEHDA (OX) que executa no ambiente pervasivo gerenciado pelo *middleware*.

Atividades clínicas – Processos realizados por humanos de forma colaborativa, coordenada e distribuída em um espaço determinado. São auxiliadas por aplicações computacionais, e seguem a forma particular de cada indivíduo de realizá-la (personalização).

Subtarefa – Operações que compõem uma tarefa. Cada subtarefa possui um descritor (ontologia) e uma aplicação pervasiva (implementação), quase sempre associada a uma aplicação do pEHS.

Tarefa – Ações que compõem uma atividade humana. Tarefas simples são compostas por subtarefas. Tarefas compostas são compostas por outras tarefas (workflow).

APÊNDICE A – Arquitetura ISAM

A Figura 14 apresenta uma visão geral da arquitetura ISAM. Segundo Yamin (2004), a representação da consciência do contexto como um módulo virtual tem por objetivo ressaltar sua importância na arquitetura e caracterizar sua presença na concepção de todos os outros componentes. Como se pode observar na Figura 14, a arquitetura ISAM é dividida em três camadas: camada de aplicação, camada de *middleware*, e camada de sistemas básicos.

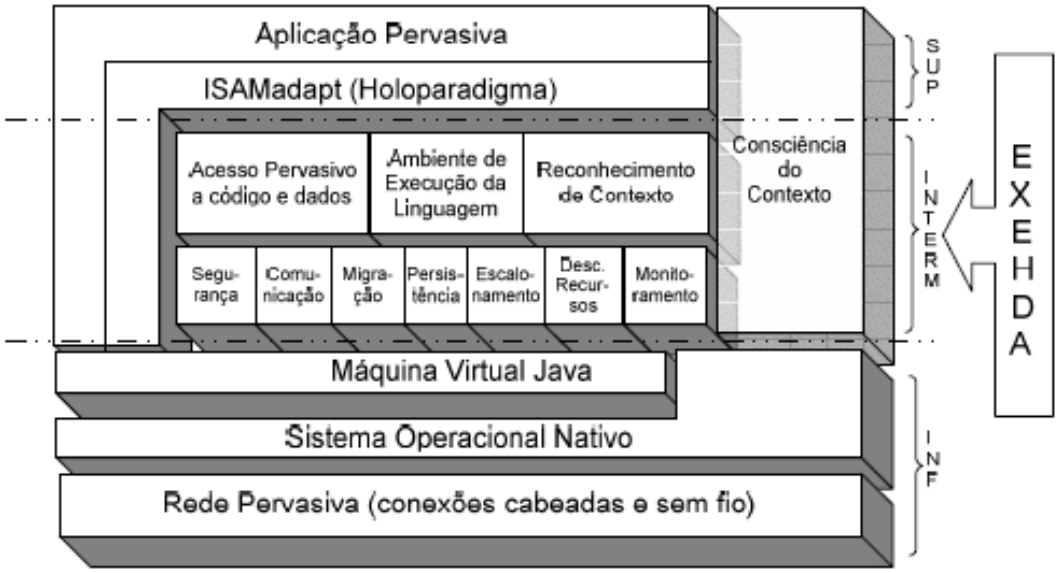

Figura 14 - Arquitetura ISAM [Fonte: (YAMIN, 2004, p. 59)]

Na camada de aplicação encontra-se a linguagem de programação ISAMadapt, a qual disponibiliza abstrações para o desenvolvimento de aplicações pervasivas conscientes do contexto. Mais detalhes sobre o ISAMadapt e suas abstrações podem ser obtidos nas publicações (AUGUSTIN, 2004), (AUGUSTIN et al, 2004a), (AUGUSTIN et al, 2006).

A camada de *middleware*, na qual estão os mecanismos de suporte à execução da aplicação pervasiva e os mecanismos de adaptação (YAMIN et al, 2002a) (YAMIN et al, 2002b) (YAMIN et al, 2003), é formada por dois níveis. O primeiro nível é composto por três módulos de serviço à aplicação: Acesso Pervasivo a Código e Dados, Reconhecimento de

Contexto, e Ambiente de Execução da Linguagem. O segundo nível, composto pelos serviços básicos do EXEHDA, provê as funcionalidades necessárias ao primeiro nível, bem como, disponibiliza serviços de migração, persistência, descoberta de recursos, comunicação, escalonamento e monitoração.

O módulo de acesso pervasivo a código e dados é responsável por disponibilizar o Ambiente Pervasivo (ISAMpe), o qual compreende o Ambiente Virtual do Usuário (AVU), o Ambiente Virtual da Aplicação (AVA) e a Base de Dados pervasiva das Aplicações (BDA). Já, o Ambiente de Execução da Linguagem é encarregado do gerenciamento da aplicação durante seu tempo de vida. E, por fim, o módulo de Reconhecimento de Contexto é responsável por informar o estado dos elementos de contexto de interesse da aplicação e do próprio ambiente de execução.

A camada de sistemas básicos é composta pela Máquina Virtual Java (JVM), podendo ser utilizada tanto a J2SE (Java Standard Edition) como a J2ME (Java Micro Edition), pelo sistema operacional sobre o qual a JVM é executada, e pela camada de rede pervasiva que deve integrar uma rede sem fio a uma rede cabeada infra-estruturada.

APÊNDICE B – EXEHDA

Controle da adaptação

Na perspectiva do EXEHDA, a adaptação não é uma propriedade funcional da aplicação. Assim, o tratamento da adaptação é realizado pela gerência de execução da aplicação de modo autônomo. Para isso, o código da aplicação é implementado de forma adaptativa e reflete as possibilidades de comportamento da aplicação frente às alterações do estado do elemento de contexto para o qual a aplicação é sensível (Augustin, 2004).

A adaptação refere-se à modificação do comportamento, da estrutura ou da interface da aplicação, em reposta a mudanças de estado dos elementos de contexto. Os tipos de adaptação que podem ser utilizados pela aplicação dependem da natureza dela e dos recursos que ela requer. Além disso, a adaptação deve ser dinâmica, devido às flutuações do estado do contexto que ficam potencializadas quando há deslocamento do usuário ou dos componentes da aplicação, e gerenciada de forma colaborativa com a aplicação.

