Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Proposta de um *Middleware* para Monitoramento de situações de emergência em Ambientes Físicos ou Lógicos Cientes de Contexto

ALUNA: GISLAINE CRISTINA MICHELOTI

ORIENTADORA: REGINA B. ARAÚJO

SÃO CARLOS - SP

JULHO – 2004

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Proposta de um *Middleware* para Monitoramento de situações de emergência em Ambientes Físicos ou Lógicos Cientes de Contexto

Gislaine Cristina Micheloti

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Departamento de Computação, da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Processamento de Imagens e Sinais - PIS.

SÃO CARLOS - SP

JULHO – 2004

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

M623pm

Micheloti, Gislaine Cristina.
Proposta de um *middleware* para monitoramento de situações e emergência em ambientes físicos ou lógicos cientes de contexto / Gislaine Cristina Micheloti. -- São Carlos : UFSCar, 2005.
103 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Sistemas - tipos. 2. Middleware. 3. Ciência do contexto. 4. Segurança crítica. 5. Computação ubíqua. I. Título.

CDD: 003.7 (20ª)

Morre lentamente quem se transforma em escravo do hábito, repetindo todos os dias os mesmos trajetos, quem não muda de marca, não arrisca vestir uma cor nova e não fala com quem não conhece.

Morre lentamente quem faz da televisão seu guru.

Morre lentamente quem evita uma paixão, quem prefere a sombra em vez da luz e os pingos nos iis a um redemoinho de emoções, exatamente a que resgata o brilho nos olhos, o sorriso nos lábios e coração aos tropeços.

Morre lentamente quem não vira a mesa quando está infeliz no trabalho, quem não arrisca o certo pelo incerto, para ir atrás de um sonho.

Morre lentamente quem não se permite, pelo menos uma vez na vida, ouvir conselhos sensatos.

Morre lentamente quem não viaja, não lê, quem não ouve música, quem não encontra graça em si mesmo.

Morre lentamente quem passa os dias queixando-se da sua má sorte, ou da chuva incessante.

Morre lentamente quem destrói seu amor próprio, quem não se deixa ajudar.

Morre lentamente quem abandona um projeto antes de iniciá-lo, nunca pergunta sobre um assunto que desconhece e nem responde quando lhe perguntam sobre algo que sabe.

Evitemos a morte em suaves porções, recordando sempre que estar vivo exige um esforço muito maior que o simples ar que respiramos.

Somente com infinita paciência conseguiremos a verdadeira felicidade.

Pablo Neruda

À minha querida mãe, Lourdes, pela lição de paciência que me dá através de seus atos e pelo exemplo de coragem e garra que me incentivou a lutar por meus objetivos quando tudo parecia muito difícil.

Ao meu pai, Florivaldo, pelo incentivo e oportunidade que sempre me deu.

Ao Fabrício, meu noivo, por toda paciência que teve comigo e pelo apoio em todos os momentos.

Agradecimentos

À Deus por estar sempre comigo, me guiando e ajudando em todos os momentos da minha vida.

À profª Drª Regina Borges de Araújo, minha orientadora, pela oportunidade e confiança na realização deste trabalho.

À minha família que sempre me apoiou em todas as dificuldades.

Ao meu noivo, Fabrício Pini Rosales, que me ajudou a superar todos os obstáculos que surgiram no decorrer do mestrado.

À Lucilda pelo apoio constante e por toda força que me deu desde o início.

À todos os meus colegas do laboratório (Richard, Goiano, Fernando, Diego, Erlon e Carlos) e um agradecimento especial para a Taciana que sempre soube me ouvir e dizer palavras de carinho e amizade quando eu precisei.

À todos os funcionários, em especial a tia Ofélia, Vera, Cristina e Mirian pelos incentivos constantes.

RESUMO

A computação ubíqua, onde informações e serviços são oferecidos aos usuários de forma contínua e não intrusiva, é amplamente explorada em aplicações de sala de aula, sala de reuniões, guias turísticos, residências e também na área de segurança crítica: segurança industrial, segurança na aviação, segurança no controle metro-ferroviário, monitoramento de pacientes, etc. O avanço nas áreas de dispositivos heterogêneos de visualização e acesso à informação, de sensores e de redes de comunicação sem fio, tem impulsionado o surgimento de aplicações que necessitam de monitoramento de granularidade fina na ajuda à prevenção, combate e avaliação de situações de emergência. No primeiro caso (prevenção), informações podem ser capturadas fazendo com que condições críticas possam ser evitadas, no segundo caso (combate), podem ajudar com informações mais detalhadas para tomadas de decisão no gerenciamento de equipes de resgate e salvamento (por exemplo, na localização de pessoas em ambientes tomados por fumaça e/ou chamas), e no terceiro caso (avaliação), auxiliar perícias por parte de seguradoras, treinadores de equipes de combate a acidentes, e até os responsáveis pela segurança do ambiente monitorado.

Sistema de Monitoramento de Condições de Emergência em Ambientes Cientes de Contexto é, considerado neste trabalho, um ambiente físico ou lógico povoado por sensores que capturam informações de contexto que são interpretadas, gravadas e visualizadas em ambientes virtuais 3D que refletem o ambiente real.

Este trabalho apresenta a proposta de um *middleware* independente de plataforma e linguagem de programação que atende requisitos funcionais de sistemas cientes de contexto, tais como interpretação de contexto, localização, adaptação, etc., bem como requisitos não funcionais, tais como confiabilidade, eficiência, modularidade, portabilidade, interoperabilidade e extensibilidade. O *middleware* é especificado na forma de serviços que interagem com a aplicação, com a rede de sensores e entre si, através de mecanismo de publicação/subscrição baseado em tópicos em que subscrições e notificações são emitidas e descritas em XML, e transmitidas pela rede através do protocolo SOAP, tornando o sistema acessível via *web*. Os serviços podem estar integrados e executando em um único computador, ou então distribuídos pela rede, e podem ser executados em nós da rede de sensores.

ABSTRACT

Ubiquitous computing in which information and services are offered to users in a continuous and non-intrusive way, is widely explored for classroom, meeting rooms, tourist guides, intelligent home and also safety-critical applications: industrial, aviation, metro-train safety, patient monitoring, etc. The advances in heterogeneous visualization and information access devices, sensors and wireless networks has propelled the emergence of applications that need fine grain monitoring to prevention, combat and evaluation of emergency situations. In the first case (prevention), information is captured from sensor network in order to prevent critical conditions. In the second case (combat), it can aid with more detailed information to help decision making in the management of rescue teams (for instance, in the localization of people and objects in environments with heavy smoke and/or flames), and, in the third case (evaluation), to help auditory for insurance companies, training teams, and even people responsible for the safety of the environment being monitored. In this work, a system for Monitoring Emergency Conditions in context aware environments is considered as being a physical or logical environment with sensors across it that capture context information, which are interpreted, recorded and visualized in 3D virtual environments that mimic the real environment.

This work presents a proposal for a Middleware, which is platform and programming language independent that meets functional requirements of context aware systems, such as context interpretation, localization, adaptation, etc., as well as non-functional requirements, such as: reliability, efficiency, modularity, portability, interoperability and extensibility. The middleware is specified as a set of services that interact with the application, with a sensor network and with services themselves. The interaction is achieved through a publish/subscribe mechanism based on topics in that subscriptions and notifications are issued and described in XML, and transmitted through the SOAP protocol, making the system accessible from the web. The services can be integrated and executing in a single computer, or they can be distributed across the network and even ran in nodes of a sensor network.

Sumário

Lista de Figuras

1 Introdução

1.1 Considerações Iniciais

Com a maior abrangência das tecnologias de comunicação sem fio, da evolução da tecnologia de sensores, da microeletrônica, da miniaturização de dispositivos móveis, e do surgimento de arquiteturas de *software* flexíveis, novas áreas de pesquisas estão sendo vislumbradas **[LYY 02],** possibilitando que os computadores sejam usados cada vez mais em ambientes variados e tenham maior ciência do mundo dinâmico que eles fazem parte **[WEI 91]**. De fato, durante os últimos anos, foi desenvolvida uma nova classe de aplicações, dentre elas a computação ubíqua ciente de contexto, que faz uso destas tecnologias, em que computadores empregam noções elementares de localização, identidade, proximidade, entrega da aplicação apropriada a cada usuário presente em um ambiente de computação baseando-se na captura dos contextos percebidos do ambiente físico ou lógico de interação **[ABO 00]**.

Domínios de aplicações mais úteis em computação ubíqua ciente de contexto são voltados basicamente para as áreas de ensino e reuniões **[BAL 03]** e na área de segurança de modo geral. O foco deste trabalho, no entanto, é o monitoramento de condições de emergência em ambientes de interação cientes de contexto, que a partir da evolução de sensores, cada vez mais poderosos e baratos, tornam viável um monitoramento de granularidade fina destes ambientes auxiliando na prevenção, combate e avaliação de situações de emergência com o objetivo de minimizar situações de perigo (como incêndio, vazamento de substâncias tóxicas, inundação, explosão) que colocam em risco vidas humanas e perda de patrimônio. Estes ambientes físicos ou lógicos devem estar povoados por sensores

que capturam informações de contexto de todas as entidades que forem definidas como relevantes para o monitoramento, inclusive do próprio ambiente. O sistema de monitoramento de granularidade fina pode auxiliar, em tempo real, equipes de resgate na tomada de decisões durante o salvamento e combate e, posteriormente, em situações de perícia e avaliação para apuração dos fatos e causas que ocasionaram uma situação perigosa ou crítica.

A classe de aplicações que categoriza situações críticas, em que se enquadra o monitoramento de condições de emergência em ambientes cientes de contexto, depende do atendimento de certos requisitos não funcionais considerados críticos, uma vez que a não observação destes requisitos, tais como confiabilidade, eficiência e integridade das informações, conduzem ao mau funcionamento do sistema que podem levar a situações inseguras. Estes requisitos incluem a proteção à vida humana, ao meio ambiente ou dos recursos materiais.

Para sistemas de monitoramento de condições de emergência, observa-se a necessidade de uma estrutura que ofereça suporte e serviços que atendam aos requisitos de computação ubíqua e os requisitos não funcionais mencionados.

1.2 Objetivos do Trabalho e Justificativas

O objetivo deste trabalho é apresentar um ***middleware*** para suporte a sistemas de monitoramento de condições de emergência em ambientes cientes de contexto. Atualmente, existem várias soluções para prover suporte a ambientes ubíquos cientes de contexto, e algumas dessas soluções **[ROM 02]** focam o desenvolvimento de *middlewares*. No entanto, a maioria é voltada para o domínio de aplicações de sala de aula, reuniões e guias turísticos, e

não abordam diretamente as necessidades do monitoramento de condições de emergência, tais como a captura precisa dos contextos por sensores, a interpretação e consistência desses contextos e a notificação imediata das mudanças que ocorrem no ambiente, mesmo na presença de falhas de nós sensores.

Outro objetivo deste trabalho é especificar serviços do ***middleware*** que são independentes entre si, porém podem ser combinados para atender solicitações de forma mais eficiente e confiável, como no caso do Serviço de Manutenção de Consistência, responsável por tratar eventos puros que chegam da rede de sensores de forma a minimizar o tráfego de informações através de eliminação de eventos redundantes e ambíguos.

Os Serviços especificados para o *middleware* são: Serviço de Manutenção de Consistência; Serviço de Interpretação de Contextos, Serviço de Repositório de Dados, Serviço de Gerenciamento de Tópicos, Serviço de Ordenação de Eventos, Serviço de Localização e a identificação da necessidade do Serviço de Adaptação de Conteúdo. O objetivo da especificação destes serviços é atender aos requisitos de computação ubíqua ciente de contexto e aos requisitos não funcionais de aplicações de monitoramento de condições críticas.

1.3 Organização da Dissertação

Este trabalho está organizado da seguinte forma: o **capítulo 2** descreve computação ubíqua, seus principais desafios e os domínios de aplicação com ênfase no monitoramento de condições de emergência em ambientes cientes de contexto. O **capítulo 3** apresenta um estudo na literatura sobre as estruturas existentes para o suporte às aplicações de computação

ubíqua ciente de contexto. O **capítulo 4** introduz e classifica os requisitos de sistemas, especificando os principais requisitos não funcionais para o sistema de monitoramento de condições de emergência em ambientes cientes de contexto. O **capítulo 5** apresenta o *middleware* e os seus serviços para o sistema de monitoramento. O **capítulo 6** apresenta as conclusões deste trabalho e propostas futuras, seguido de Referências Bibliográficas.

2 Computação Ubíqua

2.1 Considerações iniciais

A computação ubíqua objetiva proporcionar informações e serviços computacionais a usuários de forma contínua e transparente a partir da integração de tecnologia a ambientes físicos de interação. Weiser **[WEI 94]** iniciou uma mudança radical no paradigma computacional, ao definir que as pessoas utilizam melhor aquilo que está efetivamente invisível e chamou computação ubíqua essa forma transparente de integração entre tecnologia e atividades humanas. Nesse contexto os computadores são embarcados[1] de forma transparente no ambiente do usuário e essa característica de invisibilidade permite aos usuários perceberem as funcionalidades, mas não quais dispositivos provêem estas funcionalidades **[ROM 00]**.

Este capítulo introduz a computação ubíqua abordando os principais temas de pesquisa e os principais domínios de aplicações explorados nesta área enfocando especialmente o emprego de computação ubíqua no monitoramento de condições de emergência em ambientes físicos ou lógicos de interação.

2.2 Computação Ubíqua

O conceito de computação ubíqua foi introduzido em 1991 pelo pesquisador Mark Weiser **[WEI 91]** que visualizou pessoas e ambientes acrescidos de recursos computacionais que forneceriam informações e serviços quando e onde as pessoas e os ambientes assim os desejassem.

[1] Para fins deste trabalho, o termo *embedded* será tratado como embarcado.

Na visão de Frank Stajano computação ubíqua é "*a visão de um mundo no qual o custo do poder computacional e das comunicações digitais torna-se tão barato a ponto de poderem ser embutidos em todos os objetos que nos cercam no dia-a-dia*" **[STA 02]**.

Uma divisão da evolução da computação, apresentada por Weiser e Brown **[WEI, 1996],** mostra que em **1970** (era do *mainframe*) existiu o conceito de um único recurso computacional compartilhado para várias pessoas, já em **1980** (era do computador pessoal) existiu o conceito de um computador para cada pessoa. A partir da década de **90**, houve uma proliferação de dispositivos, variando em tamanho e formato, unida aos avanços da computação distribuída e da computação móvel que alavancaram para uma nova era, a da Computação Ubíqua, caracterizada pela disponibilidade de vários recursos computacionais para uma única pessoa em um ambiente. Essa afirmação foi confirmada mais tarde (**2002**) por Lyytinen **[LYY 02]** e descreve que a computação ubíqua amplia as capacidades do dispositivo para obter informações do ambiente que está incorporado onde o ambiente também é "inteligente" a ponto de detectar outros dispositivos computacionais que estão "entrando" nele. Esta dependência e interação entre ambiente e dispositivos, resulta em uma nova capacidade de computadores agirem inteligentemente dentro e fora dos ambientes em que nos movemos.

Villate [VIL 02] propõe que num futuro próximo, os dispositivos que hoje nos cercam serão enriquecidos de novas facilidades de uso, pouca manutenção, serão leves e portáteis, e terão capacidades ilimitadas em termos de tamanho, poder de consumo e conectividade intermitente.

As seções seguintes deste capítulo apresentam alguns temas de pesquisa da Computação Ubíqua e os principais desafios nesta área.

2.3 Interfaces Naturais

Interfaces naturais, um tema bastante explorado em computação ubíqua, têm o objetivo de facilitar a comunicação entre usuários e computadores; fornecendo suporte às formas mais comuns de expressão humana e utilizando ações implícitas que ocorrem durante essa comunicação **[ABO 00]**.

