

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

**UM AMBIENTE DE CONTEXTO
PERSONALIZADO E ORIENTADO A TAREFAS NA
ARQUITETURA CLINICSPACE**

DISSERTAÇÃO DE MESTRADO

Tiago Antônio Rizzetti

**Santa Maria, RS, Brasil
2009**

UM AMBIENTE DE CONTEXTO PERSONALIZADO E ORIENTADO A TAREFAS NA ARQUITETURA CLINICSPACE

por

Tiago Antônio Rizzetti

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

**UM AMBIENTE DE CONTEXTO PERSONALIZADO E ORIENTADO A
TAREFAS NA ARQUITETURA CLINICSPACE**

elaborada por
Tiago Antônio Rizzetti

como requisito parcial para obtenção do grau de
Mestre em Computação

COMISSÃO EXAMINADORA:

Iara Augustin, Dra.
(Presidente/Orientadora)

Mario Antonio Ribeiro Dantas, Dr. (UFSC)

Benhur de Oliveira Stein, Dr. (UFSM)

Santa Maria, 21 de Agosto de 2009.

RESUMO

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

UM AMBIENTE DE CONTEXTO PERSONALIZADO E ORIENTADO A TAREFAS NA ARQUITETURA CLINICSPACE

Autor: Tiago Antônio Rizzetti

Orientadora: Iara Augustin

Data e Local da Defesa: Santa Maria, 21 de Agosto de 2009.

O projeto ClinicSpace tem por objetivo preencher as lacunas existentes nos sistemas clínicos atuais, no que tange às características de pervasividade e apoio de tarefas computacionais às atividades clínicas que o usuário (médico) realiza. A arquitetura do modelo ClinicSpace, construída a partir da perspectiva dada pela teoria da atividade, é composta por vários módulos que interligados oferecem as características necessárias a um sistema clínico pervasivo orientado ao usuário-final. Um desses módulos é o tratamento de contexto das tarefas clínicas. Esse trabalho realiza uma discussão sobre os requisitos presentes para o tratamento de contexto das tarefas clínicas, definindo uma arquitetura para associá-los às tarefas do usuário, permitindo personalização de contexto e entrada automática de dados. A personalização é obtida através da utilização de Elementos Programáveis de Contexto, que são representados por atuadores, físicos ou lógicos, responsáveis por dotar o sistema de capacidade de execuções automáticas, baseadas em parâmetros de contexto especificados pelo usuário. Já a entrada automática de dados trata da obtenção destes de maneira implícita, obtendo as informações utilizadas pelas aplicações que o usuário executa, no decorrer de suas tarefas. Para isso, definiu-se uma arquitetura com suporte à personalização e especificação semântica dos dados nela utilizados. Para construir tais funcionalidades, estendeu-se o *middleware* pervasivo EXEHDA, modificando serviços existentes e agregando novos serviços. A principal contribuição do trabalho está na interligação existente entre os componentes que integram a arquitetura, construindo uma visão única do contexto de uma tarefa sob a perspectiva dos dados necessários a ela e da capacidade de personalização pelo usuário. Dessa forma, reduz-se a necessidade da entrada explícita de dados, e contribui-se para a redução da rejeição da adoção dos sistemas clínicos em ambientes altamente dinâmicos, como os hospitalares.

Palavras-chave: computação pervasiva; computação ubíqua; computação sensível ao contexto; middleware; computação orientada a tarefas; tarefas clínicas; personalização de contexto; entrada implícita de dados.

ABSTRACT

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

UM AMBIENTE DE CONTEXTO PERSONALIZADO E ORIENTADO A TAREFAS NA ARQUITETURA CLINICSPACE

Author: Tiago Antônio Rizzetti
Advisor: Iara Augustin
Santa Maria, August 21st, 2009

The project ClinicSpace aims to fill gaps in current clinical systems, regarding to the characteristics of pervasive computing tasks and clinical activities support to the user (physician). The architecture of the model ClinicSpace, built from the perspective given by the activity theory, it is composed of several modules that interconnected offer the features needed in a system geared to clinical pervasive end-user. One of these modules is the treatment of the clinical tasks. This work holds a discussion on the present requirements for the treatment of the clinical tasks, defining an architecture to link them to the tasks of the user, allowing context customization and automatic entry of data. The customization is achieved through the use of Programmable Elements of Context, which are represented by actuators, physical or logical, responsible for providing the system capacity of automatic executions, based on the parameters specified by the user. Yet the automatic data comes from the implicit way of obtaining these, the information used by applications that the user performs in the course of their duties. For this, an architecture was set up to support the customization and the semantic specification of data used. Building such features extended the pervasive middleware EXEHDA, modifying the already existing services and adding new ones. The main contribution of this work is the interconnection between the components that make up the architecture, building a unique view of the context of a task from the perspective of the necessary data for it and the ability to be customized by the user. Thus, it reduces the need for explicit data entry, and it contributes to the reducing rejections of its adoption of clinical systems in highly dynamic environments such as hospitals.

Keywords: pervasive computing, ubiquitous computing, context-sensitive computing, middleware, task-based computing; clinical tasks; context customization, implicit data entry.

LISTA DE FIGURAS

Figura 1 - Componentes da Teoria da Atividade. Fonte: (Kaenampornpan, 2005).....	32
Figura 2 - Sistema Operacional Aura. Fonte: (SOUSA, 2002).....	39
Figura 3 - Modelo de Contexto CIS (Context Information Service) do Aura. Fonte: (JUDD, 2003).....	40
Figura 4 - Arquitetura Gaia. Fonte: (ROMÁN, 2002).....	41
Figura 5 - Arquitetura de Contexto do Gaia. Fonte: (RANGANATHAN, 2003).	42
Figura 6 - Arquitetura projeto ABC. Fonte: (BARDRAM, 2007).....	44
Figura 7 - Diagrama de Estados de uma Tarefa.....	51
Figura 8 - Arquitetura ClinicSpace.....	54
Figura 9 - Relacionamento entre Tarefas, Subtarefas, Interface de Edição de Tarefas (IET), SGT, SGCT e pEHS.....	55
Figura 10 - Organização celular da arquitetura ISAM. Fonte: (AUGUSTIN, 2004).....	59
Figura 11 - Relação entre Aplicação, Templates e Arquétipos.....	65
Figura 12 - Interface de Personalização do Contexto da Tarefa - Uso dos Atuadores.....	69
Figura 13 - Dados do Atuador.....	70
Figura 14 - Interface de Personalização do Contexto da Tarefa – Atuador Agendamento Associado à tarefa.....	70
Figura 15 - Arquitetura base do sistema de gerenciamento de contexto de tarefas - outros componentes da arquitetura ClinicSpace, sem relação direta com o serviço foram omitidos...	71
Figura 16 - Hierarquia de Abstração de Sensores no Coletor.....	75
Figura 17 - Diagrama de classe para suporte a múltiplas instâncias no EXEHDA - vários métodos, atributos e classes foram omitidos.....	77
Figura 18 - Diagrama de Sequência para obtenção automática de dados pelo sistema de Contexto.....	80
Figura 19 - Digrama de Classes (reduzido) ilustrando a implementação dos elementos personalizáveis de contexto – os atuadores.....	81
Figura 20 - Gráfico mostrando a relação entre o número de sensores instanciados pelo tempo necessário para sua criação.....	90
Figura 21 - Gráfico mostrando a relação do número de atuadores criados pelo tempo.....	92

Figura 22 - Simulação de um Formulário com informações sobre Status do Paciente.....93

LISTA DE QUADROS

Quadro 1 - Exemplo da especificação de um dado de entrada para o SGCT.....	66
Quadro 2 - Exemplo do código de criação de um Sensor Abstrato.....	78
Quadro 3 - Interface sobre a qual os sensores devem serem implementados.....	79
Quadro 4 - Interface modelada para um Atuador.....	83
Quadro 5 - Código XML para criação do contexto associado ao Atuador de Agendamento..	85
Quadro 6 - Exemplo de implementação de um Sensor para obtenção de data e hora.....	89
Quadro 7 - XML Schema de uma Classe de Dados.....	117

LISTA DE APÊNDICES

APÊNDICE A - Conjunto Mínimo de Tarefas.....	105
APÊNDICE B – Middleware EXEHDA.....	110
APÊNDICE C – Definição XML Schema para uma classe de dado.....	116

ÍNDICE DE TABELAS

Tabela 1 - Comparativo entre Projetos Relacionados.....	45
Tabela 2 - Comparativo entre Projetos Relacionados e o ClinicSpace.....	88
Tabela 3 - Relação entre os tempos de Entrada Implícita e Explícita para o mesmo conjunto de dados.....	94
Tabela 4 - Conjunto Mínimo de Tarefas.....	105

LISTA DE ABREVIATURAS E SIGLAS

EHS	<i>Electronic Health-care System</i>
pEHS	<i>pervasive EHS</i>
EXEHDA	<i>Execution Environment for Highly Distributed Applications</i>
ISAM	Infra-estrutura de Suporte às aplicações Móveis Distribuídas
ISAMpe	ISAM <i>Pervasive Environment</i> - Ambiente Pervasivo do ISAM
OX	Objeto eXehda
SGCT	Serviço Gerenciamento de Contexto de Tarefas
SGDT	Subsistema de Gerenciamento Distribuído de Tarefas
SGT	Serviço de Gerenciamento de Tarefas

SUMÁRIO

1. INTRODUÇÃO.....	15
1.1. Definição do Problema.....	17
1.2. Objetivos	18
1.3. Estrutura do Texto.....	19
2. IDENTIFICANDO O CONTEXTO DAS TAREFAS CLÍNICAS.....	20
2.1. Computação Sensível ao Contexto.....	20
2.2. Sistemas Clínicos de Registro Eletrônico.....	22
2.2.1. Requisitos de um Sistema Ideal de Cuidados Clínicos.....	23
2.2.1.1. Requisitos sob o ponto de vista dos registros.....	24
2.2.1.2. Requisitos sobre o ponto de vista clínico.....	25
2.2.1.3. Interoperabilidade entre os Sistemas.....	25
2.2.2. Terminologias Clínicas.....	26
2.3. Características das Atividades Clínicas.....	28
2.3.1. Descoberta Automática de Atividades	29
2.3.2. Requisitos para Modelar Atividades Clínicas.....	30
2.3.2.1. Dinamismo e Mobilidade das Tarefas.....	30
2.3.2.2. Parada e Reativação de Tarefas.....	30
2.3.2.3. Compartilhamento de Tarefas.....	31
2.4. Computação Baseada na Atividade Humana.....	31
2.4.1. Teoria da Atividade	31
2.4.2. Teoria da Atividade Aplicada ao Contexto Clínico.....	32
2.4.3. Atividades versus Tarefas Clínicas.....	33
2.5. Contexto em Atividades Clínicas: Análise de Trabalhos Relacionados.....	34
2.5.1. Identificação de Contexto na Área Clínica.....	34
2.5.2. Sistemas Sensíveis ao Contexto considerando a Teoria da Atividade Humana.....	36
2.5.3. Sistemas Pervasivos Pró-Ativos com Reconhecimento de Contexto.....	37
2.5.3.1. Projeto Aura.....	37
2.5.3.2. Projeto Gaia.....	40
2.5.3.3. Projeto ABC (Activity-based Computing).....	43
2.5.4. Comparativo entre os Projetos Relacionados	45

3. PROJETO CLINICSPACE.....	46
3.1. Apresentação do Projeto ClinicSpace.....	46
3.1.1. Requisitos da Arquitetura ClinicSpace.....	47
3.1.2. Modelo de Tarefa ClinicSpace.....	49
3.1.3. Definições.....	50
3.1.1.1. Políticas de criação e gerenciamento de tarefas.....	52
3.1.1.2. Conjunto mínimo de tarefas.....	52
3.2. Contexto Associado às Tarefas.....	53
3.3. Arquitetura para a programação e gerenciamento das tarefas.....	54
3.3.1. Componentes da Arquitetura ClinicSpace.....	55
3.3.2. Interação do usuário com o ambiente ClinicSpace.....	56
3.4. Middleware EXEHDA.....	58
3.5. Resultados esperados pelo projeto ClinicSpace.....	59
4. GERENCIADOR DE CONTEXTO DE TAREFAS - SGCT.....	61
4.1. Tratamento de Contexto no ClinicSpace.....	61
4.1.1. Contexto das Tarefas.....	62
4.1.2. Funcionalidades	62
4.2. Modelo Proposto	63
4.2.1. Entrada automática de Dados para a aplicação.....	63
4.2.2. Personalização do Contexto de Tarefa.....	67
4.3. Implementação do SGCT.....	72
4.3.1. Serviço de Monitoramento – Coletor.....	72
4.3.2. Serviço de Gerenciamento de Contexto.....	73
4.3.3. Modelando e Construindo o Suporte a Sensores Personalizados no EXEHDA.....	74
4.3.3.1. Implementação da Nova Funcionalidade no EXEHDA.....	75
4.3.3.2. Integrando os componentes do sistema para entrada automática de dados.....	79
4.3.4. Modelagem e Implementação dos Elementos Personalizáveis de Contexto (Atuadores).....	79
5. DISCUSSÃO DOS RESULTADOS.....	86
5.1. Trabalhos Relacionados.....	86
5.2. Avaliação do Protótipo.....	88
5.2.1. Personalização de Sensores.....	88
5.2.2. Atuadores.....	91
5.2.3. Entrada de Dados Explícita versus Entrada de Dados Implícita.....	92

6. CONSIDERAÇÕES FINAIS.....	95
6.1. Trabalhos futuros.....	96
6.2. Publicações.....	97

1. INTRODUÇÃO

Sistemas computacionais vêm sendo empregados como ferramentas de apoio às mais diversas atividades humanas há algum tempo. Porém, a computação tradicional, até então, sempre se delineou pela perspectiva de adaptação do usuário às aplicações. Mark Weiser (1991), propôs a nova visão dos sistemas computacionais onde o usuário passa a ser o centro da computação, os sistemas computacionais, se tornam onipresentes e, pela perspectiva do usuário, invisíveis. O usuário, segundo a visão de Weiser, estará tão habituado aos dispositivos computacionais inseridos no ambiente, que eles passariam despercebidos, como se eles fossem parte do próprio ambiente. Essa visão ficou conhecida como Computação Ubíqua ou Pervasiva.

Há diversos desafios para tornar a Computação Ubíqua uma realidade, tanto do ponto de vista dos sistemas computacionais, que devem ser pró-ativos e onipresentes, quanto da inserção de questões subjetivas da vida cotidiana do usuário em um ambiente computacional, que, atualmente, tem na objetividade sua premissa. Uma das questões a tratar para oferecer onipresença da computação é a capacidade de migração de aplicações para sistemas diversos, utilizando uma abordagem “siga-me”: a aplicação segue o usuário em seu deslocamento (AUGUSTIN, 2004). Por outro lado, para ofertar a subjetividade, uma questão a ser abordada é a personalização das tarefas cotidianas a partir do perfil de cada usuário. A dinâmica de adaptação, inserida em ambas questões, é especialmente influenciada pela capacidade de captação e interpretação do contexto no qual o usuário está inserido.

Contexto é um termo bastante abrangente, onde, dependendo da perspectiva, aplicações, pessoas, objetos, estados emocionais ou ambientais, entre outros, podem ser considerados parte dele. Alguns aspectos são mais facilmente identificáveis do que outros, e na literatura existe uma diversidade de tratamentos de contexto (CHEN, 2005). Dey (2000) define contexto como “qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é considerado relevante para interação entre o usuário e a aplicação, incluindo o próprio usuário e aplicação”. Já a visão de contexto do middleware EXEHDA (*Execution Environment for High Distributed Applications*) (YAMIN, 2004) (YAMIN, 2005) considera como contexto “toda informação relevante para a aplicação que pode ser obtida por ela” (AUGUSTIN, 2004).

Da mesma forma que sua definição, as abordagens de contexto nas aplicações são variadas (CHEN, 2005), partindo da entrada implícita de dados, passando pela adaptação ao ambiente e dispositivos participantes do contexto do usuário, até a inferência de ações futuras baseada na captação de dados, no histórico e no contexto corrente, cada qual com seus desafios particulares.

O emprego dos sistemas pervasivos, nas mais diversas áreas das atividades humanas, deve melhorar a usabilidade dos sistemas, diminuir o tempo de interação dos seus usuários, tornar as informações presentes nos sistemas mais confiáveis (em função da entrada implícita de dados) e a interação com o usuário mais natural. Esses objetivos podem ser alcançados através da interseção de conceitos oriundos da Computação Pervasiva, bem como conceitos oriundos da Computação Baseada em Tarefas.

A Computação Baseada em Tarefas busca modelar as tarefas que o usuário realiza na sua rotina de trabalho, agregando diversas fontes heterogêneas de dados, interligadas pelo contexto de utilização do usuário. Essa abordagem promove maior usabilidade, já que as informações que pertencem a uma tarefa estão logicamente interconectadas e, portanto, com maior facilidade de acesso. Isso promove redução no tempo gasto pelo usuário na interação com o sistema, podendo gerar um aumento na produtividade da atividade realizada.

Tão importante quanto modelar a tarefa é modelar o contexto associado à tarefa. Para isso, busca-se suporte na área da psicologia, através da teoria da atividade, a qual procura identificar os componentes de uma tarefa e a relação existente entre eles. Associar as soluções da Computação Baseada em Tarefas com Computação Ubíqua produz sistemas com suporte à mobilidade, foco no usuário, pró-atividade, entre outros fatores que contribuem para a construção de sistemas computacionais de uma nova geração.

Uma área de aplicação onde a onipresença da computação é bastante útil, pela natureza da atividade, é a área de Saúde. A área clínica é uma das áreas que pode se beneficiar extensivamente da utilização da Computação Ubíqua aliada aos princípios da Computação Baseada em Tarefas. A área de Cuidados Clínicos (*HealthCare*), além de abordar sistemas hospitalares, trata da extensão de cuidados médicos para a residência do paciente utilizando um sistema pervasivo para monitoramento, como apresentado em (JAHNKE, 2005).

O fator determinante para o emprego da Computação Ubíqua para cuidados clínicos é a identificação dos requisitos desse tipo de sistema, assim como informações relevantes para identificação de contextos. Esse processo é uma importante etapa, a qual deve nortear o desenvolvimento de um projeto dessa natureza, buscando modelar as características peculiares e relativas ao usuário e aos pacientes envolvidos. Um aspecto particularmente importante

refere-se à grande dinâmica e mobilidade existente nas atividades clínicas, onde, normalmente, um profissional atende diversos pacientes. Assim, o sistema deve agir de maneira pró-ativa, evitando que o usuário gaste tempo efetuando procedimentos que podem ser realizados de forma automática no sistema, para efetivamente ajudar o profissional na realização de procedimentos clínicos e reduzir a incidência de erros.

1.1. Definição do Problema

O uso da computação tradicional na área clínica tem deixado várias lacunas a serem preenchidas. Algumas são criadas pelos próprios sistemas computacionais, onde o acesso às aplicações é disponibilizado de maneira pouco eficiente. Muitas vezes, o usuário gasta mais tempo na navegação de menus do que preenchendo dados de interesse efetivo, o que é uma atividade frustrante para usuários de qualquer área. Essa característica é uma consequência natural da vastidão dos tipos de informações, bem como sua quantificação, nos sistemas clínicos, sobrecarregando ainda mais o usuário através de uma interface pouco amigável.

A computação tradicional, por ser orientada à serviços, não é capaz de atender aos inúmeros desafios presentes em sistemas computacionais empregados em áreas com uma elevada dinâmica, tais como suporte as prerrogativas de mobilidade, disponibilidade e foco no usuário (e suas tarefas). Portanto, faz-se necessária a busca por novos paradigmas que possam tratar de tais desafios. A Computação Ubíqua proposta por Mark Weiser mostra-se como uma alternativa viável para fornecer uma nova abordagem na modelagem dos sistemas computacionais, de uma nova geração, focando especialmente no usuário, e deixando o tratamento das características técnicas da computação para os profissionais que de fato são responsáveis por isso.

Na execução de uma tarefa clínica há uma grande quantidade de informações normalmente utilizadas nas aplicações, onde seu registro, em sistemas computacionais tradicionais, é deixado exclusivamente a cargo do usuário-final (KOCH, 2007). Isso gera maior probabilidade de registro de informações erradas, devido a erros de digitação, nos diversos campos que compõem um formulário ou aplicação clínica. Além disso, esta é uma tarefa que demanda um tempo considerável por parte do usuário, podendo reduzir a quantidade de atividades clínicas por ele realizadas em virtude do tempo gasto com as tarefas computacionais.

Assim, o problema a ser aprofundado é **como fornecer um ambiente clínico pervasivo, onde o sistema de contexto atue de forma a reduzir a entrada explícita de dados pelo usuário**. A solução proposta é usar o sensoriamento dessas informações (entrada implícita de dados) e da capacidade pró-ativa do sistema, baseado em elementos de contexto programáveis pelo usuário-final. Assim, são oferecidos mecanismos tanto para obtenção automática dos dados necessários por uma aplicação quanto para o registro automático de informações, personalizável pelo usuário-final.

1.2. Objetivos

Este trabalho se desenvolve no escopo do projeto ClinicSpace, em desenvolvimento no grupo de pesquisa de Sistemas de Computação Móvel (GMob) da UFSM, o qual procura contribuir para reduzir o alto grau de rejeição dos sistemas clínicos computacionais em ambientes dinâmicos, como os hospitalares. Para tal, propõe-se preencher as lacunas existentes nos sistemas clínicos atuais, no que tange às características de pervasividade (ubiquidade) e modelagem de tarefas computacionais baseado nas atividades clínicas que o usuário realiza e na forma individualizada com que o faz. Para realizar essa modelagem, buscou-se suporte no paradigma de Computação Baseada em Tarefas e na Programação orientada ao Usuário-Final (*End-User Programming*) (SILVA, 2009). A arquitetura do modelo ClinicSpace é composta por vários módulos de serviços (SILVA, 2009) (FERREIRA, 2009), que estendem os subsistemas do *middleware* pervasivo EXEHDA (YAMIN, 2004) para promover a capacidade de manipulação de tarefas, além de módulos externos ao *middleware*, como a interface de composição de tarefas, o sistema pEHR (*Pervasive Electronic Health Record*) utilizado e o módulo de adaptação da interface do usuário. Devido à complexidade da arquitetura, esta é explicada mais detalhadamente no capítulo 3.

O *middleware* EXEHDA original não provê suporte a tarefas e seu contexto; logo, um dos serviços implementados sobre o *middleware* é o serviço de gerenciamento de contexto de tarefas (SGCT), tema deste trabalho, que tem por objetivo fornecer:

- uma arquitetura de tratamento de contexto, para que os dados necessários por uma tarefa sejam obtidos de maneira automática junto ao sistema de descoberta de contexto. Para que isso seja possível é necessária a modificação do *middleware* EXEHDA para permitir a personalização de sensores;

- mecanismos para o disparo automático de tarefas, baseado em eventos gerados pelo sistema de contexto (gatilhos) e personalizáveis pelo usuário-final. Para tal, permite-se o registro automático de informações, vinculado ao contexto da tarefa e contendo programação temporal para execução de uma tarefa.

1.3. Estrutura do Texto

No capítulo 2 são introduzidos os conceitos que compõem a base de conhecimentos sobre os sistemas sensíveis ao contexto, sistemas eletrônicos de pacientes, terminologias e características das atividades clínicas, e os conceitos relativos à Computação Baseada em Tarefas. Nesse capítulo também são analisadas as principais pesquisas envolvendo contexto na atividade clínica. No capítulo 3 é apresentado o projeto ClinicSpace, através de uma discussão sobre sua motivação, ideias e conceitos utilizados na modelagem da arquitetura, assim como ela própria. Nesse capítulo também são apresentados os conceitos principais do EXEHDA que estão relacionados diretamente com o trabalho descrito aqui. No capítulo 4 é apresentado o modelo de contexto de tarefa clínica proposto, sendo apresentadas suas funcionalidades e as soluções adotadas para promover o suporte à personalização de sensores e personalização de contextos associados às tarefas. No capítulo 5 são discutidos os resultados obtidos na implementação do protótipo do modelo e os resultados dos testes efetuados. No capítulo 6 é apresentada a conclusão do trabalho seguida das referências bibliográficas. Por fim, são apresentados os apêndices referentes ao Conjunto Mínimo de Tarefas, *Middleware* EXEHDA e Definição *XML Schema* para uma classe de dado.

2. IDENTIFICANDO O CONTEXTO DAS TAREFAS CLÍNICAS

Para procurar identificar o contexto em atividades clínicas foram estudados os sistemas de registro de saúde (EHR - *Electronic Health Record*) (OPENEHR.ORG, 2007) (HL7, 2008) (IHE, 2008) existentes, as terminologias usadas, as atividades clínicas e a teoria da atividade humana. Conhecer os aspectos gerais dos sistemas eletrônicos de registro de saúde (EHR) confere melhor capacidade de discernimento sobre as funcionalidades requeridas para um sistema clínico contextualizado, bem como suas características de implementação.

2.1. Computação Sensível ao Contexto

Contexto é um termo bastante amplo, tendo diversas definições. Sua semântica vai desde definições concretas, como temperatura e pressão, até um nível altamente abstrato como o contexto emocional do usuário ao realizar uma tarefa. Alguns aspectos são facilmente identificáveis, outros não. Por causa deste fato, na literatura existe uma diversidade de tratamentos de contexto (CHEN, 2005). Dey (2000) define contexto como qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é considerado relevante para interação entre o usuário e a aplicação, incluindo o próprio usuário e aplicação. Já a visão de contexto do *middleware* EXEHDA considera como contexto “toda informação relevante para a aplicação que pode ser obtida por ela” (AUGUSTIN, 2004).

Segundo Chen (2002), o contexto pode ser classificado em diferentes subconjuntos:

- **Contexto de Computação:** refere-se às informações de rede, conectividade, custos de comunicação, impressoras, *displays*, computadores e todo o conjunto de informações referentes aos dispositivos;
- **Contexto de Usuário:** refere-se às informações sobre localização, perfil, proximidade, situação social e outras informações relevantes sobre o usuário;
- **Contexto Físico:** refere-se às informações sobre os aspectos físicos, dentre eles, iluminação, barulho, tráfego, temperatura, pressão atmosférica, umidade relativa

do ar, e todos os demais elementos físicos mensuráveis;

- **Contexto de Tempo:** refere-se às informações temporais incluindo hora, mês, ano e estação do ano.

Outro aspecto que pode ser útil à aplicação, além das informações atuais de contexto, são informações históricas. Por exemplo, o comportamento de uma aplicação pode ser influenciado por variações no histórico de determinado contexto. No entanto, Chen (2002) observou que esse tipo de informação dificilmente é utilizada pelas aplicações, salvo raras exceções.

A Computação Sensível ao Contexto emerge como uma forma de prover as funcionalidade de pró-atividade e reatividade, dotando as aplicações de certa inteligência, no que tange à capacidade de interação com o usuário.

Há vários projetos (SOUSA, 2002) (ROMÁN, 2002) (YAMIN, 2004) que tratam da questão de contextualização das aplicações que o usuário executa, cada um adotando políticas distintas no tratamento e na utilização de contexto. O *middleware* Aura (SOUSA, 2002) deixa a cargo da aplicação efetuar o monitoramento de contexto, fornecendo somente os mecanismos necessários para permitir que ela acesse as informações de contexto. Já a abordagem adotada pelos *middlewares* Gaia (RANGANATHAN, 2002) e EXEHDA (YAMIN, 2004) é baseada no paradigma de subscrição (*publish-subscribe*), onde o sistema de contexto é o responsável por notificar a aplicação sobre as alterações ocorridas. Esses trabalhos serão discutidos de forma mais detalhada neste capítulo.

A Computação Sensível ao Contexto é utilizada com diversas funcionalidades:

- i. Adaptação de aplicações, mediante identificação do contexto do usuário, adaptando o ambiente computacional aos parâmetros do equipamento (capacidade de rede, *display*, etc.) que o usuário está utilizando, por exemplo;
- ii. Entrada automática de dados, onde os dados necessários para as aplicações, outrora informados pelo usuário, são obtidos via o sistema de monitoramento de contexto;
- iii. Ações pró-ativas, antecipando-se ao usuário na interação com o sistema computacional, através do monitoramento do contexto corrente em conjunto com as informações obtidas do histórico de utilização e perfil do usuário, bem como modelos pré-estabelecidos para o desenvolvimento das atividades do usuário no sistema.

Cada uma dessas formas de tratamento de contexto oferece vantagens distintas aos usuários, as quais podem atuar de forma complementar em um sistema.

As características do ambiente onde o sistema irá atuar influencia de forma decisiva na abordagem de contexto necessária para tratá-lo. Na próxima seção serão estudadas as características presentes em sistemas clínicos de registros eletrônicos.