O *middleware*, além de cooperar na adaptação da aplicação, também se adapta ao contexto. Atualmente, os principais elementos de contexto considerados pelo EXEHDA para adaptação são: o tipo de equipamento, o estado de ocupação dos seus recursos e a situação de sua conectividade no momento (YAMIN, et al., 2002a) (YAMIN et al, 2002b). É importante salientar que a adaptação ocorrerá se o sistema dispuser dos recursos necessários para sua efetivação, descartando-se situações onde o nível de disponibilidade de recursos inviabiliza o processo de adaptação.

Gerenciamento da execução de aplicações

Cada usuário, com uma sessão de trabalho ativa, está apto a executar as aplicações de seu interesse. Assim, os comandos de controle de sessão são necessários para viabilizar a implementação da semântica *siga-me* das aplicações. Os comandos de controle de sessão

podem ser manipulados através da aplicação ISAM Desktop, e os principais são descritos abaixo.

Login/Logout – o comando *login* é responsável pela autenticação do usuário junto ao serviço *Gatekeeper*. Esse procedimento é baseado num mecanismo de chave pública/privada, na qual a chave privada é mantida pelo usuário em um meio de armazenamento portátil, enquanto que a chave pública é armazenada na forma de um certificado no serviço CIB da célula onde o usuário foi cadastrado. No caso de o usuário efetuar *login* em outra EXEHDAcel, o CIB fará a busca transparente do certificado do usuário. O resultado de uma operação *login* realizada com sucesso é a ativação da sessão padrão do usuário, recolocando em operação as aplicações anteriormente interrompidas, caso existam. Geralmente, a sessão padrão inclui a aplicação ISAM Desktop, a qual permite acesso às demais aplicações instaladas no ambiente virtual do usuário. Por sua vez, o comando l*ogout* libera os recursos empregados na gerência dos contextos de execução das aplicações que integram a sessão ativa do usuário. Esse comando implica operação dos serviços *ContextManager* e *AdaptEngine*, além de armazenar o estado da sessão no AVU para recuperação na próxima vez que o usuário entrar no sistema.

Save/Restore Session – permite ao usuário possuir várias sessões além da sessão padrão. O comando *save session* permite mover a sessão padrão (atual) para uma sessão alternativa, armazenando seu estado. Uma sessão alternativa armazenada no AVU pode ser restaurada com o comando *restore session*. Esses comandos estão relacionados ao serviço *SessionManager*.

Disconnect/Reconnect – correspondem a chamadas ao serviço *AdaptEngine*, atuando sobre o estado de conectividade do dispositivo do usuário, através do disparo do procedimentos associados à desconexão planejada.

No EXEHDA, toda a aplicação é caracterizada pelo seu descritor de disparo, que é um documento XML gerado durante o desenvolvimento da aplicação. Esse descritor agrupa metadados que habilitam a execução da aplicação a partir de EXEHDAnodos em qualquer EXEHDAcel que integra o ISAMpe. Entre os metadados estão uma descrição da funcionalidade da aplicação, o desenvolvedor, os parâmetros fixos utilizados pela aplicação, e uma referência, independente de localização (BDA), ao arquivo JAR que contém as classes da aplicação.

Considerando que o EXEHDAnodo no qual está sendo disparada a aplicação não pertence à EXEHDAcel onde essa está armazenada, será necessário um acesso intercelular,

que é realizado de forma transparente pelo serviço BDA do *middleware*. Esse comportamento operacional viabiliza, no que se refere a acesso a código, a utilização da semântica *siga-me*.

O disparo de aplicações pode ser feito de maneira manual, através da aplicação *isam-run*, a qual não é uma aplicação pervasiva, portanto, não possui as características definidas para esse tipo de aplicação. Assim, o *isam-run* deve ser usado para disparar uma aplicação pervasiva que habilite ao usuário manipular seu AVU.

Normalmente, a primeira aplicação efetivamente pervasiva a ser disparada é o utilitário ISAM Desktop, cuja interface gráfica é adaptada ao tipo de dispositivo em uso. O ISAM Desktop provê acesso às aplicações instaladas pelo usuário em seu ambiente virtual. A instalação dessas aplicações é feita adicionando-se o respectivo descritor de disparo no AVU, e inserindo-se uma entrada para a aplicação no arquivo *contents.xml*, o qual informa ao ISAM Desktop sobre as aplicação instaladas.

Perfil de Execução

O perfil de execução de cada EXEHDAnodo é especificado através de um documento XML. Esse documento associa nomes de serviços a componentes que implementam a interface definida para eles. Ainda é possível definir propriedades para determinados serviços, as quais serão recuperadas durante o processamento. No documento XML, um bloco <PROFILE> define um perfil de execução, que tem seu nome definido pelo atributo *name*. Dentro desse bloco, são utilizados blocos <SERVICE> para definir os serviços de farão parte do perfil de execução. Nesse caso, o atributo *name* define o nome canônico do serviço, enquanto o atributo *loadPolicy* determina o momento de ativação do serviço (*boot* ou *demand*). Por fim, dentro dos blocos <SERVICE> podem ser usados elementos <PROP>, os quais possuem os atributos *name* e *value* que servem para definir propriedades dos serviços, personalizando sua execução. No Quadro 13 é apresentado um exemplo de perfil de execução.

```
<PROFILE name="base">
    <PROP name="localhost.id" value="1"/>
    <PROP name="localhost.name" value="sussurro"/>
    <PROP name="localhost.cell" value="gmob"/>

    <SERVICE name="logger" loadPolicy="boot">
       <PROP name="impl" value="org.isam.exehda.services.logging.BasicLogger"/>
       <PROP name="logLevel" value="5000"/>
    </SERVICE>

    <SERVICE name="executor" loadPolicy="boot">
       <PROP name="impl" value="org.isam.exehda.services.primos.exec.ExecutorImpl"/>
    </SERVICE>

    <SERVICE name="scheduler" loadPolicy="boot">
       <PROP name="impl" value="org.isam.exehda.services.tips.TipsCellScheduler"/>
    </SERVICE>

    <SERVICE name="oxmanager" loadPolicy="boot">
       <PROP name="impl" value="org.isam.exehda.services.oxm.DualOXManagerCell"/>
    </SERVICE>

    <SERVICE name="gatekeeper" loadPolicy="boot">
       <PROP name="impl" value="org.isam.exehda.services.GatekeeperService"/>
    </SERVICE>

    <SERVICE name="ctxmanager" loadPolicy="demand">
       <PROP name="impl" value="org.isam.exehda.services.ctxm.CtxManagerImplNode"/>
    </SERVICE>
  </PROFILE>
```

Quadro 13 - Exemplo de perfil de execução do EXEHDA

Administração das células

O administrador de uma EXEHDAcel é responsável por: (*i*) manter os recursos de uso compartilhado da célula, definindo suas políticas de acesso; (*ii*) manter a EXEHDAbase; (*iii*) adicionar e remover usuários da EXEHDAcel.