A computação ubíqua estuda interfaces naturais com a finalidade de facilitar a interação entre usuários e computador, minimizando quaisquer barreiras de interface física e tornando cada vez mais distante o paradigma *teclado-mouse-tela*. A interface física como, por exemplo, o mouse, não é transparente para o usuário, violando a visão não intrusiva da computação ubíqua.

O estudo que envolve técnicas de interação transparente, incluindo técnicas de reconhecimento de escrita manual e de gestos, interação com canetas, técnicas de fala e percepção computacional, interação com sensores e manipulação de artefatos eletrônicos e, estão começando a complementar ou substituir os elementos do paradigma de interfaces gráficas tradicionais. Essas novas interfaces facilitam o aprendizado e o uso de sistemas sem modificar drasticamente a estrutura das tarefas humanas que são executadas. Como exemplos de pesquisas em interfaces naturais podem-se citar as técnicas de interação baseada em voz de Witbrock e Hauptmann **[WIT 98]** e em escrita de Schilit et al. **[SCH 98]**.

2.4 Computação Ciente de Contexto

A computação ciente de contexto integra atividades humanas e serviços computacionais no sentido de expandir e facilitar tais atividades **[DEY 00]**. Seres humanos naturalmente fornecem informações de contexto na sua forma própria de comunicação, seja por sinais de frustração, confusão, alegria ou outros quaisquer. Computadores não possuem a capacidade de sentir, perceber e utilizar esses sinais de maneira independente. Por isso, se fazem necessárias as informações de contexto.

Na literatura encontram-se muitas definições para contexto. Schilit e Theimer **[SCH 94]** definem contexto como localização, identidade de pessoas e objetos próximos e mudanças nesses objetos. Em uma definição similar, Brown et al. **[BRO 97]** definem contexto como localização e identidade de pessoas próximas do usuário, adicionando as informações de hora, período e temperatura do ambiente. Ryan et al. **[RYA 98]** definem contexto como localização e identidade do usuário e de objetos, informações físicas do ambiente e tempo.

Dentre as definições existentes, a mais abrangente é a dada em **[Dey 01]**, onde *"contexto é qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade, onde uma entidade é uma pessoa, um lugar, ou um objeto considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação*". Em ambientes de monitoramento de condições de emergência, por exemplo, informação de contexto pode ser baseada em exemplos como identificação, localização e orientação do usuário, presença de chamas, fumaça, vazamento de gases e temperatura.

Abowd & Mynatt sugerem cinco dimensões (cinco W's) para especificação e modelagem de informações de contexto **[ABO 00]**: ***Who*** (quem): o sistema deve prover informações de contexto de todas as pessoas envolvidas em uma dada atividade assistida por

computador; ***Where*** (onde): a noção de localização é a mais utilizada pelos sistemas cientes de contexto; ***When*** (quando): o contexto temporal tem sido utilizado para indexar um registro capturado ou para informar por quanto tempo um usuário esteve em um determinado local; ***What*** (o quê): o objetivo é obter informações, normalmente via sensores, e interpretar o que o usuário está fazendo; ***Why*** (porquê): mais complexo que perceber o que o usuário está fazendo é entender o porquê de sua ação. Interpretar informações de contexto que possam caracterizar o estado de uma pessoa é talvez o maior desafio da computação ciente de contexto.

2.5 Desafios da Computação Ubíqua

A computação ubíqua apresenta diversos desafios e conta com a ajuda e contribuições de diversas áreas de pesquisa, como Interação Homem Computador, Engenharia de *Software*, Sistemas Distribuídos, Redes de Computadores e Inteligência Artificial.

Com relação ao desenvolvimento de interfaces, o desafio é o desenvolvimento de ferramentas de construção de interfaces multimodais **[ABO 00]**, ou seja, ferramentas com suporte operacional não apenas para dados textuais e primitivas originadas de interações via mouse e teclado, mas também com operações para a manipulação de linguagem natural e outras formas de entrada de dados, como vídeo, escrita, gestos e informações oriundas de sensores.

Outro desafio importante em computação ubíqua é o desenvolvimento de mecanismos padronizados para descrição das características físicas e funcionais devido à multiplicidade dos dispositivos de captura e acesso móveis ou não. Também é desejável conhecer os perfis e

preferências de seus usuários para que um serviço ou informação seja adaptado em conformidade **[BAN 02]** para cada usuário.

Sistemas de computação ubíqua requerem requisitos que sirvam de orientação durante o ciclo de vida desses sistemas **[DEY 00]** como, por exemplo, comunicação distribuída e transparente, mecanismos de tolerância à falhas e descoberta de recursos. Desenvolvedores de sistemas de computação ubíqua têm identificado ainda a necessidade de dividir responsabilidades entre aplicações e infra-estruturas de *software* especializadas quanto ao armazenamento, consulta, processamento e recuperação de informações de um ambiente de interação. Essa divisão de responsabilidades facilita, dentre outras coisas, a modularização e a reutilização de componentes de *software* para o desenvolvimento de novos sistemas.

A segurança e a privacidade das informações capturadas em ambientes de interação instrumentados com câmeras, microfones e sensores também são requisitos em sistemas de computação ubíqua **[JIA 02]**. Entre outras informações os sensores podem identificar onde pessoas estão, o que estão fazendo e quando chegam e saem de casa. Ao serem combinadas, essas informações expõem a privacidade dessas pessoas.

2.6 Áreas de Aplicações da Computação Ubíqua

A computação ubíqua está presente em diversos domínios de aplicação com intuito de auxiliar usuários em suas atividades cotidianas. Como exemplo destas aplicações podem-se citar sistemas cientes de localização de usuários que fazem uso de dispositivos como o *Active Badge* (**Figura 2.1**) e o PARCTab (**Figura 2.2**) e oferecem possibilidade de transferências automáticas de ligações telefônicas e atualizações automáticas de mapas com informações de localização dos usuários.

Figura 2.1- Exemplos de dispositivos Active Badges. Fonte: [WAN 92].

O *Active Badge System* é um sistema desenvolvido nos laboratórios da *Olivetti Research Labs* para localizar e identificar pessoas e objetos dentro da área onde é instalado. O sistema apresenta como componentes principais:

- *Active badge*: dispositivo que transmite sinais infravermelhos, cada um associado a uma pessoa ou objeto a localizar, que permite ao portador mover-se dentro da área controlada pelo sistema;
- Sensores: dispositivos receptores de raios infravermelhos associados a uma localização, normalmente espalhados por setores de um recinto fechado, como salas e corredores de um prédio.

Cada indivíduo com seu *active badge* transmite um sinal de raios infravermelhos, que consiste em um código de identificação único desse dispositivo que é detectado a cada dez segundos pelos sensores. Essa informação é enviada pela rede e associada com a informação contida em uma base de dados para posterior processamento.

Este sistema pode ser também utilizado para controle de acesso a áreas de segurança, controle de dispositivos de forma automática (ex: redirecionamento de chamadas telefônicas), auxiliar em emergências médicas e controle de visitantes.

O *PARCTab*, por sua vez, é um protótipo de pesquisa desenvolvido pela *Xerox Parc* com o objetivo de explorar o potencial e o impacto de computadores móveis em um ambiente de escritório. O sistema *PARCTab* consiste de um dispositivo portátil do tamanho de um *Palm* com potencial de comunicação sem fio, através de transceptores infravermelho, com aplicações em estações de trabalho.

O projeto do sistema *PARCTab* foi guiado pelos seguintes princípios **[SCH 95]**:

- *Alta portabilidade* – o dispositivo foi projetado para ser transportado e usado o tempo todo. Seu tamanho, peso e capacidades de *hardware* são adequados para promover computação casual, atendendo necessidades instantâneas de serviço, como a impressão de documento na impressora mais próxima do local onde está o usuário do dispositivo, por exemplo;
- *Conectividade constante* – o sistema assume que o dispositivo está sempre conectado à infra-estrutura de rede;
- *Relatório de localização* – o *software* do sistema sempre sabe onde está cada dispositivo *PARCTab*.

O *PARCTab* tem um conjunto de fontes para textos, funções que enviam imagens para sua tela, geram sons e permitem interação com canetas sobre sua tela (**Figura 2.2**).

Figura 2.2 - Dispositivos PARCTab. Fontes: [LAE 03] e [SCH 93].

Há três tipos de componentes de *software* no sistema: *gateways*, agentes e aplicações. Os *gateways* enviam e recebem pacotes de dados via sinais infravermelhos. Cada dispositivo é representado por um agente responsável por rastrear sua localização. As aplicações são adaptadas para acomodarem-se à baixa largura de banda de comunicação e à pequena tela.

Um grande número de aplicações *PARCTab* foi construído para explorar os problemas no acesso da informação ubíqua, o uso de interfaces baseadas em canetas e o controle do ambiente, conforme apresentado a seguir:

- *Controle do ambiente* – um dispositivo pode ser usado como controle remoto, por exemplo, para controlar as luzes e temperatura de um ambiente;
- *Acesso à informação* – previsão do tempo (obtida da *Web*), temperatura local e velocidade do vento;
- Dicionários, calendários e controle de migração, dentre outras.

A próxima seção apresenta domínios de aplicação tradicionais da computação ubíqua, como reuniões, ensino, residências, guias turísticos e segurança de ambientes.

2.6.1 Aplicação da Computação Ubíqua em Reuniões

Basicamente, aplicações de computação ubíqua que dão suporte a reuniões formais ou informais têm dois objetivos: (a) registrar o andamento de um projeto e (b) identificar as contribuições individuais de cada membro do grupo.

O sistema *DUMMBO* (*Dynamic, Ubiquitous, Mobile Meeting Board*) **[BRO 98]** é um projeto do Instituto de Tecnologia da Geórgia nos EUA voltado para a captura de atividades de uma reunião informal. Este sistema registra os desenhos feitos em uma lousa eletrônica compartilhada e a voz de cada membro da reunião, além de disponibilizar essa informação sob a forma de hiperdocumentos multimídia sincronizados.

O sistema *TeamSpace* **[RIC 01]** é um projeto colaborativo entre o Instituto de Tecnologia da Geórgia, IBM e Boeing com foco na captura de atividades envolvidas em uma reunião formal local ou distribuída. Esse sistema registra uma parcela maior dos artefatos apresentados durante uma reunião, incluindo slides, anotações, itens de agenda, vídeo e áudio de cada membro da reunião.

O projeto *iRoom* (*Interactive Room*) **[JOH 02]** da Universidade de Stanford nos EUA tem foco nas interações com dispositivos computacionais (ex: computadores pessoais, laptops, PDAs e superfícies eletrônicas) em um espaço de trabalho conectado a uma rede sem fio (**Figura 2.3**). Um de seus principais desafios envolve o suporte à movimentação de dados e de controle entre esses vários dispositivos de interação de forma transparente, de forma a não interromper o processo de colaboração.

Figura 2.3 - Sala interativa iRoom – lousas eletrônicas sensíveis a toque, mesa de projeção, *notebooks*. Fonte: [ICS 04].

2.6.2 Aplicação da Computação Ubíqua no Ensino

Aplicações de computação ubíqua que dão suporte ao ensino têm dois objetivos: (a) reduzir a sobrecarga que alunos têm ao anotar informações passadas pelo professor em sala de aula, (b) auxiliar professores na tarefa de produção de material didático.

Como exemplos dessas aplicações, tem-se o projeto *eClass* **[ABO 99]**, formalmente conhecido por *Classroom 2000*. Esse projeto teve início em 1995 no Instituto de Tecnologia da Geórgia nos EUA e compreende um conjunto de tecnologias de *hardware* e *software* para a captura de aulas presenciais (**Figura 2.4a**) e posterior disponibilização do material capturado sob a forma de hiperdocumentos multimídia customizados (**Figura 2.4b**).

(a) **(b)**

Figura 2.4 - Sala de aula com lousas eletrônicas, projetores e suporte multimídia (a) e um hiperdocumento Web (b) gerado automaticamente a partir do material apresentado em sala. Adaptado de [ABO 99].

O *Lecture Browser* **[MUK 99]** é um projeto da Universidade de Cornell nos EUA que difere do *eClass* ao utilizar técnicas de visão computacional e combinação de mais de uma fonte de vídeo para produzir seus hiperdocumentos multimídia resultantes da aula capturada.

O *Authoring on the Fly* **[HÜR 99]** é um projeto de suporte ao ensino da Universidade de Freiburg na Alemanha que difere de outros sistemas dessa natureza em função do poderoso modelo de sincronização das mídias capturadas da aula, das diversas formas de representação do material capturado e das funcionalidades fornecidas para acesso posterior a esse material.

2.6.3 Aplicação da Computação Ubíqua em Residências

Basicamente, aplicações de computação ubíqua que abrangem o domínio doméstico têm por objetivos: (a) conhecer as atividades dos moradores de uma casa e (b) fornecer serviços que aumentem a qualidade de vida deles.

Aware Home **[KID 99]** é um projeto desenvolvido pelo Instituto de Tecnologia da Geórgia nos EUA (**Figura 2.5**). Suas metas são: (a) investigar que tipos de serviços podem

ser fornecidos quando um sistema é ciente das atividades dos moradores da casa, (b) construir modelos de comportamento humanos para auxiliar os computadores nas tomadas de decisão e (c) apoiar pessoas com mais idade a morar sozinhas, ajudando-as a manter uma dieta saudável, lembrando-as de tomar remédios nos horários exigidos, ou mesmo mantendo seus familiares informados sobre o bem-estar delas. Uma aplicação monitora o cotidiano de pessoas idosas ou doentes que moram sozinhas. Metáforas aplicadas a interfaces usuário-computador representam o grau de atividade dessas pessoas, sem invadir sua privacidade com informações de áudio e/ou vídeo.

Figura 2.5 - Vista frontal da casa. Fonte: [GTI 04].

EasyLiving **[BRU 00]** é um projeto da *Microsoft Research* que se preocupa com o desenvolvimento de arquiteturas e tecnologias para ambientes inteligentes, focando particularmente em uma sala de estar residencial. O ambiente contém um computador, telas eletrônicas, caixas de som, sofás, mesa de café, entre outros itens. Serviços são fornecidos para melhorar o ambiente, como a automatização do controle de luz, tocar música conforme a localização e a preferência do usuário, e ainda transferir automaticamente o conteúdo de uma tela para outra.

Adaptive House **[MOZ 98]** é um projeto da Universidade de Colorado e tem como objetivo desenvolver uma casa que se programe observando o estilo de vida e os desejos de seus habitantes e aprendendo a se antecipar às necessidades dos mesmos. O sistema desenvolvido no *Adaptive House* é chamado *ACHE* (*Adaptive Control of Home Environments*), e foi desenvolvido basicamente para controlar o sistema de AVAC (Aquecedor, Ventilador, Ar-condicionado), iluminação e água.

2.6.4 Aplicação da Computação Ubíqua em Guias Turísticos

Guias turísticos eletrônicos trabalham normalmente com informações de localização dos usuários e dispositivos portáteis para conferir maior mobilidade aos usuários. Os principais objetivos dessa classe de aplicações são: (a) registrar os locais visitados e (b) identificar a posição atual do usuário dentro de um espaço de interação (ex: campus universitário, museu).

O sistema *CyberGuide* **[ABO 97]** é um guia turístico eletrônico ciente de localização que fornece serviços a usuários, como a posição atual de um usuário, o conjunto de locais visitados e que direção esse usuário está tomando. As informações de localização baseiam-se no sistema de posicionamento global (GPS – *Global Positioning System*).