2.2. Sistemas Clínicos de Registro Eletrônico

Os sistemas clínicos são conhecidos por sistemas EHR (*Electronic Health Record* ou *Registro Eletrônico de Saúde*, anteriormente chamados de *Sistemas Eletrônicos de Pacientes*). Um sistema EHR pode ser definido como um repositório digital de informações sobre cuidados clínicos abrangendo toda a vida dos pacientes, com o propósito de suportar continuamente tratamento, educação e pesquisa, e garantir a confidencialidade dessas informações, o tempo todo (IAKOVIDIS, 1998). Um sistema EHR inclui dados e informações acerca de todo tipo de tratamento e exames na área clínica, como observações, exames laboratoriais, relatórios de diagnóstico por imagem, tratamentos, terapias, medicamentos ministrados, informações sobre identificação do paciente, permissões legais e alergias. Atualmente, existem diversas soluções implementadas comercialmente, porém não existe um padrão de fato consolidado para intercâmbio dessas informações através de instituições que utilizem diferentes sistemas EHR.

A área de cuidados clínicos consiste num campo multidisciplinar onde a Computação torna-se um importante meio de prover organização e agilidade no registro e acesso às informações dos pacientes, e uma ferramenta de auxílio à tomada de decisões do profissional clínico. Nessa área, além da responsabilidade em lidar com a saúde das pessoas, há fatores críticos como o tempo. Esse é um ramo de atividade onde existe uma grande dinâmica e é necessário otimizar a utilização do tempo, minimizando-o na execução de tarefas. Exemplo onde o tempo é um fator fundamental são as emergências médicas, as quais devem receber atenção do profissional clínico em um curto espaço de tempo; do contrário, pode-se anular completamente a eficácia do atendimento, possivelmente com graves consequências. Esses são alguns dos argumentos que denotam a importância de fornecer ferramentas que auxiliem os profissionais na execução das suas tarefas diárias, da melhor forma, e no menor tempo possível.

Esses aspectos não são plenamente atendidos com a adoção do modelo de computação tradicional, pois esse possui, como característica inerente, o foco da computação no processo

computacional (serviços) e não no usuário-final e em como ele realiza a tarefa. Uma das consequências do modelo tradicional é a sobrecarga gerada sobre o usuário do sistema, visto que ele deve aprender e adaptar-se ao sistema computacional implementado, utilizando-o mais como uma forma de organização de informações do que como um sistema de auxílio às suas tarefas diárias. Nota-se que os ganhos obtidos com essa abordagem são bastante limitados, pois além de o usuário realizar a atividade como ele normalmente faria, é preciso conhecer e simular suas etapas em um sistema computacional; etapas essas, normalmente, não-personalizáveis, repletas de informações não vitais para aquele dado momento e, não raro, com uma interface pouco amigável (KOCH, 2007).

Por outro lado, a Computação Pervasiva (ou Ubíqua) propõe uma inversão nessa lógica, tendo seu foco no usuário-final e suas atividades diárias. Em virtude disto, vê-se um mundo onde os dispositivos computacionais dominam o ambiente, estando presentes em todo lugar, acessíveis a qualquer momento e fornecendo as aplicações específicas que o usuário precisa naquele instante de forma invisível (não presente no foco de atenção do usuário), ou seja, totalmente integrados ao ambiente real (AUGUSTIN, 2004). Disponibilizar sistemas adaptativos ao usuário levando em conta seu contexto é uma premissa fundamental em sistemas ubíquos ou pervasivos.

Em virtude desse novo foco de priorização, atividades humanas cotidianas do usuário-final, identificar os aspectos importantes para sistemas clínicos é uma atividade essencial no decorrer do processo de construção de qualquer sistema pervasivo para essa atividade humana.

2.2.1. Requisitos de um Sistema Ideal de Cuidados Clínicos

Existem alguns trabalhos que buscam identificar as funcionalidades que os sistemas EHR devem possuir, seus aspectos e características de segurança e acesso. A seguir são resumidas algumas dessas funcionalidades e questões que devem ser tratadas, a partir dos resultados do projeto OpenEHR (OPENEHR.ORG, 2007), que sumariza os anseios encontrados em projetos de sistemas computacionais de saúde.

2.2.1.1. Requisitos sob o ponto de vista dos registros

A prioridade dos sistemas clínicos deve ser no profissional clínico e no paciente, fornecendo interface adequada para a interação. Além disso, deve-se permitir a rastreabilidade, auditoria e controle legal das ações médicas, através de registro de procedimentos realizados, de forma detalhada e permanente no sistema. O software deve estar em constante aperfeiçoamento, portanto, flexibilidade e facilidade de manutenção são requisitos básicos para um sistema deste tipo. Outro fator importante é fornecer o suporte para todos os tipos de dados clínicos: tabelas, listas, eventos, etc.

Um dos principais componentes constituintes de um sistema EHR são os registros (prontuário) eletrônicos do paciente (PEP) ou *Electronic Patient Record* (EPR). Ele descreve todo histórico do paciente; em consequência, deve dar suporte a todas as linguagens naturais, visto que deve suportar dados oriundos da linguagem de qualquer país. Fornecer suporte a traduções entre as linguagens do registro fornece um meio eficiente de globalizar o registro, permitindo que um paciente seja tratado em mais de um país no decorrer da sua vida, sem que informações sejam desconsideradas em função de estarem em uma linguagem desconhecida. Para que isso seja possível, deve ser utilizada uma terminologia que dê suporte também à integração com outras terminologias clínicas, visto que um sistema EHR não deve estar atrelado a uma única terminologia.

Um sistema EHR também deve suportar mecanismos de compartilhamento de informações entre diferentes fontes de dados. No tratamento de dados clínicos, existe uma preocupação especial com a privacidade do paciente, disponibilizando suas informações somente para os profissionais que de fato estiverem autorizados a visualizá-las. Um mecanismo possível, em algumas investigações, é exibir as informações do prontuário do paciente, sem identificá-lo.

Pela característica colaborativa e multidisciplinar do ambiente clínico, dar suporte a mecanismos automáticos ou semi-automáticos para distribuição de fluxos de trabalho fornece uma característica flexível ao sistema e eficaz para seus usuários, além de fornecer uma forma de auditoria nos fluxos de trabalho.

2.2.1.2. Requisitos sobre o ponto de vista clínico

Do ponto de vista clínico, o sistema EHR deve ser centrado no paciente, e seus registros eletrônicos devem durar toda a vida, proporcionando uma visão holística das necessidades do paciente. Deve conter informações relevantes de todas as especialidades e serviços de emergência, como também informações gerais sobre o paciente. O sistema deve agregar as diferentes visões existentes sobre os dados clínicos do paciente, tais como emergência, patologia, radiologia, receituário, etc. Esses dados, não raro, encontram-se registrados de forma independente, e o sistema EHR deve ser capaz de agregá-los em uma visão única.

Uma característica desejável, porém complexa, é dar suporte a decisões clínicas, promovendo a segurança do paciente e a redução de custos gerados pela repetição de investigações médicas. O suporte a decisões é possível através da combinação de modelos de procedimentos clínicos, baseados em contextos históricos e da observação e exames correntes do paciente, aliados a mecanismos de inferência.

2.2.1.3. Interoperabilidade entre os Sistemas

A integração das ferramentas EHR é uma grande promessa para generalizar e tornar amplamente disponíveis os benefícios da computação, já demonstrados para casos individuais e isolados. Esta integração deverá promover a redução dos erros médicos como interações, duplicações e tratamentos inapropriados, assim como os custos associados a essas questões, visto que o registro atualizado e disponível permite aos profissionais clínicos manter um controle rígido sobre os procedimentos realizados, evitando as questões de duplicação de procedimentos pela falta de comunicação entre eles. O acesso às informações críticas sobre o paciente torna-se disponível, mesmo que nas mais variadas fontes, através da interoperabilidade dos sistemas, promovendo facilidade no acesso.

Em função do monitoramento do histórico do paciente durante toda sua vida, é possível construir sistemas de prevenção e detecção precoce de doenças, baseado em análise preditiva de fatores de risco, obtidos das informações registradas na base de dados do EHR. Com essa disponibilidade de informações sobre o paciente, as admissões hospitalares também

podem diminuir em função da consulta a diagnósticos anteriores, diminuindo o tempo de resposta às necessidades do paciente e os custos envolvidos na operação.

Essa interoperabilidade dos sistemas é um desafio há muito estudado no domínio da Tecnologia da Informação. Diversas abordagens já foram desenvolvidas, tais como: ODBC, *gateways* de dados, listas de mensagens e motores de interface, adaptadores de software e *web services* (EICHEMBERG, 2005). Visando alcançar essa interoperabilidade há diversos projetos em andamento para realizar uma padronização desses sistemas (OPENEHR.ORG, 2007) (HL7, 2008) (IHE, 2008). Um estudo sobre os atuais EHRs foi realizado no GMob (Grupo de pesquisa em Sistemas de Computação Móvel) e está disponível na referência (VICENTINI, 2009).

2.2.2. Terminologias Clínicas

Os primeiros esforços na construção de terminologias clínicas datam do século XVII (CASTILHA, 2007). Desde, então, diversos trabalhos têm sido desenvolvidos. Segundo Castilha, “...o objetivo das diversas terminologias criadas tem sido, entre outras coisas, coletar nomes de substâncias, qualidades, estruturas e processos utilizados tanto no ambiente de pesquisa como na prática clínica. As terminologias médicas refletem a diversidade dos campos biomédicos e principalmente os motivos que nortearam seu desenvolvimento ...”.

A utilização de terminologias torna possível o intercâmbio de informações de diversas fontes, sendo um parâmetro importante na construção de sistemas computacionais clínicos. Algumas das classificações produzidas, para uso genérico, em sistemas clínicos são:

- i. Cadastro Internacional de Doenças (CID). Criado e mantido pela Organização Mundial da Saúde, tem por objetivo a classificação de doenças para as mais diversas finalidades clínicas;
- ii. Systematized Nomenclature of Medicine - Clinical Terms (SNOMED – CT). Atualmente, mantido pela *International Health Terminology Standards Development Organization (IHTSDO)*, a qual é uma organização sem fins lucrativos, mantida por um consórcio de países associados;
- iii. MEDCIN é uma terminologia clínica proprietária, desenvolvida pela Medicomp Systems, Inc. Utilizada na construção de sistemas EHR, define mais de 250 mil termos, e mantém compatibilidade com SNOMED e CID.

Algumas terminologias são mais abrangentes que outras. CID, por exemplo, não especifica recursos no meio clínico, mas fornece um cadastro eficiente para especificação de doenças. Dentre as terminologias atuais, SNOMED desponta como sendo uma das mais abrangentes e proeminentes.

SNOMED-CT (*Systematized Nomenclature of Medicine - Clinical Terms*) é uma terminologia clínica, baseada em quatro componentes principais: conceitos, descrições de conceitos, hierarquia e relações. Esta terminologia permite especificar doenças, tratamentos, diagnósticos e outros procedimentos, mapeando de forma hierárquica todos os componentes envolvidos nessa atividade. Foi originalmente desenvolvida pelo *College of American Pathologists*; atualmente, é mantida pela IHTSDO, organização situada na Dinamarca e formada por um consórcio de países, sem fins lucrativos.

Os conceitos utilizados pela SNOMED-CT são definidos por Wang (2002) da seguinte forma: “Um conceito é uma unidade de pensamento, a qual é representada usando uma ou mais descrições de contexto ou termos”. Um subconjunto de conceitos, descrições ou relações do SNOMED-CT são apropriados para utilização em uma linguagem particular, dialeto, país, especialidade, organização, usuário ou contexto.

A estrutura hierárquica permite que o sistema computacional possa também efetuar validação dos dados, baseado nas faixas aceitáveis descritas pela terminologia. Essa característica pode ser integrada aos sistemas de registros clínicos (EHR) onde é possível realizar a validação das informações de entrada, oferecendo um mecanismo eficiente para manter a consistência dos registros clínicos.

Existe um mecanismo utilizado na terminologia que permite sua personalização, são as extensões. Através delas é possível regionalizar a terminologia, definindo sinônimos locais para conceitos já definidos ou definir novos conceitos, ainda não suportados pelo padrão internacional da terminologia. Esse mecanismo possibilita também o mapeamento de outras terminologias aos termos do SNOMED, o que possibilita maior capacidade de integração, mesmo com bases de dados de outras terminologias. O padrão internacional de SNOMED já tem amplo suporte ao CID (Classificação Internacional de Doenças), por exemplo.

Sua ampla abrangência, definindo termos clínicos, relações entre esses termos e as extensões, que permitem além de personalização maior flexibilidade de integração com outras ontologias, credencia SNOMED como uma das mais aceitas terminologias clínicas em desenvolvimento.

Embora a base da terminologia não seja disponibilizada para uso livre, existem

navegadores WEB onde se pode visualizá-la (NCI, 2008) (BT, 2008). O fonte UML da linguagem pode ser obtido para pesquisa e desenvolvimento, através do site da *National Library of Medicine* (NLM, 2009).

2.3. Características das Atividades Clínicas

As atividades clínicas consistem em atividades desenvolvidas por diversos profissionais de campos multidisciplinares e correlatos. Um cenário clínico, usualmente, é composto por médicos de diferentes especialidades, como cirurgiões e radiologistas, enfermeiros, auxiliares de enfermagem, farmacêuticos, etc. Para o melhor resultado de qualquer tratamento de cuidados clínicos, assim como eficiência na utilização dos recursos, é essencial um alto grau de interação entre esses profissionais, principalmente no que se refere ao compartilhamento de informações sobre procedimentos realizados, status e histórico do paciente (BARDRAM, 2004).

Outra característica que pode ser facilmente observada na área de cuidados clínicos é que esta consiste em uma atividade de natureza bastante dinâmica e descontínua. Os profissionais envolvidos nesse trabalho, normalmente, atendem um número elevado de pacientes, inclusive o mesmo paciente várias vezes em um mesmo dia, como no caso de um hospital.

Essa dinâmica e a interação dos profissionais da área da saúde levantam algumas questões a serem tratadas pelos sistemas computacionais clínicos, sendo as principais:

- **Segurança.** Informações clínicas normalmente são confidenciais e devem ser tratadas como tal, impedindo o acesso por pessoas não autorizadas. Além disso, é necessário atestar a procedência das informações registradas, certificando-se de que elas proveem de pessoas autorizadas e credenciadas a fazê-lo;
- **Praticidade no Acesso.** Um sistema para cuidados clínicos deve ser ágil e pró-ativo, evitando que os profissionais percam tempo, realizando procedimentos complexos de registro, ou esquecendo (evitando) registrar tais informações em virtude do pouco tempo disponível;
- **Colaboração.** O diagnóstico médico passa por várias etapas, por exemplo: raio-x, resultados de análise sanguínea, histórico do paciente, discussão com colegas, etc. Esses estágios de diagnóstico, normalmente, são executados em diferentes períodos de

tempo e em diferentes localizações.

Os atuais EHR, baseados em processos, não atendem a todos esses requisitos devido a limitações impostas pelo modelo de computação tradicional. Acredita-se que a Computação Baseada em Atividades (ver seção 2.4) pode ser vista como a maneira mais adequada de promover a construção de sistemas de cuidados clínicos pervasivos, em virtude do constante deslocamento dos usuários e da rápida comutação de dispositivo possível durante tal deslocamento, além do compartilhamento de informações (BARDRAM, 2004).

2.3.1. Descoberta Automática de Atividades

A descoberta automática de atividades agrega maior grau de pró-atividade para os sistemas computacionais orientados aos usuários-finais. A capacidade de o sistema inferir corretamente as tarefas que o usuário está realizando, antecipando as próximas etapas, é diretamente proporcional à precisão das informações que o sistema contém acerca das atividades normalmente realizadas pelo usuário, ou seja, pelo conhecimento das atividades do campo de aplicação. Além do conhecimento das atividades, o sistema deve saber em que ponto da tarefa o usuário se encontra e qual seu contexto. Visto isso, pode-se descrever três fatores primordiais no que tange à descoberta de atividades:

- Contexto: os objetos e pacientes, incluindo atributos destes, próximos ao profissional, abrangendo informações atuais e históricas;
- Modelos de atividades: conhecimento sobre as atividades da área de atuação do sistema. Na área clínica, por exemplo, a ação de um médico visualizando um raio-x indica que, provavelmente, uma tarefa de diagnóstico está sendo realizada, naquele momento;
- Técnicas de programação: baseado nas informações de contexto e no modelo de atividades utilizam-se técnicas de programação para inferir a atividade a ser realizada pelo profissional, permitindo ao sistema agir de maneira pró-ativa.

A descoberta de atividades contém vários desafios, entre eles o fato de que a descoberta de certas atividades, às vezes, não é passível de ser definida com precisão. Um exemplo típico na área clínica é saber qual paciente o médico vai atender logo em seguida, pois essa informação somente pode ser obtida com precisão no momento em que o paciente

estiver entrando no consultório para realizar a consulta.

2.3.2. Requisitos para Modelar Atividades Clínicas

De acordo com as pesquisas de Bardram (2004), existem dois fatores especialmente presentes nas atividades clínicas: (i) dinamismo e mobilidade e (ii) parada e reativação de tarefas.

2.3.2.1. Dinamismo e Mobilidade das Tarefas

As atividades clínicas têm uma natureza particular, sendo alguns pontos importantes a observar:

- as atividades clínicas são muito dinâmicas e os profissionais se deslocam de um ambiente para outro rapidamente, inclusive saindo e entrando no mesmo ambiente várias vezes por turno. Para se ter uma ideia, uma enfermeira chega a caminhar 14 km durante um turno num hospital (BARDRAM, 2007);
- em virtude dessa mobilidade, as tarefas devem acompanhar esses profissionais onde quer que eles estejam dentro do hospital, seja através de um dispositivo portátil ou não;
- dispositivos com baixa resolução de tela não são apropriados para apresentar informações médicas (telefones celulares, PDA's).

2.3.2.2. Parada e Reativação de Tarefas

Médicos tratam de 10 a 15 pacientes no hospital e 100 ou mais em clínicas (BARDRAM, 2007). Em virtude disso, as interrupções são frequentes e é necessário que o profissional possa sair da tarefa A para a tarefa B e, depois, possa voltar para a tarefa A, exatamente no ponto em que havia parado, inclusive utilizando para isso outro dispositivo.

2.3.2.3. Compartilhamento de Tarefas

Devido à natureza e ao número de profissionais das mais diversas áreas envolvidos no ambiente clínico, o compartilhamento de informações entre os profissionais é um ponto-chave. O trabalho médico é altamente colaborativo devido à natureza especialista do tratamento: médicos, enfermeiros e assistentes devem colaborar através do tempo e espaço. Outro aspecto importante a ser observado é a característica assíncrona da comunicação entre esses profissionais, essencial numa mudança de turno, por exemplo, ou na discussão de um médico com um especialista em radiologia, que está em outra localização.

2.4. Computação Baseada na Atividade Humana

As atividades humanas são atividades complexas, que envolvem o aspecto psicológico de comportamento e cognição, bem como fatores sociais e econômicos. Diversos estudos já foram realizados, alguns dos quais propuseram modelos de como essas atividades são desenvolvidas, como Cognição Distribuída, Modelo de Ações Situadas, e Teoria da Atividade (NARDI, 1996). Sendo esse último mais amplamente utilizado.

2.4.1. Teoria da Atividade

A teoria da atividade é um modelo no qual se busca identificar um conjunto de componentes comuns constituintes da atividade e o relacionamento existente entre esses componentes. A teoria original, proposta por pesquisadores russos na década de 1920, definia três componentes nas atividades: objeto, sujeito e ferramentas (NARDI, 1996). Esses componentes são intrinsecamente interligados. A razão de ser da atividade, ou seja, sua meta, corresponde ao objeto, o qual o usuário se propõe na realização da tarefa, utilizando-se, para isso, de um conjunto de instrumentos de mediação, ou ferramentas.

Aperfeiçoada ao longo do tempo, essa teoria foi expandida (ENGESTROM, 1999) para incluir os aspectos sociais, passando a identificar seis elementos na atividade: sujeito, objeto, ferramentas, regras, comunidade e colaboração. O sujeito realiza uma atividade

buscando atingir uma meta, ou seja, seu objetivo, utilizando-se para isso ferramentas, norteadas por regras, sendo esse usuário inserido em uma comunidade, onde se pode utilizar a colaboração entre indivíduos para que uma atividade seja realizada. A Figura 1 mostra esse relacionamento.

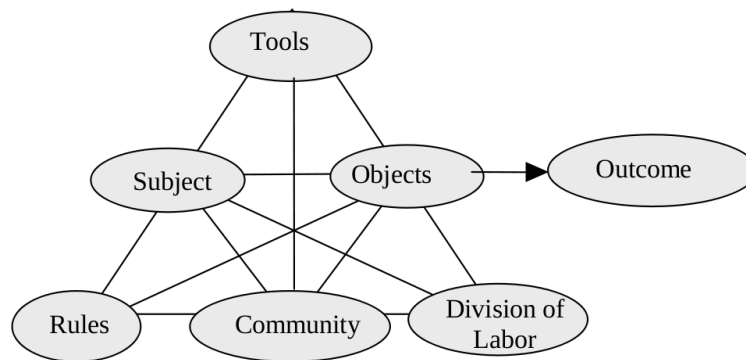


Figura 1 - Componentes da Teoria da Atividade. Fonte: (KAENAMPORNPAN, 2005).

2.4.2. Teoria da Atividade Aplicada ao Contexto Clínico

Aplicando ao ambiente clínico o modelo genérico proposto pela teoria da atividade tem-se o usuário clínico atuando como o sujeito, o qual tem por objetivo diagnosticar e/ou tratar o paciente que atua como objeto da tarefa. Para realizar sua atividade, o clínico utiliza-se de ferramentas de mediação, as quais consistem em registros, procedimentos, equipamentos e recursos, os quais propiciam a execução da tarefa. Essa mediação, normalmente, segue protocolos especificados através de guias clínicos, constituindo-se nas *regras*. A tarefa é executada no âmbito clínico, sendo esta a sua comunidade, onde há uma diversidade de profissionais que, usualmente, realizam uma divisão do trabalho para realização de uma tarefa. Portanto, nesses ambientes onde uma tarefa envolve mais de um profissional, existem vários pontos de vista para a tarefa, dependendo do profissional que se toma como referência.

Por exemplo, na grande maioria dos casos, o procedimento de diagnóstico clínico, baseado nos sintomas informados pelo paciente e pelo seu histórico, poderá requerer alguns

exames complementares para auxiliar no diagnóstico definitivo. Esses exames são realizados por outros profissionais, um farmacêutico, por exemplo, o qual produz um laudo que deverá ser analisado pelo médico. Este, se considerar ter informações suficientes, produz um diagnóstico, podendo receitar um tratamento específico ou, se julgar necessário, poderá requisitar outros exames para complementar as informações e produzir um diagnóstico mais preciso. Nota-se aqui uma relação de colaboração entre os procedimentos realizados pelo médico e pelo farmacêutico, sendo partes complementares de uma mesma tarefa, porém com pontos-de-vista totalmente diferentes. O médico não precisa saber como é o processo de realização do exame, basta conhecer o resultado e o farmacêutico, por outro lado, não necessita conhecer o histórico do paciente ou mesmo os motivos que levaram àquele exame ser requisitado.

2.4.3. Atividades *versus* Tarefas Clínicas

Para executar uma atividade humana, frequentemente as pessoas devem executar tarefas computacionais. Elas auxiliam no controle, consistência e validação de uma atividade humana através da utilização de ferramentas computacionais, registrando, recuperando e processando dados de e para a atividade humana. Ranganathan (2005) define uma atividade como “um conjunto de ações realizadas colaborativamente por humanos e por um sistema computacional pervasivo para obter um objetivo”.

Semanticamente, a palavra atividade remete ao conceito de ramo de atividade ou área de atuação, onde para cada qual há um conjunto de atividades comuns. Considerando essa ideia, definiu-se na arquitetura ClinicSpace (ver Capítulo 3) que o termo atividade refere-se exclusivamente às atividades realizadas por humanos, ao passo que as ações que o usuário realiza, utilizando um sistema computacional de apoio à atividade, denomina-se tarefa computacional.

Uma tarefa computacional pode ser de variados tipos, por exemplo: considerando uma consulta clínica, o médico deverá, através de entrevista ou exames, coletar dados sobre o paciente e, se houver disponível um sistema computacional para isso, registrar tais dados em um formulário eletrônico. Essa última tarefa consiste em uma atividade de apoio (tarefa computacional), não sendo objetivo principal do médico, embora necessário.

2.5. Contexto em Atividades Clínicas: Análise de Trabalhos Relacionados

Este trabalho é composto pelo conjunto de duas vertentes de pesquisas em contexto, os sistemas sensíveis ao contexto de forma geral e os sistemas de contexto na área clínica, que, embora distintas, são complementares para o objetivo proposto neste trabalho – definir um sistema de contexto pervasivo para a área clínica. A seguir são discutidos alguns dos trabalhos relacionados com essas áreas.

2.5.1. Identificação de Contexto na Área Clínica

Pesquisas recentes buscam identificar e modelar as informações sensíveis ao contexto no ambiente clínico, como o trabalho desenvolvido por KOCH (2007). Ele trata sobre a questão da recuperação de informações e adaptação dos sistemas clínicos baseados no contexto do usuário, através do conhecimento das atividades médicas, as quais consistem em um fluxo de operações executadas pelo usuário. Conhecendo-se esse *workflow* e a atividade que o usuário realiza no momento, é possível prever quais as informações que efetivamente serão necessárias na etapa seguinte, filtrando-as. KOCH defende a disponibilização das informações seguindo o paradigma: “entregar a informação certa, no montante adequado, no local e tempo em que elas são necessárias”. Como resultado, espera-se a redução da sobrecarga de informações, imposta aos profissionais da área da saúde, relativo a dados não essenciais, em um dado momento.

Koch (2007) propõe um modelo no qual se identificam cinco categorias de contextos: (i) contexto do paciente, (ii) contexto do médico, (iii) contexto do processo, (iv) contexto do ambiente e (v) contexto da busca. Esse modelo é centralizado na recuperação e exibição de informações médicas, concentrando os esforços de modelagem na questão adaptativa do sistema clínico à atividade do médico.

Essa é uma abordagem válida para um sistema de contexto adaptativo, porém dotar o sistema de contexto pervasivo dessa capacidade aumenta significativamente as dificuldades de projeto, em virtude da complexidade dos dados clínicos. Esse tipo de informação é melhor conhecida e tratada por um sistema clínico propriamente dito, ou sistema EHR.

Existem outros trabalhos que tendem para essa abordagem, tal como os trabalhos

apresentados por Bardram (BARDRAM, 2004) (BARDRAM, 2007). O trabalho de Bardram é focado na questão da adaptação das aplicações ao contexto, sendo um recurso utilizado de diferentes formas, dependendo do contexto que ele se encontra. Um exemplo muito utilizado na descrição da pesquisa é a utilização de um *display* no quarto do paciente no hospital. Esse *display* pode servir para entretenimento do paciente, exibindo canais de TV ou filmes, bem como pode ser utilizado para mostrar o prontuário do paciente na ocasião de um médico estar presente, ou exibir o esquema de medicação na presença de uma enfermeira com um recipiente para medicamentos.

Através da realização de *workshops*, contando com a presença de profissionais da área da saúde, foram construídos cenários para auxiliar na compreensão das questões de contexto do sistema. As conclusões obtidas, por Bardram, através desses cenários são: (i) o contexto é especialmente útil para apresentar dados; (ii) o sistema deve conhecer e dar suporte às atividades do usuário. (iii) a arquitetura de contexto deve ser extensível e flexível para permitir a acoplagem de novos recursos e sensores ao sistema, o que é uma consequência natural em uma área dinâmica, com constantes aperfeiçoamentos; (iv) serviços devem ser distribuídos e colaborativos, mantendo baixa acoplagem entre os serviços de contexto disponíveis. (v) segurança e privacidade devem ser fornecidos, pois os dados do paciente são confidenciais, e mecanismos de segurança devem ser implementados para não permitir que pessoas não autorizadas tenham acesso. Como se pode perceber, esse trabalho busca aproximar a abordagem computacional do contexto com as atividades humanas, através da teoria da atividade, sendo esse o objetivo principal do projeto ABC (*Activity-Based Computing* – (ABC, 2009)) conduzido por Bardram.

Semelhantes às conclusões de Bardram, o trabalho apresentado por Dahl (2004) identifica algumas necessidades dos ambientes clínicos dado sua intensa dinâmica, através da colaboração e da constante parada e reativação de tarefas. Um médico, pode estar tratando de um paciente, realizando um diagnóstico, e durante a realização dessa tarefa (A), uma emergência com outro paciente seu pode ocorrer, tal como um ataque cardíaco. Imediatamente o médico deverá parar a tarefa que está realizando (A) e começar a tarefa de tratamento do outro paciente (B). Quando esta última tarefa estiver concluída, a tarefa anterior (A) deverá ser retomada. Porém, há situações em que tarefas não devem serem paradas, como por exemplo, o atendimento de um paciente em situações de emergência. Nesse caso, se houver outra emergência com um outro paciente, esta deverá ser redirecionada a outro clínico que esteja tratando de uma tarefa menos prioritária.