Com o objetivo de facilitar as tarefas do administrador da célula, foi prototipada uma ferramenta para manutenção das células e de seus serviços. Esta ferramenta foi denominada EXEHDA-AMI (EXEHDA *Architecture Management Interface*). A EXEHDA-AMI é constituída por um módulo base, ao qual podem ser agregados, dinamicamente, outros componentes para ampliar suas funcionalidades.

Entre as funcionalidades oferecidas pela EXEHDA-AMI estão o suporte à navegação pelas células atualmente ativas no ISAMpe, e o serviço de adição e remoção de usuários. Os usuários podem ser comuns, apenas executam aplicações, ou desenvolvedores, os quais podem instalar e remover aplicações no serviço BDA da EXEHDAcel.

APÊNDICE C – Subsistemas do EXEHDA

Subsistema de Execução Distribuída

O Subsistema de Execução Distribuída é responsável pelo suporte ao processamento distribuído no EXEHDA. Esse subsistema interage com outros subsistemas, como o de reconhecimento do contexto e de adaptação, para promover uma execução efetivamente pervasiva. Os serviços que compõem esse subsistema são:

Executor – possui as funções de disparo de aplicações, e de criação e migração de seus objetos. Na implementação desse serviço é empregada a instalação de código sob demanda, e sua interface define métodos para controle do ciclo de vida das aplicações e dos objetos;

Cell Information Base (CIB) – implementa a base de informação da célula, e sua principal funcionalidade está relacionada à manutenção da infra-estrutura que forma o ISAMpe. Esse serviço mantém os dados estruturais da EXEHDAcel, como informações sobre recursos, vizinhança e atributos que descrevem as aplicações em execução. Esses atributos são registrados pelo serviço *Executor* quando ele realiza o disparo da aplicação, e incluem identificação do proprietário, referência para o código da aplicação, descritor de disparo da aplicação. O CIB ainda é responsável pela manutenção das informações dos usuários registrados na célula, como um certificado assinado para autenticação;

OXManager – sua atribuição é a gerência e a manutenção da meta-informação associada a um OX, conferindo às operações de consulta e atualização dos atributos de um OX o caráter pervasivo necessário para que se possa acessá-los a partir de qualquer nodo do ISAMpe. A abstração OX (Objeto eXehda) é uma instância de objeto criada pelo *Executor*, a qual pode ser associada meta-informação em tempo de execução. Essa abstração provida pelo EXEHDA é útil no mapeamento entre o modelo de computação do ISAMadapt e as funcionalidades oferecidas pelo *middleware*;

Discoverer – é o servido de descoberta de recursos do EXEHDA, sendo responsável pela localização de recursos no ISAMpe a partir de especificações abstratas dos mesmo, as quais caracterizam o recurso a ser descoberto por meio de atributos e seus respectivos valores.

Para isso, a interface desse serviço disponibiliza métodos para registro e remoção de recursos, e para pesquisa de recursos que atendam a um determinado critério. Os recursos registrados são catalogados no CIB, ficando visíveis no ISAMpe. Além disso, o registro do recurso deve ser periodicamente renovado, caso contrário, será automaticamente removido do conjunto de recursos registrados no CIB. Deve-se observar que a localização de um recurso não implica a alocação ou reserva do mesmo. O serviço responsável pela gerência dos recursos é o *ResourceBroker*, abordado a seguir;

ResourceBroker – é responsável pelo controle da alocação de recursos para as aplicações no EXEHDA, atendendo tanto requisições originárias da própria célula quanto oriundas de outras células do ISAMpe. No tratamento das requisições de localização e alocação de recursos, o *ResourceBroker* interage com os serviços *Discoverer* e *Scheduler* para, respectivamente, localização de recursos especializados e alocação de nodos de processamento;

Gateway – é responsável pela intermediação da comunicação entre os nodos externos à célula e os recursos internos a ela. A interação desse serviço com o *ResourceBroker* promove o controle de acesso aos recursos de uma EXEHDAcel. A interface do serviço *Gateway*, o qual mantém uma tabela que registra quais recursos são visíveis por cada aplicação, define três métodos direcionados a concessão, revogação e renovação de permissões de acesso;

StdStreams – provê suporte ao redirecionamento dos *streams* padrões de entrada, saída e erro. Esse serviço associa a cada aplicação um *Console* que agrupa os três *streams* padrões. Assim, podem-se redirecionar esses *streams* sem a necessidade de modificar a aplicação, bastando apenas configurar os atributos de execução do serviço;

Logger – provê a funcionalidade do registro de rastro de execução, e pode ser usado na depuração de programas (fase de desenvolvimento) ou no registro de operações críticas, facilitando o controle da segurança do sistema. A interface desse serviço disponibiliza métodos para registro de mensagens, as quais podem ser classificadas segundo níveis de prioridade;

Dynamic Configurator (DC) – tem a função de realizar a configuração do perfil de execução do *middleware* em um EXEHDAnodo de forma automatizada. Para isso, o nodo deve ser inicializado com um perfil de ativação base que contempla referência ao serviço DC, informado ao *ServiceManager* que deve ser gerado um profile para o nodo. Dessa forma, o serviço DC executa um processo de detecção básico para identificar as características do nodo, as quais incluem informações sobre a JVM e sobre o sistema operacional,

configurações de rede, e capacidade de memória. Além disso, o DC pode ser configurado para interagir com o serviço *Collector*, responsável pela aquisição de dados de sensores, para criar um perfil mais específico. Após isso, as informações detectadas pela instância local do DC são enviadas para a instância celular do mesmo, onde são utilizadas para a geração dinâmica do perfil adequado àquele nodo. O perfil gerado é retornado ao *ServiceManager*, que dispara uma nova inicialização do *middleware* considerando a configuração gerada automaticamente.