O *CampusAware* **[BUR 02]** também é um sistema de turismo que utiliza informações de localização para fornecer serviços aos usuários. O sistema permite que o usuário faça anotações em seus dispositivos portáteis sobre os locais que está visitando (**Figura 2.6a**), fornecendo também informações de como encontrar determinados locais de interesse e a posição atual do usuário (**Figura 2.6b**). O *CampusAware* possui um repositório central com três bancos de dados para informações contextuais e sociais, como as anotações dos usuários e

locais visitados. Assim como o sistema *CyberGuide*, o *CampusAware* utiliza o sistema GPS para obtenção de informações de localização em ambientes abertos.

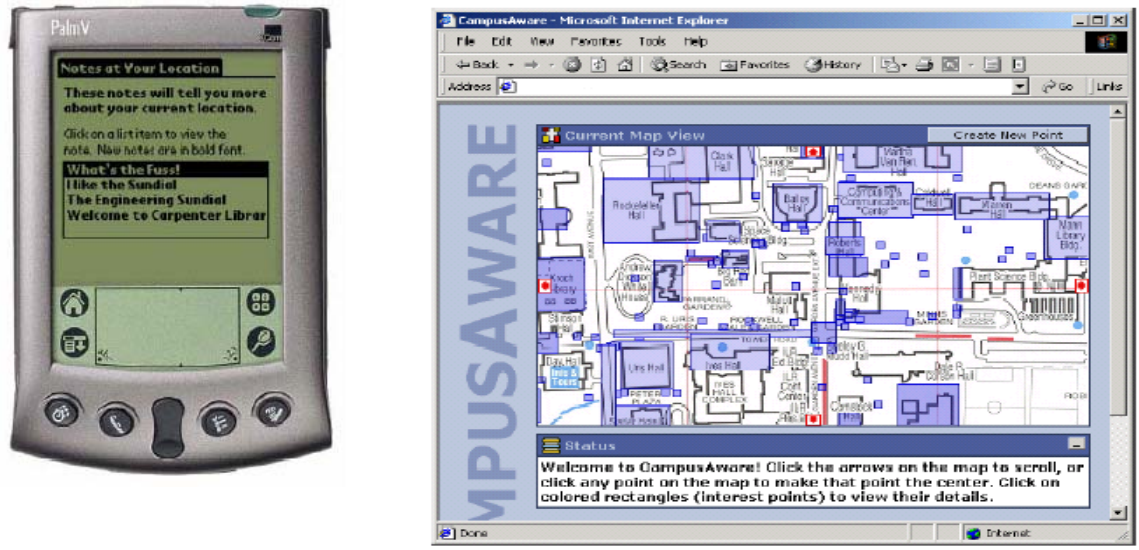


Figura 2.6 - Dispositivo móvel com dados sobre a localização do usuário (a) e interface web com o mapa do campus (b). Adaptado de [BUR 02].

O sistema *Rememberer* **[FLE 02]** é parte do projeto *Cooltown* **[KIN 01]** dirigido pela *Hewlett-Packard Laboratories*. Seu objetivo é capturar experiências pessoais e estimular discussões ou outras formas de interação pessoal através de dispositivos portáteis sem fio. Para capturar informação de localização em recintos fechados, esse sistema utiliza identificação por rádio-freqüência ou RFID (*Radio Frequency Identification)* **[AIM 03]**. Um exemplo de aplicação desse sistema é em visitas a museus, onde câmeras fotográficas registram interações dos usuários no museu e essas informações são disponibilizadas na ordem em que os locais foram visitados, permitindo ainda que o usuário faça anotações sobre esses lugares.

Sistemas de monitoramento de condições de emergência serão apresentados na próxima seção com mais detalhes por serem o foco principal deste trabalho.

2.7 Sistemas de monitoramento de condições de emergência e suas áreas de aplicações

Sistemas de segurança crítica são aqueles em que uma falha pode resultar em perdas de vidas, significantes perdas de propriedade ou prejuízos financeiros, **[KNI 02]**. Aplicações voltadas para segurança crítica têm como objetivo minimizar ocorrências de situações perigosas que resultam em acidentes[2] catastróficos, como por exemplo: situações que levam a morte, destruição de propriedade, danos à saúde ou ao meio ambiente e detecção de falhas. Como exemplos de aplicações em tais áreas pode-se citar o gerenciamento de tráfego aéreo, sistemas nucleares, monitoramento e combate a incêndios, monitoramento de manobras militares, dispositivos médicos e segurança industrial.

Em alguns casos, os serviços ou aplicações a serem disponibilizados dependem de certos requisitos de operação do sistema, requisitos estes considerados críticos, desde que um mau funcionamento no sistema pode resultar em sérias conseqüências a seus usuários. Requisitos típicos no projeto desses sistemas incluem a proteção à vida humana, ao meio ambiente ou dos recursos materiais. Em sistemas críticos, a denominação segurança refere-se sempre à expressão "*safety*" em contraposição ao termo "*security*".

Reliable Wireless Networks for Industrial Systems **[POO 02]** é um projeto da *Ember Corporation* que utiliza redes de sensores sem fio em uma fábrica de tratamento de água. A idéia é conectar sensores nas tubulações de água enviando dados a uma sala de controle da fábrica para que possam identificar e prevenir vazamentos. Questões como confiabilidade,

[2] Genericamente, o termo acidente pode ser definido como uma seqüência de eventos (acontecimentos casuais, fortuitos, imprevisíveis ou infelizes), cujos resultados (conseqüências), são traduzidos em danos às pessoas (morte ou ferimentos graves) e prejuízos à propriedade como também ao meio ambiente. O termo *incidente* é considerado como um evento em um sentido bem próximo de *quase-acidente*, desde que suas conseqüências não resultam em seqüelas ou danos graves às pessoas ou à propriedade **[WEL 01]**.

redundância e comunicação distribuída oferecidas pela rede de sensores sem fio são altamente desejáveis nesse tipo de situação.

Na área de aviação, Lampe et al. **[LAM 04]** propõem a utilização de tecnologias de computação ubíqua para melhorar o processo de manutenção de aeronaves. Qualidade, segurança, documentação e os altos custos gerados quando aeronaves estão fora de operação demandam uma execução eficiente do processo de manutenção das aeronaves.

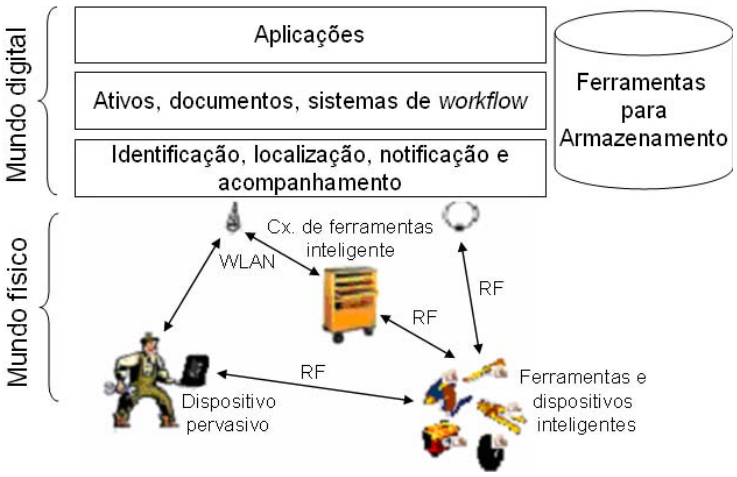


Figura 2.7 - Arquitetura de sistema de computação ubíqua para manutenção de aviões. Adaptado de [LAM 04].

A **Figura 2.7** apresenta a arquitetura de um sistema de computação ubíqua para manutenção de aviões, composto de duas partes: o mundo digital com suporte de armazenamento, aplicações, serviços e dados, e o mundo físico com ferramentas munidas de identificação por rádio freqüência. Todas essas ferramentas se comunicariam com uma espécie de caixa de ferramentas inteligente por rede sem fio descrevendo sua localização, *status* de operação, dentre outras informações.

Existem algumas aplicações de computação ubíqua que têm por objetivo monitorar as condições do ambiente e auxiliar nas decisões que precisam ser tomadas de maneira rápida

e eficaz quando situações anormais são detectadas. Exemplos de projetos dessa natureza incluem o *Siren* **[JIA 04]** e o Posto de Comando do Futuro *(The Command Post of the Future)* **[DAR 04]**.

Siren é um projeto desenvolvido pelas Universidades da Califórnia, Stanford e Washington nos EUA, voltado especificamente para o suporte às situações de incêndio que fornece comunicação entre bombeiros e mensagens de alerta **[JIA 04]**. As mensagens são codificadas utilizando XML e enviadas sobre HTTP. *Siren* trata apenas os seguintes contextos: (a) localização relativa – significa que a informação é do tipo "Sala tal"; (b) temperatura do ambiente; e (c) nível de oxigênio restante.

São gerados apenas cinco tipos de mensagens de alerta: “lugar perigoso”, “perigo para si mesmo”, “próximo a um lugar perigoso”, “outros em perigo” e “instruções”. Essas informações são apresentadas no PDA de cada bombeiro com o objetivo de manter uma comunicação entre os bombeiros e apresentar informações sobre o próprio bombeiro, e se algum outro bombeiro estiver próximo, passar informações sobre o ambiente. Outra característica é que *Siren* não se preocupa com a persistência dos dados, apenas com a utilização desses em tempo-real.

A **Figura 2.8** apresenta a arquitetura do *Siren*: um gerente de armazenamento implementa a abstração de espaço de informação; um gerente de comunicação manipula passagens de mensagens entre dispositivos em uma rede *peer-to-peer*; e uma máquina de regras de contexto que controla o *feedback* do usuário ciente de situação. Todos esses componentes e as aplicações interagem por meio da API *InfoSpace*, uma interface de programação baseado em espaços de tuplas. Aplicações que utilizam os componentes do *Siren* coletam informações de sensores carregados por bombeiros ou embutidos pelo ambiente.

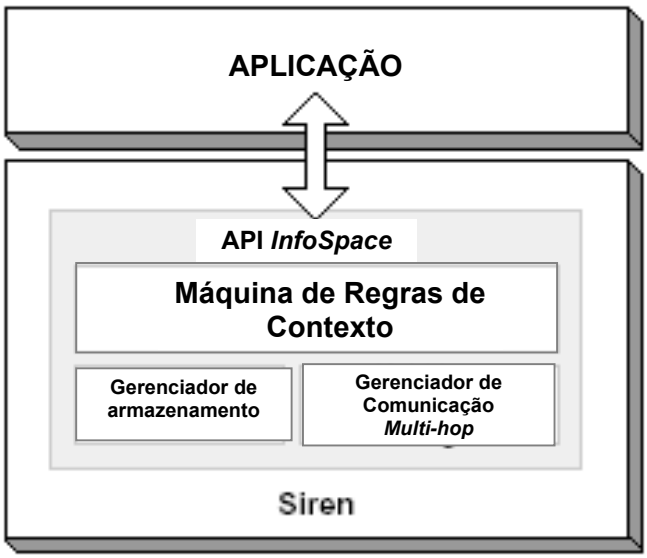


Figura 2.8 - Arquitetura do Siren utiliza a API InfoSpace para unificar as facilidades de armazenamento, comunicação e feedback adaptativo em regras de contexto. Adaptado de [JIA 04].

O Posto de Comando do Futuro (*The Command Post of the Future*) **[DAR 04]** consiste de um conjunto de projetos que se estendem desde a década de 70 sobre a investigação em situações de campo de batalha para traçar táticas e estratégias, principalmente em tempo de execução. Esses projetos focalizam o desenvolvimento de tecnologias avançadas que possibilitam a criação de *displays* interativos do ambiente, integrados com múltiplos sensores para o comandante aperfeiçoar suas decisões de forma rápida e com qualidade (o que pode ser chamado de ciência situacional). Esses *displays* podem auxiliar em tomadas de decisões facilitando a interação multimodal, a visualização de informações do ambiente e a formação de estratégias baseadas em conhecimento.

O domínio de aplicação de monitoramento de condições de emergência diferencia-se dos demais domínios (ex: sala de aula e sala de reuniões) no que se refere à consistência,

precisão e entrega dos dados do ambiente sendo monitorado, já que o monitoramento de condições de emergência envolve vidas humanas entre outras perdas substanciais.

2.8 Considerações finais

A computação ubíqua sugestiona e acrescenta novos paradigmas de interação inspirados pelo acesso difundido à informação e capacidades computacionais. Este capítulo introduziu a computação ubíqua e deu uma visão geral em suas áreas de aplicações, abordando o seu uso em sistemas de monitoramento de condições de emergência que é o foco principal deste trabalho.

Um exemplo de aplicação de supervisão de segurança crítica apresentado neste capítulo é o projeto *Siren* **[JIA 04]** que focaliza aplicações de combate a incêndios com o objetivo de auxiliar os bombeiros no momento do combate, todavia não existe a preocupação com a persistência dos dados para acesso posterior (ex: para perícia e treinamento). Além disso, a comunicação usada é exclusivamente através de PDAs, o que pode tornar a visualização um pouco inviável quando a situação é de emergência, uma vez que decisões devem ser tomadas rapidamente e o ambiente pode estar sujeito à presença de fumaça densa e altas temperaturas. Outro exemplo é o projeto "Posto de Comando do Futuro" **[DAR 04]** que objetiva o desenvolvimento de tecnologias avançadas para auxílio em tomadas de decisões facilitando a interação e a visualização de informações, as limitações neste sistema com relação à classe de aplicação que procuramos atender (sistemas de monitoramento de situações de emergência) está no custo envolvido para implementação deste projeto, por ser pretensioso e em longo prazo.

O uso de computação ubíqua pode auxiliar o monitoramento de condições de emergência em ambientes de segurança crítica fornecendo dados mais precisos e,

conseqüentemente, ajudar na prevenção de acidentes, no gerenciamento de combate de acidentes tanto para auxiliar a tomada de decisões como na supervisão de equipe e posteriormente para perícia e treinamento. Neste sentido, pretende-se alcançar um refinamento na supervisão a ponto de contribuir na prevenção e combate reduzindo perdas de vidas humanas, de patrimônio e perdas financeiras de outras naturezas, além de atender a certos critérios cujos requisitos de segurança excedem a capacidade e as habilidades humanas.

O capítulo seguinte apresenta as principais estruturas encontradas na literatura para o suporte às aplicações de computação ubíqua cientes de contexto.

3 Estruturas de suporte às aplicações de computação cientes de contexto

3.1 Considerações iniciais

A computação ubíqua ciente de contexto é amplamente explorada em aplicações de sala de aula **[ABO 99]**, sala de reuniões **[JOH 02]**, guias turísticos **[ABO 97]**, residências **[BRU 00]** e também na área de segurança crítica: segurança industrial **[POO 02],** segurança na aviação **[LAM 04],** segurança no controle metro-ferroviário **[CAM 96].** Aplicações que envolvem monitoramento de condições de emergência em ambientes físicos de interação também podem ser consideradas sob o domínio de aplicações de computação ciente de contexto **[JIA 04]**, devido ao avanço da tecnologia de sensores cada vez mais poderosos e mais baratos que oferecem um monitoramento bastante minucioso, ou de granularidade fina, com relação às informações que são capturadas. Tais aplicações tornam-se cada vez mais viáveis e importantes na ajuda à prevenção, combate e avaliação de situações de emergência, uma vez que informações capturadas por sensores podem facilitar com que condições críticas sejam evitadas (prevenção), podem ajudar com informações mais detalhadas para tomadas de decisão no gerenciamento de equipes de resgate e salvamento (combate), ou ainda auxiliar perícias por parte da seguradora ou até mesmo do próprio responsável pela segurança da empresa (avaliação).

3.2 Context Toolkit

Desenvolvido pelo Instituto de Tecnologia da Geórgia, o *Context Toolkit* **[SAL 99]** captura dados de contexto através de sensores em ambientes de computação ubíqua. Seu

objetivo principal é auxiliar o desenvolvimento de soluções para a construção de aplicações cientes de contexto.