Um ponto importante a considerar é que sempre um profissional deve ser notificado

sobre um evento ocorrido com um paciente, não podendo ignorar eventos ocorridos com quaisquer pacientes. Ou seja, deve haver uma coordenação de perspectivas dos profissionais clínicos.

Outro ponto levantado por Dahl (2004), também considerado pelas pesquisas de Bardram, é o cuidado que se deve ter com as execuções automáticas de tarefas. Um sistema, por melhor que seja, pode eventualmente falhar ou captar dados errôneos. Na área clínica, tomar uma decisão ou adotar uma ação através de uma perspectiva equivocada pode trazer sérios problemas para o paciente.

Dessa forma, os trabalhos desenvolvidos por esses pesquisadores levaram-nos a concluir que o sistema, ainda que pró-ativo, deve ser controlado pelo usuário, através da confirmação antes da execução automática de uma tarefa. Isso permite também deixar o usuário mais consciente sobre o que se está realizando, uma vez que mudanças abruptas de interface deixam o usuário confuso. Esse tipo de abordagem resulta em uma perspectiva de que o controle do sistema deve ser mantido pelo usuário.

Embora voltados para a área clínica, e identificando importantes aspectos a serem utilizados em um sistema de contexto clínico, esses sistemas não tratam da questão de personalização do contexto relativos às tarefas de forma explícita e controlada pelo usuário, tampouco da questão de definir níveis de abstração para o tratamento das informações, como Sistemas de Gerenciamento de Contexto sobre um sistema clínico EHR propriamente dito.

2.5.2. Sistemas Sensíveis ao Contexto considerando a Teoria da Atividade Humana

A construção de sistemas sensíveis ao contexto é abordada em muitos trabalhos recentes. Uma das vertentes que tem se mostrado promissora é a questão da modelagem de contexto levando em consideração as atividades realizadas pelo usuário, através do modelo lógico proposto pela teoria da atividade. Alguns dos trabalhos que buscam realizar essa aproximação podem ser vistos em (KAENAMPORN PAN, 2005) e (CASSENS, 2006).

Na literatura, os trabalhos que abordam o contexto através dessa perspectiva utilizam o modelo estendido da teoria da atividade, proposto por Engestrom (ENGESTROM, 1999). Esse modelo leva em consideração, além dos fatores básicos da teoria da atividade, a questão do aspecto social, colaborativo e das regras e guias que norteiam a atividade, fatores esses importantes na percepção do contexto da atividade humana.

Outros trabalhos, como o de Kaenampornpan (KAENAMPORNPAN, 2005) defendem que, além da utilização da modelagem das atividades humanas, o sistema de contexto deve dar suporte ao fator histórico. As atividades humanas são usualmente compostas por melhorias através do aprendizado realizado na execução anterior de tarefas similares. As variações ocorridas na tarefa, no decorrer de sua execução, influenciam suas execuções futuras, através da observação de fatores do contexto associados aos resultados da execução da tarefa. O trabalho de Kaenampornpan, assim como diversos outros, propõem um modelo lógico a ser considerado na construção de uma aplicação consciente de contexto, mas não aproxima esse tratamento especificamente para à área clínica, nem oferece a possibilidade de o usuário explicitamente configurar sua atividade através da especificação de contextos relevantes.

2.5.3. Sistemas Pervasivos Pró-Ativos com Reconhecimento de Contexto

Aura (SOUSA, 2002) e Gaia (ROMAN, 2002) são sistemas pervasivos da primeira geração que influenciaram muitos outros e sedimentaram muitos dos conceitos usados em sistema ubíquos; por outro lado, o projeto ABC é importante representante de sistemas ubíquos orientados a atividades em ambientes hospitalares.

2.5.3.1. Projeto Aura

O projeto Aura (SOUSA, 2002), desenvolvido na Carnegie Mellon University, tem por objetivo promover uma arquitetura de gerência para um sistema pervasivo, baseado nas atividades realizadas pelo usuário. Através desta perspectiva, o sistema busca uma aproximação mais intrínseca com o usuário, através da disponibilização, busca e adaptação de recursos a fim de corresponder as necessidades da tarefa que o usuário executa.

Com esse foco, busca modelar uma arquitetura que dê suporte à maximização da disponibilidade de recursos através da comunicação entre dispositivos de forma inteligente, fornecendo um ambiente pervasivo. Com isso, busca automatizar o processo de migração e reconfiguração do ambiente computacional do usuário, minimizando a distração outrora ocasionada por esse processo.

O conceito principal da arquitetura é a “aura” computacional do usuário, a qual constitui-se no mecanismo de identificação automática do usuário, da tarefa do usuário, bem como os requisitos necessários para sua realização. Baseado nas informações obtidas da aura computacional do usuário, o *middleware* Aura pró-ativamente busca os serviços e recursos disponíveis e mais adequados à realização da tarefa. Portanto, é uma arquitetura baseada em serviços, onde o usuário poderia mover-se entre diferentes ambientes, sempre levando consigo sua aura computacional, a qual é responsável por identificar os serviços necessários em um novo ambiente. Dessa forma, isenta-se o usuário da configuração de recursos computacionais, bem como das ações explícitas de migração da tarefa.

Visão geral da arquitetura

A arquitetura para fornecer o suporte pervasivo proposto por Aura foi modelada na forma de quatro componentes principais, conforme Figura 2, sendo eles:

- Gerenciador de Tarefa (*Task Manager*): consiste no componente representado pela aura do usuário, ou seja, é o componente responsável por orquestrar o processo de mobilidade necessário na arquitetura bem como todas as prerrogativas associadas. Dentre essas, incluindo descoberta e associação de novos recursos e identificação dos serviços adequados à realização da tarefa. É o componente responsável pelo monitoramento do ambiente que circunda o usuário, comunicando-se com os demais componentes da arquitetura para descobrir e utilizar os serviços e recursos necessários pela tarefa;
- Provedores de Serviço (*Service Suppliers*): constituem-se nos componentes fornecedores de serviço na arquitetura, incluindo as ferramentas e aplicações disponíveis no ambiente. Cada provedor descreve suas capacidades; na busca dos recursos adequados à realização de uma tarefa, busca-se o serviço que melhor supre suas necessidades;
- Observador de Contexto (*Context Observer*): disponibiliza a informação física de contexto e reporta eventos físicos para o *Task Manager* e *Environment Manager*. Traduz-se no componente de monitoramento de contexto da arquitetura;
- Gerenciador de Ambiente (*Environment Manager*): consiste em um *gateway* do ambiente, realizando o mapeamento necessário para que o *Task Manager* encontre os serviços correspondentes às suas necessidades no ambiente pervasivo.

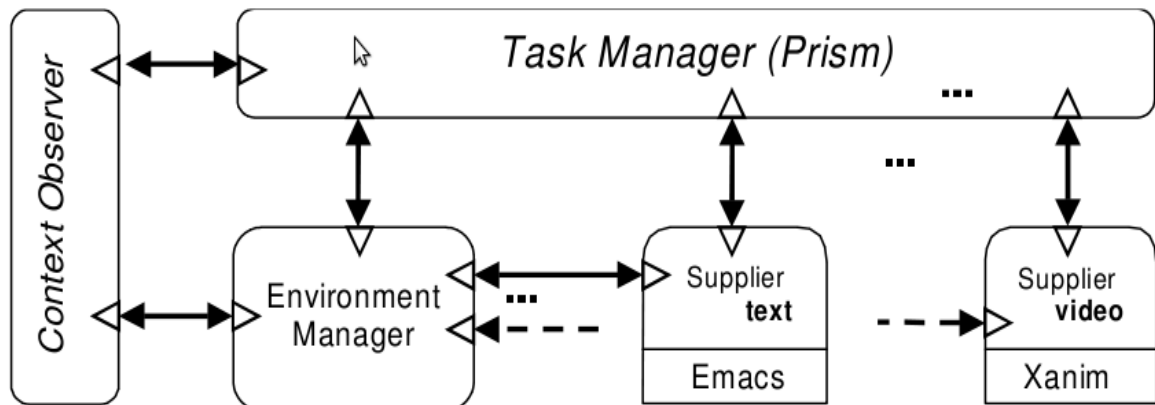


Figura 2 - Sistema Operacional Aura. Fonte: (SOUSA, 2002).

Contexto em Aura

O Sistema de Contexto proposto, monitorado através do componente *Context Observer*, consiste em um sistema baseado em múltiplas fontes de dados, as quais são diferenciadas basicamente em função da dinâmica de atualização destes. Dados estáticos, com baixa ou nenhuma frequência de atualização são disponibilizados através de um banco de dados comum, o que traz consigo toda a estrutura de gerenciamento de dados presentes no sistema. Já dados com uma elevada frequência de atualização não são adequados para esses sistemas, em virtude do gargalo que pode formar-se. Para isso, a arquitetura prevê a possibilidade de utilização de outras fontes de dados provedoras de informação, construídas de forma dedicada.

Para acesso às informações de contexto, Aura usa o CIS (*Context Information System*), o qual especifica as buscas de dados na linguagem *CSint*, similar ao *SQL* (*Simple Query Language*). As requisições de contexto, feitas nessa linguagem, são processadas por um sintetizador, responsável por adequar a requisição à entidade promotora da informação, a qual pode ser um banco de dados, ou diretamente um dispositivo que possa interpretar e responder a uma requisição realizada nessa linguagem. A Figura 3 ilustra o serviço de contexto proposto pela arquitetura.

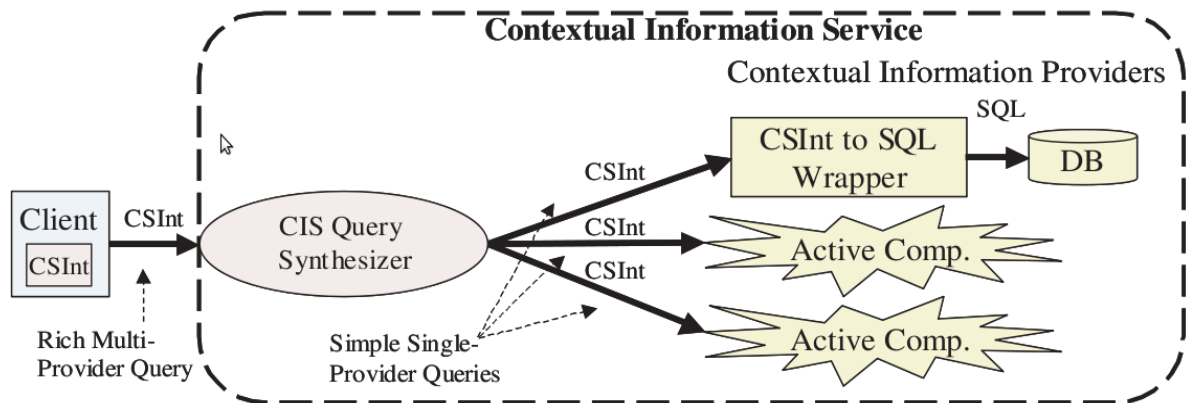


Figura 3 - Modelo de Contexto CIS (Context Information Service) do Aura. Fonte: (JUDD, 2003).

2.5.3.2. Projeto Gaia

O projeto Gaia (ROMAN, 2002) baseia-se na ideia de espaços ativos (*Active Spaces*), o qual é promovido através da extensão das funcionalidades de um sistema operacional tradicional, utilizando para isso, suporte extra a contexto, especialmente localização de entidades, dispositivos móveis e atuadores. O projeto tem a visão da construção de um ambiente computacional, composto de diversos componentes que interagem de forma transparente, como se fosse um único sistema, do ponto de vista do usuário.

Visão geral da arquitetura

A arquitetura Gaia é composta de um núcleo, sobre o qual operam os serviços de contexto, sistema de arquivos, presença, eventos e repositórios pervasivos. As aplicações fazem acesso ao *middleware* através de um *framework* que abstrai as complexidades de acesso aos demais componentes do *middleware*. As aplicações rodando sobre esse *middleware* compõem a camada de aplicações do *Active Space*, oferecendo ao usuário as características pervasivas. A construção desses componentes é ilustrada na Figura 4.

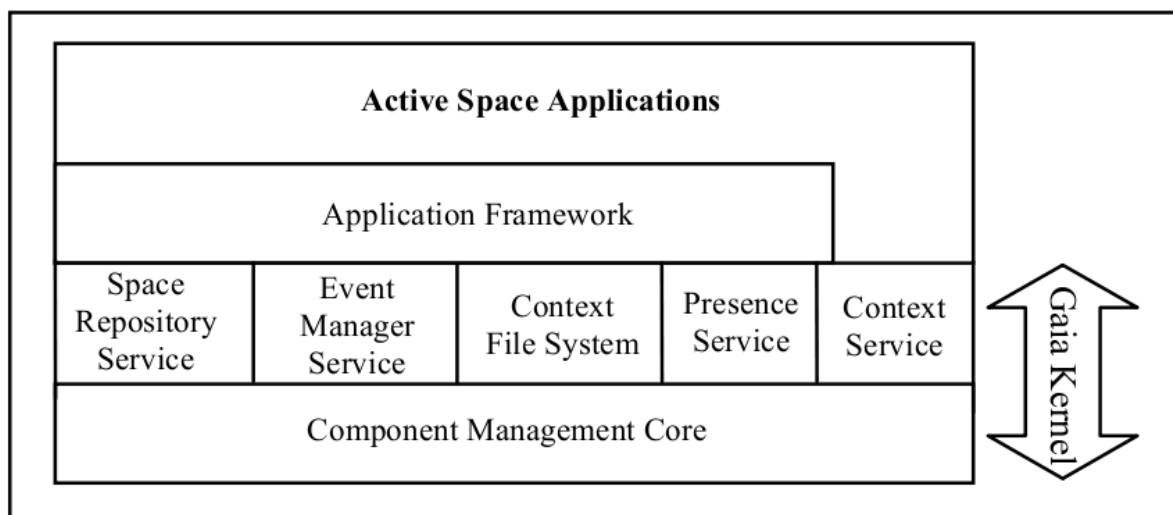


Figura 4 - Arquitetura Gaia. Fonte: (ROMÁN, 2002).

Contexto em Gaia

O contexto caracteriza um importante componente na arquitetura, sendo produzido e monitorado em diversos componentes deste. O Sistema de Eventos utiliza-se de um mecanismo de notificação de eventos monitorados, utilizando-se para isso o modelo de subscrição (*publish-subscribe*). As aplicações inscrevem-se como interessadas em um dado contexto e são notificadas quando o sistema perceber alguma modificação nos elementos monitorados. Com isso, promove-se um sistema capaz de oferecer tanto reatividade como pró-atividade ao usuário.

A arquitetura de contexto utiliza uma abordagem baseada em predicados lógicos de primeira ordem (RANGANATHAN, 2003), onde as condições de contexto são dadas por `TipoContexto(<sujeito>, <verbo>, <objeto>)`. O protocolo de comunicação entre contexto e aplicação é dado de duas formas:

a) através da requisição direta da aplicação a uma informação de contexto; nesse caso, a aplicação faz uma requisição em um formato semelhante às requisições em Prolog (ex: `Localização(usuario1, in, sala232)`);

b) através do modelo *publish-subscribe*, onde as informações são repassadas da camada de contexto à aplicação através da utilização de canais de comunicação; neste caso, quando uma aplicação necessita de um determinado dado ela deverá monitorar o canal associado com a informação desejada.

Os Provedores de Contexto, ou sistema de eventos são responsáveis pela disponibilização da informação nos canais.

Além das informações brutas de contexto, o sistema permite também a obtenção de informações de contexto agregadas a uma significação semântica deste, fornecendo assim um contexto de alto nível. O Sintetizador de Contexto é o componente responsável, nesta arquitetura, por promover a associação de um contexto de alto nível. Embora o contexto de alto nível esteja disponível, a aplicação não é obrigada a utilizá-lo, podendo acessar diretamente um Provedor, se assim o desejar.

Para abranger uma melhoria temporal no processo, a arquitetura também modela uma base de dados onde os históricos de contexto são armazenados, e sobre as quais são aplicados algoritmos de mineração de dados para obter regras, construídas dinamicamente, utilizadas no mecanismo de inferência de contexto. A Figura 5 ilustra os componentes da arquitetura de contexto do Gaia, as aplicações pervasivas são representadas através dos “*Context Consumers*”.

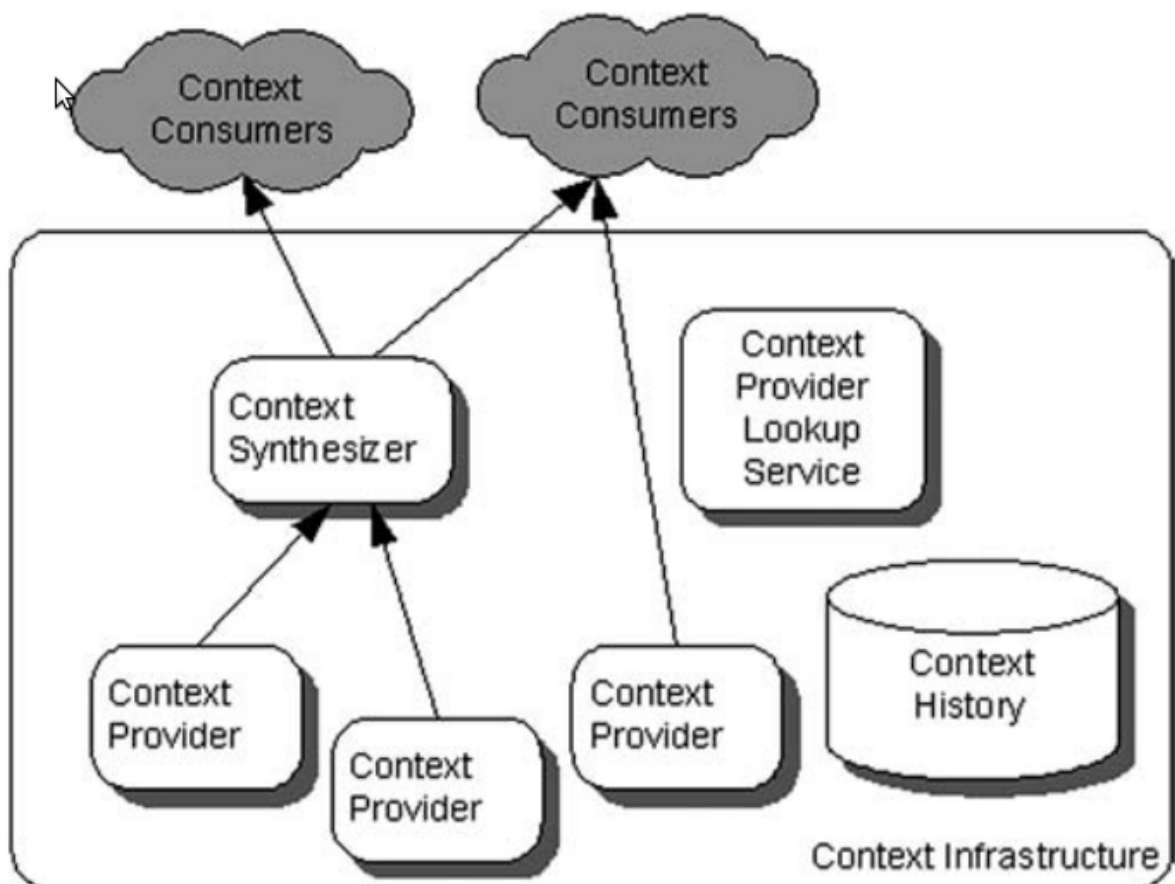


Figura 5 - Arquitetura de Contexto do Gaia. Fonte: (RANGANATHAN, 2003).

2.5.3.3. Projeto ABC (*Activity-based Computing*)

O projeto ABC (BARDRAM, 2007) foi desenvolvido na Dinamarca, através de uma parceria entre uma grande empresa atuante no desenvolvimento de sistemas EPR (*Electronic Patient Record*), um hospital universitário e o Centro de Computação Pervasiva para a Saúde (*Centre for Pervasive Healthcare*). ABC, além de basear-se na utilização das tarefas do usuário para modelar um sistema pervasivo, o faz levando em consideração os requisitos necessários para sistemas direcionados a cuidados clínicos.

Para identificar e levantar os requisitos necessários no ambiente clínico, em especial o ambiente hospitalar, foram realizados vários *workshops*, onde usuários clínicos realizaram a avaliação de vários cenários clínicos previamente construídos. Para o suporte à intensa dinâmica e mobilidade existente no meio clínico, a arquitetura ABC utiliza-se da Computação Baseada em Tarefas. O monitoramento do contexto corrente do usuário, em conjunto com modelos (previamente construídos) de tarefas comumente desenvolvidas nesse ambiente, possibilitam a descoberta das atividades realizadas pelo clínico, possibilitando ao sistema atuar de forma pró-ativa. Segundo os autores, esta abordagem minimiza o tempo gasto pelo usuário-final, principalmente no que tange à busca e execução automática de aplicações baseada na tarefa do usuário, evita que o usuário gaste tempo na navegação de complexos menus existentes em sistemas clínicos.

Visão geral da arquitetura

O projeto ABC desenvolveu a arquitetura exibida na Figura 6. A fim de dar suporte à mobilidade dos usuários, cada profissional deverá carregar um dispositivo de rádio frequência, com o qual ele realiza a autenticação no sistema quando se aproxima de um dos terminais computacionais disponíveis no ambiente. Após ser detectada a presença do usuário junto ao terminal, ele confirma que deseja se logar no sistema. Dessa forma, evita-se o problema de constantes autenticações, gerado pela utilização de vários terminais no decorrer do deslocamento. Quando uma autenticação é realizada, o usuário pode executar novas tarefas ou reativar uma tarefa anteriormente suspensa.

Através de *widgets* direcionados aos usuários, podem-se utilizar mecanismos síncronos ou assíncronos de comunicação, entre os profissionais de diferentes especialidades, permitindo o compartilhamento de informações e mecanismos colaborativos.

Contexto

Diferentemente das arquiteturas Aura e Gaia, a abordagem de contexto utilizada no projeto ABC é focada na questão da descoberta das atividades do usuário baseada no contexto em que ele se encontra e em modelos de atividades previamente conhecidos. ABC consiste em um sistema abstrato construído sobre arquiteturas que promovam a capacidade de captação e programação de aplicações conscientes de contexto, no caso, o JCAF (BARDRAM, 2005). Para permitir a descoberta de contexto, a arquitetura supõe a existência de um sistema de localização, onde pessoas e recursos são identificados através de um dispositivo de rádio frequência que os acompanha. Usuários autenticam-se no sistema através da “*Activity Bar*”, a qual, quando detecta a aproximação de um usuário, a um terminal, e oferece a autenticação deste através da confirmação de que o usuário quer utilizar o sistema. O sistema procura evitar as constantes autenticações utilizando nomes de usuário e senha, presentes em sistemas mais restritivos de autenticação. É através do componente *Activity bar* que o usuário interage com o sistema de sugestão de tarefas da arquitetura.

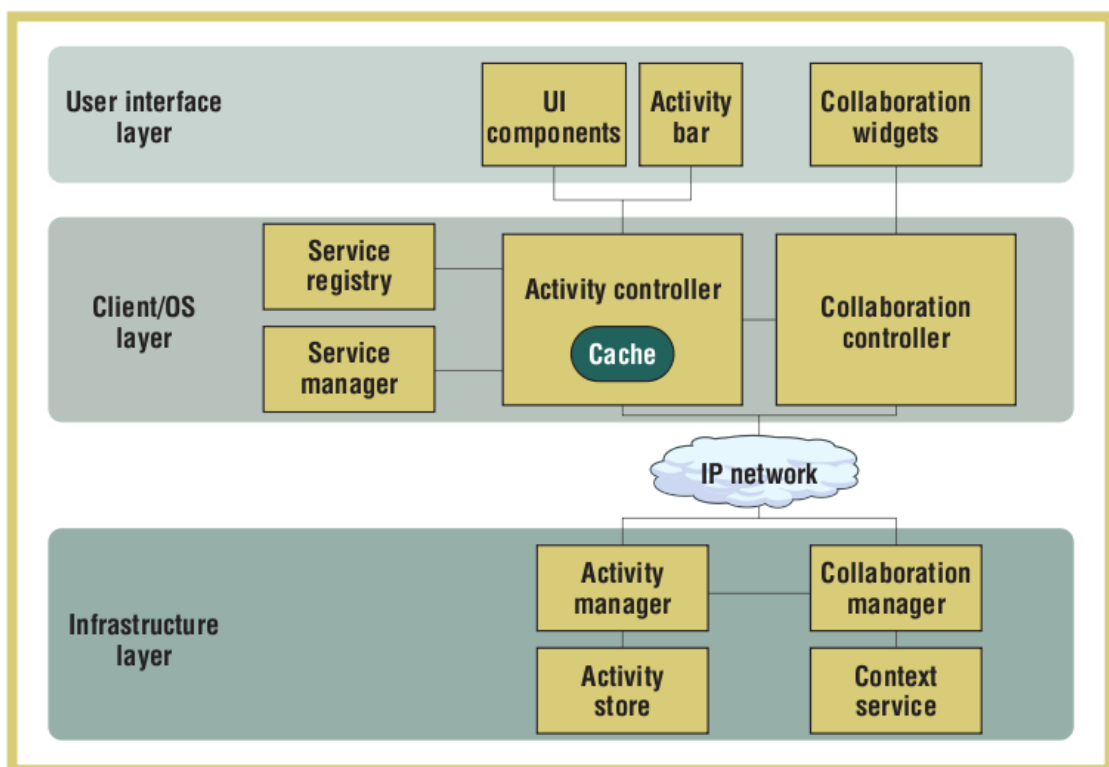


Figura 6 - Arquitetura projeto ABC. Fonte: (BARDRAM, 2007).

2.5.4. Comparativo entre os Projetos Relacionados

Os projetos Aura e Gaia foram uns dos primeiros a explorarem a questão de *middlewares* pervasivos, apresentando importantes contribuições na construção do *back-end* destes sistemas. O middleware EXEHDA, o qual será apresentado no capítulo 3, segue na mesma linha, porém promovendo uma infraestrutura pervasiva através de serviços disponibilizados em um ambiente de grade computacional.

Já o projeto ABC além de tratar do ambiente pervasivo, engloba o suporte as tarefas do usuário. A Tabela 1 sumariza as principais características destes sistemas, no que tange ao ambiente de contexto.

Pode observar-se que nenhum destes *middlewares* pervasivos abordagem a questão de personalização pelo usuário, nem tampouco fornecem mecanismos para entrada automática de dados.

Tabela 1 - Comparativo entre Projetos Relacionados.

	Ambiente Pervasivo			Suporte a tarefas	
	Mobilidade	Contexto	Contexto Assíncrono	Controle	Automação
Aura	X	X			X
Gaia	X	X	X		
EXEHDA	X	X	X		
ABC	X	X	X	X	X

Para usufruir dos benefícios da Computação Baseada em Atividades Humanas, primeiramente, devem-se identificar quais são as atividades clínicas envolvidas e como modelá-las computacionalmente. No próximo capítulo descreve-se o projeto ClinicSpace, escopo deste trabalho, em desenvolvimento no grupo de Sistemas de Computação Móvel (GMob) do PPGI/UFSM, que visa introduzir a Computação Orientada a Tarefas e ao usuário-final em um sistema de suporte às atividades clínicas.

3. PROJETO CLINICSPACE

Este capítulo resume as informações mais importantes para o entendimento do escopo dessa dissertação: o projeto ClinicSpace.

3.1. Apresentação do Projeto ClinicSpace

Os atuais sistemas de registro eletrônico da Saúde introduzem, em sua maioria, a visão organizacional/administrativa ou, mais recentemente, a visão do paciente (*Google Health*). Observa-se, no entanto, que a visão dos clínicos (profissionais da saúde) não é adequadamente atendida por ambos tipos de sistemas (VICENTINI, 2009).

Argumenta-se que o paradigma de computação atual, cuja computação é direcionada aos sistemas computacionais em si e não às atividades do usuário-final, é inadequado ao suporte aos clínicos. Por exemplo, em um sistema clínico tradicional, diversas informações (correspondente a especialidades) podem ser acessadas por diferentes softwares, assim como exames laboratoriais. Utilizando um sistema tradicional, o usuário gasta muito tempo procurando e executando diversas aplicações que venham a fornecer o conjunto de informações de que ele necessita (especialidade), além de causar uma sobrecarga com outras informações que não são necessárias naquele momento.

Como alternativa a esse antigo paradigma, pode-se optar por um sistema baseado na atividade do usuário (profissional), onde, de acordo com a atividade que ele está realizando, as informações são condensadas das diversas bases de dados, de forma transparente, em uma única aplicação. Com isso, é minimizado o tempo gasto pelo profissional na interação com o sistema computacional. Como consequência desse novo paradigma, o sistema computacional deve ser dotado de conhecimentos sobre as atividades executadas pelo usuário no seu campo de especialidade.