Subsistema de Reconhecimento de Contexto e Adaptação

O Subsistema de Reconhecimento de Contexto e Adaptação inclui serviços de aquisição de informações sobre o ISAMpe, de identificação em alto nível dos elementos de contexto e de disparo das ações de adaptação. Integram esse subsistema os serviços:

Collector – é responsável pela aquisição da informação bruta que, posteriormente, formará os elementos de contexto. Para isso, o *Collector* aglutina informações oriundas de vários monitores (*Monitor*) e as repassa aos consumidores registrados (*MonitoringConsumer*). Esse repasse pode ser feito via *callback* ou via canais de multicast providos pelo serviço *Deflector*. Um *Monitor* gerencia um conjunto de sensores parametrizáveis. Cada sensor contribui com a aquisição de um valor que descreve um aspecto específico, que pode ser estático ou dinâmico, do recurso monitorado. O conjunto de sensores de um EXEHDAnodo, bem como os parâmetros suportados por eles, integra a informação de descrição daquele nodo, disponibilizada no serviço CIB;

Deflector – tem o objetivo de disponibilizar a abstração de canais de multicast para uso na disseminação das informações monitoradas. A interface desse serviço disponibiliza métodos para criação de um canal multicast, inscrição e desinscrição dos consumidores no canal, e para disparar a disseminação de determinada informação no canal;

ContextManager – realiza o tratamento da informação bruta produzida pela monitoração, produzindo as informações abstratas referentes aos elementos de contexto. A definição dos elementos de contexto (objeto *Context*) é parametrizada por uma descrição XML que descreve como o dado referente àquele elemento de contexto deve ser produzido a partir da informação proveniente da monitoração. O método que cria o objeto *Context* também define todos os estados possíveis para o elemento de contexto criado, também com base nas informações da descrição XML. Após a definição de um elemento de contexto, os

interessados no recebimento de informações sobre esse contexto devem registrar-se junto ao *ContextManager*. O registro pode ser cancelado quando não houver mais interesse num contexto específico;

AdaptEngine – é responsável pelo controle das adaptações funcionais, provendo facilidades para definição e gerência de comportamentos adaptativos por parte das aplicações. Dessa forma, libera o programador de gerenciar aspectos de mais baixo nível de gerenciamento de contexto. Esse serviço faz a parametrização do *ContextManager*, a partir de definições em XML, para criação dos elementos de contexto de interesse de cada aplicação, e registra seu interesse nas notificações de alteração de estado desses elementos. Em face de uma notificação, o *AdaptEngine* é responsável pela gerência e notificação dos componentes registrados como adaptativos/sensível àquele elemento de contexto cujo estado foi alterado. Outra função desse serviço é prover o mecanismo de carga de código contextualizado, ou seja, selecionar e carregar o código da aplicação que melhor se adapta ao estado do contexto corrente;

Scheduler – é o serviço responsável pelas adaptações não-funcionais, isto é, que não implicam alteração de código. Para isso, o *Scheduler* emprega informações de monitoração do serviço *Collector* para orientar operações de instanciação remota, migração, ou re-escalonamento de código, de acordo com estado dos recursos de processamento.

Subsistema de Comunicação

O Subsistema de Comunicação disponibiliza mecanismos para atender, principalmente, aspectos relacionados às desconexões, muito comuns em ambiente pervasivos devido tanto à existência de enlaces sem fio como às estratégias de economia de energia dos dispositivos móveis. Fazem parte desse subsistema os serviços:

Dispatcher – disponibiliza um modelo de comunicação por troca de mensagens ponto-a-ponto com garantia de entrega e ordenamento, o qual é especializado para operação no ambiente pervasivo. Quando inicializado, o *Dispatcher* atualiza as informações do EXEHDAnodo na CIB, provendo um conjunto de protocolos e endereços que podem ser usados para alcançar o nodo. Durante os períodos de desconexões planejadas, objetivando manter a consistência da comunicação, esse serviço emprega um mecanismo de *checkpointing*

do estado dos canais para fazer o tratamento transparente dessas desconexões, procedendo à entrega das mensagens assim que ocorre a reconexão;

WORB – tem o objetivo de simplificar a construção de serviços distribuídos, permitindo ao programador abstrair aspectos de baixo nível relativos ao tratamento das comunicações em rede. Para tanto, o *WORB* oferece um modelo de comunicação baseado em invocações remotas de métodos. Esse serviço é construído sobre o serviço *Dispatcher*, tornando a invocação remota de métodos também sintonizada à premissa de desconexão do ambiente pervasivo;

CCManager – disponibiliza um mecanismo de comunicação baseado na abstração espaço de tuplas, o qual não precisa da coexistência temporal de emissor e receptor. Esse mecanismo atende a demanda de desacoplamento espacial e temporal, causada pela premissa da mobilidade lógica dos componentes que constituem as aplicações pervasivas.

Subsistema de Acesso Pervasivo

O Subsistema de Acesso Pervasivo tem por finalidade dar suporte à premissa da Computação Pervasiva de acesso em qualquer lugar e todo o tempo a dados e código. Compõem esse subsistema os serviços:

BDA – o serviço Base de Dados pervasiva das Aplicações provê a capacidade de instalação de código sob demanda, que é uma necessidade inerente à execução de aplicações pervasivas. Para isso, esse serviço mantém um repositório de código que fornece a mesma visão de software disponibilizado a partir de qualquer dispositivo do ISAMpe, mesmo após migrações. Além disso, o BDA também suporta o controle de versões, que é importante na manutenção da operacionalidade das aplicações. Assim, as requisições geradas aos EXEHDAnodos de uma determinada célula são sempre direcionadas à instância celular do serviço BDA da mesma EXEHDAcel. Essa, se necessário, realiza o acesso a instâncias remotas (BDAs de outras células);

AVU – é atribuição do serviço Ambiente Virtual do Usuário a manutenção do acesso pervasivo ao ambiente computacional do usuário, o qual compreende aplicações em execução, informações de personalização das aplicações, conjunto de aplicações instaladas e seus arquivos privados. Com esse objetivo, o AVU adota uma estratégia de utilização análoga a do BDA, ou seja, as requisições geradas pela instância local do EXEHDAnodo são direcionadas