Ele consiste em três abstrações principais, **[Dey 99]**:

- *Widgets*: assim como os *widgets* das interfaces gráficas convencionais fazem a interação entre o usuário e uma aplicação, os *widgets* de contexto fazem a interação entre o usuário e o ambiente. *Context widgets* encapsulam informações de contexto, tais como localização ou atividade, por exemplo. *Widgets* provêem uma interface uniforme para componentes ou aplicações que usam o contexto, escondendo os detalhes dos mecanismos de contexto,

- *Aggregators*: agregadores podem ser comparados como meta *widgets* que aumentam as capacidades de *widgets*, possibilitando a agregação de informações de contexto de entidades do mundo real tais como usuários e lugares,

- *Interpreters*: é um interpretador de contexto usado para abstrair ou interpretar informações de contexto de baixo nível em informações de alto nível. Por exemplo, identificação de pessoas, localização física de pessoas ou dispositivos e sons podem identificar que uma reunião está acontecendo em uma determinada sala e que, eventualmente, não pode ser interrompida. Os agregadores atuam como uma ligação entre as aplicações e os *widgets* básicos, fornecendo ainda mais transparência para o processo.

O *Context Toolkit* usa HTTP como protocolo de rede e XML para o formato de dados, alcançando um certo nível de interoperabilidade, também é bastante flexível para permitir protocolos e formato de dados diferentes.

O *Context Toolkit* não apresenta compartilhamento de sensores, nem compartilhamento do poder de processamento, nem dos dados e serviços. Quando uma aplicação é iniciada, ela se comunica com o descobridor para localizar os componentes que são relevantes para suas funcionalidades. Aplicações e agregadores precisam se subscrever aos *widgets* para receber contextos de seus interesses. As subscrições são feitas utilizando o método *subscribeTo* que envia uma mensagem addSubscriber para um *widget*. No método *subscribeTo* são definidos os dados que as aplicações ou agregadores precisam receber e em quais condições. Isto ajuda a reduzir a comunicação na rede, o que é importante em uma arquitetura distribuída por razões de desempenho.

Os *widgets* capturam contexto dos sensores e os torna disponíveis tanto para os agregadores, quanto para as aplicações. As aplicações não adquirem contexto somente de *widgets*, pois podem se subscrever também nos agregadores. *Widgets*, agregadores e aplicações ainda podem solicitar que um contexto seja transformado utilizando os interpretadores. A **Figura 3.1** mostra como aplicações e componentes no *Context Toolkit* podem interagir.

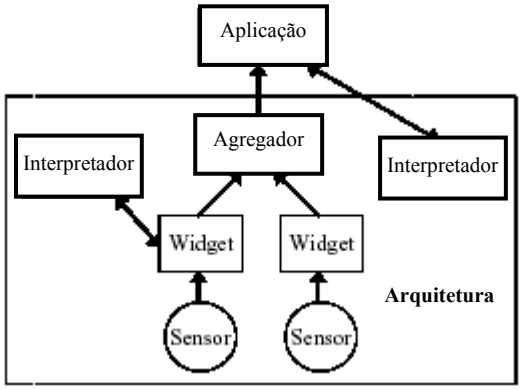


Figura 3.1 - Context Toolkit Adaptada de [DEY 99].

Uma limitação do *Context Toolkit* é que o processamento de contexto não é feito de forma automática, pois o desenvolvedor deve especificar manualmente qual caminho os dados de contexto deverão seguir, ou seja, o desenvolvedor é quem define se o contexto deve passar por um agregador, por um interpretador, e assim por diante.

3.3 Context Fabric

Context Fabric **[HON 01]** é uma infraestrutura desenvolvida pela Universidade da Califórnia – Berkeley. Uma de suas características é sua arquitetura em camadas, onde cada camada possui diferentes responsabilidades e fornece conjuntos de serviços específicos.

O *Context Fabric* é composto por quatro serviços básicos: serviço de evento de contexto (*context event service*), serviço de consulta de contexto (*context query service*), serviço de criação automática de caminho (*automatic path creation*) e serviço de gerenciamento de sensores (*sensor management service*). Existe também uma linguagem por meio da qual aplicações podem especificar as informações de contextos necessárias chamada linguagem de especificação de contexto (*Context Specification Language*). A linguagem de especificação de contexto é a interface pela qual os serviços de evento de contexto e consulta de contexto são acessados (**Figura 3.2**).

Utilizando a linguagem de especificação de contexto, aplicações podem especificar em quais eventos estão interessadas e quais desejam ser notificadas de forma assíncrona quando este evento ocorrer; para tanto elas devem subscrever-se no serviço de evento de contexto.

A subscrição necessita de duas partes:

1ª) especificação de informações relacionadas ao *subscriber*, por exemplo, como enviar um evento ao *subscriber*, o nome do evento a ser enviado e dados de sensores e contextos relevantes para a situação;

2ª) descrição do evento de contexto que deseja receber.

O serviço de consulta de contexto fornece interface para que aplicações possam consultar de forma síncrona uma informação de contexto. As consultas têm o mesmo formato das subscrições de eventos.

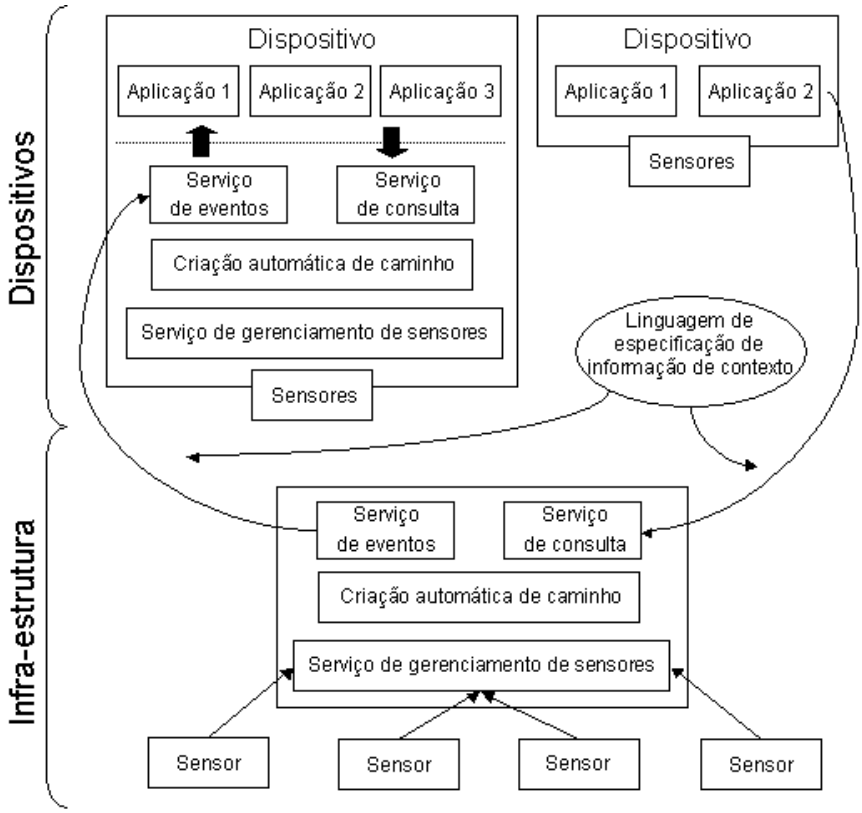


Figura 3.2 - Arquitetura do Context Fabric – Fonte: [HON 01].

3.4 GAIA

A infra-estrutura GAIA **[ROM 02]** foi desenvolvida pela Universidade de Illinois e trata um espaço ativo (*ActiveSpace*) e seus dispositivos de forma análoga a um sistema operacional tradicional. Espaço Ativo é a abstração usada para se referir a qualquer ambiente de computação ubíqua que possa ser gerenciado pela infra-estrutura GAIA **[HES 02]**.

GAIA trabalha com a idéia de "sessões" associadas a usuários independentes de espaços particulares como o "espaço virtual ativo do usuário". As informações do usuário e as aplicações não estão confinadas a qualquer espaço ativo; elas são associadas com os usuários que usam sessões permitindo que eles se movam entre diferentes espaços ativos e tenham seus dados e aplicações sempre disponíveis, [ROM 02]. Quando um usuário entra em um espaço ativo, suas sessões são dinamicamente mapeadas para os recursos ativos do espaço.

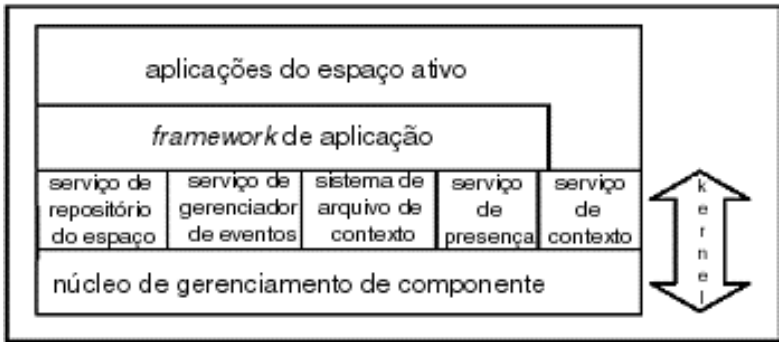


Figura 3.3 - Arquitetura da Infra-estrutura GAIA. Fonte: [ROM 02].

GAIA é definido como uma infraestrutura de *software*, ou ainda como um meta sistema operacional (Gaia OS – *Gaia Operating System*) que aponta para o suporte ao desenvolvimento e execução de aplicações portáteis para espaços ativos, **[CER 01]**. O GAIA gerencia os recursos e serviços de um espaço ativo, provê serviços para localização, contexto e eventos, e repositórios com informações sobre os espaços ativos. O sistema é construído como um objeto distribuído.

A **Figura 3.3** mostra os três maiores blocos do GAIA:

- *Núcleo de Gerenciamento de Componentes* - contém um sistema de gerenciamento e desenvolvimento para objetos distribuídos e um conjunto básico de serviços que são usados por todas as aplicações: um *Component Management Core* que carrega, descarrega, transfere, cria e destrói dinamicamente todos os componentes e aplicações do Gaia, seus componentes são distribuídos e exigem comunicação *middleware* para suportar interação remota, a implantação atual do GAIA usa CORBA, entretanto, a infraestrutura GAIA permite a possibilidade de suporte a outras arquiteturas de comunicação inclusive SOAP[8], RMI[9], ou implementações customizadas; os cinco serviços básicos do GAIA *Kernel* são: *Event Manager Service* - usado para distribuir informação entre componentes, suportar ciência de contexto e usado na descoberta de serviços para rastrear componentes de softwares, pessoas, apresentar entidades físicas em um espaço; *Presence Service* – usado para manter informações atualizadas sobre os recursos presentes em um espaço ativo; *Context Service* – usa informações de contexto para adaptar as características dos usuários e suas atividades; *Repository Service* – usado para armazenar informações sobre todos os softwares e *hardwares* presentes no ambiente e, o *Context File System* – armazena informações de contexto para minimizarem muitas das tarefas que tradicionalmente são inseridas manualmente ou requerem programação adicional*;*

- *Framework de Aplicação* - que cria um vínculo entre um usuário, os recursos, os dispositivos múltiplos, a sensibilidade de contexto e o modelo de aplicação móvel. Aplicações são particionadas entre um grupo de dispositivos, elas recebem entradas

de eventos de diferentes dispositivos, apresenta seus estados usando diferentes tipos de dispositivos e se adapta para diferentes ambientes.

- *Aplicações do Espaço Ativo* - as aplicações do Gaia usam um conjunto de componentes construídos em blocos, organizados como o *Gaia Application Framework*, para suportar aplicações que executam dentro de um espaço ativo. O *framework* provê mobilidade, adaptação e construção dinâmica. Esta camada de aplicações no espaço ativo (*Active Space Applications*) contém aplicações e provê funcionalidades para executar, registrar, gerenciar, e controlar estas aplicações pelos serviços do *Gaia Kernel.*

3.5 AURA

David Garlan, **[GAR 02],** definiu AURA como um *framework* para mobilidade de usuários em ambientes de computação ubíqua, envolvendo comunicação sem fio, computadores de mão, e espaços pequenos. “Os dois principais desafios do AURA para suportar usuários móveis são: primeiro, maximizar o uso de recursos disponíveis; e segundo, minimizar a distração, e concentrar a atenção do usuário.” **[SOU 02]**. A maior fonte de distração do usuário surge da necessidade do usuário gerenciar seus recursos de computação em cada ambiente novo, e do fato que os recursos em um ambiente particular podem mudar dinamicamente e freqüentemente.

A **Figura 3.4** mostra uma visão do AURA. Ele possui quatro tipos de componentes: primeiro, o *Task Manager,* chamado de Prisma, que é uma descrição dos serviços independente de plataforma que objetiva minimizar as distrações do usuário que pode se mover de um ambiente para outro conservando a qualidade de serviço (QoS), as mudanças

entre serviços e as mudanças de contexto capturando exigências de privacidade, atividade do usuário (sentando, dirigindo) etc. Segundo, o *Context Observer* provê informação no contexto físico e reporta eventos relevantes de contexto de volta para o Prisma e para o *Environment Manager*. O terceiro componente do AURA, o *Environment Manager,* incorpora o *gateway* para o ambiente, ele conhece quais componentes estão disponíveis para serem oferecidos com os serviços; e quarto, *Suppliers* provêem serviços que compõem as requisições dos usuários como: edição de texto, acesso a vídeo, etc.

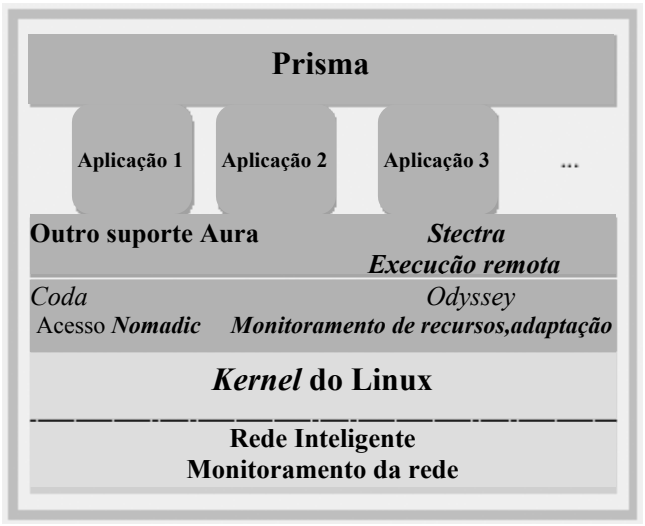


Figura 3.4 - Infraestrutura AURA. Fonte: [SOU 02].

WANT **[WAN 02]** fala a respeito da sincronização no AURA: "Sincronização é complicada e cara. É difícil escrever um *software* simultâneo. A solução é dividir a computação em pequenas tarefas, e executar cada tarefa até sua conclusão antes de executar a próxima". Essa é uma característica de modularidade presente neste sistema.

3.6 iROS

Segundo a definição de Brad Johanson **[JOH 02],** iROS (*Interactive Room Operating System*) é um meta sistema operacional, ou uma infraestrutura de *software* para suporte a interação homem-computador em ambientes fechados construída pela Universidade de Stanford. iROS provê espaços de trabalho interativos (*Interactive Workspace*) que é uma abstração que consiste de um ambiente que usa quadros ou telas grandes sensíveis a toque e compartilhadas e, que permitem a integração de dispositivos dentro deste espaço fechado onde pessoas podem trabalhar de maneira colaborativa. Na **Figura 3.5** encontra-se a arquitetura do iROS composta por três subsistemas: memória de eventos (*EventHeap*), memória de dados (*DataHeap*) e *ICrafter*.