O projeto ClinicSpace visa explorar uma nova geração de sistemas clínicos pervasivos, onde a utilização da computação esteja presente de forma mais natural no decorrer das atividades humanas realizadas nessa área. Para que isso seja possível, além de introduzir elementos de Computação Ubíqua no sistema, deve-se buscar um equilíbrio entre a pró-

atividade disponibilizada pelo sistema e a manutenção do controle do sistema pelo usuário.

3.1.1. Requisitos da Arquitetura ClinicSpace

Nos sistemas clínicos atuais, os profissionais da saúde são os responsáveis pela captação e registro das informações do paciente no sistema. Isso demanda tempo do usuário na utilização do sistema informatizado e é apontado como um dos principais fatores da alta rejeição dos sistemas computacionais no ambiente clínico, em especial em ambientes onde a dinâmica é especialmente intensa, caso dos ambientes hospitalares (VICENTINI, 2009). Uma solução para esse problema é fornecer os mecanismos de contexto para monitoramento das informações desejadas, retirando essa tarefa do usuário e transferindo-a para o sistema clínico.

Como visto no resultado das pesquisas conduzidas por Bardram (2007), é importante manter no sistema a noção de controle por parte do usuário, evitando-se as implicações envolvidas em decisões equivocadas tomadas de forma automática, sem a aprovação do usuário. Bardram argumenta que o mecanismo mais adequado para prover a pró-atividade em conjunto com a perspectiva de controle do usuário é promover a sugestão de ações e/ou dados. Dessa forma, o sistema pode realizar os esforços de busca e/ou processamento necessário para obtenção destes recursos; porém, cabe ao usuário dar aval sobre as sugestões do sistema, corrigindo-as se necessário.

Um importante aspecto para a construção de um sistema pervasivo é dotar o sistema de mecanismos de interação que permitam a forma mais transparente de uso do sistema por parte do usuário. Embora as ações das pessoas, em geral, sejam delineadas por hábitos, estes são formados individualmente - um importante argumento na modelagem de um sistema personalizável pelo usuário, permitindo adequar o ambiente aos seus hábitos diários. Essa é uma característica que implica diretamente nas tarefas do usuário, onde uma mesma tarefa pode ser realizada utilizando diferentes recursos e/ou de diferente maneira por usuários distintos. Para permitir a personalização do ambiente, o projeto ClinicSpace propõe a utilização do paradigma de Computação Orientada a Tarefas.

Os termos tarefa e atividade, embora semanticamente semelhantes, são interpretados de forma diferente no projeto ClinicSpace. Uma tarefa consiste no conjunto de ações executadas colaborativamente por humanos e por sistemas de computação pervasivos para atingir um objetivo. Estas tarefas são compostas hierarquicamente, sendo as subtarefas

situadas no menor grau abstrato do conjunto. Tarefas correspondem a um *workflow* ou fluxo de execução de trabalho realizado pelo usuário na sua atividade profissional. Essa definição de tarefa é baseada na definição de Anand Ranganathan e Roy H. Campbell em “*Supporting Tasks in a Programmable Smart Home*” (RANGANATHAN, 2005).

Uma atividade, por outro lado, refere-se às atividades humanas propriamente ditas, e não às tarefas de apoio ao usuário, fornecidas por um sistema computacional. Portanto, sempre que este texto fizer referência a uma atividade, está de fato, referenciando a atividade humana que o usuário está realizando, ou seja, a atividade real. Já quando houver uma referência à tarefa, o texto está referenciando as operações ou tarefas computacionais executadas pelo usuário com a finalidade de busca, processamento e/ou registro das informações geradas de/e para a atividade real. No ambiente ClinicSpace, atividades clínicas, como atendimento a pacientes, são processos realizados por humanos de forma colaborativa, coordenada e distribuída em um espaço determinado. Para realizar tal atividade (atendimento ao paciente), o médico pode necessitar ver o histórico do paciente, usando suporte computacional (tarefa).

A utilização do conceito de tarefas fornece uma série de benefícios obtidos pela decomposição, recombinação, reuso e interrupção de tarefas, além da contextualização e programação destas pelo usuário. Uma recombinação de um mesmo conjunto de subtarefas de forma diferente acarreta na formação de uma nova tarefa.

O modelo ClinicSpace prevê associar à cada tarefa um contexto de tarefa, dinâmico e programável pelo usuário-final. A Teoria da Atividade foi utilizada para a modelagem da definição de tarefas através de uma ontologia¹ (LIBRELOTTO et al, 2008) e também para a modelagem de contexto.

Dessa forma, adotou-se a seguinte definição: as atividades realizadas por humanos podem ser suportadas por 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*).

¹ A ontologia para atividades clínicas está em fase de desenvolvimento.

3.1.2. Modelo de Tarefa ClinicSpace

Cada ambiente possui suas particularidades que devem ser observadas na modelagem de um ambiente pervasivo dedicado. No caso do projeto ClinicSpace, buscou-se o suporte fornecido por guias clínicos e estudos realizados junto a profissionais da saúde para nortear o desenvolvimento do sistema, em especial o trabalho desenvolvido por Hallvard Laerum e Arild Faxvaag (LAERUM; FAXVAAG, 2004), do qual extraiu-se o conjunto mínimo de tarefas (ver Apêndice A), suportadas na versão piloto da arquitetura.

As tarefas do usuário são executadas em um ambiente dinâmico, o que implica na necessidade de a arquitetura oferecer suporte a um conjunto de características, necessárias às tarefas, sendo:

- Decomposição, Recombinação e Reuso – tarefas são compostas por subtarefas reusáveis. Essas subtarefas podem ser re combinadas de diferentes maneiras para formar diferentes tarefas; como diferentes tarefas possuem subtarefas em comum, elas podem ser reutilizadas para desenvolver novas tarefas e assim reaproveitar a programação das subtarefas já implementadas;
- Interrupção (preemptáveis) – as tarefas podem ser interrompidas e retomadas mais tarde (semelhante ao modelo de co-rotina);
- Mobilidade e adaptabilidade – podem migrar e se adaptar para acompanhar o usuário (semântica siga-me);
- Contextualização – as tarefas são associadas a um contexto, o que pode ocorrer dinamicamente. O contexto, assim como as próprias tarefas, são elementos modularizados, programáveis pelo usuário, capazes de oferecer suporte à recombinação e reuso entre diferentes tarefas.

Como dito, no modelo ClinicSpace 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 que podem ser unidas usando um conceito similar a *workflow*. Tarefas são decompostas até unidades elementares: as subtarefas, as quais correspondem a aplicações do sistema clínico ou do sistema de gerenciamento do ambiente pervasivo fornecidas para o usuário. Diferentes tarefas podem ter subtarefas comuns ou similares – desta forma, o reuso de atividades já programadas é essencial.

Um exemplo de tarefa no ambiente clínico é “Atendimento a Pacientes” a qual poderia ser formada por subtarefas, as quais são associadas a aplicações computacionais, tais como “buscar informações específicas no prontuário do paciente”, “inserir informações sobre o atendimento no prontuário” e “prescrever um tratamento”. O uso dessas subtarefas ou a ordem com que são utilizadas podem variar de usuário para usuário.

3.1.3. Definições

As tarefas possuem um conjunto de informações que são de fundamental importância para a sua correta utilização e gerenciamento (SILVA, 2009). 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 de execução ela se encontra. Lembrando que uma tarefa obrigatoriamente encontra-se em um dos seis (Figura 7);
- Descrição: descrição textual da tarefa;
- Recursos: responsável por informar pré-condições de execução da tarefa e/ou de aplicações assistentes a ela;
- Código: as tarefas possuem um código onde se encontram as instruções de código que serão executadas para a obtenção das funcionalidades particulares de cada tarefa;
- Contexto de Tarefa: conjunto de elementos de contexto associados à tarefa, abrangendo elementos personalizáveis e de entrada automática de dados.

Cada tarefa deve assumir um único estado específico em um determinado momento de tempo. Seu ciclo de vida pode ser resumido na forma de uma máquina de estados, onde se identificou seis possíveis estados particulares. Estes são utilizados pelo sistema de gerenciamento de tarefas (FERREIRA, 2009), o qual trata a tarefa de acordo com os seguintes estados:

- Inicializada: criada e inicializada recentemente;
- Executando: executando no momento;

- Cancelada: tarefa estava em execução e foi cancelada;
- Pausada: tarefa foi interrompida e pode ser ativada (retomada a sua execução) a qualquer momento;
- Finalizada: tarefa foi finalizada por ter encerrado a sua execução.
- Encerrada: término do ciclo de vida de uma tarefa. Esse estado pode ser decorrente de uma tarefa finalizada ou cancelada.

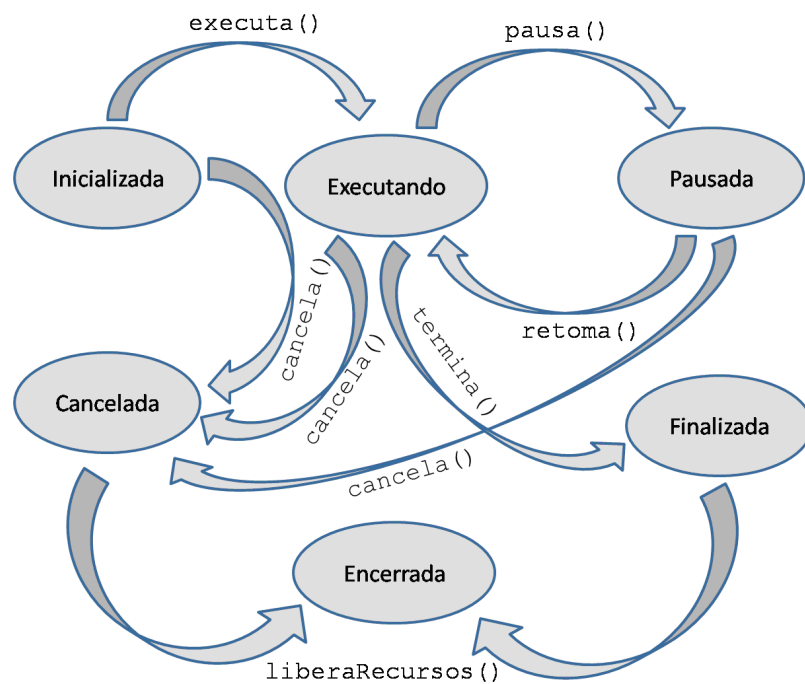


Figura 7 - Diagrama de Estados de uma Tarefa

Para simplificar o projeto do sistema que permite ao usuário final programar novas tarefas, a esse modelo de tarefa é associado 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 ClinicSpace, e até o momento não está totalmente concluída.

3.1.1.1. 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 (SILVA, 2009). 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 - modo padrão; (ii) compartilhada com profissionais da mesma especialidade; (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 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. 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 ignorá-la.

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

3.1.1.2. Conjunto mínimo de tarefas

O conjunto mínimo de tarefas, definido na primeira versão da arquitetura ClinicSpace, é baseado no estudo de Hallvard Laerum e Arild Faxvaag (LAERUM; FAXVAAG, 2004). No APÊNDICE A - Conjunto Mínimo de Tarefas 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: categoria composta por atividades relacionadas a métodos, procedimentos ou técnicas para determinar a natureza, ou identidade, da doença ou enfermidade. Não se inclui procedimentos realizados com amostras em laboratórios;

Tratamento: categoria composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para melhorar condições dos pacientes, e ainda outras quaisquer atividades para tratar doenças, enfermidades e ferimentos;

Laboratorial: categoria composta por atividades relacionadas a métodos, procedimentos ou técnicas utilizadas para determinar a composição, qualidade ou concentração de espécime. Essa categoria inclui ainda atividades que medem tempo e taxas das reações, e também as que se relacionam com o sistema Laboratorial para a obtenção de informações de exames, etc.;

Administrativa: categoria composta por atividades relacionadas a 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.

3.2. Contexto Associado às Tarefas

As subtarefas, além de terem a descrição ontológica (LIBRELOTTO et al, 2008), são implementadas diretamente como objetos Java, gerenciados pelo ambiente pervasivo. Estes correspondem a aplicações pervasivas relativas ao sistema clínico utilizado, ou a outras funcionalidades do *middleware* de gerenciamento do ambiente (FERREIRA, 2009).

Na descrição ontológica de cada subtarefa deve ser especificado o Contexto de Tarefa (CT), o qual é responsável por manter os contextos associados às tarefas, sejam eles dinâmicos ou estáticos. Os dados (informações de entrada em sistemas clínicos tradicionais) necessários a uma tarefa são mantidos de maneira estática, através de um *template* que descreve a estrutura semântica das informações de interesse da aplicação.

Esses dados são categorizados em dados obrigatórios para execução da tarefa e dados desejáveis, os primeiros correspondem a identificadores básicos, sem os quais o sistema de

contexto não é capaz de realizar qualquer esforço adicional na busca dos demais.

Outra questão também tratada pelo CT são os contextos personalizados pelo usuário e associados às tarefas, os quais permitem o disparo da tarefa quando os valores de gatilho, definidos pelo usuário, forem atingidos. O sistema de contexto é tema dessa dissertação e é abordado detalhadamente no capítulo 4.

3.3. Arquitetura para a programação e gerenciamento das tarefas

As camadas da arquitetura do ClinicSpace são mostradas na Figura 8, e estão organizadas 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.

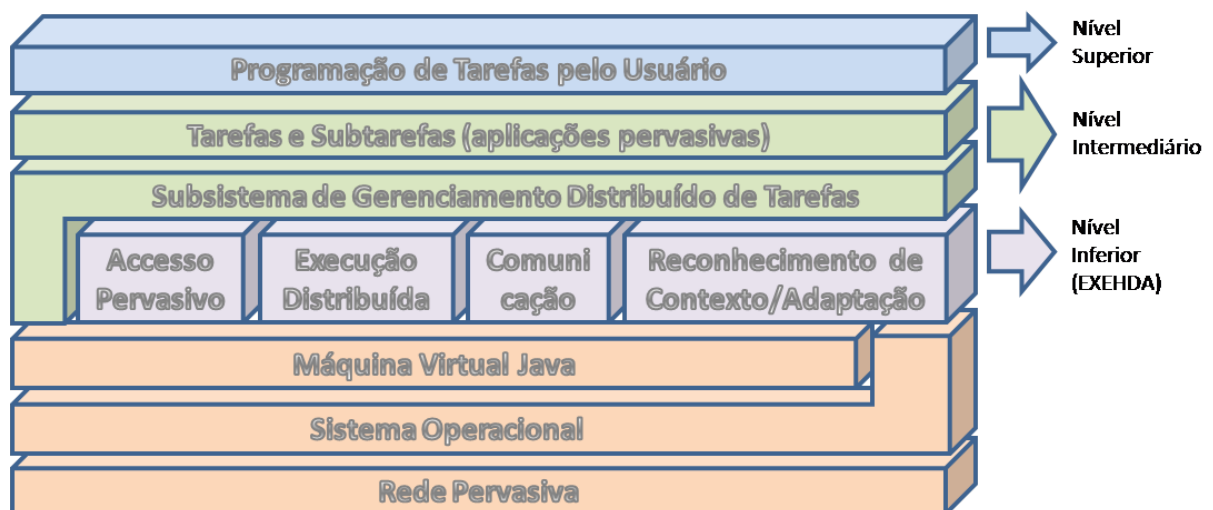


Figura 8 - Arquitetura ClinicSpace.

3.3.1. Componentes da Arquitetura ClinicSpace

A arquitetura da Figura 8 é genérica, para especializá-la ao ambiente clínico, inseriu-se outros componentes, ilustrados na Figura 9. 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*).

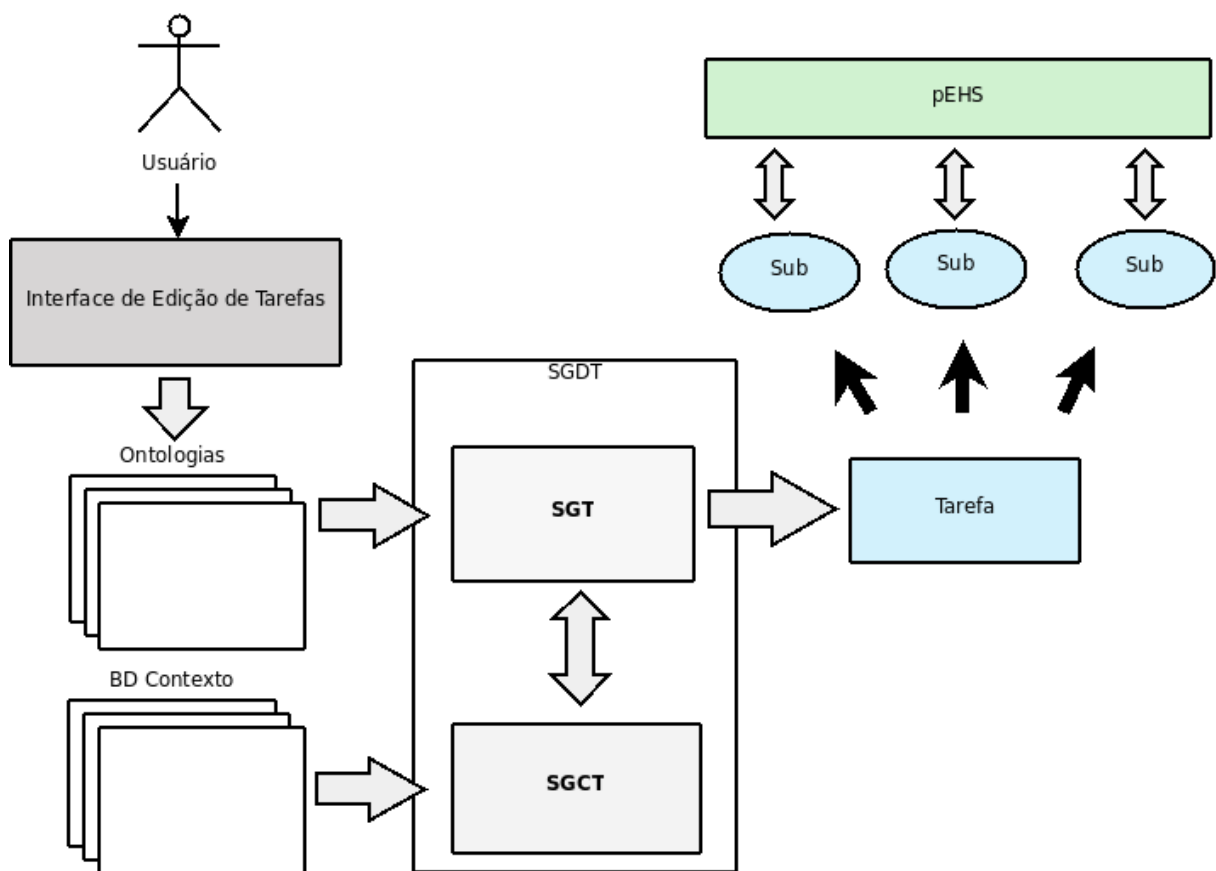


Figura 9 - Relacionamento entre Tarefas, Subtarefas, Interface de Edição de Tarefas (IET), SGT, SGCT e pEHS.

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 (fluxos de execução), a partir de suas próprias tarefas; e (iii) personalizar suas tarefas de acordo com suas necessidades, ligando elementos de contexto a elas.

Neste primeiro momento, supõem-se a existência de um sistema pervasivo de

informação de Saúde, denominado pEHS², 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.

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 (FERREIRA, 2009).

O 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.

Dentro do subsistema SGDT um novo serviço foi criado, o Serviço de Gerenciamento de Contexto de Tarefas (SGCT), o qual tem por finalidade permitir a personalização do contexto monitorado, na forma de mecanismos para entrada implícita de dados e mecanismos pró-ativos para execução de ações baseado na programação efetuada pelo usuário-final. Para que isso seja possível, foi necessária a modificação do subsistema de monitoramento do *middleware* EXEHDA, permitindo adotar o conceito de sensores personalizados.

3.3.2. Interação do usuário com o ambiente ClinicSpace

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

- i. 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.

² O pEHS está sendo modelado nos trabalhos de dissertação (em andamento) dos mestrandos Caroline Vicentini e Alencar Machado (PPGI - UFSM), também ligados ao projeto ClinicSpace.

- ii. 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.
- iii. O usuário pode trocar de dispositivo. 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, de gerenciamento de Tarefas (SGDT) 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.

O usuário interage com a arquitetura ClinicSpace através da interface de edição de tarefas (SILVA, 2009), na qual ele pode editar tarefas e associar contextos a elas. As informações sobre as personalizações realizadas pelo usuário são armazenadas nas ontologias de tarefas e no Banco de Dados de Contexto, para tarefas e contextos, respectivamente. Futuramente, o contexto também deverá ser armazenado através de uma ontologia apropriada. As informações da base de dados de contexto e das ontologias de tarefas são recuperadas pelo Subsistema de Gerenciamento Distribuído de Tarefas (SGDT), o qual é um subsistema construído no *middleware* pervasivo EXEHDA.

Dentro do subsistema de programação e gerenciamento de tarefas há dois serviços principais: (i) serviço de gerenciamento de tarefas (FERREIRA, 2009), o qual é responsável por mapear as tarefas editadas pelo usuário para serviços disponíveis no sistema pervasivo de cuidados clínicos (pEHS). (ii) serviço de gerenciamento de Contexto de Tarefas (SGCT) o qual é responsável por realizar o mapeamento dos dados necessários pelas aplicações mapeadas para realização de uma tarefa, bem como o gerenciamento dos contextos personalizáveis pelo usuário (atuadores) mapeados para a tarefa do usuário. A Figura 9 ilustra a relação existente entre os componentes da arquitetura.

Este trabalho aborda o Serviço de Gerenciamento do Contexto de Tarefas (SGCT), e

os elementos necessários para realizar o mapeamento dos dados necessários pelas aplicações e os contextos associados à elas, sendo esse assunto tratado de forma detalhada no capítulo 4.

3.4. Middleware EXEHDA

O EXEHDA (*Environment Execution for High Distributed Applications*) (YAMIN, 2004) é um *middleware* para suporte a sistemas pervasivos, desenvolvido dentro da arquitetura ISAM (ISAM, 2001) (AUGUSTIN, 2004). O principal objetivo é disponibilizar o ambiente pervasivo, atuando como o mecanismo base, promotor da migração, disponibilidade e organização da arquitetura como um todo. Esse *middleware* foi adotado para prover o ambiente computacional pervasivo da arquitetura ClinicSpace.

O núcleo do *middleware* é formado por serviços, os quais podem interagir entre si fornecendo os requisitos pervasivos de que as aplicações necessitam. Como o ambiente é organizado em termos de serviços, é possível adaptá-lo ou modificá-lo conforme necessário, através da adição de novos serviços e/ou modificação dos serviços existentes, fornecendo desta forma, flexibilidade.

A arquitetura do *middleware* é organizada na forma de células, as quais são representadas por um conjunto de nodos, podendo estes serem móveis. As células são vistas como unidades fornecedoras de serviço, os quais podem ser disponibilizados pelos nodos constituintes da célula. Cada base é organizada em função de um nodo principal, chamado de EXEHDAbase, o qual é responsável pelo gerenciamento dos serviços nela disponibilizados. A Figura 10 apresenta uma imagem que demonstra um exemplo de organização do ambiente pervasivo.

Alguns dos principais serviços providos pela arquitetura são: serviço de reconhecimento de contexto (fornecido na forma de *framework*), monitoramento, distribuição, adaptação, entre outros. Maiores detalhes sobre o *middleware* EXEHDA serão apresentados no APÊNDICE B – Middleware EXEHDA.

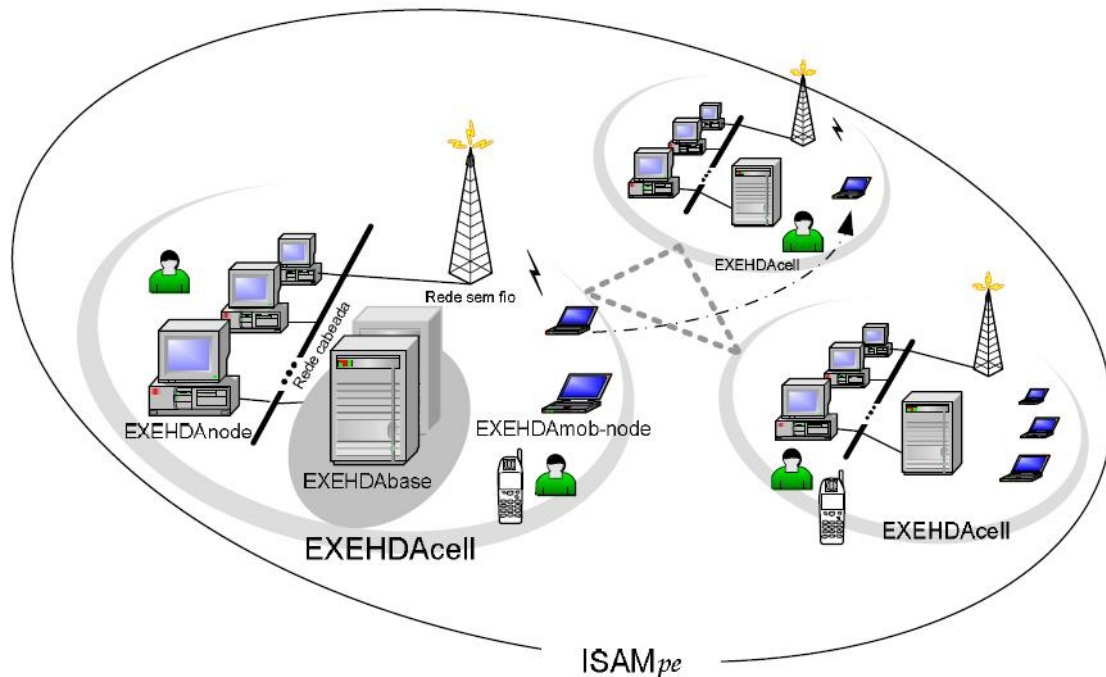


Figura 10 - Organização celular da arquitetura ISAM. Fonte: (AUGUSTIN, 2004).

3.5. Resultados esperados pelo projeto ClinicSpace

O projeto ClinicSpace está desenvolvendo uma arquitetura que visa reduzir o *gap* entre os atuais sistemas clínicos e os profissionais de Saúde, contribuindo para diminuir a rejeição dos atuais sistemas por esses profissionais, que alegam que esses demandam muito tempo para o seu uso. Ao final do projeto, espera-se ter um protótipo capaz de oferecer inovações, com a introdução de conceitos da Computação Ubíqua e Computação Orientada a Contexto, e mecanismos que permitam absorver a principal vantagem do paradigma de sistemas centrados nas atividades humanas: redução da sobrecarga colocada no usuário que usa o sistema.

Como já mencionado, a meta inicial do sistema ClinicSpace, para construção do protótipo de validação, é fixar o foco no usuário médico, ou seja, as tarefas são realizadas sob a perspectiva que o médico tem do sistema, onde, por exemplo, não é relevante saber como os exames são efetuados, mas sim conhecer seus resultados.

Esse usuário (médico) utilizará instrumentos de mediação para realizar suas atividades, entre tais ferramentas está um sistema pEHR que é acoplado ao sistema ClinicSpace, onde estão disponíveis todas informações sobre registros de saúde do paciente. Inúmeros outros recursos são utilizados pelo médico para realização da sua tarefa, em grande

parte esses recursos são físicos, como equipamentos médicos e a sala onde a consulta ou o procedimento é realizado.

A utilização desses recursos, normalmente, é vinculada a regras ou guias clínicos, que em certo nível pode ser individualizada pelo usuário através da personalização de tarefas. O usuário poderá alterar o fluxo de execução dentro de uma tarefa, obedecendo a certos limites de validação impostos pela arquitetura, dessa forma, ele estará indiretamente fornecendo uma regra para execução daquela atividade. Além disso, a utilização de uma terminologia clínica também atua como uma regra, estabelecendo nomes para procedimentos e/ou dados, além de faixas válidas. Sendo, inclusive, através da utilização destas faixas válidas, possível realizar a validação de dados, principalmente no que tange a dados obtidos pelo sistema de contexto.

Sabe-se que há uma diversidade de profissionais atuando no meio clínico, médicos de diversas especialidades, enfermeiros, farmacêuticos, entre outros, logo, o conjunto desses profissionais compõe a comunidade na qual a tarefa está sendo desenvolvida, os quais atuam de forma colaborativa para execução dela. Essa diversidade trás várias visões distintas em relação à atividade, sendo que a modelagem de tarefas, embora utilizando o mesmo esqueleto comum, possui diferenças significativas, ou seja, as subtarefas (serviços EXEHDA) executadas por cada profissional são bastante heterogêneas, cada qual exigindo um conjunto específico de subtarefas. Na versão inicial, o ClinicSpace modela as atividades clínicas, mais comuns, executadas pelo usuário médico. Embora sua arquitetura e conceitos possam ser aplicados para todos os usuários, ou seja, seus pontos de vista da tarefa.