à instância celular do serviço. A instância celular, por sua vez, consulta a célula *home* do usuário de forma a descobrir a última localização física de seu ambiente virtual, e acessá-lo para conclusão da requisição. O AVU ainda inclui operações *prefetch* e *release* para otimização do acesso aos dados armazenados face aos aspectos de mobilidade e desconexão planejada;

SessionManager – é responsável pela gerência da sessão de trabalho do usuário, que é o conjunto de aplicações correntemente em execução para aquele usuário. A informação que descreve o estado da sessão de trabalho é armazenada no AVU, estando disponível de forma pervasiva. Esse serviço trabalha com objetos *SessionDescriptor*, o qual representa uma sessão e define métodos para inclusão e remoção de aplicações. Além disso, o *SessionManager* disponibiliza métodos para salvamento e recuperação de sessões. Normalmente, esse serviço é ativado através do serviço *GateKeeper*, após o procedimento de autenticação do usuário;

GateKeeper – é responsável por intermediar o acesso entre entidades externas à plataforma ISAM e os serviços do *middleware*, realizando os procedimentos de autenticação necessários. O protocolo de autenticação baseia-se em um mecanismo de chave pública/privada, sendo a chave pública disponibilizada na CIB e a privada armazenada em um meio portátil do usuário. Em caso de sucesso na autenticação, um identificador de sessão é retornado, pelo qual o usuário poderá disparar as aplicações. O método *logout* permite invalidar um identificador de sessão.

APÊNDICE D – Conjunto mínimo de tarefas

O conjunto mínimo de tarefas, definido para o projeto ClinicSpaces, é baseado no estudo de Hallvard Laerum e Arild Faxvaag (LAERUM; FAXVAAG, 2004). Basicamente, esse estudo buscou levantar informações sobre as atividades médicas em um hospital. Assim, foram obtidas vinte e quatro tarefas clínicas. Após o levantamento das atividades (tarefas), foi realizada uma validação na qual os profissionais entrevistados respondiam, entre outras questões, se realmente realizavam essas tarefas em suas rotinas diárias.

Dessa forma, para iniciar o projeto ClinicSpaces, foram selecionadas onze das vinte e quarto tarefas. A seleção foi feita com base nas estatísticas apresentadas no referido estudo, sendo que as onze tarefas escolhidas obtiveram cem por cento de respostas positivas quando se perguntava, aos profissionais, se costumavam realizar a tarefa na sua rotina diária. A seguir são apresentadas as onze tarefas selecionadas.

Tarefa	**Definição**	**Classificação**
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
Procurar informações específicas em registros do paciente (prontuário)	Procurar específicas e limitadas quantidades de informações sobre o paciente nos registros do paciente.	Diagnóstico
Obter os resultados de novos testes ou investigações (avaliações médicas)	Identificar e obter resultados de investigações já executadas e analisadas, e que ainda não tenham sido acessadas por um clínico.	Diagnóstico
Adicionar notas diárias sobre as condições do paciente	Escritas sobre avaliações das condições do paciente, e assim formulando notas de progresso do estado do paciente.	Tratamento Diagnóstico

Tarefa	Definição	Classificação
Requisitar análises clínicas em laboratórios bioquímicos	Requisitar uma ou diversas análises clínicas em laboratórios bioquímicos. A reserva do teste pode ser feita pelo médico ou outro profissional.	Diagnóstico
Obter resultados de exames clínicos (análises laboratoriais)	Obter resultados novos, e velhos, de exames clínicos executados em laboratórios bioquímicos.	Laboratorial
Requisitar raios-X, ultra-som e investigações CT	Requisitar raios-X, ultra-sons ou investigações CT, e ainda prover um sumário clínico do paciente. A reserva da investigação pode ser feita pelo médico ou outro profissional.	Diagnóstico
Obter resultados de raios-X, ultra-som e investigações CT	Obter resultados novos, e velhos, de raios-X, ultra-sons ou investigações tomográficas computadorizadas.	Laboratorial
Requisitar tratamentos	Requisitar procedimentos de tratamentos como medicamentos, cirurgias, etc. a serem executadas no hospital, e que geralmente não sejam administradas pelos pacientes.	Tratamento
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
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

APÊNDICE E – Exemplos de subtarefas

A seguir são listadas possíveis subtarefas que irão compor as tarefas descritas no Apêndice D. As subtarefas “ID” são implementadas como solicitações ao módulo Contexto de Tarefas para identificação do profissional e/ou do paciente. As subtarefas “EHS” 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).

Revisar os Problemas do Paciente:

- ID profissional;
- ID Paciente;
- EHS - Busca Informação do Paciente;
- EHS - Visualização de Informações.

Procurar Informações Específicas no Registro do Paciente:

- ID Profissional;
- ID Paciente;
- EHS - Busca Informação do Paciente;
- EHS - Visualização de Informações.

Obter os Resultados de Novos Testes e Investigações:

- ID Profissional;
- ID Paciente;
- EHS - Busca Informação do Paciente (Parâmetro: Informações novas);
- EHS - Visualização de Informações;

Adicionar Notas Diárias sobre Condições do Paciente:

- ID Profissional;
- ID Paciente;
- EHS - Registra nota diária.

Requisitar Análises Laboratoriais:

- ID Profissional;
- ID Paciente;

- EHS - Preencher requisição laboratorial; (preenchimento + encaminhamento + registro no pEHS).

Requisitar Exames de Vídeo/Imagem (raio-x, tomografia, etc):

- ID Profissional;
- ID Paciente;
- EHS - Preencher requisição de exames de vídeo/imagem; (preenchimento + encaminhamento + registro no pEHS).

Obter Resultados Laboratoriais:

- ID Profissional;
- ID Paciente;
- EHS - Busca Informação do Paciente (Parâmetro: Requisições Laboratoriais);
- EHS - Visualização de Informações laboratoriais.

Obter Resultados de Vídeo/Imagem (raio-x, tomografia, etc)

- ID Profissional;
- ID Paciente;
- EHS - Busca Informação do Paciente (Parâmetro: Requisições de vídeo/imagem);
- EHS - Visualização de Informações de exames de vídeo/imagem.