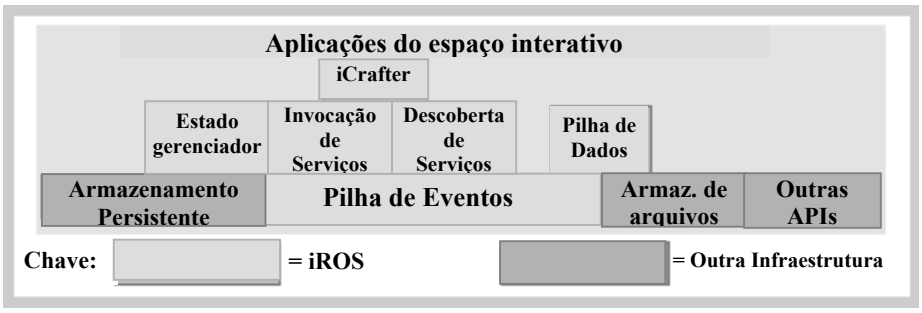


Figura 3.5 - Arquitetura da Infraestrutura iROS. Fonte: [JOH 02].

iROS é composto por três subsistemas que possibilitam a movimentação dos dados, o controle da movimentação e a coordenação da aplicação dinâmica respectivamente, conforme mostra a **Figura 3.5**: o primeiro é a Pilha de Dados que facilita a movimentação dos dados permitindo que qualquer aplicação coloque dados em um repositório associado com o ambiente local, os dados são armazenados com atributos que os caracterizam e não é informada sua localização específica, assim as aplicações não precisam se preocupar com a localização física onde os dados são armazenados, além disso, as informações de formato dos dados são armazenadas para prover transformação dos dados para o melhor formato suportado

pela aplicação; o segundo subsistema é o *iCrafter* que provê um sistema de invocação de serviços semelhante ao Jini, o *iCrafter* trabalha junto com um gerenciador de interfaces que permite ao usuário selecionar um serviço para controlar e executar automaticamente a melhor interface para o dispositivo do usuário; e o terceiro chamado de Pilha de Eventos que armazena e responde mensagens conhecidas como eventos, cada uma delas é uma coleção de campos de nome-tipo-valores, este subsistema provê um repositório central para que todas as aplicações em um espaço interativo possam armazenar esses eventos que são automaticamente removidos, as aplicações podem interagir com o Pilha de Eventos através de várias APIs, incluindo *Web*, Java, e C++.

3.7 Considerações finais

Nas estruturas para sistemas cientes de contexto apresentadas nas seções anteriores deste capítulo, é possível identificar limitações em relação a alguns aspectos importantes para sistemas de monitoramento de condições de emergência em ambientes cientes de contexto. O *Context Toolkit,* por exemplo, como o próprio nome diz, é uma ferramenta de ajuda ao desenvolvedor para criar aplicações cientes de contexto. O toolkit não atende os requisitos funcionais e não funcionais de aplicações cientes de contexto, mas sim oferece “peças” ou elementos básicos para a construção dessas aplicações.

De uma maneira geral, a principal limitação encontrada em tais estruturas é que elas não atendem requisitos importantes como confiabilidade, consistência, segurança e eficiência que são aspectos relevantes para a qualidade de sistemas de monitoramento de condições de emergência em ambientes cientes de contexto.

O próximo capítulo apresenta os requisitos de qualidade no desenvolvimento de sistemas para a classe de aplicações voltadas ao monitoramento de situações de emergência em ambientes cientes de contexto.

4 Requisitos de sistemas de monitoramento de condições de emergência em ambientes cientes de contexto

4.1 Considerações iniciais

Em alguns casos, os serviços ou aplicações a serem disponibilizados dependem de certos requisitos de operação do sistema, requisitos estes considerados críticos, uma vez que um mau funcionamento no sistema pode resultar em sérias conseqüências a seus usuários, **[KNI 02]**. Genericamente, tais sistemas são denominados de *sistemas críticos de segurança*. Requisitos típicos no projeto desses sistemas incluem a proteção à vida humana, ao meio ambiente ou dos recursos materiais. Em sistemas críticos, a denominação segurança refere-se sempre à expressão "*safety*" em contraposição ao termo "*security*".

Os requisitos importantes em sistemas computacionais de natureza crítica estão voltados, basicamente, para prevenção, controle e correção de possíveis falhas. Aplicações nesta área variam muito em relação à complexidade e às necessidades de garantia no atendimento de restrições temporais. Entre os sistemas mais conhecidos, estão os sistemas militares de defesa, os sistemas de controle de plantas industriais (químicas e nucleares) e os sistemas de controle de tráfego aéreo e metro-ferroviário. Algumas aplicações representam restrições de tempo mais rigorosas do que outras; entre essas, encontram-se os sistemas responsáveis pelo monitoramento de pacientes em hospitais e os sistemas embarcados em robôs e veículos (de automóveis até aviões e sondas espaciais).

Este capítulo tem o objetivo de introduzir e contextualizar os requisitos existentes e suas áreas de aplicação, e identificar quais os mais relevantes em sistemas de monitoramento de condições críticas em ambientes físicos cientes de contexto, porém sem entrar em questões específicas de implementação de funcionalidades dos mesmos.

4.2 Requisitos Não Funcionais

Requisitos devem definir como um sistema deve se comportar, suas funcionalidades e restrições de operação. Os **Requisitos Não Funcionais** (ou **RNF's**) são limitações globais em um sistema que especificam atributos de qualidade (desempenho, portabilidade, custo, confiabilidade, segurança), **[CHU 00]**. Esta denominação foi utilizada por Roman **[ROM 85],** sendo os RNF's também conhecidos como atributos de qualidade **[BOE 78] [KEL 90] [TUR 96]**, restrições **[ROM 85]**, objetivos **[MOS 85]** entre outros. De uma forma simplista pode-se colocar que os RNF's não expressam nenhuma função a ser realizada pelo sistema, mas sim comportamentos e restrições que ele deve satisfazer, descrevem "como o sistema deve fazer". Por outro lado, os **Requisitos Funcionais** (ou **RF's**) estão relacionados com as funcionalidades do sistema e descrevem "o que o sistema deve fazer".

Segundo a abordagem de Chung et. al. **[CHU 99]**, os RNF's incluem tanto limitações no produto (desempenho, confiabilidade e segurança) como limitações no processo de desenvolvimento (custos, métodos a serem adotados no desenvolvimento e componentes a serem reutilizados).

Em Cysneiros et. al. **[CYS 01]** é expressa a importância de considerar os RNF's no desenvolvimento de um sistema com finalidade de reduzir custos de manutenção e conseqüentemente aumentar a qualidade do mesmo.

Pode-se observar, muitas vezes, que os RNF's podem estar relacionados com um ou mais RF's, podendo ser conflitantes e contraditórios, e não existem regras universais e uma única linha de direção para determinar quando os RNF's estão corretamente estabelecidos, pois um sistema pode ser seguro, mas não confiável (um automóvel que não funciona), ou então pode ser confiável, mas não seguro (uma pistola carregada em um centro comercial) **[CYS 97]**.

Apesar deste trabalho não ter pretensão de abordar todos os RNF's para sistemas críticos, nas próximas seções é dada uma visão geral sobre as classificações propostas, bem como a identificação dos principais RNF's para a classe de sistemas de monitoramento de condições de emergência em ambientes cientes de contexto considerados neste trabalho.

4.2.1 Classificação de RNF's

A maioria dos modelos propostos obedece a uma metodologia hierárquica e respeita uma estrutura de árvore onde se identifica no 1º nível o fator a avaliar, no 2º os critérios para cada fator e no 3º a métrica a ser utilizada.

Modelo McCall

O modelo identifica três aspectos principais **[GRA 96]:** a **operação** implica que o sistema seja facilmente aprendido, eficientemente operado e que os resultados sejam os pretendidos pelo utilizador; a **revisão** tem a ver com a correção de erros e a adaptação do sistema. Sendo geralmente considerada a parte mais onerosa do desenvolvimento, torna-se, portanto, a mais importante; a **transição**, não sendo igualmente importante em todas as aplicações, é bastante relevante em substituições para processamentos distribuídos bem como na substituição de infraestruturas de *hardware*, cuja freqüência é cada vez maior.

Para a operação, McCall indica a **correção** (ou *correctness* - Cumpre as especificações?), a **confiabilidade** (executa sempre da mesma maneira?), a **eficiência** (Usa corretamente os recursos computacionais?), a **integridade** (As informações são verdadeiras? É seguro?) e a **usabilidade** (ou *usability* - É de fácil utilização? A operação, a preparação dos dados e a interpretação dos resultados exigem muito esforço?).

Para a revisão, aponta a **Flexibilidade** (É fácil fazer-lhe modificações?), a **Facilidade em fazer-lhe Testes** (*Testability* - para garantia de zero erro e do cumprimento das especificações), e a **Capacidade de Manutenção** (ou *Manutenalibity* - a facilidade de identificação de falhas em ambiente operacional).

Para a transição os critérios são a **portabilidade** (Que esforço é pedido para a transferência de um sistema para uma ou plataforma?), a **reutilização** (ou *reusability* - Em que extensão o sistema, ou partes dele pode ser utilizado como componentes de outras aplicações?) e a **inter-operabilidade** (Será capaz de interagir com outros *hardwares*?).

O modelo Boehm

O modelo de Boehm considera os requisitos em 3 níveis: os que se destinam a uso generalizado; os critérios intermediários (eficiência e a portabilidade) e no 3º os critérios primitivos, capazes de serem medidos (eficiência).

O conjunto de critérios em que se baseia este modelo, apresentado na **Figura 4.1**, é consideravelmente maior do que os do modelo McCall, a saber: a **usabilidade**, a **clareza** (ou *understandability*), a **eficiência**, a **confiabilidade**, a **modificabilidade** (*modificability*), a **portabilidade**, a **facilidade em fazer-se-lhe Testes** (ou testabilidade – tradução literal de *testability*).

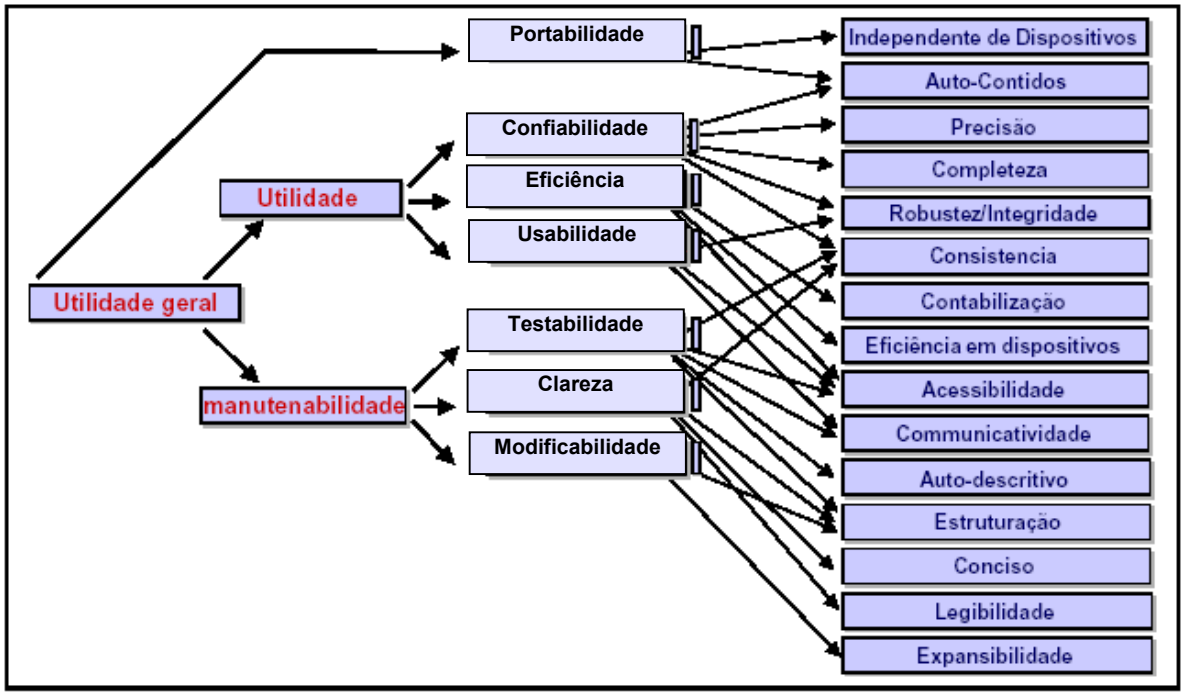


Figura 4.1 - Árvore dos requisitos não funcionais do modelo de Boehm. Adaptada de [BOE 76].

Modelo da norma ISO9126

A norma ISO9126 classifica os RNF's descritos com base em várias características escolhidas que definem a qualidade de *software* com a preocupação de serem independentes umas da outras **[ISO9126]**: **funcionalidade** (segurança e interoperabilidade), **confiabilidade**, **usabilidade** (facilidade de monitoria, operação e atratividade), **manutenabilidade** (ou *maintenability*), **eficiência** e **portabilidade**.

Pode-se observar, quanto aos requisitos mencionados, que alguns têm maior importância que outros dependendo de sua utilidade, todavia suas características devem ser combinadas sob a perspectiva global do domínio de aplicação que deseja atender, como por exemplo: a **confiabilidade,** no controle de sistemas críticos como (os de gestão do tráfego aéreo, controle metros-ferroviários ou centrais nucleares); a **eficiência,** em sistemas em tempo real não críticos (sistemas de gestão de bases de dados, protocolos de comunicações ou

centrais de comutação telefônica); a **facilidade de utilização** e a **funcionalidade,** na perspectiva dos utilizadores de sistemas interativos; a **facilidade de manutenção,** na perspectiva dos elementos responsáveis pela evolução e operação do sistema; a **portabilidade,** na perspectiva de quem suportou financeiramente o desenvolvimento de um dado produto de *software* estratégico.

Os requisitos mencionados sugerem a possibilidade da construção de um sistema perfeitamente confiável e permanentemente disponível. Infelizmente, estas técnicas apenas aumentam a **dependabilidade** de um sistema, não fornecendo nenhuma garantia de que todas as falhas possíveis possam ser toleradas.

O termo dependabilidade é uma tradução literal do termo inglês *dependability,* que indica a qualidade do serviço fornecido por um dado sistema e a confiança depositada no serviço fornecido. As principais medidas de dependabilidade, segundo PRADHAN **[PRA 96]**, são confiabilidade, disponibilidade, segurança de funcionamento (*safety*), segurança (*security*), testabilidade e comprometimento do desempenho.

Observa-se que grande parte dos sistemas implementa funcionalidades com o objetivo de melhorar a confiabilidade, a portabilidade, e outros aspectos de qualidade do sistema, no entanto, não identificam quais RNF's estão sendo atendidos através destas implementações. A próxima seção apresenta alguns exemplos que reforçam a necessidade de identificar e atender os RNF's.

4.3 RNF's de Sistemas Tradicionais

As principais áreas onde se empregam os RNF's têm o objetivo de tornar o sistema mais próximo possível do conceito de computação tolerante a falhas, onde mesmo na ocorrência de falhas em algum componente, é possível prover serviços de forma adequada.

Como exemplo podem-se citar as aplicações de sistemas de tempo real crítico como medicina, controle de processos e transporte aéreo que, comumente, consideram a disponibilidade de apenas um curto intervalo de tempo para reconhecimento de erros, de forma a não prejudicar o processamento em tempo real; a impossibilidade de emprego de recuperação por retorno, uma vez que eventos passados não são reproduzíveis em sistemas de tempo real; a exigência de redundância para garantir a continuidade do processamento em tempo real em caso de falha de um componente; e o comportamento livre de falhas (*fail-safe*), ou seja, em caso de ocorrência de uma falha, o sistema deve ir imediatamente para um estado seguro.