No próximo capítulo é detalhada a arquitetura que adiciona contexto às tarefas clínicas definidas no projeto ClinicSpace, de forma a capturar automaticamente as informações do ambiente e introduzi-las no sistema como uma forma de entrada automática de dados.

4. GERENCIADOR DE CONTEXTO DE TAREFAS - SGCT

O projeto ClinicSpace, conforme discutido anteriormente, visa construir uma arquitetura para facilitar a execução de tarefas computacionais no meio clínico. Para isso, utiliza-se do paradigma de computação orientada à tarefa (*Task Computing*), com forte foco na questão de personalização de tarefas pelo usuário médico. Um dos aspectos-chaves do ambiente pervasivo é fornecer a capacidade do ambiente atuar como fornecedor de informações para a arquitetura, liberando o usuário da entrada explícita de dados tanto quanto possível, através da obtenção destes por um sistema de monitoramento, o qual pode ser implementado através de sensores físicos ou lógicos que permitem ao sistema obter os dados de que necessita.

Esse capítulo detalha o Sistema Gerenciador de Contexto de Tarefas (SGCT) projetado dentro da Arquitetura ClinicSpace.

4.1. Tratamento de Contexto no ClinicSpace

Uma aplicação clínica, usualmente, é composta por um conjunto de atributos que conferem quantização a uma determinada característica de uma entidade, seja esta um paciente, equipamento, recurso ou usuário. O tamanho desse conjunto de dados varia de aplicação para aplicação, mas pode-se afirmar que na maioria das vezes o usuário demora um tempo considerável para interagir com o sistema realizando essa atividade (KOCH, 2007).

A modelagem de contexto desenvolvida neste trabalho, foi concebida de forma a atender a **demanda de entrada implícita de dados**, fornecendo mecanismos para personalização do contexto pelo usuário final. É importante ressaltar que se está tratando de tarefas computacionais, ou seja, a modelagem das tarefas que o usuário terá que realizar no sistema computacional para registro/consulta/controle das atividades reais por ele executadas. O conhecimento sobre como as atividades humanas são desenvolvidas é um importante ponto para construir um modelo válido e funcional, o qual pode ser obtido através do suporte oferecido pela teoria da atividade (NARDI, 1996) (KOFOD-PETERSEN, 2006).

4.1.1. Contexto das Tarefas

A tarefa computacional está intrinsecamente ligada às atividades desenvolvidas, especificando recursos e procedimentos nela utilizados, além dos dados utilizados no registro da tarefa. Estas informações fazem parte do Contexto da Tarefa Computacional (CT), através do qual é possível efetuar o registro automático destas informações no sistema. Sempre que o usuário realizar a tarefa, as informações do contexto a ela associado poderão ser utilizadas para fornecer os mecanismos necessários para entrada de dados implícita.

As informações de contexto associadas a uma tarefa constituem-se basicamente dos dados associados aos campos de formulários correspondentes à aplicação clínica, bem como dados para registros automáticos associados à tarefa, tal como o registro de procedimentos e recursos necessários durante a execução da tarefa, porém de forma condicional à ocorrência de uma situação específica de contexto.

Pode-se perceber que essa associação é bastante dinâmica, mudando de tarefa para tarefa ou de um usuário para outro em uma mesma tarefa. O Serviço de Gerenciamento de Contexto de Tarefa (SGCT) resolve essa questão através da utilização de uma interface personalizável pelo usuário, para especificação do contexto associado à tarefa.

4.1.2. Funcionalidades

O contexto associado a uma tarefa é dinâmico, e é permitido ao usuário, através de uma interface de programação amigável, especificar quais dos elementos de contexto existentes no sistema que devem ser associados à tarefa.

Um Elemento Personalizável de Contexto (EPC) pode ser visto como um atuador, visto que, através da identificação de um contexto, uma ação é realizada, seja ela um registro no sistema, ou mesmo o disparo de uma aplicação ou processo. Esses elementos possibilitam a criação de tarefas que possuem um comportamento dinâmico, reagindo às mudanças de contexto programadas pelo usuário.

O usuário interage com o sistema através da interface de edição de tarefas, nessa interface ele escolhe as tarefas que irá disparar, associa elementos de contexto disponíveis às tarefas e, edita os fluxos de execução na composição de tarefas. Quando uma tarefa é

disparada, é acionado o serviço de gerenciamento de tarefas (SGT) do EXEHDA (FERREIRA, 2009), o qual identifica os serviços necessários para execução do fluxo requisitado pelo usuário. Esses serviços constituem-se nas subtarefas e, usualmente são mapeados para aplicações clínicas do sistema pEHS³, no qual estão disponíveis os formulários e aplicações do sistema clínico.

Não obstante, essas funcionalidades devem ser disponibilizadas no ambiente pervasivo, que entre outras características deve oferecer suporte à mobilidade, disponibilidade e escalabilidade. Para oferecer tais características, buscou-se suporte no *middleware* EXEHDA (YAMIN, 2004). Porém, entre seus requisitos de modelagem não estava previsto o suporte ao conceito de Computação Orientada a Tarefas; portanto, demandou-se projetar extensões e alterações no *middleware*. Este trabalho trata exclusivamente das alterações pertinentes às questões de contexto e a disponibilidade do contexto da tarefa. Para isso, serviços existentes no *middleware* tiveram de ser alterados, assim como novos serviços criados. As alterações efetuadas serão discutidas na seção referente à implementação do modelo (seção 4.3).

4.2. Modelo Proposto

O modelo proposto para realizar a entrada de dados de forma implícita pode ser dividido em duas questões principais: a) Entrada automática de Dados para os campos do formulário de uma aplicação e; b) elementos personalizáveis de contexto, os quais são associados às tarefas pelo usuário final, que determina os valores para os quais o atuador (EPC) deverá ser disparado, realizando a ação à qual ele se propõe.

Nas seções a seguir são discutidas essas duas funcionalidades.

4.2.1. Entrada automática de Dados para a aplicação

As aplicações clínicas são compostas por vários dados, normalmente agregados em função de uma perspectiva, como visualização de histórico do paciente, requisição de novos exames, etc. O conjunto destes dados é variado sendo intrinsecamente atrelado ao propósito

³ Tema de Dissertação da mestranda Caroline Vicentini.

da aplicação clínica que os contém.

Um dos objetivos do SGCT é permitir que o máximo de dados de entrada sejam obtidos através do sistema de reconhecimento de contexto, liberando o usuário tanto quanto possível da entrada de dados. Além de possibilitar maior eficiência na interação entre usuário e sistema, culminando em uma economia de tempo, a obtenção automática de dados facilita a acurácia destes, evitando erros de digitação ou omissão de informações nos registros clínicos, o que pode ocorrer devido a interfaces mal projetadas, onde o usuário é obrigado a gastar muito do seu tempo navegando em menus e/ou digitando dados de forma repetitiva (KOCH, 2007). Para que esse processo seja transferido para o sistema de contexto há diversos desafios a serem vencidos, entre eles a modelagem de como esses dados devem ser definidos em cada aplicação.

Visando prover essa automatização, torna-se necessário que o sistema de contexto conheça quais são os dados de que a aplicação necessita, para que possa monitorá-los. Na maioria das vezes, não basta efetuar o monitoramento dos dados brutos de contexto, é necessário agregar uma significação semântica a eles, baseado em faixas de valores válidos, nomenclaturas clínicas ou outras informações que agregam valor semântico à informação.

Para tornar isso possível, utilizou-se o conceito de arquétipos, proposto pela arquitetura OpenEHR (BEALE, 2002), o qual consiste em uma estrutura que permite especificar a semântica de uma determinada informação através da especificação de faixas de valores válidos e estados baseados em faixas de valores, por exemplo. O conjunto de dados necessários para uma aplicação é especificado através de um *template*, que consiste na agregação de um conjunto de arquétipos com vistas a formar uma visão destes, diretamente relacionada a um formulário clínico. A Figura 11 ilustra o relacionamento existente, entre a aplicação, arquétipos e *templates*.

A principal vantagem na utilização dos arquétipos e *templates* é promover uma arquitetura logicamente dividida em duas camadas: a) a primeira representando os dados computacionais brutos, o que é tradicionalmente feito através da programação de estruturas, utilizando tipos básicos de dados, em uma linguagem de programação; b) e outra camada representando a descrição semântica da informação.

Esse modelo oferece robustez e extensibilidade, visto que alterações na estrutura semântica de uma informação, tal como estados baseados em faixas válidas, podem ser alterados ao longo do tempo sem a necessidade de programação ou recompilação de aplicações. Para alterar essas informações, um usuário especialista, com conhecimento sobre a construção de arquétipos e da linguagem empregada em sua descrição, deve acessar a base de

dados e alterá-lo. Não são necessárias intervenções extras, pois a arquitetura do sistema automaticamente utiliza as novas informações estabelecidas em execuções futuras.

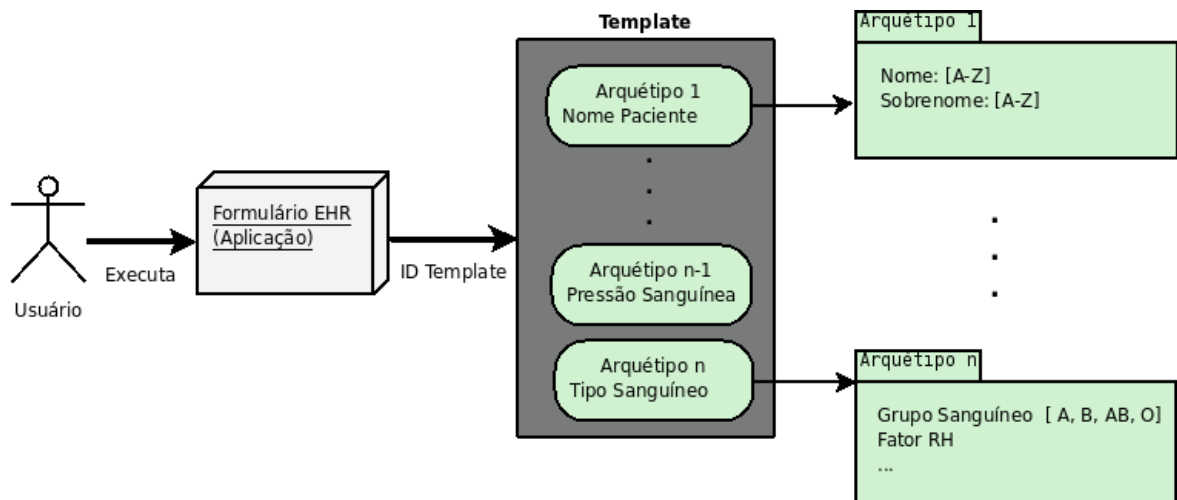


Figura 11 - Relação entre Aplicação, *Templates* e Arquétipos.

A abstração fornecida por esse modelo oferece a flexibilidade de utilização de terminologias clínicas variadas, visto que o sistema atua em função do arquétipo e não de uma terminologia específica, oferecendo dessa forma flexibilidade também quanto ao sistema pEHS utilizado. A arquitetura de contexto foi concebida de forma a ser independente do protocolo utilizado para comunicação entre o sistema pEHS e os demais componentes da arquitetura ClinicSpace, não assumindo suposições sobre o protocolo utilizado, nem sobre a terminologia utilizada para representar as informações clínicas. Essas questões estão sendo tratadas por outros trabalhos, ainda em desenvolvimento, relacionados ao projeto ClinicSpace⁴.

Os serviços responsáveis pelo mapeamento das aplicação clínicas do sistema pEHS para o meio clínico, são os responsáveis por obter, junto a aplicação, quais são os dados por ela necessários, através da identificação do *template* que os representa. Esse *template* tem por finalidade especificar todos os arquétipos, os quais contêm a descrição semântica dos dados necessários pela aplicação pEHS vinculada à sub tarefa. Dessa forma, quando uma sub tarefa é inicializada, deverá notificar o SGCT, informando também qual o identificador do *template* utilizado. Com isso, é possível a busca aos dados necessários para a construção da

⁴ Assuntos abordados como tema de dissertação de Caroline Vicentini e Avelar Machado.

especificação de contexto da tarefa, especificando filtros, estados, e valores padrões para os elementos de contexto.

O serviço de gerenciamento de contexto de tarefas, quando notificado sobre a inicialização de uma tarefa, efetua a busca no Banco de Dados de Contexto baseado no identificador de *template* informado pela aplicação. Este, por sua vez, deverá fornecer uma lista, em formato XML, contendo as classes de dados necessárias, bem como os filtros e estados de contexto por ela necessários. Cada elemento de dado deve ser representado por uma string XML, cujas regras de construção são expressas através a linguagem XML Schema (<http://www.w3.org/XML/Schema>). Maiores informações estão disponíveis no APÊNDICE C – Definição XML Schema para uma classe de dado. Um exemplo desse formato é exibido no Quadro 1, modelando um contexto associado a um sensor de temperatura.

```

<DATACLASS n='Temperature'>
  <STATES>
    <STATE n='hot' />
    <STATE n='nice' />
    <STATE n='cold' />
  </STATES>
  <FILTER type='NumericRangeMapping'>
    <MAPPINGS>
      <RANGE ub='15' state='cold' />
      <RANGE lbo='16' ub='24' state='nice' />
      <RANGE lbo='25' state='hot' />
      <DEFAULT state='nice' />
    </MAPPINGS>
  </FILTER>
</DATACLASS>

```

Quadro 1 - Exemplo da especificação de um dado de entrada para o SGCT.

O formato é derivado da especificação XML tratada pelo serviço de contexto incluído na arquitetura EXEHDA (SILVA, 2003), porém alguns elementos foram retirados, como a especificação do sensor. Essa informação deverá ser buscada e tratada pelo sistema de gerenciamento de tarefas junto ao serviço de monitoramento do EXEHDA, sendo responsabilidade do SGCT realizar as adaptações necessárias para que essa especificação seja compatível com o sistema de contexto base utilizado.

Baseado no valor informado no atributo da *tag* “*DataClass*”, o sistema de contexto de tarefa (SGCT) busca um sensor, junto ao serviço de monitoramento, que possa fornecer essa informação. Se o sensor for encontrado, é criada uma instância dele para a aplicação, sendo

registrado um contexto baseado nas informações obtidas junto ao banco de contextos. Se não for encontrado um sensor compatível com o dado requisitado, o monitoramento não é realizado. Dentro da arquitetura, isso não é visto como um problema, visto que a proposta é tanto quanto possível monitorar o ambiente para fornecer a entrada implícita de dados, porém, sem a obrigatoriedade de fornecer tais informações. Dessa forma, segue-se a definição de contexto proposta no projeto ISAM, a qual define como contexto “toda informação relevante para a aplicação que pode ser obtida por ela” (AUGUSTIN, 2004).

No momento em que o gerenciador de tarefas (SGTD) requisitar a criação de um contexto para a tarefa, através do SGCT, deverá ser repassada também uma interface de retorno para notificação dos dados monitorados, o que é implementado sobre a forma de uma *callback* ou um canal de comunicação. Os dados são monitorados pelo sistema de gerenciamento e notificados à aplicação pEHS sempre que algum dos dados monitorados sofrer alteração de estado. Essa ligação entre o SGCT e o pEHS é realizada através do serviço de gerenciamento de tarefas (SGT), visto que ele é o componente responsável pela interface entre os demais serviços do EXEHDA e o sistema pEHS (FERREIRA, 2009).

Embora o monitoramento de contexto seja um recurso útil para reduzir a entrada de dados, sua função é sugerir valores para os dados desejados, e não o de registrá-los sem o consentimento do usuário. Um sistema de contexto pode eventualmente sugerir valores errôneos, que se registrados de forma automática, sem o conhecimento do usuário, poderiam trazer consequências indesejáveis, prejudicando sua atividade. Dessa forma, cabe ao usuário validar os dados monitorados pelo sistema de contexto, mesmo que essa validação seja uma ação global, como pressionar o botão “salvar” em um formulário contendo diversos dados monitorados. Com isso, mantém-se o controle do sistema nas mãos do usuário, ao mesmo tempo em que se fornecem mecanismos pró-ativos na descoberta de dados. Maiores detalhes de como essas características foram integradas ao EXEHDA são descritos nas seções a seguir.

4.2.2. Personalização do Contexto de Tarefa

Segundo os conceitos propostos pela Teoria da Atividade, uma atividade é composta por um usuário que realiza um conjunto de ações, muitas vezes com o auxílio de ferramentas, norteado a atingir um objetivo (NARDI, 1996) (KOFOD-PETERSEN, 2006). Uma mesma tarefa pode possuir uma composição completamente diferente, baseado na ótica de cada

usuário envolvido na sua realização. No projeto ClinicSpace, projetou-se a composição e gerenciamento de tarefas baseados na visão do usuário clínico, ou seja, o médico. Sob essa ótica, na realização de uma tarefa, algumas vezes, o médico pode executar procedimentos que, embora não façam parte da tarefa regular, podem vir a ser necessários em algumas situações específicas. O sistema de contexto de tarefas pode agir pró-ativamente nesse caso, disponibilizando elementos de contexto, que o usuário possa associar à tarefa e definir valores de gatilhos para seu disparo, sendo executado somente quando o sistema de monitoramento de contexto identificar os valores definidos pelo usuário para sua ativação.

Esse conceito é associado ao contexto de atuadores, que podem ser ligados às tarefas, mas executados de forma dinâmica, somente na ocorrência dos valores programados pelo usuário. Com isso, ClinicSpace vai além de permitir ao usuário personalizar as etapas ou ações de sua tarefa, através da Interface de Edição de Tarefas (SILVA, 2009), possibilitando também especificar um contexto para a tarefa, o que torna possível a personalização de eventos para disparo de tarefas no sistema.

Essa capacidade torna o sistema mais flexível, fornecendo suporte para o usuário programar fluxos de execução lineares e não-lineares, representados pelo fluxo de execução normal de uma tarefa e o fluxo condicionado a uma situação de contexto personalizada, respectivamente.

O sistema foi modelado de forma a permitir que atuadores sejam construídos para fornecer elementos e contexto associáveis às tarefas pelo usuário. Cabe salientar que um atuador no sistema é uma entidade genérica, estabelecida via programação, onde parâmetros de execução e gatilhos (*triggers*) são especificados.

A abstração sobre a qual os atuadores são construídos permite que estes sejam tanto implementados sobre a forma de um atuador físico, quanto sob a forma de um atuador lógico. Efetivamente, o que motivou o suporte aos atuadores é a capacidade de construir mecanismos de disparos de tarefas programados pelo usuário, sejam eles mecanismos globais, associados a uma data e hora, ou mecanismos verificados no escopo da tarefa, como uma situação específica de contexto encontrada no decorrer da tarefa e que é responsável pelo disparo do atuador.

Essa perspectiva permite a construção de mecanismos para suporte a procedimentos comuns nas atividades cotidianas do usuário, tal como programar o agendamento, para disparo automático de tarefas. Outro possível uso está na implementação do mecanismo de registro automático de dados da tarefa, quando situações específicas ocorrerem, fornecendo, dessa forma, outro mecanismo de entrada implícita de dados baseado no contexto da tarefa.

Com isso, adiciona-se um novo mecanismo de pró-atividade no sistema, baseado no monitoramento de elementos de contexto. Inclusive, vê-se como uma possibilidade a expansão deste conceito para dar suporte a mecanismos de inferência de contexto. Porém, não é objetivo deste trabalho aprofundar-se no tema de inferência, esse assunto está sendo estudado e é tema de outra dissertação⁵ desenvolvida também dentro do escopo do projeto ClinicSpace.

Os atuadores são disponibilizados ao usuário pela Interface de Edição de Tarefas, através da guia “Elementos de Contexto”. A associação destes à tarefa, bem como a parametrização para sua execução é realizada pelo usuário clínico, fazendo uso dos atuadores implementados no sistema, em fase de desenvolvimento.

Quando um atuador é associado a uma tarefa, é exibida uma janela ao usuário questionando sobre os valores desejados para o disparo do mecanismo. Os mecanismos *dnd* (*drag and drop*) utilizados na Interface de Edição de Tarefas (IET) estão disponíveis também aos atuadores, uma discussão completa sobre a IET pode ser encontrada em (SILVA, 2009). A Figura 12 ilustra um exemplo simples desse ambiente, mostrando os elementos de contextos disponíveis no sistema para o usuário (observe que não há elementos de contexto ligados à tarefa).

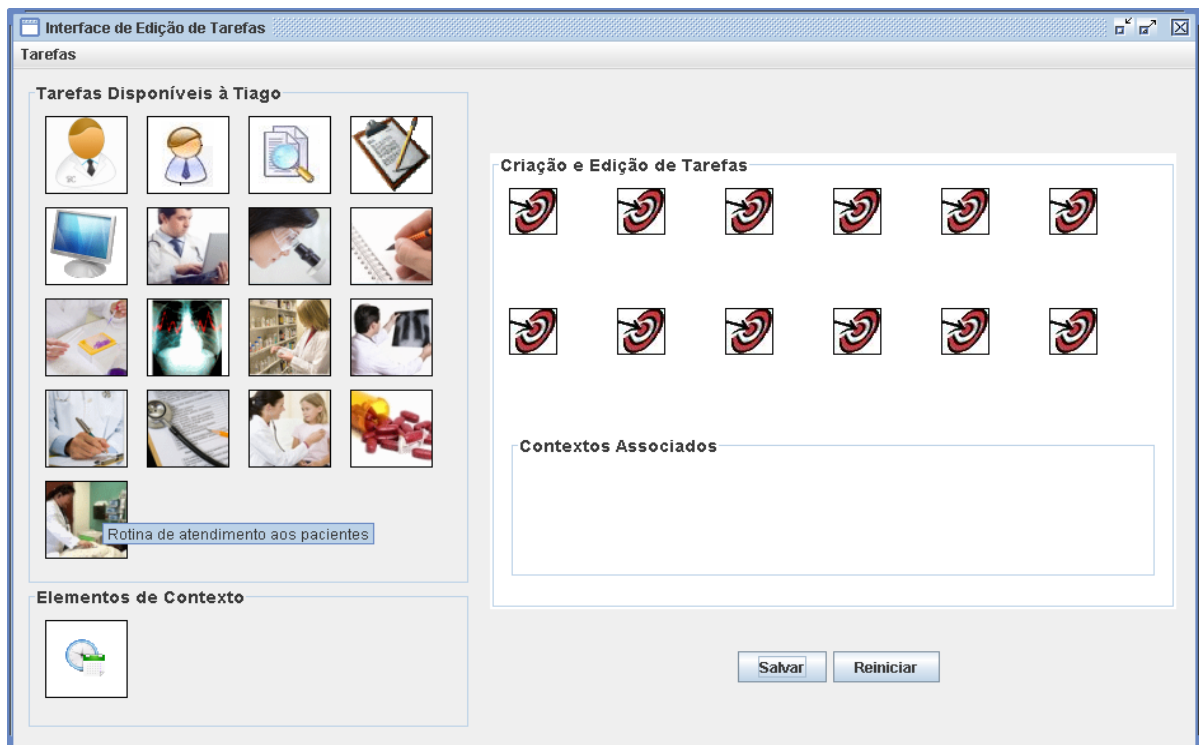
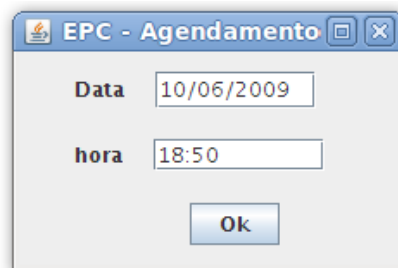


Figura 12 - Interface de Personalização do Contexto da Tarefa - Uso dos Atuadores

⁵ Tema de Dissertação de Marcos Vinícius B. de Souza

Como se pode observar, nesse exemplo, um elemento de contexto para agendamento de tarefas é disponibilizado ao usuário, sendo previamente definido via programação. O usuário, pode arrastar esse elemento sobre a área de contextos associados à tarefa, representado na seção direita da tela. Nesse instante, o sistema busca junto ao SGCT os parâmetros necessários pelo Elemento Personalizável de Contexto (EPC) e mostra ao usuário a tela requisitando esses valores, conforme exemplo mostrado na Figura 13.



The image shows a small dialog box titled "EPC - Agendamento". It has a title bar with standard window controls. Inside, there are two text input fields. The first is labeled "Data" and contains the text "10/06/2009". The second is labeled "hora" and contains the text "18:50". Below these fields is a single button labeled "Ok".

Figura 13 - Dados do Atuador

Logo em seguida, juntamente com as informações sobre o fluxo interno da tarefa é mostrado o novo contexto associado à tarefa. Esse elemento permite múltiplas associações à tarefa, representando diferentes instâncias de agendamento. A Figura 14 ilustra essa operação.



Figura 14 - Interface de Personalização do Contexto da Tarefa – Atuador Agendamento Associado à tarefa.

A arquitetura resultante dos componentes necessários para entrada automática de dados e elementos personalizáveis de contexto (atuadores) é mostrada na Figura 15. O usuário interage através da edição de tarefas, o qual realiza chamada a métodos disponibilizados pelo SGCT para realizar a personalização do contexto da tarefa. Quando uma tarefa é iniciada, o serviço de Gerenciamento de Tarefas (SGT) comunica ao SGCT sobre a tarefa inicializada, informando o *template* que representa o conjunto de dados necessários pela aplicação, do sistema pEHR, correspondente. Com essa informação o serviço de gerenciamento de Contexto de Tarefas busca as Classes de Dados necessárias pela aplicação, bem como a semântica associada à elas. Com isso, obtém as instâncias de sensores necessários junto ao serviço de monitoramento e os registra no serviço de gerência de contexto do EXEHDA. Da mesma forma, os atuadores associados a uma tarefa são carregados junto ao sistema de monitoramento e gerenciados pelo SGCT.

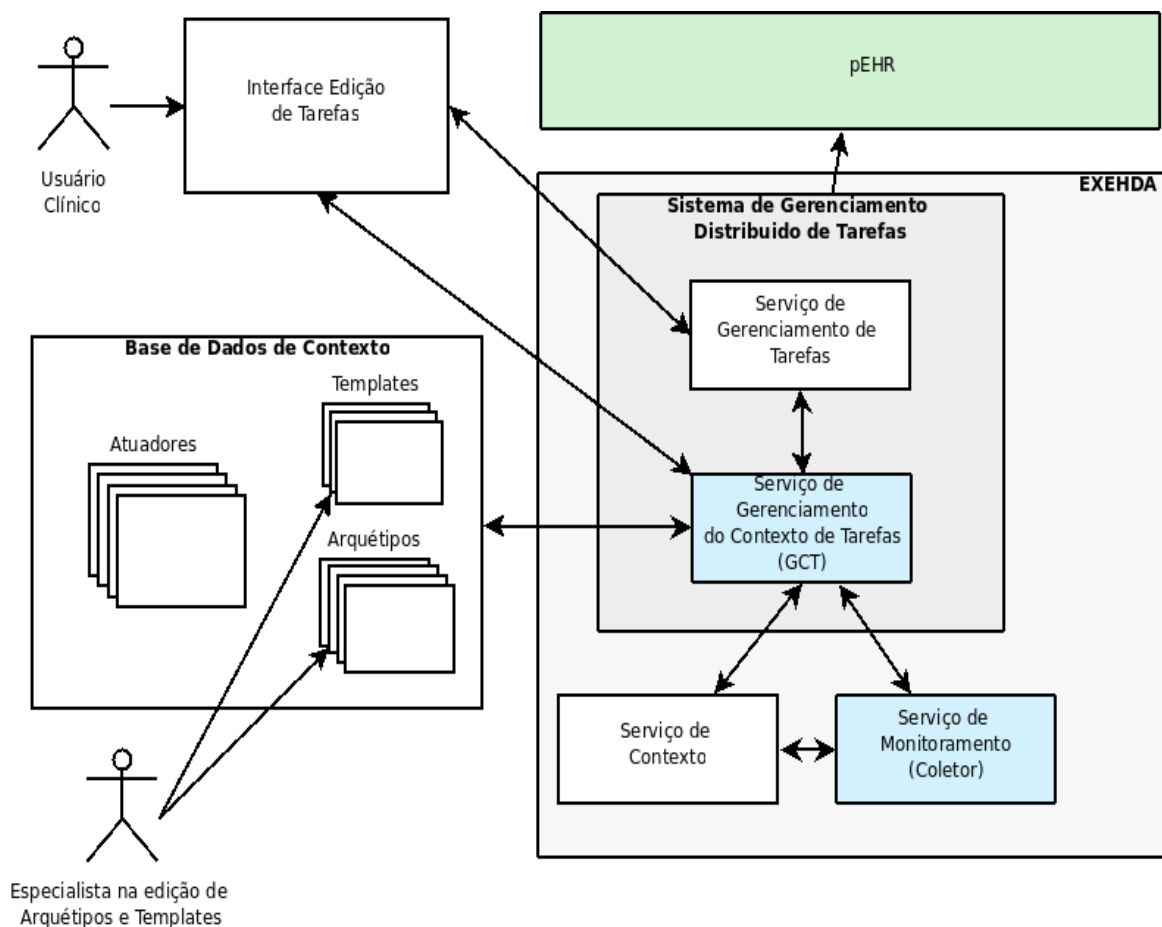


Figura 15 - Arquitetura base do sistema de gerenciamento de contexto de tarefas - outros componentes da arquitetura ClinicSpace, sem relação direta com o serviço foram omitidos.