Requisitar Tratamentos:

- ID Profissional;
- ID Paciente;
- EHS - Preencher de requisição de tratamento (preenchimento + encaminhamento + registro no pEHS).

Escrever Prescrições:

- ID Profissional;
- ID Paciente;
- EHS - Preencher prescrição (preenchimento + registro no pEHS + impressão).

Registrar Códigos para Diagnósticos e Procedimentos Executados:

- ID Profissional;
- ID Paciente;
- EHS - Registro de diagnósticos e procedimentos executados.

APÊNDICE F – Diagrama de Classes

O diagrama de classes do Subsistema de Gerenciamento Distribuído de Tarefas (SGDT) foi dividido em três partes para simplificar a visualização. A Figura 15 mostra as classes criadas para representar exceções e as classes *Handler*.

A classe *ValidationErrorHandler* é usada pelo Parser XML para manipular erros encontrados durante a validação dos arquivos XML (descritor das tarefas e subtarefas). Já a classe *Handler* é usada para manipular requisições ao Serviço de Tarefas Ativas. Essa classe transforma as requisições ao serviço em conexões HTTP.

As classes que representam exceções foram criadas para facilitar o tratamento de exceções e a depuração do protótipo do SGDT. Na Figura 15 elas estão classificadas em dois grupos: exceções que são levantadas para indicar uma falha específica e exceções que encapsulam outras exceções, simplificando o tratamento delas pelos métodos superiores.

No primeiro grupo estão: *ParameterNotFoundException* e *ContextNotFoundException*, as quais indicam que um parâmetro/contexto necessário a uma subtarefa não foi encontrado na tarefa correspondente; *ParameterNotSupportedException* e *ContextNotSupportedException*, as quais indicam que tentou-se configurar um parâmetro ou contexto que não é suportado pela subtarefa; e *TaskTypeNotSupportedException*, o qual indica que tentou-se instanciar um tipo de tarefa não suportado pelo SGDT. Os tipos atualmente suportados são Tarefa Simples e Tarefa Composta.

No segundo grupo estão: *ParserException*, que encapsula exceções ligadas ao parser XML, dentre elas, exceções e entrada/saída (causadas por problemas na leitura de arquivos ou conexão HTTP) e exceções de processamento do XML; *SubTaskInstantiationException*, que encapsula exceções do Java *Reflection*, usado para instanciar as subtarefas de acordo com parâmetros de seu descritor; *SubTaskLoadException* e *TaskLoadException* que encapsulam todas as exceções levantadas durante a busca, instanciação e configuração das subtarefas e tarefas, sendo que a segunda encapsula, ainda, a primeira.

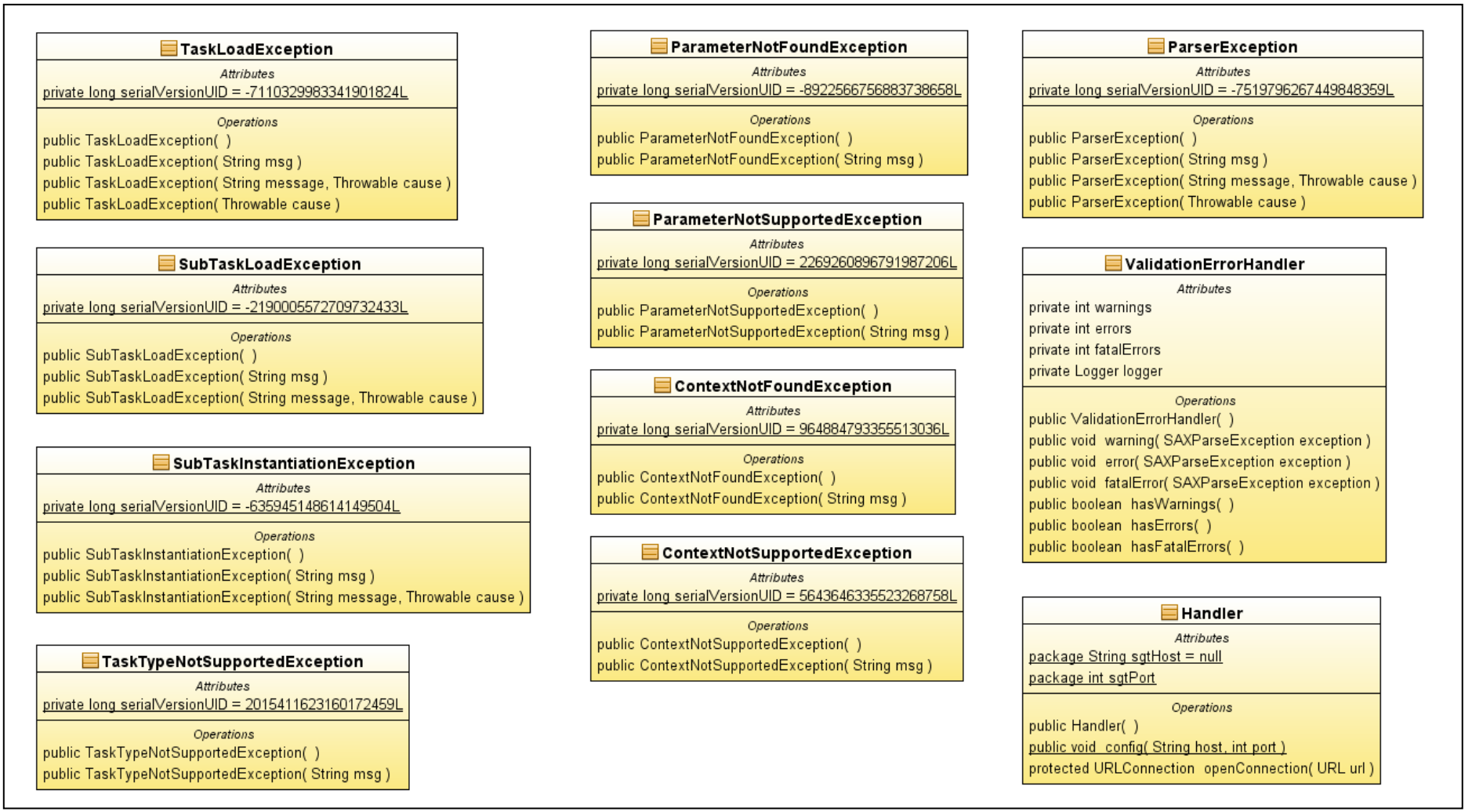

Figura 15 - Diagrama de Classes - parte 1

Todas as exceções de ambos os grupos são levantadas por falha no carregamento da tarefa (incluindo o carregamento de suas subtarefas). Por isso, todas são levantadas, tratadas e encapsuladas no Serviço de Acesso a Tarefas (SAT). Assim, as exceções são empilhadas hierarquicamente, pelos métodos do SAT, até atingir a de maior hierarquia (*TaskLoadException*), a qual é a única a ser repassada ao Serviço de Gerenciamento de Tarefas (SGT). Dessa forma, esta é a única exceção que o SGT deve tratar.