A próxima seção descreve exemplos de sistemas tradicionais e seus principais RNF's correspondentes.

4.3.1 Sistemas digitais de telefonia

Os sistemas eletrônicos para telefonia empregam RNF's devido às rigorosas exigências quanto à **disponibilidade** do sistema. É especificado, por exemplo, um tempo máximo em falha não maior que 2 horas em 30 anos.

Os requisitos para a aplicação nessa área são: a detecção e localização automática de erros tanto de *software* como de *hardware*; o tratamento automático de erros por reconfiguração do sistema; e o isolamento e substituição de componentes faltosos durante o período de operação normal do sistema. Outros dois requisitos igualmente importantes são a

duplicação de componentes que tem sido uma técnica muito empregada em sistemas telefônicos para substituição de componentes com falhas permanentes, e também a detecção de erros, onde duas unidades executam de forma sincronizada e um comparador verifica os resultados aumentando, desta forma, a **confiabilidade**.

4.3.2 Sistemas de transações

Muitas aplicações comerciais são realizadas na forma de processamento de transações. Esse processamento implica na existência de uma base de dados comum usada interativa e concorrentemente por um grande número de usuários através de terminais de acesso. Exemplos incluem bancos de dados para aplicações financeiras, bancárias e de bolsa de valores, e para reserva de vôos.

Requisitos indispensáveis para aplicações nessa área são: a garantia de **integridade** e **consistência** de dados na base de dados; a alta **disponibilidade** para o processamento contínuo de transações. Adicionalmente aos requisitos citados, são características desejáveis: o tratamento automático de erros no processamento de transações e de erros de *hardware*, sem interrupção do funcionamento normal; a reconfiguração tanto de *hardware* como de *software*, sem interrupção do processamento normal. Integridade e consistência de dados são os requisitos mais importantes.

Os sistemas comerciais, de forma geral, operam baseados no modelo *fail-stop*. Em caso de erro, o sistema interrompe o processamento evitando assim a propagação do erro com conseqüente dano à base de dados. Exemplos incluem os sistemas desenvolvidos pela *Tandem Computers* (fundada em 1976) que apesar de sua arquitetura estar em desuso suas técnicas continuam sendo usadas e o sistema *Continuous Processing* produzidos pela *Stratus*

Computers **[HEN 83]** que teve seus modelos também comercializados por outros fabricantes de computadores, como a Olivetti *(Olivetti* CPS32) e a IBM (IBM/88).

4.3.3 Redes de serviço locais

Os RNF´s para aplicações na área de redes de serviços são semelhantes aos requisitos para sistemas de transação: a garantia de **integridade** e **redundância** de dados; a alta **disponibilidade** do servidor para prover serviços de forma contínua na rede; o tratamento automático de erros de *hardware* e no serviço da rede; a reconfiguração da rede em caso de erro e estabelecimento de uma nova configuração com a entrada de outras estações (clientes e servidores), sem interrupção do processamento normal.

O crescimento do comércio eletrônico e a popularidade de redes impulsionaram o mercado para servidores que empregam RNF's. Os servidores comerciais são baseados em redundância de componentes de *hardware*, troca de módulos sem necessidade de desligar o sistema (*hotswap*), espelhamento de discos ou RAID etc.

Os RNF´s de Sistemas de monitoramento em ambientes condições de emergência serão identificados na próxima seção.

4.4 RNF's de Sistemas de monitoramento de condições de emergência

Como já definido no capítulo 2 **(seção 2.7)** deste trabalho, sistemas de monitoramento de condições de emergência em ambientes físicos de interação cientes de contexto são aqueles em que seu mau funcionamento e a falta de confiabilidade dos serviços oferecidos podem colocar em risco vidas humanas, além de perdas de propriedade e prejuízos financeiros. Este trabalho considera como sistema de monitoramento de condições de emergência um ambiente

físico povoado por sensores que capturam informações de contexto conforme ocorrências no ambiente e remete essas informações (eventos) para quem solicitou (aplicações ou serviços). Em uma prisão, por exemplo, é importante ter um monitoramento confiável do ambiente físico, especialmente em situações de emergência, tais como rebeliões de prisioneiros que podem provocar incidentes ou acidentes que coloca em risco a vida de pessoas e perda de patrimônio. Em tais situações, é importante que informações possam ser "sentidas" a partir do ambiente físico em tempo real, visto que informações precisas podem ser usadas pela equipe de segurança para gerenciamento de operação e decisões sobre a melhor estratégia a ser tomada em tempo real.

Para aplicações deste tipo, é preciso analisar as informações vindas de sensores durante as situações de emergência e considerar requisitos como, por exemplo, a **confiabilidade** dos dados, uma vez que o sistema pode ser colocado em risco caso os sensores enviem informações incorretas que podem ser causadas por falhas devido a interferências, tais como água ou presença de fumaça densa no ambiente, ou pelo seu mau funcionamento. Desta forma, aplicações de monitoramento de condições de emergência devem ser tolerantes a falha e confiáveis, e prover baixa latência na entrega dos eventos.

Sistemas de monitoramento de condições de emergência em ambientes físicos e lógicos cientes de contexto é o foco principal deste trabalho. Por esse motivo, a seção seguinte identifica os requisitos mais importantes para esta classe de aplicação, considerados neste trabalho.

4.4.1 RNF's importantes em sistemas de monitoramento de condições de emergência em ambientes cientes de contexto

Esta seção destaca os principais requisitos que devem estar presentes em estruturas que oferecem suporte e serviços para aplicações de monitoramento de condições de emergência em ambientes físicos ou lógicos cientes de contexto. Não é objetivo deste trabalho identificar todos os requisitos e nem mesmo especificar todas as funcionalidades desta classe de aplicações, entretanto é importante apresentar os aspectos mais relevantes que envolvem o gerenciamento de eventos, através dos principais requisitos identificados.

- **Segurança** - Os sistemas de monitoramento de condições de emergência podem ser definidos sob o ponto de vista de Segurança ("*Safety*") quando a conseqüência de pelo menos uma falha temporal exceda em muito os benefícios normais do sistema podendo levar a uma catástrofe. Os sistemas críticos de tempo real podem ser subdivididos em: Sistema de tempo real crítico Seguro em caso de falha ("*fail safe*") onde um ou vários estados seguros podem ser atingidos em caso de falha (p.ex., parada obrigatório de trens na ocorrência de falha do sistema de sinalização ferroviário); e Sistemas Operacionais de Tempo Real Crítico em caso de falha ("*fail operational*") que, na presença de falhas parciais, podem se degradar fornecendo alguma forma de serviço mínimo (por exemplo, sistema de controle de vôo com comportamento degradado, mas ainda seguro.) **[FAR 00]**. O sistema de monitoramento de situações de emergência em ambientes físicos é considerado, neste trabalho, como crítico em relação à segurança, uma vez que qualquer falha no sistema seja falha de componentes de *hardware,* de *software* ou até mesmo transmissão de informações inconsistentes para quem requisitou o serviço pode colocar em risco a vida de pessoas e trazer sérios prejuízos financeiros e de

patrimônio. Por exemplo, para garantir a segurança em ambientes sujeitos a vazamento de gases tóxicos, um sistema de monitoramento de condições de emergência deveria não apenas detectar incidentes (como o vazamento de gás) que poderiam levar a uma explosão, como também emitir sinal de aviso quando desta ocorrência (sistemas de supervisão e controle).

- **Eficiência e Desempenho** – Eficiência e desempenho são requisitos importantes em sistemas que precisam ter resposta em tempo-real. Estes envolvem também requisitos de **tempo de resposta** (tempo aceitável do ponto de vista do usuário para que alguma operação seja concluída); de **vazão** (***throughput*** **-** quantidade de dados que precisam ser processados em um intervalo pré-definido de tempo) e de temporização. Requisitos não-funcionais de desempenho são aqueles que se referem à velocidade de operação do sistema. De acordo com SOMMERVILLE et. al. **[SOM 98]** o **tempo de resposta** e a **temporização** especificam quão rápido o sistema precisa coletar, por exemplo, dados de sensores antes que sejam sobrescritos por outros dados da mesma natureza, ou quão rápido o sistema precisa realizar o processamento para que dados não sejam perdidos. Para aumentar a eficiência, por exemplo, pode-se especificar um serviço responsável pela filtragem do montante de eventos, possivelmente redundantes, que estão chegando da rede de sensores, e com isso, diminuir o tráfego na rede aumentando, conseqüentemente, o desempenho do sistema.
- **Modularidade** – Em sistemas críticos a subdivisão do sistema em vários módulos permite que a separação de conceitos seja aplicada em duas fases: (1) ao se lidar com cada módulo isoladamente (ignorando os detalhes dos outros módulos); e (2) ao

se lidar com as características gerais de cada módulo e suas relações com os outros, de forma a integrá-los e a construir um sistema coerente.

- **Confiabilidade** - Em sistemas de monitoramento de condições críticas é indispensável a garantia de confiabilidade e certeza de que o sistema irá operar de acordo com o esperado. Mesmo na ocorrência de falha de algum nó da rede de sensores, ou falha durante a comunicação, os dados deverão estar corretos. Informalmente, um sistema é confiável se o usuário pode depender dele. A integridade e a consistência dos dados também devem ser garantidas neste tipo de sistema. Pode-se especificar, por exemplo, que um serviço de tratamento de eventos não pode falhar quando o sistema está monitorando uma condição crítica de alta prioridade. Uma das funcionalidades para garantir esta exigência poderia ser a aplicação de redundância no serviço (cópia do serviço roda paralelamente e assume o controle no caso de falha).

Outros requisitos desejados em sistemas de monitoramento de condições de emergência são: **interoperabilidade, usabilidade**, **manutenabilidade, extensibilidade** e **portabilidade**.

4.5 Considerações finais

Este capítulo introduziu os conceitos de RF's e RNF's e identificou os principais requisitos não funcionais de alguns sistemas tradicionais em que a qualidade de serviço é fundamental. Alguns RNF´s importantes para a classe de aplicações baseadas no monitoramento de condições de emergência em ambientes cientes de contexto foram identificados. As exigências destes requisitos, tais como confiabilidade, segurança, modularidade e eficiência, além da correção temporal de eventos vindos da rede de sensores e

entregues através de comunicação assíncrona, colocam em xeque as metodologias convencionais, sob pena de perdas em termos financeiro, humano e ambiental. Essas aplicações necessitam de suportes computacionais e metodologias que vão além das estruturas tipicamente utilizadas e lançam novos desafios para os projetistas desse tipo de sistemas.

Este trabalho não objetiva garantir a dependabilidade de um sistema, mas sim identificar requisitos importantes para a qualidade de sistemas de monitoramento de condições de emergência em ambientes cientes de contexto, e propor soluções para garantir a manutenção da consistência dos eventos que chegam da rede de sensores, aumentando a confiabilidade do sistema. A Qualidade de Serviço – QoS está relacionada ao atendimento de requisitos não funcionais de sistemas, tais como eficiência e confiabilidade, oferecendo, por exemplo, mecanismos para atender esses requisitos (exemplos incluem *buffers*, filas, mecanismos de escalonamento etc). Não foi utilizado aqui, em nenhum momento, o termo QoS, que está sendo tratado na rede de sensores, por outro aluno de mestrado do LRVNet. Entretanto, entende-se que as soluções propostas aqui deverão ser parte das soluções que garantirão QoS ao sistema.

O próximo capítulo apresenta a proposta de um ***middleware*** que atende estes requisitos não funcionais para aplicações de monitoramento de condições de emergência em ambientes cientes de contexto.

5 *Middleware* para Monitoramento de situações de emergência

5.1 Considerações iniciais

A computação ubíqua ciente de contexto é amplamente explorada em aplicações de sala de aula, sala de reuniões, guias turísticos, residências e também na área de segurança crítica: segurança industrial, segurança na aviação, segurança no controle metro-ferroviário, monitoramento de pacientes, etc. O avanço nas áreas de dispositivos heterogêneos de visualização e acesso à informação, de sensores e de redes de comunicação sem fio, tem impulsionado as aplicações que necessitam de monitoramento de granularidade fina na ajuda à prevenção, combate e avaliação de situações de emergência. No primeiro caso (prevenção), informações podem ser capturadas fazendo com que condições críticas possam ser evitadas, no segundo caso (combate), podem ajudar com informações mais detalhadas para tomadas de decisão no gerenciamento de equipes de resgate e salvamento (por exemplo, na localização de pessoas em ambientes tomados por fumaça e/ou chamas), e no terceiro caso (avaliação), auxiliar perícias por parte de seguradoras, treinadores de equipes de combate a acidentes, e até os responsáveis pela segurança do ambiente monitorado.

Este trabalho considera sistema de Monitoramento de Condições de Emergência em Ambientes Cientes de Contexto, um ambiente físico ou lógico povoado por sensores que capturam informações de contexto conforme ocorrências no ambiente, e remetem essas informações (eventos) para quem solicitou (aplicações ou serviços).

Como já apresentado no capítulo 3, as estruturas de captura e acesso existentes oferecem, tipicamente, suporte para ambientes de salas de reunião e ensino, não atendendo assim, alguns requisitos importantes de sistemas de monitoramento de condições de emergência em ambientes físicos ou lógicos cientes de contexto tais como manutenção da consistência de eventos, baixa latência (uso de interfaces do tipo *publish/subscribe* baseada em tópicos para simplificar e agilizar o processo de captura e acesso) etc.

Este capítulo apresenta a proposta de um ***middleware*** que é definido para prover serviços independentes de plataforma, sistema operacional e/ou linguagem de programação que atende os requisitos funcionais e não funcionais da classe de aplicações de monitoramento de situações de emergência em ambientes cientes de contexto.

5.2 Projeto de um sistema de Monitoramento de Condições de Emergência em Ambientes Cientes de Contexto

O Sistema de Monitoramento de Ambientes Cientes de Contexto que integra realidade virtual, redes de sensores e computação ubíqua é um projeto que está sendo desenvolvido no Laboratório de Realidade Virtual em Rede (LRVNet) no Departamento de Ciência da Computação da UFSCar.

O sistema objetiva o monitoramento de condições críticas em ambientes físicos e ambientes lógicos (programas e ambientes computacionais) sujeitos a situações de emergência, além do tratamento dos contextos obtidos a partir destes ambientes, a localização de pessoas e objetos no ambiente físico e a visualização em tempo real e posterior dos eventos em progresso no ambiente físico. A **Figura 5.1** mostra a visão geral do projeto.

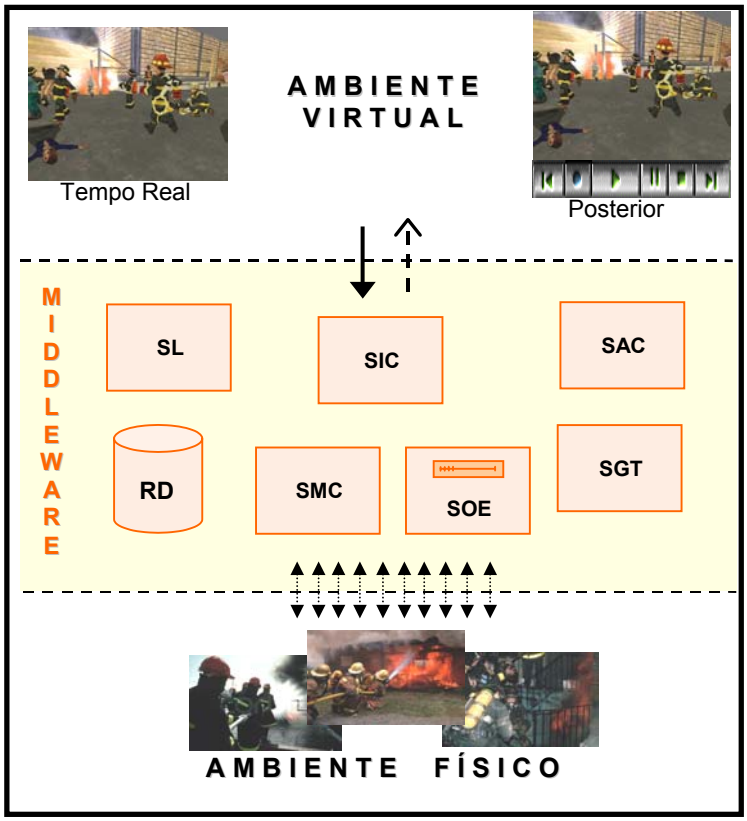


Figura 5.1 - Visão Geral do Sistema de Monitoramento.