As próximas seções descrevem maiores detalhes de como esse modelo foi implementado.

4.3. Implementação do SGCT

O modelo de contexto proposto executa no meio pervasivo fornecido pelo *middleware* EXEHDA, usufruindo dos serviços nele implementados. Os principais serviços do EXEHDA diretamente relacionados com o contexto são (i) o sistema de monitoramento, através do qual as informações de sensores são obtidas e (ii) o serviço de contexto, o qual especifica uma interface para realizar um monitoramento de contexto baseado em estados, permitindo dessa forma o tratamento de contextos de alto nível no *middleware*.

Para adequar esses serviços ao modelo proposto, foram necessárias modificações nos serviços do *middleware*, bem como a criação um novo serviço, denominado Serviço de Gerenciamento de Contexto de Tarefas (SGCT), o qual é responsável pela interface entre os demais componentes, além de promover o gerenciamento necessário para obter o contexto relacionado à tarefa. Outros serviços do EXEHDA são utilizados pela arquitetura ClinicSpace, porém não serão tratados neste trabalho. No APÊNDICE B – Middleware EXEHDA encontra-se uma descrição sucinta dos serviços disponibilizados pelo *middleware*.

4.3.1. Serviço de Monitoramento – *Coletor*

O *middleware* promove uma separação entre as camadas produtoras de informação e as camadas consumidoras. A interligação entre elas ocorre através de um processo de subscrição-notificação (*publish-subscribe*). O serviço coletor (SILVA, 2003) é responsável por todo o aspecto de monitoramento de sensores no EXEHDA, promovendo as prerrogativas necessárias para subscrição de aplicações e serviços interessados nos dados produzidos pelos sensores do sistema de monitoramento. O serviço, em intervalos pré-definidos de tempo, realiza o *polling* das informações produzidas por esses sensores e armazena esses dados na memória *cache* do serviço. Com isso, quando diversas aplicações requisitarem os dados eles serão lidos da *cache*, e não do sensor propriamente, produzindo um mecanismo para proteção da sobrecarga eventualmente gerada por muitos acessos a um mesmo sensor.

Os sensores são registrados no EXEHDA sob a responsabilidade de um componente denominado *monitor*, o qual tem por finalidade coordenar a operação dos sensores. Através do acesso ao *monitor*, o sistema de *polling* age continuamente e, quando mudanças nos valores monitorados são detectadas, todas as aplicações inscritas naquele sensor são notificadas, através de uma função de *callback* ou *listener*.

Essa abordagem é interessante, porém apresenta uma necessidade intrínseca de a aplicação ou o serviço conhecer o nome do sensor associado. Além disso, esse sistema dá suporte somente a sensores gerais, ou seja, se houver mais de um sensor monitorando um mesmo tipo de atributo, de diferentes entidades, há a necessidade de conhecer-se previamente o nome do sensor ligado a cada entidade, o que do ponto de vista do projeto ClinicSpace, não é viável. Em função disso, é parte integrante deste trabalho modificar o serviço *coletor* de forma que seja possível a utilização de sensores personalizados através do conceito de instâncias de sensores. Essa modificação será descrita na subseção 4.3.3.

4.3.2. Serviço de Gerenciamento de Contexto

O sistema de monitoramento, descrito anteriormente, promove a capacidade de a aplicação receber os dados brutos produzidos pelos sensores do EXEHDA, porém, sem nenhuma significação semântica agregada. Esta última, é função do sistema de gerenciamento de contexto do *middleware* EXEHDA, no qual as aplicações subscrevem-se para que um determinado contexto associado a um sensor seja monitorado.

Para oferecer um mecanismo flexível e adaptável, a semântica envolvida nos dados é especificada pela aplicação, através de um padrão XML estabelecido por esse serviço. Dessa forma, a aplicação não mais se inscreve junto ao *Coletor*, mas sim ao sistema de contexto, que atua como um intermediário entre o *Coletor* e a aplicação, oferecendo uma semântica aos valores monitorados e notificando a aplicação somente sobre as mudança de estados deste dado. Os estados são o núcleo do padrão XML fornecido pela aplicação e podem corresponder a um valor pontual ou intervalo de valores (SILVA, 2003).

Dentro do projeto ISAM foi desenvolvido outro subsistema de contexto, denominado MULTIS (FEHLBERG, 2007), que difere do primeiro na questão da composição de contexto, utilizando múltiplos sensores para descoberta de um dado. Tanto um quanto outro sistema de contexto pode ser utilizado sem alterações significativas pelo ClinicSpace. As adaptações

cabíveis são no nível da especificação adequada pelo mecanismo de subscrição do serviço ao contexto.

4.3.3. Modelando e Construindo o Suporte a Sensores Personalizados no EXEHDA

Em um sistema automático de coleta de dados, deverão existir muitas situações onde um mesmo tipo de atributo é requisitado por n diferentes aplicações e usuários. Sendo a busca, no sistema de monitoramento, norteadas em função do nome do sensor que produz o dado, cria-se uma grande dependência entre os nomes de sensores disponibilizados pelo sistema de monitoramento e a aplicação. Essa situação é melhor descrita através do cenário a seguir:

O usuário A1 está executando uma aplicação, chamada “Obtém status do paciente”. Esta é responsável pelo monitoramento dos batimentos cardíacos e pressão sanguínea do paciente, neste caso, o paciente P1. Os dados necessários pela aplicação podem ser obtidos através de sensores do EXEHDA. Reproduz-se esse processo, com n usuários executando essa mesma aplicação, sendo os dados por ela tratados específicos para n pacientes.

Esse tipo de cenário não é suportado pelo EXEHDA, visto que a aplicação é estaticamente programada para tentar obter os dados através de sensores específicos, cuja identificação é dada pelo nome com que foram submetidos junto ao sistema de monitoramento. Não haveria forma de distinguir à que paciente esses dados pertencem, tampouco obter os mesmos tipos de dados para pacientes distintos, através de um mesmo sensor.

Uma solução para resolver essa questão seria parametrizar a aplicação para que ela busque um sensor específico, com um nome único, que monitora um determinado tipo de dado, para um determinado paciente. Porém, essa solução implica na necessidade de, no nível de aplicação, manter uma lista mapeando nomes de sensores relacionando-os com as entidades sensoreadas.

Uma solução mais adequada é permitir o conceito de instâncias de sensores, onde através da parametrização de um sensor geral, relativo ao dado desejado, possam ser criados

sensores personalizados para obtenção do dado específico para a entidade desejada. Essa solução foi escolhida e representa o modelo adotado para promover a personalização de sensores no ClinicSpace.

4.3.3.1. Implementação da Nova Funcionalidade no EXEHDA

Para dotar o EXEHDA dessa nova funcionalidade, modelaram-se novas classes e modificou-se a interface de serviços, bem como sua implementação, de forma a suportar os novos requisitos. O serviço `Coletor` oferece níveis de hierarquia no acesso aos sensores, disponibilizando uma versão para acesso público e a versão privada, de uso interno. Essa diferenciação ocorre em função, principalmente, dos aspectos de segurança de acesso. O acesso externo a API vê os sensores através de uma classe genérica, `Sensor`. Conforme mostra a Figura 16.

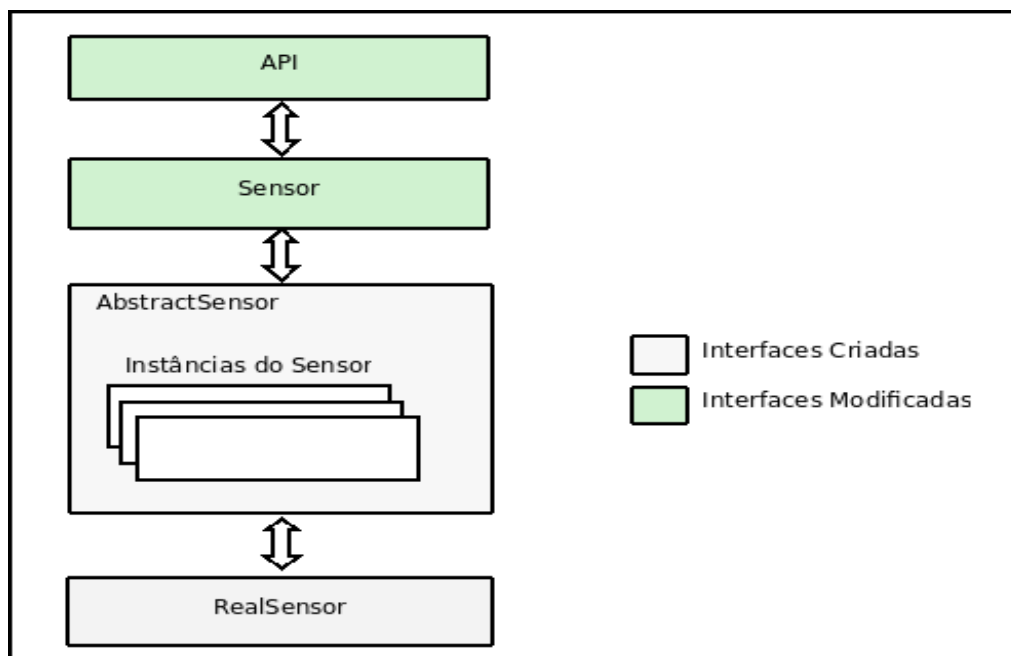


Figura 16 - Hierarquia de Abstração de Sensores no Coletor.

O suporte à personalização é obtido através do conceito de Classe de Dado, implementado no sistema, sob a interface `AbstractSensor`, a qual representa as operações

que devem ser suportadas por um Sensor Abstrato. Cada sensor disponível no EXEHDA dá suporte à obtenção de uma determinada Classe de Dado, que, através da parametrização, possibilita o sensoriamento de múltiplas entidades que tenham, entre seus atributos, uma Classe de Dados comum.

No desenvolvimento de um sensor são especificados quais os parâmetros que ele suporta, especificando um nome e tipo para eles. Em tempo de execução, sempre que um sensor deste tipo for requisitado por uma aplicação, deverá também ser especificados os valores dos parâmetros para a instância deste.

Quando uma aplicação requisitar um sensor personalizado, ela deverá verificar junto ao serviço `Coletor` se existe um *AbstractSensor* disponível no sistema para oferecer suporte a Classe de Dado requisitada. Se houver, a aplicação requisita a criação ou obtém uma instância deste sensor, para que ela possa obter os dados que deseja.

Na construção de uma nova instância, um nome único de sensor é criado, baseado na classe de dados monitorada e nos parâmetros repassados pela aplicação. No caso de mais de uma aplicação requisitar o mesmo sensor personalizado, ambas são subscritas na primeira instância existente e passam a receber notificações sobre as alterações ocorridas em seus valores. Com isso, resolve-se a questão dos nomes únicos para o sistema de sensoriamento, ao mesmo tempo em que é fornecida a capacidade de criação dinâmica de sensores baseadas em um sensor abstrato para a Classe de Dados desejada.

O controle de monitoramento dos sensores é realizado através de entidades chamadas *monitores*, estes, aplicam políticas de monitoramento comum a um conjunto de sensores construídos sobre ele. Dessa forma, os Sensores Abstratos estão contidos logicamente dentro de um monitor e, sempre que o último disparar o processo de *polling* sobre um Sensor Abstrato, é realizado *polling* em todas as instâncias pertencentes a ele, sendo os valores obtidos armazenados na *cache* do serviço.

Embora os sensores personalizados sejam requisito essencial para o suporte ao modelo proposto por ClinicSpace, há também a necessidade de manter a funcionalidade de sensores globais. O modelo implementado permite a coexistência de ambas funcionalidades de maneira natural.

Embora todos os sensores sejam implementados obedecendo a interface de um Sensor Abstrato, os parâmetros especificados para esse sensor global, no momento de sua programação, podem ser nulos, ou seja, nenhum parâmetro. Nesse caso, o sensor é genérico e permite a criação de uma única instância para o monitoramento daquela Classe de Dado, fornecendo um sensor absoluto.

Um exemplo de sensor deste tipo seria um sensor de data e hora, que embora sejam utilizados por todas as aplicações, não necessitam de personalização, pois tratam de atributos globais do ambiente. Para os sensores deste último tipo, a primeira aplicação que requisitá-los criará uma instância global deste, que será utilizada em todos os acessos àquela classe de Dados a partir daquele momento, independente de qual aplicação o requisita. Essa característica torna o sistema de personalização de sensores genérico, agregando novas funcionalidades ao serviço *Coletor*, ao mesmo tempo em que mantém a compatibilidade com a ideia inicial, de sensores globais, presentes no EXEHDA.

O EXEHDA permite o crescimento dinâmico do ambiente virtual, ou seja, o acréscimo de novos sensores e elementos no decorrer de sua execução. Elementos derivados de um sensor abstrato não podem serem adicionados individualmente no sistema, mas sua adição é permitida através de monitores. Para tornar essa ideia um padrão mais formal, foi construído um conjunto de interfaces e classes que as implementam, tornando o modelo mais claro.

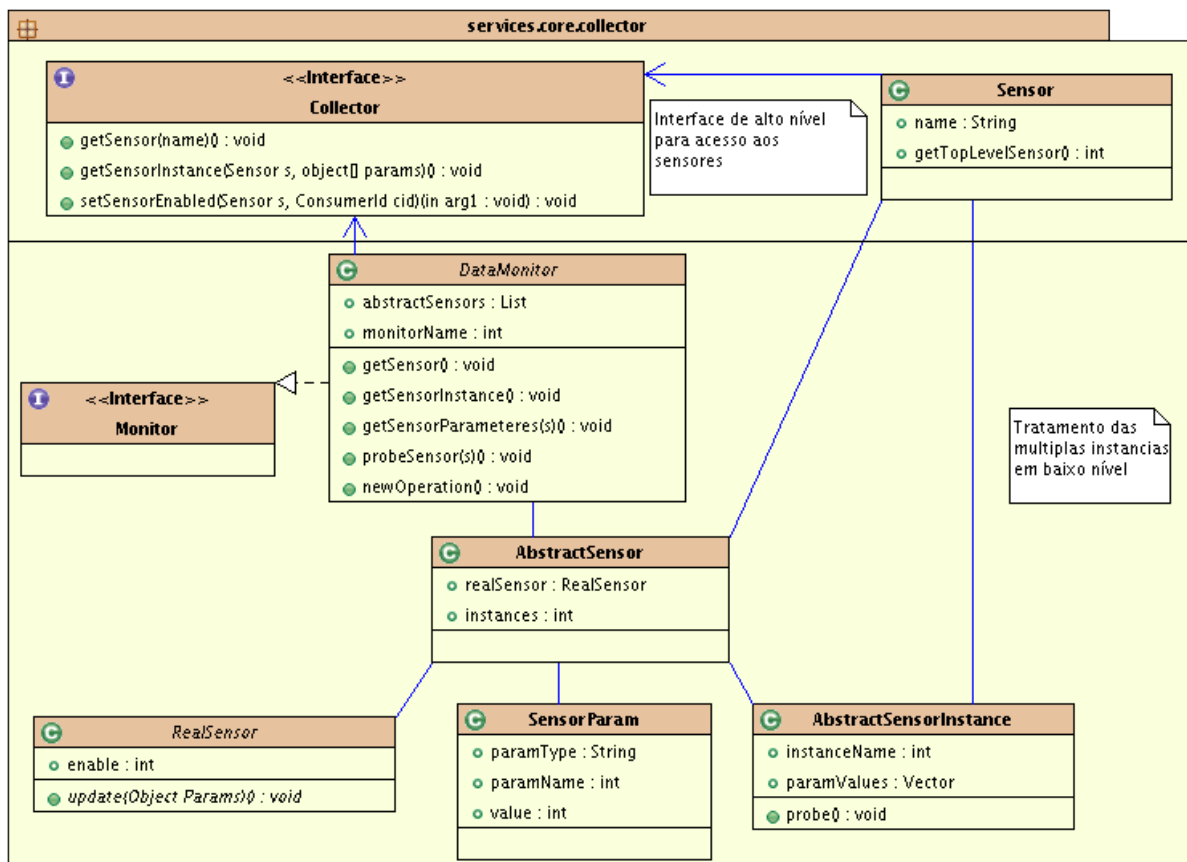


Figura 17 - Diagrama de classe para suporte a múltiplas instâncias no EXEHDA - vários métodos, atributos e classes foram omitidos.

Para permitir as abstrações necessárias para esse novo modelo do serviço coletor, especificou-se interfaces definidas que devem ser implementadas tanto pelos monitores quanto pelos sensores reais. O modelo principal é mostrado na Figura 17, mas por uma questão de clareza são omitidas várias classes e métodos que compõem o modelo, focando nas principais classes e seus relacionamentos.

Quando um monitor é implementado, são adicionados sensores abstratos que fornecem a interface padrão de um sensor no sistema, implementada pela classe *AbstractSensor*. Estes são ligados a um sensor real, que efetivamente produz os dados desejados. Para ilustrar esse processo, o Quadro 2 mostra um exemplo simples, que simula o processo de construção e registro de sensores no *middleware*.

```

public class GenericMonitor extends DataClassMonitor {
    public final static String name = "ENVIROMENT_PATIENT";

    public GenericMonitor() {

        super.setName(name);
        AbstractSensor sensor;
        RealSensor rs;

        //número de sensores distintos que haverão neste monitor
        this.abstractSensors = new AbstractSensor[2];

        //adiciona o sensor bloodpressure
        SensorParameter[] params = new SensorParameter[1];
        //fica encapsulado dentro do abstractsensor
        rs = new Sensor_BloodPressure();

        //especifica o parâmetro "pacienteID", que consiste em um
        //inteiro, e é utilizado para identificação das instâncias
        //particulares deste sensor.
        params[0] = new SensorParameter("pacienteID", Integer.class,
null);
        sensor = new AbstractSensor("Paciente_Pressure", params, rs,
this);
        this.abstractSensors[0] = sensor;

        //adiciona o sensor datetime
        params = null;
        rs = new Sensor_DateTime();
        sensor = new AbstractSensor("DateTime", params, rs, this);
        this.abstractSensors[1] = sensor;

        //outros sensores poderiam ser adicionados aqui...
    }
}

```

Quadro 2 - Exemplo do código de criação de um Sensor Abstrato.

O código de um sensor deve obedecer a interface estabelecida para a classe *RealSensor*, conforme mostra o Quadro 3.

```
public interface RealSensor {  
    public Object update(Object[] params);  
}
```

Quadro 3 - Interface sobre a qual os sensores devem ser implementados.

4.3.3.2. Integrando os componentes do sistema para entrada automática de dados

O modelo implementado no serviço SGCT, criado junto ao EXEHDA, é o responsável por orquestrar a ligação entre as aplicações e o serviço de monitoramento de contexto, utilizando a interface modificada do serviço coletor, para permitir a obtenção das instâncias de sensores desejados. A partir da instância de sensor obtida, juntamente com o conjunto de dados obtidos da base de dados de contexto, monta a especificação necessária para acesso ao serviço de contexto do *middleware*. O diagrama de sequência exibido na Figura 18 torna gráfico esse processo.

4.3.4. Modelagem e Implementação dos Elementos Personalizáveis de Contexto (Atuadores)

Os eventos programáveis pelo usuário, atuadores, consistem em uma nova funcionalidade disponibilizada no EXEHDA, responsável por receber os parâmetros informados pelo usuário e realizar as operações necessárias para sua efetivação na arquitetura. Os atuadores são controlados pelo SGCT, utilizando os serviços base de monitoramento de sensores e, contexto do EXEHDA. No momento de sua construção, é especificado um ou mais gatilhos monitorados pelo *middleware* e que, quando atingidos, são responsáveis por executar a ação prevista pelo atuador. Salienta-se que os atuadores são entidades construídas

com um propósito específico, onde é permitido ao usuário ligá-los às tarefas que desejar, personalizando o valor dos elementos de disparo. O agendamento é um exemplo de atuador, onde seu gatilho principal é constituído pela data e hora.

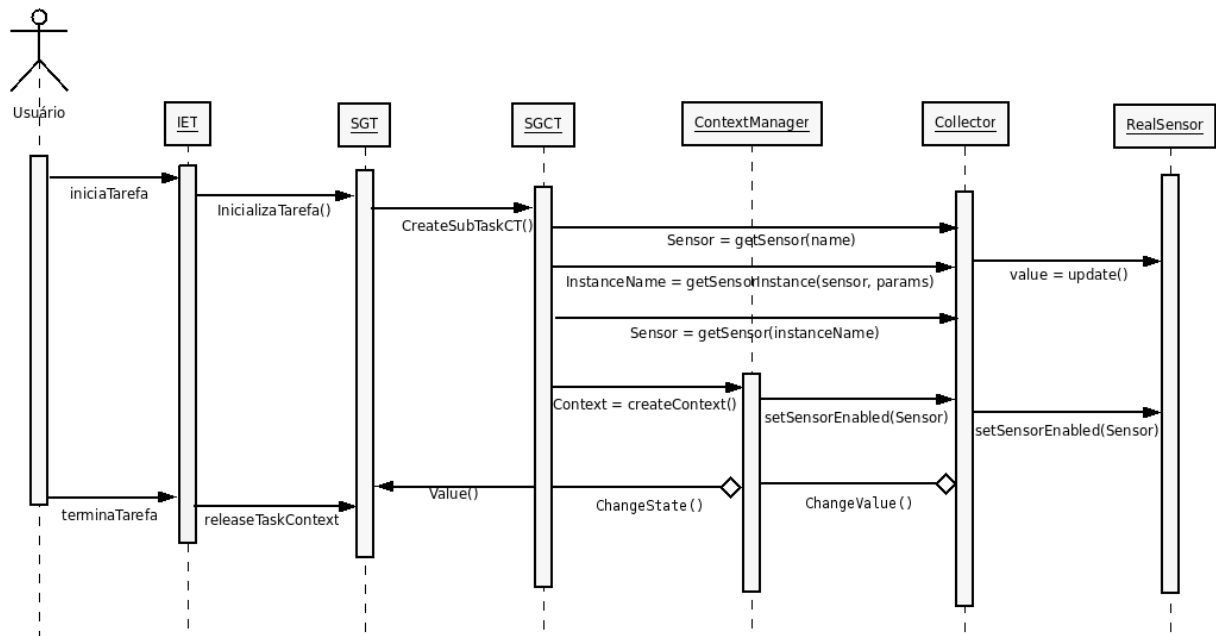


Figura 18 - Diagrama de Sequência para obtenção automática de dados pelo sistema de Contexto.

Existem basicamente dois escopos para os atuadores: os diretamente relacionados ao usuário e os diretamente relacionados à tarefa. Atuadores relacionados ao usuário tem seu processo de monitoramento norteado pela simples presença do usuário no sistema, não importando qual a tarefa que o usuário está realizando no momento. Um atuador para agendamento é um exemplo de atuadores de escopo de usuário, visto que se houver uma tarefa programada para ser disparada em uma determinada data e hora, independente da tarefa que o usuário está realizando no sistema, ele deverá ser notificado sobre esse agendamento.

Outro escopo possível para os atuadores é a própria tarefa. Pertencem a esse escopo os elementos de contexto que são definidos somente em função de uma situação particular, no decorrer de uma determinada tarefa. Os atuadores deste tipo somente são instanciados no sistema quando a tarefa estiver sendo executada. Para exemplificar, um atuador que detecta alterações significativas na pulsação ou pressão sanguínea do paciente, no decorrer de uma tarefa, pertencem ao escopo desta, e podem efetuar o registro dessas informações de forma automática no sistema clínico, através de uma aplicação, como a responsável pelo registro de notas diárias. A definição do escopo de um atuador é estabelecida no momento do seu

desenvolvimento.

De forma semelhante à solução adotada para os sensores personalizados, os atuadores são instanciados a cada associação destes, realizada pelos usuários. Uma instância de um atuador deverá conter o usuário, tarefa e o conjunto de valores para os elementos de contexto que efetuam o disparo da ação desejada. Para implementar um atuador, deve-se estender a classe *AbstractActuatorGeneric*, a qual implementa o conjunto de operações comuns aos atuadores, seguindo a interface fornecida pela classe *AbstractActuator*. Um subconjunto do diagrama de classes mostrando a relação entre os componentes do sistema é ilustrado na Figura 19 (vários elementos foram omitidos para facilitar o entendimento).

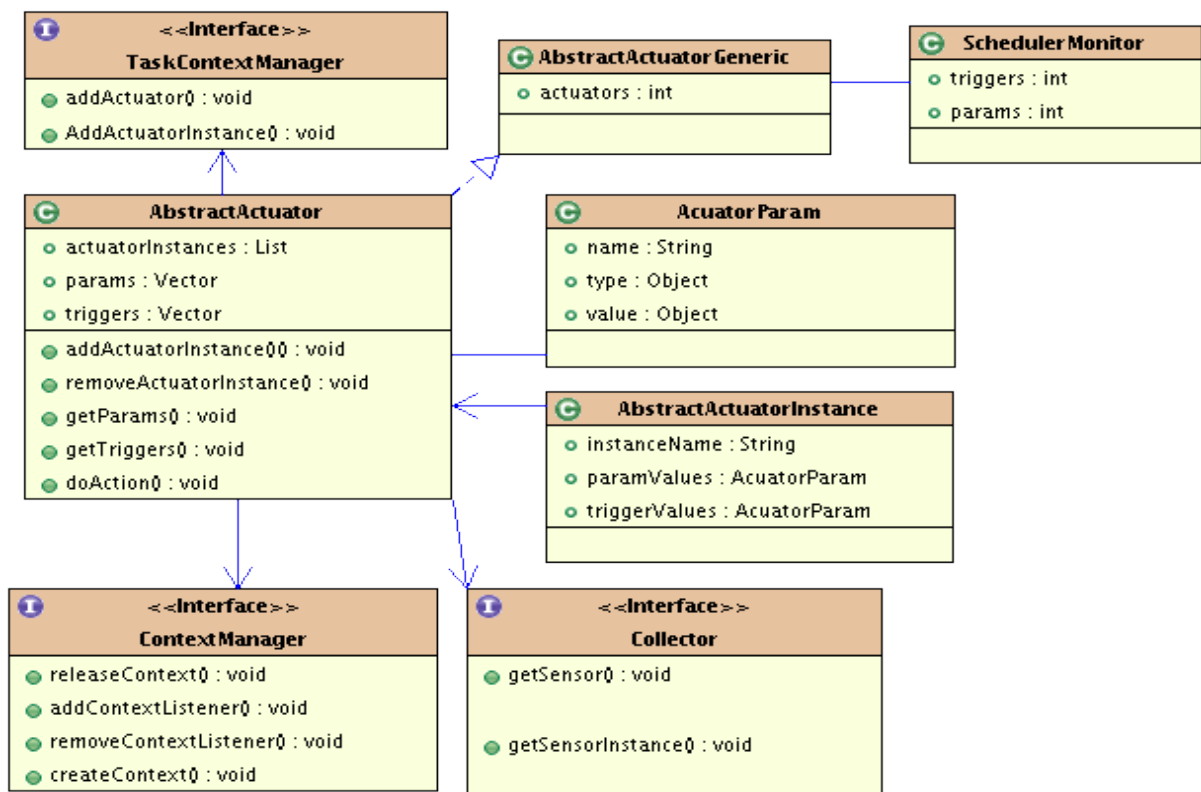


Figura 19 - Diagrama de Classes (reduzido) ilustrando a implementação dos elementos personalizáveis de contexto – os atuadores.

Os atuadores são armazenados no banco de dados de contexto, e carregados conforme seu escopo de ação, no instante da carga da tarefa desejada ou, no instante da autenticação do usuário junto ao sistema. Os atuadores são componentes gerais, disponíveis no sistema, e que devem ser instanciados pelo usuário quando este deseja utilizá-lo, esse processo é realizado através do mecanismo DND (*Drag'n'Drop*) mostrado na seção 4.2.

Da mesma forma que o sistema automático de entrada de dados, os atuadores utilizam-se do serviço de monitoramento para criar os sensores personalizados de que necessita. Cada atuador possui um conjunto de um ou mais gatilhos que passam a serem monitorados no momento da criação de uma nova instância do atuador. Esse monitoramento é realizado através da obtenção dos sensores abstratos que representam as classes de dados relativas aos gatilhos desse. Sendo os sensores abstratos encontrados, são criadas instâncias destes, as quais passam a ser monitoradas pelo sistema de contexto que, por sua vez, recebe como parâmetros do monitoramento os valores que o usuário especificou no momento da criação da instância do atuador, para sua execução.