Essa abordagem hierárquica facilita o tratamento de exceções no código dos serviços, pois cada método trata suas exceções específicas, repassando uma exceção mais genérica para seu "superior" e delegando também a responsabilidade de tomar a decisão adequada. Com isso, diminui-se o código necessário para tratar as exceções e o processamento não é interrompido até que a exceção chegue ao método que iniciou a execução, o qual tem melhores condições de tratar a exceção com menor impacto ao usuário.

Outra vantagem dessa abordagem é com relação à depuração de erros. Como as exceções são encapsuladas hierarquicamente, o programador pode acessar o log do sistema e pesquisar na pilha de exceções, seguindo-a do nível mais alto até chegar ao motivo que gerou a exceção. Além disso, essa pilha reflete a mesma pilha de execução dos métodos, o que facilita encontrar o erro.

A Figura 16 mostra o diagrama de classes da modelagem de tarefas e subtarefas. A interface Tarefa define métodos para permitir que o SGT controle seu ciclo de vida, e métodos para que a tarefa possa gerenciar suas subtarefas. Assim, a ligação do SGDT com esse modelo é através da interface Tarefa, o que permite certo grau de independência entre as implementações.

Essa interface é implementada pelas classes TarefaSimples e TarefaComposta, sendo que a segunda não foi implementada no protótipo. A classe TarefaSimples possui um objeto da classe *TaskState*, o qual representa o estado da tarefa, e através do qual o SGT gerencia seu ciclo de vida. Além disso, a classe TarefaSimples possui uma lista de objetos que implementam a interface SubTarefa. Portanto, essa lista pode conter qualquer instância de qualquer classe que implemente essa interface, o que permite a criação de subtarefas sem a necessidade de modificar o SGDT.

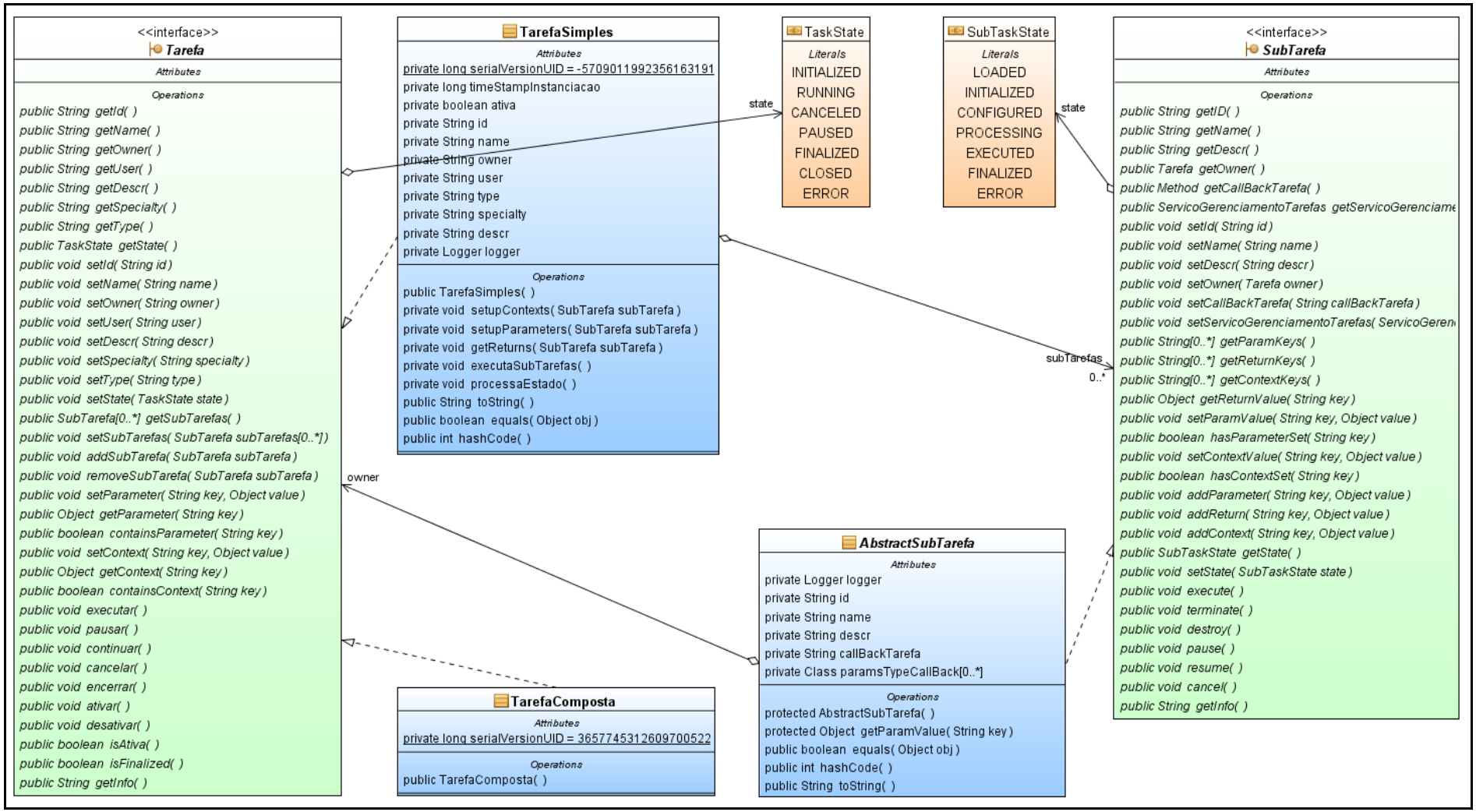

Figura 16 - Diagrama de Classes - parte 2

A classe abstrata AbstractSubTarefa foi criada para facilitar a implementação de subtarefas. Ela implementa os métodos da interface SubTarefa necessários ao gerenciamento, deixando a implementação dos métodos de processamento para as subtarefas. Além disso, essa classe abstrata possui um objeto da classe *SubTaskState*, o qual indica o estado da subtarefa, e um objeto de qualquer classe que implemente a interface Tarefa, o qual representa a tarefa a qual pertence a subtarefa.