O Projeto de monitoramento de condições críticas em ambientes cientes de contexto é dividido em três subprojetos:

1. Algoritmos para redes de sensores que atendam às necessidades da classe de aplicações de monitoramento, isto é, protocolos tolerantes a falhas, cientes de consumo de energia, e com garantia de qualidade de serviço (entrega garantida e dentro de prazo pré-estabelecido);

2. Projeto e avaliação de estruturas de suporte às aplicações de monitoramento em ambientes cientes de contexto (*frameworks*, *middlewares*, *toolkits* e bibliotecas). Um *middleware* está sendo projetado e deverá ser avaliado, que atende os requisitos funcionais e não funcionais da referida classe de aplicações de monitoramento. Estes serviços incluem: Serviço de Interpretação de

3. Contexto(**SIC**), Serviço de Localização (**SL**), Serviço de Adaptação de Conteúdo (**SAC**) e um Repositório de Dados (**RD**). Os requisitos não funcionais são relacionados ao tratamento dos eventos da rede de sensores: Manutenção de Consistência (**SMC**), Ordenação de Eventos (**SOE**), além de serviços de gerenciamento de Tópicos (**SGT**), que dá suporte ao esquema de subscrição/publicação adotado como o mecanismo de interface entre todos os módulos do sistema, a aplicação e a rede de sensores. A proposta de um *middleware* é parte deste sub-projeto (parte destacada da **Figura 5.1**);
4. Ambientes de visualização de situações de monitoramento no ambiente físico e lógico em dispositivos heterogêneos (de PDAs a CAVERNAS digitais), em tempo real e posterior.

A próxima seção apresenta a proposta de uma arquitetura composta por um ***middleware*** e seus respectivos serviços para o monitoramento de situações de emergência em ambientes físicos povoados por sensores.

5.3 Arquitetura de um *Middleware* para classe de aplicações de monitoramento de situações de emergência em ambientes cientes de contexto

Esta seção apresenta a descrição de uma proposta de arquitetura composta por um *middleware* e seus serviços e a comunicação baseada no mecanismo *Publish/Subscribe* em eventos.

5.3.1 Descrição da Arquitetura do *Middleware*

Middleware refere-se a uma plataforma distribuída de interfaces e serviços que reside entre a aplicação e o sistema operacional e facilita o desenvolvimento, disponibilidade e gerenciamento de aplicações distribuídas **[COU 02]**.

Os seguintes serviços são oferecidos pelo *middleware,* conforme mostra a **Figura 5.1:**

- **Serviço de Gerenciamento de Tópicos (SGT):** responsável por descobrir se existe determinado tópico, por criar novos tópicos, editar os tópicos já existentes e excluir tópicos (através de políticas que são ditadas durante a configuração do sistema). Por exemplo, quando uma aplicação ou serviço deseja receber notificações sobre a ocorrência de eventos específicos (ocorrência de vazamento de gás). Para isso há a necessidade de subscrever-se em um tópico que forneça esse tipo de notificação, a aplicação ou o serviço interessado solicita informações do serviço de gerenciamento de tópicos que sabe sobre a existência deste e, caso não exista ele é criado. Edição de tópicos podem reaproveitar tópicos antigos que não estão mais em uso ou então esses tópicos antigos podem ser excluídos de acordo com a política escolhida;
- **Serviço de Localização (SL):** responsável por receber subscrições de consultas sobre localização de pessoas e objetos. Dependendo da tecnologia envolvida nos sensores de localização, as tarefas deste serviço podem ser mais ou menos complexas. Por exemplo, a aplicação ou serviço interessado pode solicitar a localização de uma pessoa e usar uma política de confiabilidade da informação dizendo que deseja que se trace a trajetória da pessoa para comprovar se realmente é possível esta pessoa estar no lugar onde foi informado. Desta forma a notificação

- sobre a localização da pessoa é informada de forma precisa e com maior confiabilidade;
- **Serviço de Interpretação de Contexto (SIC):** responsável por receber subscrições de aplicações que se interessam por informações que forneçam mais do que somente os dados puros vindos dos sensores. O serviço de interpretação de contextos notifica os contextos (eventos) depois de interpretá-los segundo regras que são fornecidas pela aplicação que requisitou o serviço;
- **Serviço de Ordenação de Eventos (SOE):** responsável por ordenar os eventos baseando-se na ordem temporal de ocorrência e/ou por políticas informadas por quem fez a subscrição (aplicação ou serviço);
- **Serviço de Manutenção de Consistência (SMC):** responsável por aplicar mecanismos que aumentam a confiabilidade da informação resolvendo problemas tais como ambigüidade e redundância entre eventos. O conceito de redundância neste sentido está relacionado com a multiplicidade dos eventos recebidos;
- **Serviço de Adaptação de Conteúdo (SAC):** responsável por adaptar conteúdos em função das capacidades dos dispositivos, do meio de comunicação e da aplicação;
- **Repositório de Dados (RD):** responsável pelo armazenamento dos dados de contexto puros e interpretados.

O foco deste trabalho é a especificação dos módulos de serviço de interpretação de contextos, serviço de manutenção de consistência, serviço de localização, serviço de ordenação de eventos e serviço de gerenciamento de tópicos. A **Figura 5.2** apresenta o diagrama *use-case* que descreve as funcionalidades do *Middleware*.

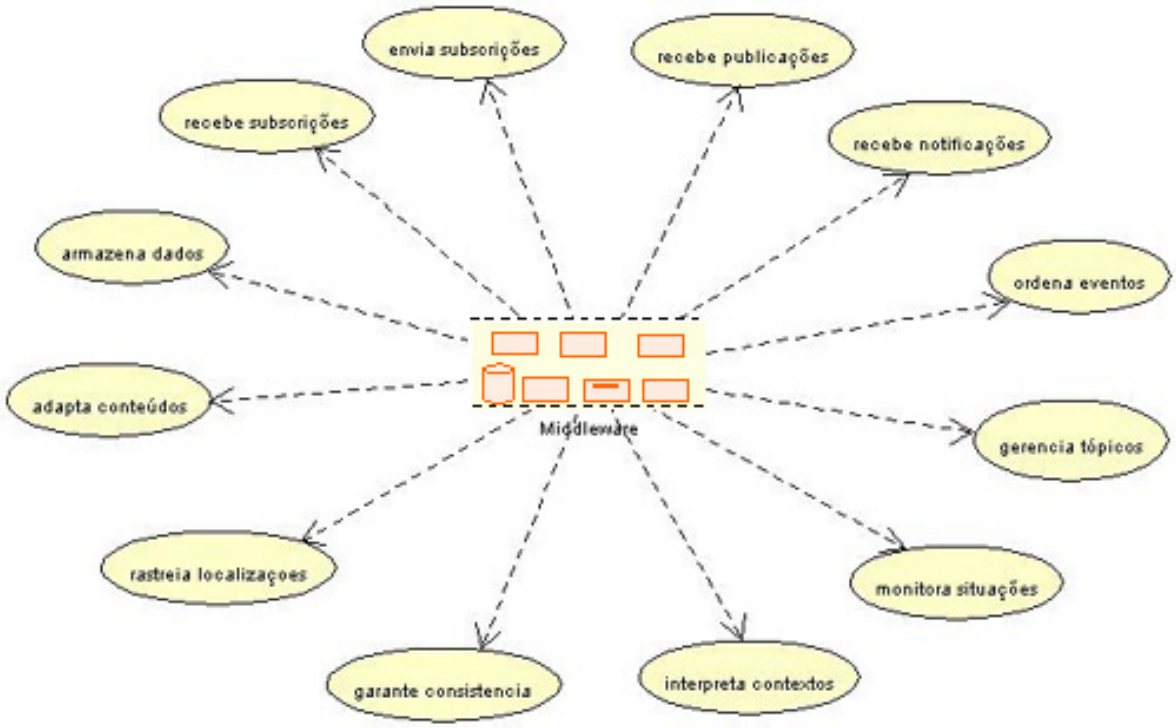


Figura 5.2 – Diagrama Use-Case das funcionalidades do *Middleware.*

Os serviços fornecidos pelo *middleware* são independentes uns dos outros e podem ser solicitados entre si ou por aplicações. O **SIC**, por exemplo, pode ser solicitado por uma aplicação que deseja receber informações toda vez que a pressão interna de uma determinada caldeira estiver acima do normal. Para que esta interpretação seja feita (acima do normal), é necessária a entrada dos dados que geram essa regra de interpretação (ex: Pressão > 500). Depois que esses dados são fornecidos, o **SIC** subscreve-se nos sensores que monitoram a caldeira para receber notificações sobre pressão. Quando a regra for satisfeita, ou seja, quando a pressão "sentida" pelos sensores for maior que 500, o serviço de interpretação notifica a aplicação subscrita sobre o evento.

Assim como uma aplicação pode solicitar notificações ao **SIC** fornecendo-lhe condições para estas notificações, ela também pode solicitar os serviços de localização do módulo **SL** quando seu interesse é na localização de pessoas e/ou objetos. Neste caso, os

eventos capturados, através de rastreamento de sensores, são notificados ao **SL** que, em posse dessa informação, pode gerar localizações mais precisas ou mais próximas do contexto requerido pela aplicação.

Inconsistência de eventos é um problema observável em redes de sensores, uma vez que o sensor notifica todos os eventos assim que eles são "sentidos" no ambiente, gerando, desta forma, muitos eventos redundantes ou ambíguos. O **SMC**, que é o responsável por eliminar esses problemas, pode se comunicar com qualquer um dos outros módulos ou diretamente com a rede de sensores para receber todos os eventos, conforme eles são gerados e notificados, e tratá-los de forma a eliminar redundância e ambigüidade entre os eventos.

Da mesma forma que o **SMC**, o **SOE** também pode se comunicar com qualquer dos outros módulos do *middleware* ou diretamente com a rede de sensores. A função do **SOE** é ordenar temporalmente os eventos antes que eles sejam processados, minimizando este processamento para os outros módulos.

A próxima seção apresenta a comunicação baseada no mecanismo *publish/subscribe* em eventos utilizada neste trabalho para comunicação entre aplicações, *middleware* e os módulos do *middleware*.

5.3.2 Comunicação baseada no mecanismo *Publish/Subscribe*

Um **evento** é uma ocorrência instantânea em um momento de tempo **[HAN 99]** e está relacionado ao objeto no qual ele ocorreu. O mecanismo de comunicação baseado em eventos utilizado neste trabalho entre as aplicações e os serviços do *middleware* é baseado no mecanismo *Publish/Subscribe*. Neste mecanismo, os *subscribers* (entidades interessadas em receber eventos) podem definir seus interesses em cada evento e serem notificadas quando um

evento de seu interesse for gerado por um *publisher*. Um evento é propagado para todos os *subscribers* interessados naquele evento, de forma assíncrona[3], diminuindo assim o tráfego de dados desnecessários. Vários paradigmas de comunicação podem ser usados para promover a interação entre as entidades que produzem eventos e os destinos que consomem estes eventos. Exemplos de tais paradigmas incluem troca de mensagens, invocações remotas, notificações, espaços compartilhados e enfileiramento de mensagens. O problema básico com estes paradigmas é que eles falham em promover um desacoplamento total entre as entidades participantes, tornando o sistema menos flexível e menos escalável. Eugster et al. **[EUG 00]** fazem um excelente estudo destes paradigmas e os compara com o paradigma de publicação/subscrição, o qual têm recebido crescente atenção por oferecer desacoplamento entre produtores e consumidores no **tempo** (publicadores e subscritores não precisam estar ativos na interação ao mesmo tempo), **espaço** (publicadores e subscritores não precisam conhecer um ao outro) e **fluxo** (publicadores e subscritores não precisam estar sincronizados para interagir). Os modelos mais conhecidos de Subscrição e Publicação são **[EUG 00]**:

1. **Baseado em tópico ou assunto** – as subscrições identificam apenas classes de eventos pertencentes a um dado canal ou assunto, em que tópicos representam palavras-chave, do tipo: cotação de ações, ou jogos do campeonato brasileiro. Participantes podem publicar notificações e subscrever para recebê-las. São mais simples de implementar, mas menos flexíveis e expressivos;
2. **Baseado em conteúdo** – eventos são classificados de acordo com propriedades, especificadas pelos *susbscribers,* que os eventos podem ter – apenas os eventos que satisfazem estas propriedades serão entregues aos *subscribers.* Isto dá às entidades

[3] A abordagem assíncrona foi, mais usualmente, introduzida por R. Milner **[Mil 80]** e trata a ocorrência e a percepção dos eventos independentes numa ordem arbitrária, mas não simultânea.

subscritoras muito mais opções de subscrição, além de filtragem de informação, o que reduz o tráfego da rede, porém aumenta a complexidade do sistema;

3. **Baseado em Tipo** – os eventos são objetos, como, instâncias de tipos nativos em uma linguagem de programação orientada a objetos, o que leva a maior segurança em relação a tipos, além de melhor encapsulamento de eventos. No modelo baseado em tipo, o *subscriber* de um tipo específico de objeto só vai receber eventos daquele tipo.

O mecanismo de publicação/subscrição escolhido, neste trabalho, é o baseado em tópico por ser simples de implementar, o que pode representar menor latência na entrega dos eventos, um requisito importante em ambientes de monitoramento de condições de emergência. A **Figura 5.3** mostra a interação entre a aplicação, os serviços do *middleware* e a rede, através do paradigma de publicação/subscrição.

Aplicações podem demonstrar interesse em eventos do ambiente, através da emissão de subscrições para o *middleware,* em tópicos específicos, do tipo média de temperatura do ambiente, umidade de uma área do ambiente, situação de incêndio, rebelião, explosão, localização de pessoas e/ou de objetos, etc.