Dessa forma, existem dois contextos de alto nível possíveis para o estado monitorado dos gatilhos de um atuador: *executar* e *aguardar*. Até que o valor definido pelo usuário, para os sensores monitorados, seja atingido, o sistema de contexto atribui o estado “aguardar” ao parâmetro estabelecido; no momento que o valor atingido for identificado pelo sistema de monitoramento, o sistema de contexto será acionado, modificando o contexto de alto nível para o estado “executar”.

Devido à mudança de estado, o sistema de contexto irá invocar a *callback* registrada pelo atuador, executando dessa forma a ação para a qual o atuador foi programado.

Na ocasião da associação de um atuador a uma tarefa, o SGCT busca os parâmetros relativos ao atuador associado. Esses parâmetros são obtidos através do método *getTriggers()*, o qual retorna uma lista de objetos derivados da classe *ActuatorParam*, que contém, entre outras informações, o tipo (classe) que representa o elemento que se deseja monitorar, bem como uma descrição deste. Essas informações são utilizadas na interface de interação com o usuário, para determinar a mensagem mostrada ao usuário e para validar o tipo de informação recebida do usuário (*string*, *integer*, *float*, etc.).

Após obter-se essas informações, ocorre a criação de uma instância para o atuador em questão através do método *addActutorInstance()*, que dentre seus parâmetros constam identificador da tarefa, usuário, além dos valores de disparo à serem monitorados pelo atuador.

Podem ser criadas quantas instâncias se quiser de um atuador, desde que elas tenham nomes únicos no sistema, o que oferece mais um mecanismo de validação para a associação dos atuadores, visto que o nome da instância do atuador é dado pelo conjunto formado pelo nome do atuador abstrato bem como usuário, tarefa e valores de parâmetros de disparo. Se um usuário tentar duplicar a associação de atuadores para uma tarefa, baseado nos mesmos

parâmetros de disparo, ele será informado sobre esse fato, através de uma mensagem. No Quadro 4 é exibida a interface genérica de um atuador.

```

public interface AbstractActuator {

    public abstract ActuatorInstance getActuatorInstance(
        String instanceName);

    public abstract String makeInstanceName(int taskid, int userid,
        Object triggers[]);

    public abstract ActuatorInstance removeActuatorInstance(
        String instanceName);

    public abstract ActuatorInstance addActuatorInstance(int taskid,
        int userid, String userDescription,
        Object[] triggersValues,
        Object[] paramValues);

    public abstract Hashtable getActuatorInstances();

    public abstract LinkedList getTriggers();

    public abstract LinkedList getParams();

    public abstract int getParamsSize();
}

```

Quadro 4 - Interface modelada para um Atuador.

Essa arquitetura permite tanto a utilização de atuadores lógicos quanto atuadores físicos, embora na modelagem inicial da arquitetura somente os primeiros tenham sido contemplados.

Um atuador por ser utilizado para diversos fins, (i) agendamento de tarefas, (ii) registro automático de procedimentos e informações baseado em situações particulares de execução do decorrer de uma tarefa ou (iii) mecanismos de colaboração, repassando tarefas para outros usuários, se a tarefa atual de um usuário tiver de ser interrompida em função de uma tarefa mais urgente. Com isso, demonstra-se sua flexibilidade, possibilitando a percepção de que a sua capacidade de programação e utilização é diretamente baseada na quantidade de sensores disponíveis no ambiente, fornecendo as informações necessárias.

Mesmo que um atuador esteja disponível no sistema e, por algum motivo, os sensores fornecedores das informações correspondentes aos seus gatilhos não puderem ser obtidos pelo sistema de contexto, ele poderá ser utilizado. Porém, existem duas situações

potencialmente problemáticas:

a) Tentar instanciar um atuador na inicialização de uma tarefa e não ser possível obter os sensores abstratos correspondentes aos dados necessários pelo atuador;

b) O sensor abstrato está disponível e o atuador é instanciado; porém, durante a execução da tarefa o usuário se move, por exemplo, e no novo ambiente esses sensores não estão mais disponíveis, sendo que o contexto necessário a inicialização do atuador não será alcançado e, portanto, sua ação não será executada. Esse atuador ficará registrado no sistema, consumindo os recursos associados e não realizando a ação que o usuário espera que ele realize, previamente programada por ele.

O tratamento dado para esses problemas pelo SGCT é feito da seguinte forma: para o caso (a) onde não é possível obter os sensores necessários pelo atuador, o usuário é notificado dessa limitação, ficando ciente da incapacidade do sistema realizar as ações programadas. Já o caso (b) decorre normalmente da movimentação do usuário no ambiente onde encontra-se o sistema, ou da retomada de tarefas previamente suspensas. Nesse caso, liberar os atuadores no momento da suspensão de uma tarefa, realocando-os no momento que ela for retomada, auxilia na resolução dessa questão.

Para que isso seja possível, sempre que uma tarefa for suspensa ou ativada, o gerenciador de tarefas (SGT) deverá informar ao gerenciador de contexto de tarefas (SGCT) sobre esse fato; os atuadores serão alocados ou desalocados, conforme o caso, para a tarefa requerida. A alocação de um atuador segue a ideia de instâncias tal como o sistema de monitoramento, sendo o nome da instância dado pelo nome do atuador abstrato mais o identificador de usuário, tarefa e os parâmetros necessários para sua execução, formando dessa forma uma chave única para seu nome, que permitirá buscas posteriores.

Para exemplificar, implementou-se o atuador responsável pelo agendamento, o qual tem por finalidade associar informações temporais, especificadas pelo usuário, para efetuar o lançamento de uma tarefa. A informação monitorada por ele, utiliza o sensor `DateTime` (exemplificado no Capítulo 4.3.4), cuja informação monitorada é a data e hora do sistema. Esse atuador possui o escopo do usuário, já que o evento responsável por sua criação é gerado pela presença do usuário no sistema. Sempre que o usuário autenticar-se, o sistema de gerenciamento do contexto de tarefas busca junto à base de dados de contexto quais atuadores deve instanciar, baseado no escopo do usuário.

Os gatilhos de um atuador são registrados no sistema de contexto em um formato similar ao descrito no Quadro 5 (atuador de agendamento), substituindo os valores de `taskid`, `userid`, `userdate` e `trigger_value` pelos parâmetros relativos à

instância de utilização.

```
<CONTEXT n='scheduler:taskid:userdate'>
  <STATES>
    <STATE n='standby' />
    <STATE n='execute' />
  </STATES>

  <FILTER syntax='xml:numericRange'>
    <INDEX>
      <SENSOR n='dateTime' />
    </INDEX>
    <MAPPINGS>
      <MATCH op='equals' v='trigger_value' ignoreCase='true' state='execute' />
      <DEFAULT state='standby' />
    </MAPPINGS>
  </FILTER>
</CONTEXT>
```

Quadro 5 - Código XML para criação do contexto associado ao Atuador de Agendamento.

Os testes realizados para validação da arquitetura através de um protótipo é tema do próximo capítulo.

5. DISCUSSÃO DOS RESULTADOS

Neste capítulo são discutidos os testes realizados para validação da arquitetura, bem como uma análise comparativa com os trabalhos relacionados.

5.1. Trabalhos Relacionados

O *middleware* pervasivo Aura (SOUSA, 2002) é precursor dos ambientes pervasivos baseados no conceito de tarefa do usuário. Seguindo a ideia de que o usuário possui uma “aura computacional”, a qual o acompanha para os variados ambientes e age de forma pró-ativa descobrindo recursos e atuando na migração de tarefas do usuário para esse novo ambiente. Em Aura, a obtenção de contexto é tratada de forma síncrona, ou seja, os dados são requisitados pela aplicação e obtidos junto aos provedores naquele momento, não sendo utilizado o mecanismo de subscrição presente em outros *middlewares*. Dessa forma, não permite a comunicação assíncrona de eventos de contexto. O contexto tratado por Aura é focado na descoberta de serviços adequados as características dos equipamento utilizados pelos usuários.

O projeto Gaia (ROMAN, 2002) originou o *middleware* pervasivo Gaia, o qual estabelece o conceito de Espaços Ativos (*Active Spaces*). Estes são construídos através da extensão das funcionalidades de um sistema operacional tradicional, provendo os serviços responsáveis pelo tratamento das características pervasivas. Este *middleware* tem por objetivo fornecer um ambiente de programação pervasivo, fornecendo uma visão transparente da interação entre os serviços, através da disponibilização de uma API de programação. A adaptação e monitoramento de eventos em Gaia é baseada no paradigma de publicação-subscrição, o qual fornece informações de contexto baseado na alteração de seus estados, de forma assíncrona. O contexto de Gaia leva em consideração diversos fatores presentes no ambiente e sua forma de utilização, o que permite inferir tarefas baseado no conjunto de informações obtidas junto ao monitoramento.

De forma similar, o projeto ISAM (ISAM, 2001), deu origem ao *middleware* pervasivo EXEHDA (YAMIN, 2004), o qual tem por finalidade promover uma ambiente pervasivo,

utilizando uma arquitetura na forma de uma grade computacional móvel. Esta é constituída de nodos, os quais formam células, onde existem nodos responsáveis por coordenar a distribuição de serviços e a comunicação entre estes, visto que a arquitetura modela a composição de serviços descentralizados. Com isso, há um melhor aproveitamento dos recursos disponíveis no ambiente pervasivo. Os serviços da arquitetura interagem entre si, fornecendo as características pervasivas para a programação de aplicações. Nesta arquitetura, o serviço de contexto utiliza os mecanismos de subscrição, para notificação de alterações nos estados representantes dos dados monitorados pela aplicação.

Os três projetos anteriormente descritos abordam a construção de um ambiente pervasivo de propósito geral, adotando diferentes formas de solução para as questões pervasivas envolvidas. Porém, não levam em consideração as particularidades existentes nos ambientes clínicos. Diferenciando-se dos anteriores, o projeto ABC (*Activity-Based Computing*) (BARDRAM, 2007) propõe a construção de um ambiente pervasivo direcionado ao ambiente clínico, identificando aspectos importantes e as particularidades dos requisitos existentes na área clínica. Para isso, ABC utiliza o paradigma da Computação Orientada a Tarefas, aliados a técnicas de modelos para programação orientadas ao ambiente pervasivo. A utilização dos recursos presentes no ambiente é realizada de forma contextualizada, onde dependendo dos indivíduos e recursos próximos, diferentes comportamentos podem ser observados para o mesmo recurso. As informações de localização das entidades que compõem o ambiente e do conhecimento de modelos de atividades são extensivamente utilizadas na arquitetura ABC.

O projeto ClinicSpace realiza uma abordagem semelhante à utilizada no projeto ABC, porém com o diferencial inovador de promover a personalização das tarefas e do contexto associado à elas pelo usuário-final. Argumenta-se que o modo de realização das tarefas entre diferentes usuários não pode ser estático, visto que cada usuário tem suas preferências na realização desta, podendo modificar o fluxo de execução original estabelecido para a tarefa. Para compreender os componentes envolvidos na realização de uma atividade humana, buscou-se o suporte oferecido pela Teoria da Atividade. Esta define uma atividade como sendo um conjunto de operações e ações executadas por um usuário visando alcançar um objetivo, utilizando-se para isso de ferramentas de apoio. No ClinicSpace, o usuário pode realizar a programação de tarefas conforme queira, seguindo regras de validação, utilizando a composição de tarefas previamente realizadas (operações) e de subtarefas programadas no sistema (ações). Para construir o ambiente pervasivo associado, ClinicSpace estendeu as funcionalidades presentes no *middleware* EXEHDA, promovendo os serviços necessários,

para o suporte ao paradigma de Computação Orientada a Tarefas, já aproveitando a base pervasiva por ele oferecida. A Tabela 2 ilustra uma comparação entre esses projetos e as características tratadas por cada um deles.

Tabela 2 - Comparativo entre Projetos Relacionados e o ClinicSpace

	Ambiente Pervasivo			Suporte a tarefas			
	Mobilidade	Contexto	Contexto Assíncrono	Controle	Automação	Personalização Contexto	Entrada Automática
Aura	X	X			X		
Gaia	X	X	X				
EXEHDA	X	X	X				
ABC	X	X	X	X	X		
ClinicSpace	X	X	X	X	X	X	X

5.2. Avaliação do Protótipo

Para avaliar o modelo proposto de contexto, bem como suas características de funcionamento, criaram-se exemplos de sensores e atuadores os quais foram utilizados em aplicações de teste (experimentos). Os testes realizados visaram identificar, além do correto funcionamento dos componentes, questões relacionadas à escalabilidade do sistema. Nas próximas seções serão descritos os testes realizados individualmente para cada componente do sistema de contexto de tarefas.

5.2.1. Personalização de Sensores

Modelou-se sensores no sistema, um para obtenção de data e hora, sua implementação corresponde, com fidelidade, a implementação de um sensor real para essa finalidade, e outro sensor (fictício, implementação lógica correspondente a um sensor físico) para o monitoramento da pressão sanguínea do paciente. Esse último simula o funcionamento do

sistema de monitoramento através da geração de números aleatórios correspondentes ao valor do dado buscado. O sensor de data e hora foi implementado utilizando funções nativas da linguagem Java para obtenção destas informações do sistema operacional, conforme pode ser verificado no Quadro 6.

```

public class Sensor_DateTime extends RealSensor {

    @Override
    public Object update(Object[] params) {

        Format formatter;
        Date date;
        String stringDate;

        date = new Date();

        formatter = new SimpleDateFormat("yyyy.MM.dd.HH.mm");
        stringDate = formatter.format(date);
        return stringDate;
    }
}

```

Quadro 6 - Exemplo de implementação de um Sensor para obtenção de data e hora.

Em ambos os casos, os valores monitorados mudam constantemente com o tempo, o que permite verificar o funcionamento do sistema de subscrição e notificação das aplicações pelo sistema de monitoramento. Da mesma forma, permite a verificação do funcionamento do contexto de alto nível, associando faixas dos valores monitorados a estados, o que na mudança destes, acarreta a notificação da aplicação.

A escalabilidade do sistema foi testada através da criação sistemática de várias instâncias de um dos sensores implementados. Testaram-se diversas configurações de instanciamento, começando com um conjunto de 10 instâncias, aumentadas por esse mesmo número, até atingir um total de 1000 instâncias criadas. Para a criação de cada conjunto, o anterior é deletado, promovendo independência entre os resultados obtidos. Portanto, o tempo de criação, mostrado no gráfico, refere-se somente ao tempo de criação de cada conjunto.

Nota-se que o gráfico (Figura 20) segue uma tendência exponencial, o que significa que para um grande número de instâncias criadas, principalmente em sistemas com muitos usuários e sensores, deve-se implementar algum mecanismo de otimização.

Outro aspecto relevante é as variações abruptas ocorridas no gráfico nos conjuntos maiores que 500 instâncias de sensores. Isso é explicado pelo funcionamento do sistema de

polling utilizado na camada de monitoramento. Em virtude do acesso serial presente nas estruturas representantes do sistema de monitoramento, incluindo os sensores e suas instâncias, o acesso concorrente entre o sistema de *polling* e a criação de grandes blocos de sensores acarreta em acessos exclusivos, o que gera a variação de tempos visualizada no gráfico. Para conjuntos de sensores menores que 500 instâncias esse problema é imperceptível, visto que o tempo para criação é pequeno, menor que 1 segundo, o que implica em menor probabilidade do sistema de monitoramento realizar o *polling* durante o tempo de criação dos sensores.

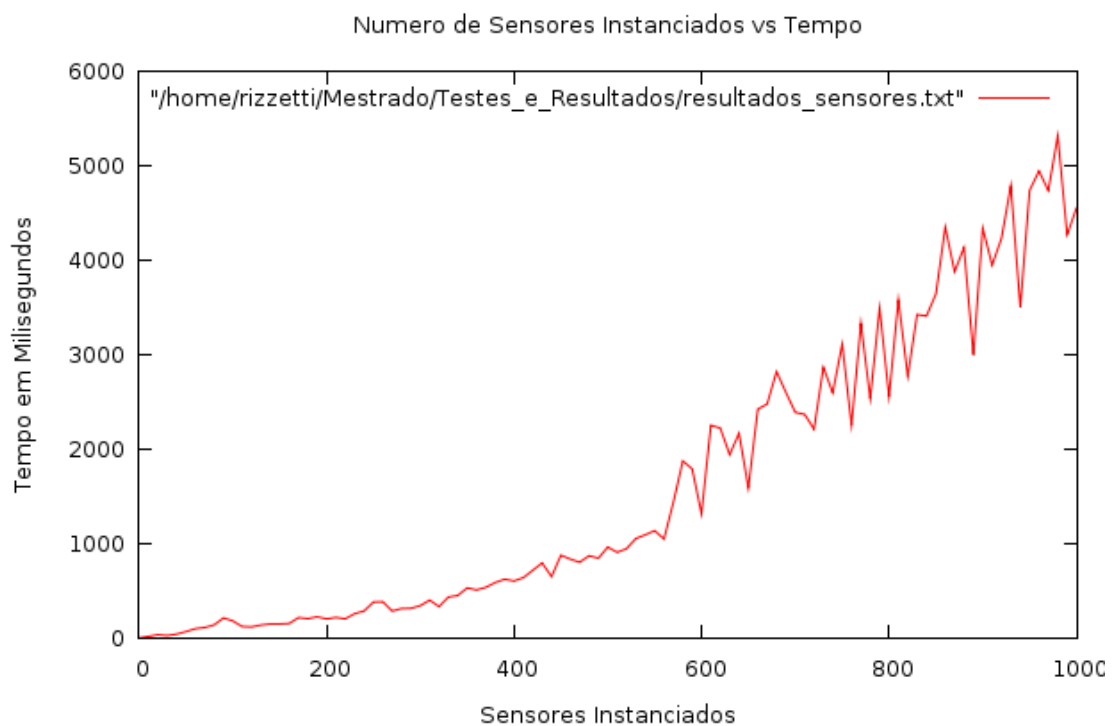


Figura 20 - Gráfico mostrando a relação entre o número de sensores instanciados pelo tempo necessário para sua criação.

A questão da concorrência entre o sistema de *polling* e a criação de grandes conjuntos de sensores, embora conhecida, não deve ocorrer com frequência na prática, já que cada aplicação geralmente deverá usar um conjunto de dados menores do que os utilizados no teste.

5.2.2. Atuadores

Os atuadores, conforme discutido no capítulo anterior, oferecem suporte a um variado conjunto de operações no ambiente oferecido pelo ClinicSpace. Semelhante aos sensores personalizados, instâncias de atuadores deverão ser criadas com frequência no ambiente de ClinicSpace, o que implica na necessidade de o sistema suportar escalabilidade. Para testar seu funcionamento, bem como o comportamento apresentado diante de uma carga intensa de instanciação, utilizou-se um formato similar ao empregado no teste dos sensores personalizados. Os quais de fato, também são utilizados pelos atuadores para monitorar os seus gatilhos.

Para realizar o teste da solução, implementou-se o atuador de agendamento, o qual utiliza o sensor descrito na seção anterior. Esse atuador é responsável por monitorar a informação de data e hora, sendo que, quando as informações cadastradas no atuador forem iguais as obtidas pelo sistema, esse realizará a ação correspondente ao seu método *doAction()*.

Utilizou-se a criação de conjuntos diversos de atuadores, medindo-se o tempo necessário da criação destas instâncias, incluindo o tempo necessário para que o gatilho de cada instância do atuador seja registrado junto ao sistema de monitoramento de contexto. Foram criados conjuntos de atuadores do intervalo entre 50 e 5000, variando o tamanho de cada conjunto em 50 elementos. O gráfico obtido é apresentado na Figura 21.

Conforme se observa no gráfico, o tempo gasto na criação dos atuadores é diretamente proporcional à quantidade de instâncias criadas, resultando em um gráfico linear, com uma reta de inclinação constante. Esse comportamento linear é decorrente do fato que todas as instâncias do atuador de agendamento monitoram um mesmo sensor, o sensor `DateTime`, que, em função do tipo de dado monitorado, não necessita utilizar instâncias separadas para cada usuário. Isso significa que o pior caso é gerado apenas na criação da primeira instância do atuador, quando é buscada a instância global do sensor `DateTime`, e criada se não existir. Após isso, todas as instâncias farão uso da mesma instância desse sensor.

Observa-se que o tempo para criação de atuadores é reduzido em relação ao tempo necessário para criação de sensores. Isso se deve ao fato de os atuadores possuírem estruturas únicas e menos complexas que o sistema de monitoramento. No entanto, quando vários usuários no sistema estiverem utilizando diversas instâncias de um mesmo atuador, cada um deverá criar uma instância personalizada para o sensor responsável por monitorar o gatilho do atuador. Isso acarreta no somatório do tempo gasto da criação do sensor que, como se pode

observar pelos gráficos, é o maior tempo necessário no processo, com o tempo gasto na criação do atuador. O que manifesta um comportamento gráfico semelhante ao apresentado na Figura 20.

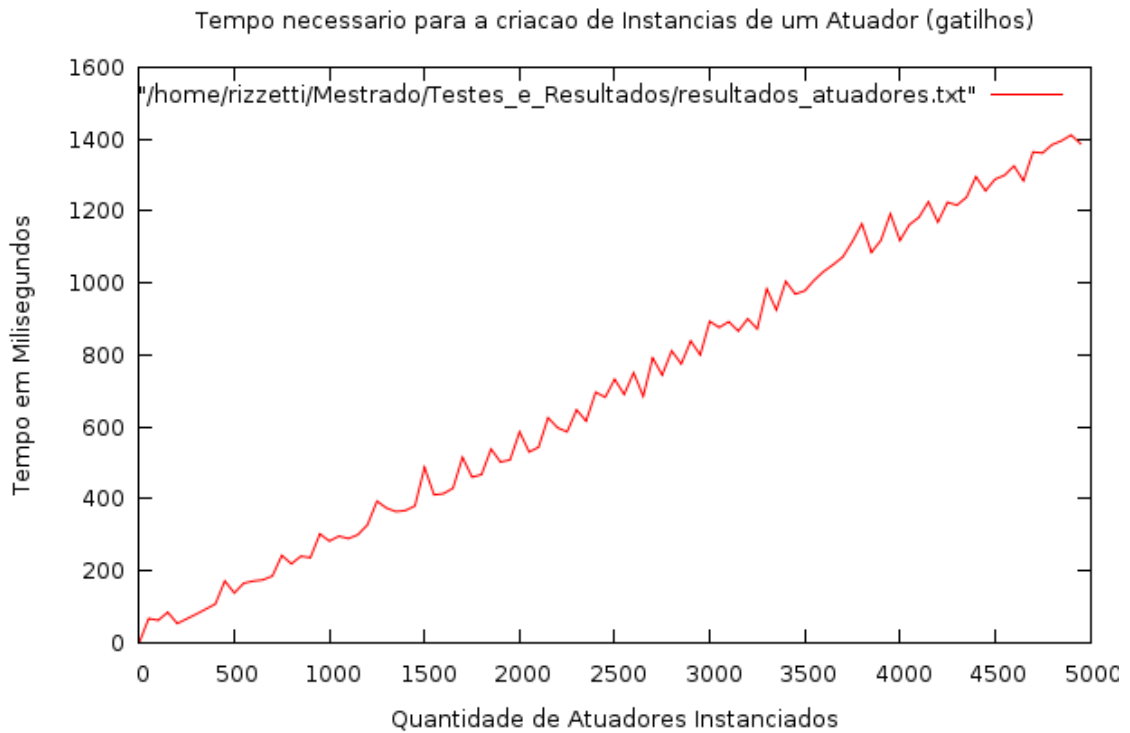


Figura 21 - Gráfico mostrando a relação do número de atuadores criados pelo tempo.

5.2.3. Entrada de Dados Explícita *versus* Entrada de Dados Implícita

A principal finalidade do modelo de contexto tratado neste trabalho, conforme discutido, é propiciar a entrada automática de Dados, minimizando o tempo gasto pelo usuário no preenchimento de informações. Para mensurar os ganhos que esse modelo traz, simulou-se uma aplicação clínica, que monitora os dados do paciente. O formulário simula, hipoteticamente, oito informações sobre o paciente que podem ser monitoradas com uma elevada frequência. A Figura 22 exhibe o formulário mostrado ao usuário.

Status do Paciente	
ID Paciente	12
Temperatura (°C)	37,1
Hematócrito (%)	40
Pressão Sanguínea	
Sistólica (mmHg)	120
Diastólica (mmHg)	80
Glicose (mg/dl)	89
Frequência Cardíaca	79
Hemoglobina (g/dL)	13
Salvar	

Figura 22 - Simulação de um Formulário com informações sobre Status do Paciente.

A entrada explícita de dados foi simulada através do preenchimento do formulário (Figura 22), medindo-se o tempo necessário desde o preenchimento do primeiro campo, informando os valores de todos os demais, finalizando com a seleção do botão “Salvar”. Para medir esse tempo utilizou-se o próprio código da linguagem de programação Java, sendo a seleção do campo “ID Paciente” o evento responsável pela inicialização da contagem, e, o pressionar do botão “Salvar”, o evento responsável pela sua finalização. A diferença existente entre esses dois momentos do tempo informa o tempo gasto no preenchimento do formulário.

Para simulação da entrada implícita de dados, construiu-se sensores lógicos responsáveis por monitorar cada um dos campos constantes no formulário. Os valores monitorados por estes sensores são resultado da geração de números aleatórios, que simulam o funcionamento de um sensor real. O funcionamento do sistema foi simulado através de uma chamada ao método `createSubTaskContext()`, do SGCT, o qual tem, como um dos parâmetros, o *template* representando o conjunto de dados necessários pelo formulário. Em função das ontologias de contexto não estarem disponíveis no momento, a busca dessa informação foi simulada através de retornos fixos em código. Com isso, a simulação cobriu todos os passos necessários por uma utilização real.

O tempo necessário na entrada implícita também foi medido via programação, abrangendo desde a chamada ao método `createSubTaskContext()` até a identificação do primeiro valor aleatório válido lido de cada um destes sensores requisitados pelo formulário.

Cada um dos tipos, de entrada de dados, foi medido 10 vezes, sendo os tempos obtidos mostrados na Tabela 3.

Tabela 3 - Relação entre os tempos de Entrada Implícita e Explícita para o mesmo conjunto de dados

	Tempo para Entrada Implícita (s)	Tempo para Entrada Explícita (s)
Execução 1	0,997	32,000
Execução 2	1,446	30,000
Execução 3	2,983	29,000
Execução 4	4,193	27,000
Execução 5	1,685	26,000
Execução 6	3,829	31,000
Execução 7	0,790	29,000
Execução 8	1,336	27,000
Execução 9	1,754	30,000
Execução 10	0,849	26,000
Tempo Médio	1,986	28,700

As diferenças de tempo entre cada execução do teste para entrada implícita de dados é explicado pelo intervalo de *polling* do sistema de monitoramento. Em função da configuração do mecanismo de *polling* estar setado para execução a cada 4 segundos, observa-se que o pior caso de execução é pouco além desse valor (4,193 segundos), mostrando que o teste foi lançado instantes após o mecanismo de *polling* ter sido executado. Normalmente os valores obtidos são bem menores, em média, pouco menos do que a metade do tempo de *polling*, neste caso 1,9 segundos.

Como observa-se na Tabela 3, os ganhos obtidos através da entrada automática de dados são significativos, no caso do teste, representando um tempo aproximadamente 15 vezes menor, do que o necessário para a entrada explícita, realizada pelo usuário. Validando, dessa forma, o argumento defendido, de que a entrada automática de dados contribui para um melhor aproveitamento do tempo do usuário, reduzindo o tempo dedicado por ele para realizar entrada de dados.

Todos os testes abordados neste capítulo realizaram-se em um computador com processador Turion X2 1,6 Ghz, 1 GB de memória RAM (533 Mhz), utilizando o sistema operacional Linux Ubuntu 9.04, e Sun Java JDK 6.0.14.

6. CONSIDERAÇÕES FINAIS

Existe uma diversidade de frentes de pesquisas buscando adequar a computação aos sistemas clínicos. Do ponto de vista organizacional dos dados, e da intercambialidade das informações, há diversas propostas de terminologias clínicas, entre elas SNOMED-CT, CID e MEDCIN. Estas buscam efetuar uma padronização dos dados dos ambientes clínicos, dessa forma buscando uma unificação dos termos empregados. Esses dados porém, são tratados por sistemas eletrônicos de saúde, dentre os quais o grande desafio está no intercâmbio das informações, pesquisas relacionados ao tema podem serem encontrados em (OPENEHR.ORG, 2007) e (HL7, 2008).