A Figura 17 apresenta o diagrama de classe dos serviços do SGDT implementados no protótipo. Seguindo a arquitetura do EXEHDA, os serviços do SGDT possuem uma interface, que define as funcionalidades do serviço, e uma implementação, que pode variar de acordo com o tipo de dispositivo. As implementações do Serviço de Gerenciamento de Tarefas (SGT), do Serviço de Acesso a Tarefas (SAT) e do Serviço de Contexto de Tarefas (SCT) estendem o *AbstractService* do EXEHDA, o qual provê funcionalidades básicas para os serviços.

O SGT, que é o núcleo do SGDT, utiliza o SAT para buscar e carregar as tarefas, e o SCT para obter informações de contexto para as tarefas. Este último foi implementado somente como forma de prever sua integração com os outros serviços, pois suas funcionalidades estão sendo estudadas em outro trabalho. Já a implementação do SAT utiliza a classe ParserXml para processar o descritor das tarefas e das subtarefas.

O Serviço de Tarefas Ativas está isolado no diagrama de classes porque ele não é usado diretamente pelos outros serviços. O acesso a esse serviço é feito por meio de requisições HTTP, e por isso, a interface ServicoTarefasAtivas é implementada por duas classes: ServicoTarefasAtivasClient, a qual recebe as requisições locais do SGT e converte em requisições remotas ao servidor; e ServicoTarefasAtivasServer, responsável por manter as informações sobre as tarefas ativas dos usuários e por atender às requisições do cliente. Para tanto, a classe servidora estende a classe *HTTPService* do EXEHDA, a qual implementa algumas funcionalidades para esse tipo de comunicação.

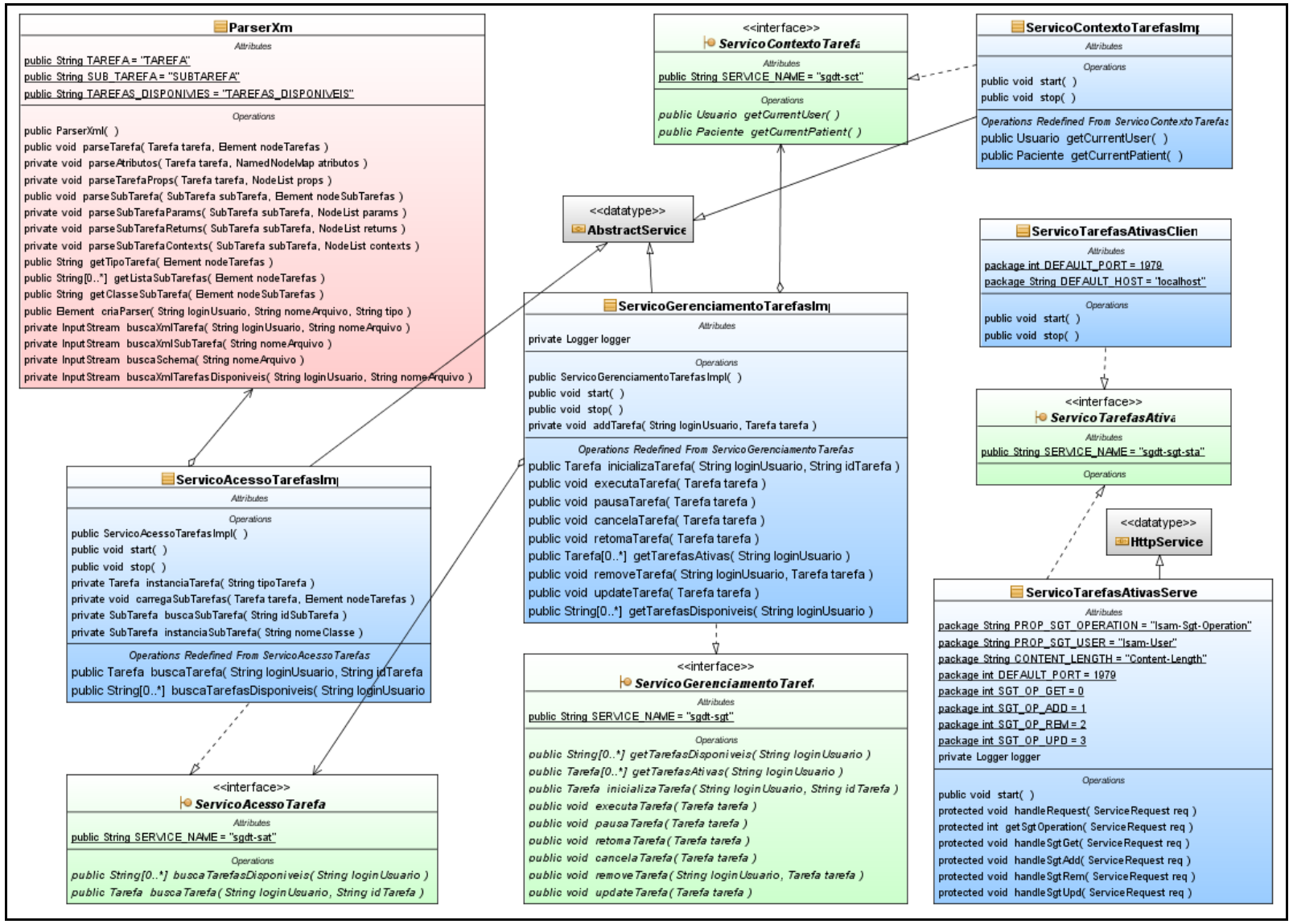

Figura 17 - Diagrama de Classes - parte 3