Já na área da computação pervasiva, várias pesquisas vem sendo realizadas buscando construir o *back-end* computacional necessário para o ambiente pervasivo tornar-se uma realidade, alguns dos trabalhos que tratam do tema são (ROMÁN, 2002) (SOUSA, 2002) e (YAMIN, 2004).

Neste trabalho é tratada a integração destes conceitos, sob uma visão única, capaz de tratar o contexto associado as tarefas clínicas executadas pelo profissional clínico (médico). Para se conseguir atingir tal objetivo vários desafios foram encontrados.

O primeiro deles foi identificar o que realmente o sistema de contexto deveria tratar. Para se chegar ao modelo proposto neste trabalho foi necessário buscar informações e analisar diversos aspectos referentes aos sistemas clínicos, incluindo terminologias clínicas, modelos arquiteturais (OPENEHR.ORG, 2007) (HL7, 2008), modelos de comunicação, características do meio clínico, além dos *middlewares* pervasivos mencionados no decorrer deste trabalho. Para cada um deles há um conjunto bastante grande de informações, sendo que identificar quais ideias e conceitos poderiam serem reaproveitados ou modificados para serem inseridos na arquitetura ClinicSpace foi um grande desafio.

Posteriormente, vencida a etapa de especificação e projeto da arquitetura, o tratamento da personalização dos sensores no EXEHDA foi uma tarefa bastante desafiadora. Em função da camada de monitoramento de sensores do EXEHDA ter sido projetada somente para tratamento de sensores globais, várias modificações foram necessárias para adequa-lo ao uso relativo aos sensores personalizados, proposto neste trabalho. As modificações necessárias foram modeladas de forma a integrar os diversos componentes que compõem a arquitetura

para entrada automática de dados. Nesse aspecto julgou-se importante interferir o mínimo possível no código já implementado, agregando tanto quanto possível camadas sobre ele que promovam as funcionalidades (abstrações) buscadas. Mesmo assim, foram necessárias alterações no próprio *middleware*, o que foi um processo moroso em função deste ser um esforço independente deste trabalho, sendo necessário resgatar todos os conceitos envolvidos na sua construção.

Dentro deste escopo, esse trabalho propôs e implementou o protótipo de um modelo de contexto associado às tarefas, onde o foco está na capacidade de programação orientada ao usuário-final, bem como na automação da entrada implícita de dados, liberando o usuário de fornecer explicitamente tais informações. Através dos testes realizados sobre o protótipo desenvolvido, verificou-se que a arquitetura é viável, atendendo aos requisitos aos quais se propôs. A entrada implícita de dados, permitida através da personalização dos sensores do sistema de monitoramento, agrega pró-atividade ao sistema, antecipando-se ao usuário na busca de informações baseadas no perfil da aplicação executada.

6.1. Trabalhos futuros

Sob os aspectos discutidos na modelagem apresentada no capítulo 4, identifica-se vários aperfeiçoamentos que podem ser realizados, sugerindo-se como trabalhos futuros:

a) aprofundar a questão do tratamento de arquétipos e *templates*. O primeiro tem como modelo inicial o uso da linguagem ADL (*Archetypes Definition Language*) (BEALE, 2009), sugerindo-se como trabalho futuro a validação dessa linguagem através da construção de arquétipos de um conjunto piloto, contendo os dados mais utilizados no meio clínico. Da mesma forma, sugere-se a modelagem de uma linguagem para expressão dos *templates*, bem como a modelagem de um conjunto piloto de *templates* correspondentes ao conjunto inicial de tarefas propostas.

b) construir um modelo ontológico, responsável por expressar o contexto de tarefa, incluindo os dados de atuadores, *templates* e arquétipos associados a uma tarefa clínica. Esse modelo deverá fornecer como resultado o protocolo de especificação para os dados do sistema de contexto expressos no APÊNDICE C – Definição XML Schema para uma classe de dado.

Vale frisar, que o sistema no modelo como ele foi concebido, permite que seja

possível a adição dinâmica de novos sensores, atuadores e equipamentos ao ambiente, dessa forma estendendo-o ou mesmo convergindo diversos ambientes anteriormente separados para um único, através de uma integração realizadas pelos serviços do ClinicSpace de forma transparente ao usuário.

Outros componentes da arquitetura ClinicSpace poderão utilizar-se das funcionalidades oferecidas pela arquitetura desenvolvida neste trabalho, visto que a arquitetura é dinâmica e permite a adição de sensores e atuadores em tempo de execução, possibilitando sua utilização, por exemplo, em mecanismos de inferência, que poderão registrar atuadores para determinadas situações de contexto encontradas no sistema.

O modelo desenvolvido neste trabalho, bem como os pontos sugeridos como trabalhos futuros deverão formalizar um modelo completo do sistema, o qual deverá possibilitar sua aplicação, na prática, em um ambiente clínico. Contribuindo para construir um ambiente que promova maior facilidade de uso, eficiência na busca automática de dados, e adaptação ao usuário através da personalização.

Por fim, cabe salientar que a modelagem inicial da arquitetura ClinicSpace não possui atualmente todos seus componentes construídos, em virtude da sua complexidade adotou-se uma abordagem de desenvolvimento incremental; logo, alguns componentes encontram-se em fase de desenvolvimento, outros em fase de projeto. Os sistemas constituintes da arquitetura estão sendo testados de forma individual, através da simulação da interação com os demais componentes. Em função disso, a validação do sistema junto aos usuários clínicos é deixada como uma tarefa futura, a qual deverá ser realizada tão logo estejam disponíveis os componentes atualmente inacabados.

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.

CLEI 2009 - “Ferramenta para a Programação pelo Usuário-Final de Tarefas Clínicas em um Ambiente de Saúde Ubíquo”. In: XXXV Conferencia Latino americana de Informática (XXXV CLEI).

(a submeter) “Projeto de um Ambiente de Contexto Personalizável e Orientado a Tarefas na Arquitetura ClinicSpace”.

REFERÊNCIAS

ABC. Activity-Based Computing. 2009. Disponível em: <<http://www.activity-based-computing.org/>>. Acesso em: 14 mai. 2009.

AUGUSTIN, Iara. **ClinicSpace: auxílio às tarefas clínicas em um ambiente hospitalar do futuro baseado em tecnologias da Computação Ubíqua/Pervasiva**, projeto CNPQ. 2008-2011;

AUGUSTIN, Iara. **Abstrações para uma linguagem de Programação visando Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing**. 2004. 194 p. Tese de doutorado, Universidade Federal do Rio Grande do Sul. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS.

BARDRAM, Jakob E. et. al. “Applications of Context-Aware Computing in Hospital Work – Examples and Design Principles”, **Symposium on Applied Computing 2004**. Páginas: 1574 – 1579, ISBN:1-58113-812-1. 2004a.

BARDRAM, Jakob; E., CHRITENSEN, H.B., and OLESEN, A. K. Activity-driven computing infrastructure – pervasive computing in healthcare. **Technical Report CfPC 2004-PB-65**, Centre for Pervasive Computing, Aarhus, Denmark. 2004b.

BARDRAM, Jakob E. “The Java Context Awareness Framework (JCAF) A Service Infrastructure and Programming Framework for Context-Aware Applications”. **IEEE Pervasive Computing (2005)**, pp. 98-115. 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, no. 1, pp. 44-51, Jan-Mar, 2007.

BEALE, T. 2002. Archetypes: Constraint-based domain models for future-proof information systems. In **Workshop on Behavioural Semantics (OOPSLA'02)**. Disponível em <http://www.deepthought.com.au/it/archetypes/archetypes-new.pdf>. Acesso em 3 de janeiro de 2009.

BT - SNOMED BROWSER by BT. Acesso em dezembro de 2008. Disponível em <http://www.jdet.com>. 2008.

CASSENS, J. and A. Kofod-Petersen, 2006, "Using Activity Theory to Model Context Awareness: a Qualitative Case Study", **Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference**, Florida, USA. 2006.

CASTILHA, André Coutinho. "**Instrumento de Investigação Clínico-Epidemiológica em Cardiologia Fundamentado no Processamento de Linguagem Natural**". Tese de Doutorado, para obtenção do Título de Doutor em Ciências, 2007, Faculdade de Medicina da Universidade de São Paulo, BR. 2007.

CEN - European Committee for Standardization. Disponível em <http://www.cen.eu/CENORM/Sectors/TechnicalCommitteesWorkshops/CENTechnicalCommittees/CENTechnicalCommittees.asp?param=6232&title=CEN%2FTC+251>, acesso em janeiro de 2008.

CHEN, G. **Solar: Building A Context Fusion Network for Pervasive Computing**. 2004. Tese de doutorado – Dartmouth College, Hanover, New Hampshire, USA. 2004.

CHEN, G. Li M, KOTZ, D. Design and implementation of a large-scale context fusion network. 2004 **The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. MOBIQUITOUS 2004**. p 246- 255. Disponível em http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1331731>. 2004b.

CHEN, G. KOTZ, D. Solar: A pervasive-computing infrastructure for context-aware mobile applications. **Technical Report TR2002-421**, February 2002. Disponível em <ftp://ftp.cs.dartmouth.edu/TR/TR2002-421.pdf>>. Acesso em: jun. 2004. 2002.

CHEN, G.; KOTZ, D. A Survey of Context-aware Mobile Computing Research. 2000. **TECHNICAL REPORT TR2000-381**. Dartmouth Computer Science. Disponível em <http://www.cs.dartmouth.edu/~solar/>>. Acesso em julho de 2005. 2005.

DAHL, Yngve. Et al. "Context in Care – Requirements for Mobile Context-Aware Patient Charts". **MEDINFO 2004**. M. Fieschi et Al (Eds). Amstern IOS Press. 2004.

DATALINE - "SNOMED Browser Version 2.0", disponível em <http://snomed.dateline.co.uk/>,

acessado em janeiro de 2009.

EICHELBERG, Marco; ADEN, Thomas; RIESMEIER, Jorg; "A Survey and Analysis of Electronic Healthcare Record Standards" **ACM Computing Surveys**, Vol. 37, No. 4, pp. 277–315., December 2005.

ENGESTROM, Y., Miettinen, R; PUNAMAKI, R.L. (eds.). "Perspectives on Activity Theory." **Cambridge University Press**, 1999.

FEHLBERG, Felipe w. "**MultiS: Um servidor de Contexto voltado à Computação Pervasiva**". Dissertação de Mestrado em Ciência da Computação na UFRGS, abril de 2007.

FERREIRA, Giuliano Lopes. "**Adaptando o Middleware EXEHDA para o Suporte a Aplicações Orientadas à Atividades**". Dissertação de Mestrado, para obtenção do Título de Mestre em Computação, 2009, UFSM, BR.

HL7 - Health Level 7, obtido em <http://www.hl7.org/>, acesso em janeiro de 2008.

HANSEN, Tomas R. Et al. "Pervasive Interaction – Using movement and speech to interact with computers". **Workshop - "What is the Next Generation of Human-Computer Interaction?"** held on Sunday, April 23, 2006, at CHI 2006 in Montreal. 2006.

IAKOVIDIS, I. 1998. "Towards personal health records: Current situation, obstacles and trends in implementation of Electronic Healthcare Records in Europe". **Int. J. Medical Informatics** 52, 128, p. 105–117. 1998.

IHE - Integrating Healthcare Enterprise, Disponível em: <http://www.ihe.net/>, acesso em janeiro de 2008.

IHETF - Integrating Healthcare Enterprise Technical Framework, http://www.ihe.net/Technical_Framework/, acesso em janeiro de 2008.

ISAM. Infra-estrutura de Suporte às Aplicações Móveis. 2001. Disponível em: <http://www.inf.ufrgs.br/~isam>>. Acesso em: 10 agos. 2008.

ISOTC 215 - ISO Technical Committee for Health Informatics, <http://isotc.iso.org/livelink/livelink?func=ll&objId=529137&objAction=browse&sort=name>, acesso em janeiro de 2008.

JAHNKE, Jens H., "Toward Context-Aware Computing in Clinical Care ", **OOPLSA 2005 (Object-Oriented Programming, Systems, Languages, and Applications.)**, outubro de 2005.

JUDD, Glenn; STEENKISTE, Peter. "Providing Contextual Information to Pervasive Computing Applications", **Proceedings of the First IEEE International Conference on Pervasive Computing and Communications**, Página 133, 2003. ISBN:0-7695-1893-1 . 2003.

LAERUM, Hallvard; Faxvaag, Arild. " Task-oriented evaluation of eletronic medical record systems: development and validation of a questionnaire for physicians". **BMC Medical Informatics and Decision Making 2004**, 4:1. Published: 09 February 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.

KAASINEN, Eija. "User needs for location-aware mobile services", **Personal and Ubiquitous Computing**, Volume 7, Issue 1, May 2003, Pages 70 – 79 . 2003.

KAENAMPORNPAN, Manasawee; O'Neill, Eamonn. "Integrating History and Activity Theory in Context Aware System Design". In **Proceedings of the 1st International Workshop on Exploiting Context Histories in Smart Environments (ECHISE)**. 2005.

KOCH, Oliver. "Process-based and context-sensitive information supply in medical care ", **Proceedings of the 2nd International Workshop on Context-Based Information Retrieval** , August, 2007.

KOFOD-PETERSEN, Anders; Cassens, Jorg. "Using Activity Theory to Model Context Awareness" ISBN: 978-3-540-33587-0, **Springer Berlin / Heidelberg**, Volume 3946/2006. 26 de Abril de 2006.

NARDI, Bonnie. Context And Consciousness: Activity Theory And Human-computer Interaction, **Chapter 4 - Studying Context: A comparison of Activity Theory, Situated Action Models, and Distributed Cognition**, Cambridge, Ma: Mit Press, 1996, 400 pp.

NCI - National Cancer Institute US. "NCI Terminology Browser". Disponível em <http://nciterms.nci.nih.gov/NCIBrowser/>, acessado em dezembro de 2008.

NLM - Unified Medical Language System "SNOMED CT Release Files", disponível em <http://www.nlm.nih.gov/research/umls/licensedcontent/snomedctfiles.html>, obtido em janeiro de 2009.

NI, Hongbo. Et al. "Context-Dependent Task Computing in Pervasive Environment" .Y. Youn, M. Kim, and H. Morikawa (Eds.): UCS 2006, LNCS 4239, pp. 119 – 128, 2006. Springer-Verlag Berlin Heidelberg 2006.

OPENEHR.ORG - Architecture Overview, obtido em <http://www.openehr.org/>, acessado em Agosto de 2007.

RANGANATHAN, Anand; Campbell, Roy H. Supporting Tasks in a Programmable Smart Home. In: ICOST 2005 : 3rd **International Conference On Smart homes and health Telematic**. 2005.

RANGANATHAN, Anand. Campbell, Roy H . An infrastructure for context-awareness based on first order logic. **Personal and Ubiquitous Computing** archive Volume 7, Issue 6 (December 2003). Pages: 353 – 364. Year of Publication: 2003. ISSN:1617-4909. 2003.

ROMÁN, Manuel, et. al. Gaia: a middleware infrastructure to enable active spaces. **IEEE Pervasive Computing**, pp. 74-83, Oct-Dec 2002.

SILVA, Luciano Cavalheiro da. **Primitivas para Suporte à Distribuição de Objetos Direcionadas à Pervasive Computing**, Dissertação de Mestrado em Ciência da Computação na UFRGS, maio de 2003.

SILVA, Fábio Lorenzi. **Modelagem de uma Ferramenta para Definição de tarefas clínicas em um ambiente de computação baseada em Tarefas e Direcionada ao Usuário Final**.

Dissertação de Mestrado, para obtenção do Título de Mestre em Computação, 2009, UFSM, BR.

SOUSA, J. and Garlan, D. Aura: An architectural framework for user mobility in ubiquitous computing environments, **The Working IEEE/IFIP Conference on Software Architecture (WICSA) 2002**.

UMLS. Unified Medical Language System Fact Sheet. Disponível em <http://www.nlm.nih.gov/pubs/factsheets/umls.html>, acesso em janeiro de 2009.

VICENTINI, C. et al. Proposta de Arquitetura para Sistemas de Registro de Saúde Ubíquo. 2009 (submetido para publicação).

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

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.

WANG, Amy Y. et al. The SNOMED Clinical Terms Development Process: Refinement and Analysis of Content. **Proc AMIA Symp.** 2002; p. 845–849.

WEISER, M. The Computer for the 21st Century. **Scientific American**, September 1991.

APÊNDICE A - Conjunto Mínimo de Tarefas

O conjunto mínimo de tarefas é baseado no estudo “Task-oriented evaluation of electronic medical record systems: development and validation of a questionnaire for physicians.” (LAERUM, 2004).

A.1 Conjunto Mínimo de Tarefas

Na Tabela 4 - Conjunto Mínimo de Tarefas é apresentado o conjunto mínimo de tarefas, bem como sua definição e classificação.

Tabela 4 - Conjunto Mínimo de Tarefas

N.	Tarefa	Definição	Classificação da Atividade
1	Revisar os problemas do paciente	Obter informações suficientes para formular os principais problemas dos pacientes para então poder realizar, ou requisitar, novas investigações ou tomar decisões clínicas.	Diagnóstico
2	Procurar informações específicas em registros do paciente (prontuário)	Procurar específicas e limitadas quantidades de informações sobre o paciente nos registros do paciente.	Diagnóstico
3	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

N.	Tarefa	Definição	Classificação da Atividade
4	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
5	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
6	Obter resultados de exames clínicos (análises laboratoriais)	Obter resultados novos, e velhos, de exames clínicos executados em laboratórios bioquímicos.	Laboratorial
7	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
8	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
9	Requisitar tratamentos	Requisitar procedimentos de tratamentos como medicamentos, cirurgias, etc. a serem executadas no hospital, e que geralmente não sejam administradas pelos pacientes.	Tratamento
10	Prescrever receitas	Receitar medicamentos (ou outros tipos de tratamentos auto-	Tratamento

N.	Tarefa	Definição	Classificação da Atividade
		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.	
11	Registrar códigos para diagnósticos ou procedimentos executados	Executar seleções em vários sistemas de classificação para os procedimentos clínicos executados ou para diagnósticos, e documentar a seleção.	Administrativa

A.2 Sub-Tarefas

Nesta seção são apresentadas as tarefas que compõem o conjunto mínimo de tarefas, bem como suas sub-tarefas internas.

1. Revisar os Problemas do Paciente:

1. ID profissional;
2. ID Paciente;
3. EHR - Busca Informação do Paciente;
4. EHR - Visualização de Informações;

2. Procurar Informações Específicas no Registro do Paciente

1. ID Profissional;
2. ID Paciente;
3. EHR - Busca Informação do Paciente;
4. EHR - Visualização de Informações.

3. Obter os Resultados de novos Testes e Investigações

1. ID Profissional;
 2. ID Paciente;
 3. EHR - Busca Informação do Paciente (Parâmetro: Informações novas);
 4. EHR - Visualização de Informações;
4. Adicionar notas diárias sobre condições do paciente
1. ID Profissional;
 2. ID Paciente;
 3. EHR - Registra nota diária;
5. Requisitar análises clínicas em laboratórios de análises clínicas.
1. ID Profissional;
 2. ID Paciente;
 3. EHR - Preencher requisição laboratorial; (preenchimento + encaminhamento + registro no PEP);
6. Requisitar raios-X, ultra-som e tomografias computadorizadas
1. ID Profissional;
 2. ID Paciente;
 3. EHR - Preencher requisição de exames de vídeo/imagem; (preenchimento + encaminhamento + registro no PEP);
7. Obter resultados laboratoriais
1. ID Profissional;
 2. ID Paciente;
 3. EHR - Busca Informação do Paciente (Parâmetro: Requisições Laboratoriais);
 4. EHR - Visualização de Informações laboratoriais;
8. Obter resultados de raios-X, ultra-som e tomografias computadorizadas
1. ID Profissional;
 2. ID Paciente;
 3. EHR - Busca Informação do Paciente (Parâmetro: Requisições de vídeo/imagem);

4. EHR - Visualização de Informações de exames de vídeo/imagem.

9. Requisitar Tratamentos

1. ID Profissional;
2. ID Paciente;
3. EHR - Preencher de requisição de tratamento (preenchimento + encaminhamento + registro no PEP).

10. Prescrever receitas

1. ID Profissional;
2. ID Paciente;
3. EHR - Preencher prescrição (preenchimento + registro no PEP + impressão).

11. Registrar Códigos para diagnósticos e procedimentos executados

1. ID Profissional;
2. ID Paciente;
3. EHR - Registro de diagnósticos e procedimentos executados.

APÊNDICE B – MIDDLEWARE EXEHDA

O EXEHDA (*Execution Environment for High Distributed Applications*) é um middleware pervasivo, construído sobre a ótica de interação entre diversos serviços disponibilizados de forma modular. A arquitetura pervasiva é fornecida através da utilização de quatro subsistemas distintos: subsistema de execução distribuída, Subsistema de Reconhecimento de Contexto e Adaptação, Subsistema de Comunicação e Subsistema de Acesso Pervasivo. A seguir são apresentados os serviços constituintes desses sistemas.

B.1 Subsistema de Execução Distribuída

Através da interação com os demais subsistemas do *middleware*, o Sistema de Execução Distribuída é responsável por fornecer o suporte à execução e ao processamento distribuído. É constituído pelos serviços relacionados a seguir.

Executor: Tem como finalidade tratar do disparo de aplicações bem como criação e migração de seus objetos. Constrói e fornece uma interface que dá suporte as operações realizadas no ciclo de vida de uma aplicação.

Cell Information Base (CIB): Implementa a base de informação da célula, mantendo as informações sobre usuários e recursos, bem como as características dessas entidades. Controla a tipificação dos nomes dessas entidades, agregando informações dos espaços onde estão presentes, gerando nomes únicos, os quais representam um mapeamento hierárquico do ambiente.

OXManage: Fornece a gerência e manutenção das meta-informação associadas a um OX (*Object eXehda*), o qual representa uma aplicação criada pelo serviço *executor* sobre o *middleware*. Este serviço confere às operações de consulta e atualização dos atributos do OX, oferecendo o caráter pervasivo necessário, permitindo que estes sejam acessados a partir de qualquer nodo do ISAMpe.

Discoverer: É responsável pela localização de recursos especializados no ISAMpe a partir de especificações abstratas dos mesmos. As especificações caracterizam o recurso a ser descoberto por meio de atributos e seus respectivos valores. Adicionalmente, a requisição de

descoberta de recurso incorpora um parâmetro que define a amplitude da pesquisa. A interface do serviço `Discoverer` define operações para o registro e remoção de recursos além da pesquisa destes. Os recursos registrados no `Discoverer` são catalogados no serviço CIB (*Cell Information Base*), ficando a partir de então visíveis no ISAMpe.

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, localizar recursos especializados e alocar nodos de processamento;

Gateway: Realiza a intermediação das comunicações entre os nodos externos à célula e os recursos internos a ela. Da sua ação integrada com o *ResourceBroker* decorre o controle de acesso aos recursos de uma `EXEHDAcell`. Pode-se sintetizar que o controle de acesso entre as células é realizado através da união das funcionalidades oferecida pelo *Gateway* com o *ResourceBroker*. A interface do serviço *Gateway*, define três métodos direcionados à concessão, revogação e renovação de permissões de acesso.

StdStreams: Provê suporte ao redirecionamento dos *streams* padrões: entrada, saída e erro. Através da associação destes à aplicação, agrupados sob a forma de um Console. Assim, podem-se redirecionar esses *streams* sem a necessidade de modificar a aplicação, bastando apenas configurar os atributos de execução do serviço;

Logger: Disponibiliza métodos para registro de mensagens, classificando-as segundo níveis de prioridade. Além do método básico `log`, através do qual o usuário do serviço pode explicitar o nível de prioridade das mensagens, inclui também facilidades para o registro de mensagens com níveis de prioridade mais usualmente encontrados: situação crítica, erro, alerta, mensagem informativa e depuração. Esse sistema é especialmente útil na fase de desenvolvimento da aplicação, bem como rastreamento e auditoria para uma ambiente em produção.

Dynamic Configurator (DC): Tem a função de realizar a configuração do perfil de execução do *middleware* em um `EXEHDA`nodo 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.

B.2 Subsistema de Reconhecimento de Contexto e Adaptação

O Subsistema de Reconhecimento de Contexto e Adaptação inclui serviços de aquisição de informações sobre o ambiente pervasivo, identificação em alto nível dos elementos de contexto e de disparo das ações de adaptação. Essas funcionalidades são disponibilizadas pelos serviços descritos à seguir.

Collector: É responsável pela extração da informação bruta (diretamente dos recursos envolvidos) que, posteriormente refinada, dará origem aos elementos de contexto. Para isto, o serviço *Collector* agrupa informações oriundas de vários monitores e as repassa para os consumidores registrados. Esse repasse pode ser feito via *callback* ou via canais de *multicast* providos pelo serviço *Deflector*. O conjunto de sensores de um *EXEHDA*nodo, 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: É responsável pelo tratamento das informações brutas produzidas pelo sistema de monitoramento (*Collector*), produzindo informações com uma semântica agregada para os 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*. A aplicação interessada é notificada

sobre eventos que geram mudança nos estados de contexto, podendo cancelar sua subscrição à qualquer momento.

AdaptEngine: Fornece o controle das adaptações de cunho funcional. Este serviço, oferece uma interface que provê facilidades para definição e gerência de comportamentos adaptativos por parte das aplicações. Deste modo, libera o programador de gerenciar os aspectos de mais baixo nível envolvidos na definição e liberação dos elementos de contexto junto ao *ContextManager*.

Scheduler: Responsável pela gerência das adaptações de cunho não-funcional no EXEHDA, isto é, que não implicam alteração de código. Para isso, o *Scheduler* emprega a informação de monitoração, obtida junto ao serviço *Collector*, para orientar operações de mapeamento. Essas operações decorrem de instanciações remotas ou migrações realizadas pelo serviço Executor, ou quando de chamadas de re-escalonamento, originadas do estado atual de um recurso não satisfazer mais as necessidades de um objeto anteriormente a ele alocado.

B.3 Subsistema de Comunicação

A natureza da mobilidade do hardware e, na maioria das vezes, também a do software, não garante a interação contínua entre os componentes da aplicação distribuída. As desconexões são comuns, não somente devido à existência de alguns *links* sem fio, mas sobretudo como uma estratégia para economia de energia nos dispositivos móveis. O subsistema de comunicação do EXEHDA disponibiliza mecanismos que atendem estes aspectos da Computação Pervasiva. Integram este subsistema os serviços *Dispatcher*, *WORB*, *CCManager*, os quais contemplam modelos com níveis diferenciados de abstração para as comunicações.

Dispatcher: Esse serviço oferece um modelo de comunicação através da troca de mensagens ponto-a-ponto, oferecendo ordenamento e garantia de entrega. Quando inicializado, o *Dispatcher* atualiza as informações do *EXEHDA* nodo 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. Fornece essa funcionalidade através de um modelo de comunicação baseado em invocações remotas, semelhante ao RMI, porém se a necessidade de manutenção da conexão durante a execução.

CCManager: Disponibiliza um mecanismo de comunicação baseado na abstração espaço de tuplas, o qual oferece um mecanismo assíncrono de comunicação entre 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.

B.4 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 a qualquer tempo, tanto para dados quanto código. Compõem esse subsistema os serviços:

BDA: Consiste no serviço Base de Dados para as aplicações pervasivas. O código destas é disponibilizado nessa base permitindo que seja carregado e instanciado, sob demanda, a qualquer tempo e em qualquer dispositivo.

AVU: Corresponde ao serviço de Ambiente Virtual, o qual é responsável pela manutenção do acesso pervasivo ao ambiente computacional do usuário. Este é constituído por 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 *fetch* 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.

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.

APÊNDICE C – Definição XML *Schema* para uma classe de dado

O formato do XML interpretado pelo Sistema de Gerenciamento de Contexto de Tarefas, segue as regras estabelecidas pelo XML *Schema* listado no Quadro 7. Nota-se que esse formato é gerado por um componente que faz a tradução dos arquétipos expressos na base de dados de contexto, o qual é sugerido como um trabalho futuro, sendo simulado para realização deste trabalho.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="DATACLASS">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="STATES"/>
        <xs:element ref="FILTER"/>
      </xs:sequence>
      <xs:attribute name="n" use="required" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="STATES">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="STATE"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="STATE">
    <xs:complexType>
      <xs:attribute name="n" use="required" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="FILTER">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="MAPPINGS"/>
      </xs:sequence>
      <xs:attribute name="type" use="required" type="xs:NCName"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="MAPPINGS">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="RANGE"/>
        <xs:element ref="DEFAULT"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<xs:element name="RANGE">
  <xs:complexType>
    <xs:attribute name="lbo" type="xs:integer"/>
    <xs:attribute name="state" use="required" type="xs:NCName"/>
    <xs:attribute name="ub" type="xs:integer"/>
  </xs:complexType>
</xs:element>
<xs:element name="DEFAULT">
  <xs:complexType>
    <xs:attribute name="state" use="required" type="xs:NCName"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Quadro 7 - XML Schema de uma Classe de Dados.