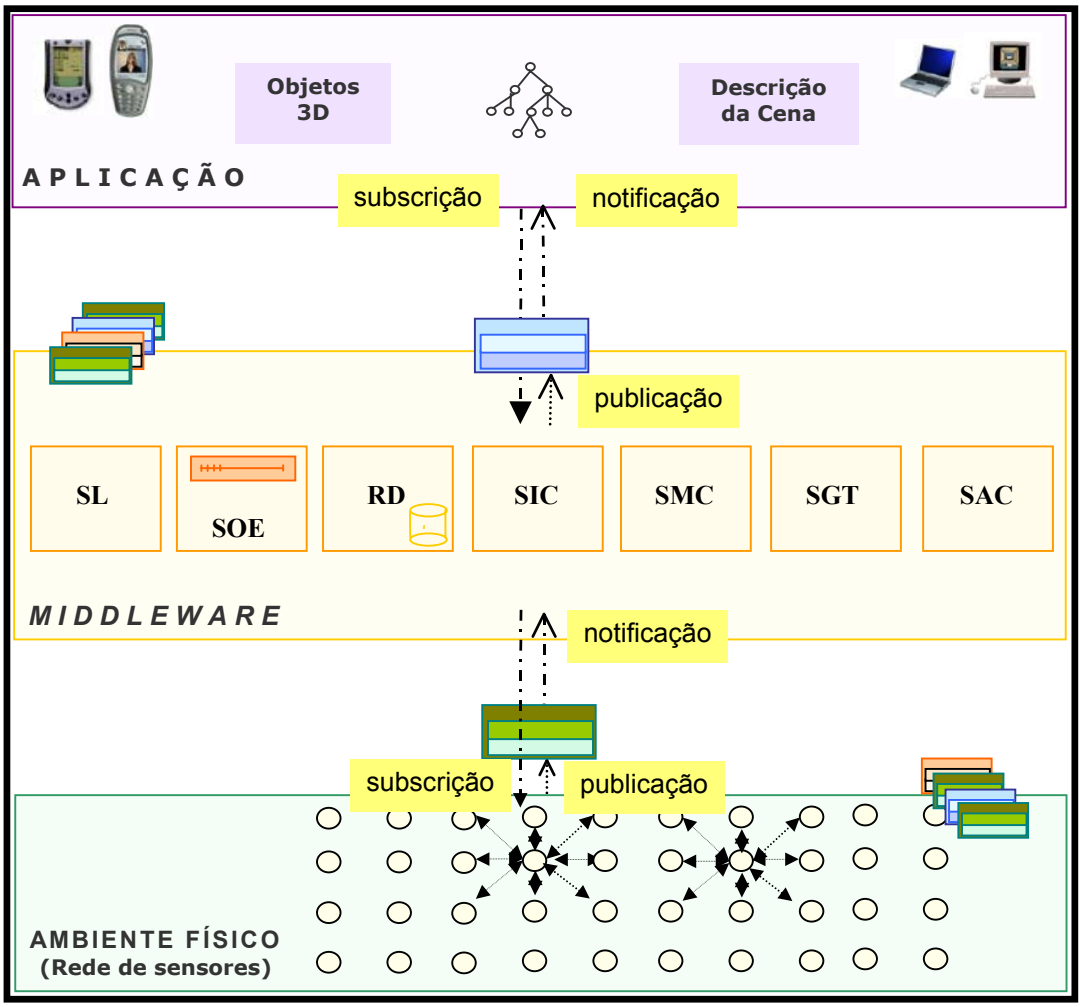


Figura 5.3 - Interação entre Aplicação, Serviços do Middleware e Rede de Sensores.

Para isto, é importante que um sistema de monitoramento de condições de emergência forneça suporte para os seguintes tipos de monitoramento: **(1) periódico:** é especificada uma peridiocidade para receber informações (deseja-se receber a cada 10 minutos informações sobre a temperatura do SETOR-1); **(2) orientada a eventos:** é especificada uma condição em que tem interesse para receber informações relacionadas a uma entidade e a um tipo de sensor (deseja-se receber dados sobre a temperatura do SETOR-1 quando a temperatura estiver acima de 50 ºC); e **(3) baseada em consulta:** são definidos a entidade e o tipo de sensor de

interesse (deseja-se consultar a temperatura do SETOR-1: entidade = SETOR-1; tipo de sensor = temperatura).

5.3.2.1 Subscrições, Publicações e Notificações

As subscrições, publicações e notificações de eventos são especificadas como uma coleção de tópicos sobre os quais entidades participantes (aplicações ou serviços do *middleware*) mostram interesse e podem ser solicitadas a qualquer momento. As aplicações de monitoramento de condições críticas têm interesse em informações capturadas do ambiente que possam sinalizar, em especial, condições anormais, sejam estas de tipo simples, como temperatura ou pressão muito altas, ou complexas como um incêndio ou explosão, ou ainda a localização de vítimas em situações de incêndio.

As situações de monitoramento são definidas com base nas condições que um responsável pela segurança definiu no momento da configuração do sistema e podem ser re-definidas a qualquer momento. Uma lista de interesses da classe de aplicações considerada inclui: qualquer tipo de informação para a qual exista um sensor capaz de capturá-la ("senti-la"); condições críticas inerentes a um ambiente específico de aplicação (os interesses de uma aplicação de monitoramento de condições críticas em uma indústria de produtos químicos certamente serão diferentes em um museu ou em um avião).

Com o objetivo de fornecer ao usuário uma maneira de entrar com informações importantes para configuração das situações que devem ser monitoradas no ambiente físico foi construída uma Interface de Configuração e Monitoramento como parte de um outro trabalho de mestrado apresentado em maio deste ano **[KUD 04]**. Através desta interface o

usuário (responsável pela segurança) deverá fornecer as seguintes informações para a configuração:

- O que deve ser protegido (será considerada uma entidade para o sistema);
- Quais os tipos de sensores existentes no ambiente, relacionados com cada entidade em particular e o número de identificação de cada sensor;
- Contra quais situações as entidades devem ser protegidas (ex: incêndio, explosão, curto-circuito), quais os tipos de sensores responsáveis pela detecção dessas situações (ex: temperatura, chama, fumaça, corrente-elétrica), e as condições de monitoramento (ex: T>40º).

As aplicações e/ou serviços subscrevem-se para receber eventos, e são notificados assim que os eventos se tornam disponíveis. A publicação dos eventos ocorre quando eles são produzidos por entidades produtoras de eventos (puros ou processados). Eventos puros são produzidos por sensores e publicados em tópicos que mostraram interesse. Eventos processados são produzidos pelos serviços do *middleware* (**SOE, SIC, SL ou SMC**) após algum processamento e são publicados em todos os tópicos que mostraram interesse em recebê-los. Estes tópicos, por sua vez, notificam as entidades interessadas. Desta forma, observa-se que um tópico recebe **subscrições** de entidades interessadas em receber notificações de eventos (aplicação e/ou serviços) e recebe **publicações** de entidades produtoras de eventos (sensores publicam eventos puros e, serviços publicam eventos processados).

Subscrições em tópicos para notificações de eventos puros, por exemplo, são emitidas para a rede de sensores que capturam informações diretamente do ambiente físico. Quando estas coincidirem com os critérios descritos nas subscrições, são publicadas nos tópicos pelos sensores e estes tópicos decidem, através do identificador da subscrição, quem deve ser notificado.

5.3.2.2 A definição da entidade Tópico num modelo de publicação/subscrição baseado em Tópico

Um tópico é representado por uma abstração que representa, em sua essência, um motor de notificação **[EUG, 2000]**. No modelo clássico de interação *Publish/Subscribe*, os Tópicos costumam serem vistos como grupos, em que a subscrição para um tópico T é vista como uma associação a um grupo T. Entretanto, eles têm uma natureza mais dinâmica, uma vez que os participantes podem subscrever em vários tópicos diferentes enquanto os grupos, normalmente têm um conjunto disjunto de membros.

Em um modelo de *Publish/Subscribe* baseado em Tópicos, um participante mostra interesse em um tópico, subscrevendo no tópico. A **Figura 5.4** mostra a dinâmica do modelo *Publish/Subscribe* baseado em Tópicos.

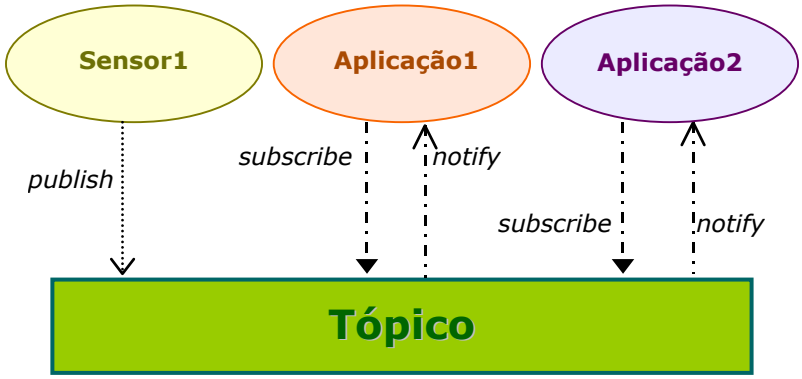
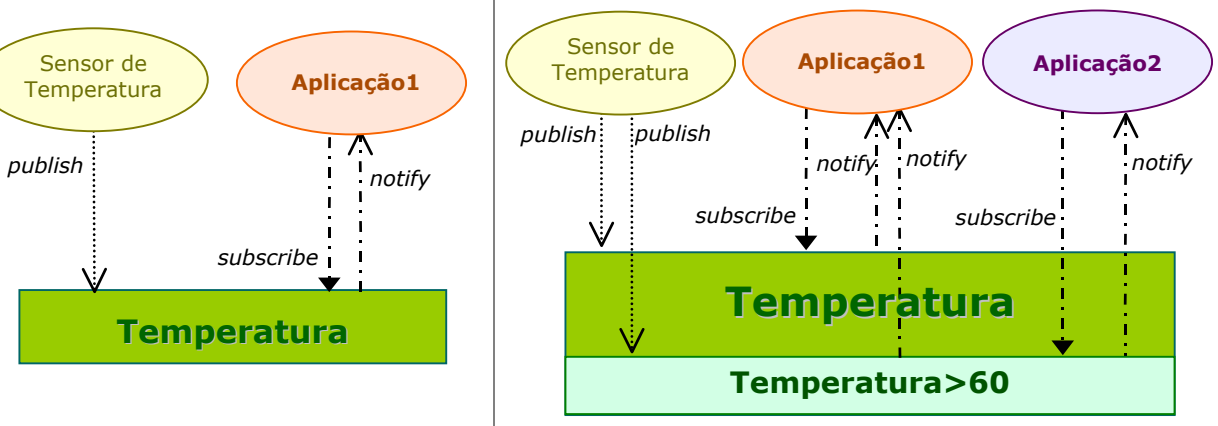


Figura 5.4 - *Publish/Subscribe* baseado em Tópicos – adaptado de [EUG, 2000].

Os tópicos podem ser estruturados hierarquicamente em sub-tópicos, por exemplo, um tópico pode ser derivado ou especializado de um outro tópico, como mostra a **Figura 5.5 (a)** e **(b)**.



(a) **(b)**

Figura 5.5 - Exemplo de Publish/Subscribe baseado em Tópicos para um tópico de temperatura (a), e exemplo de Publish/Subscribe baseado em Tópicos para um tópico de temperatura e um sub-tópico de temperatura>60 (b).

5.3.2.3 Implementação de Tópicos

Mensagens que representam as subscrições, publicações e notificações são trocadas na rede entre os módulos do *middleware* e com a aplicação em formato XML e são formatadas e

empacotadas através do protocolo SOAP **[W3C 00]**. A utilização de XML e do protocolo SOAP, possibilita que os serviços do *middleware* sejam disponibilizados como serviços *web*, podendo ser acessados por qualquer aplicação com acesso a *Internet*. Desta forma, aumentam-se as possibilidades de acesso ao sistema de monitoramento. A escolha de XML foi motivada pelo seu alto poder de representatividade e sua capacidade de extensão. A linguagem oferece flexibilidade suficiente para representar consultas e tarefas de sensores. SOAP foi escolhido por ser um protocolo leve e versátil. Além disso, ambos são padrões de *facto* na *Web*.

A subscrição da aplicação nestes tópicos dá-se através da estrutura mostrada na **Figura 5.6**. Essas subscrições são emitidas ao serviço de eventos do *middleware.*

```
<subscribe>
      <subscription_id></subscription_id>
      <subscription_type></subscription_type>
      <subscription_value></subscription_value>
      <entity_type></entity_type>
      <entity_value></entity_value>
      <policy_type></policy_type>
      <policy_value></policy_value>
      <text></text>
      <value></value>
</subscribe>
```

Figura 5.6 - Descrição XML para subscrição.

Onde:

- **subscription_id -** associa uma subscrição a uma determinada notificação;
- **subscription_type** – indica qual o tipo de subscrição desejada (periódica, orientada a eventos e baseada em consulta);
- **subscription_value** – indica o tipo de subscrição relacionada ao serviço solicitado (de uma lista a ser descrita em cada um dos serviços abaixo);

- **entity_type** – indica o tipo de entidade sobre a qual se requer informação (Ex: *person*; *object*; *environment*);
- **entity_value -** refere-se à entidade sobre a qual se requer informação;
- **policy_type** – indica o tipo de política que um serviço deve atender, por exemplo, manutenção de consistência (*Consistency*) e ordenação de eventos (*Order*);
- **policy_value -** refere-se a uma política particular. Por exemplo, se alguns eventos dentre vários que representam a temperatura de uma sala chegam com valores muito discrepantes em relação à maioria, a política requisitada de manutenção da consistência (policy_type = *consistency*), a ser explicada mais abaixo no texto, pode ser a de considerar como valor correto a média aritmética dos valores recebidos (ARITH_AVRG), ou ainda, o valor que ocorreu em maior número (MODA). As políticas serão descritas nas seções de Serviços correspondentes.
- **text e value -** descrevem informação dependente do tipo de subscrição (periódica, orientada a eventos, baseada em consulta)**.**

A notificação da aplicação e do *middleware* nestes tópicos dá-se através da estrutura mostrada na **Figura 5.7**.

```
<notify>
      <subscription_id></subscription_id>
      <value></value>
<notify >
```

Figura 5.7 - Descrição XML para notificação.

Onde:

- **subscription_id -** subscrição associada a esta notificação;
- **value –** valor a ser notificado como resultado de uma subscrição.

As próximas seções descrevem os serviços do *middleware* para aplicações de monitoramento de condições de emergência, a saber: Serviço de Gerenciamento de Tópicos, Serviço de Interpretação de Contexto, Serviço de Manutenção de Consistência, Serviço de Localização e Serviço de Ordenação de Eventos. O Serviço de Adaptação de Conteúdo não é foco principal deste trabalho e por isso não será abordado com maiores detalhes. No entanto, observa-se a importância de agregar um módulo para este fim devido à evolução e diversidade de dispositivos, interfaces e aplicações que podem fazer uso do sistema de monitoramento de condições de emergência em ambientes físicos cientes de contexto.

5.3.3 Serviço de Gerenciamento de Tópicos

Um tópico é representado por uma abstração ou um motor de notificação **[EUG 00]**. Em um modelo *Publish/Subscribe* baseado em Tópicos, aplicações e serviços mostram interesse em um tópico, subscrevendo-se nele. O *subscriber*, que neste caso é uma aplicação ou um serviço que emitiu uma subscrição, é notificado dos eventos correspondentes ao tópico em que se subscreveu.

O Serviço de Gerenciamento de Tópicos possui controle sobre todos os tópicos existentes, e pode ser consultado por *subscribers,* que desejam receber notificações sobre determinadas situações, para saber se o tópico de interesse já existe. O SGT oferece as seguintes primitivas:

- **Create_topic ( )** – cria um tópico novo, caso este não exista no diretório de tópicos;

- **Destroy_topic ( )** – elimina um tópico existente. Um tópico só pode ser removido quando não houver *subscribers* aguardando notificação;

- **Find_topic ( )** – busca um tópico no diretório de tópicos. Retorna um "*handler*" para o tópico encontrado. Este "*handler*" é utilizado pelos *subscribers* para subscreverem-se no referido tópico;

- **Add_topic ( )** – adiciona um tópico no diretório de tópicos;

- **Remove_topic ( )** – remove um tópico do diretório de tópicos;

- **Edit_topic ( )** – permite a edição da descrição de um tópico no diretório de tópicos.

5.3.4 Serviço de Interpretação de Contexto

Quando as informações solicitadas pelos serviços do *middleware* são simples, estes podem se subscrever diretamente nos eventos que geram estas informações. Em casos mais complexos do tipo consultas sobre localização ou monitoramento de condições de emergência, então as subscrições são emitidas para os serviços correspondentes. Da mesma forma, serviços do *middleware* podem emitir subscrições em outros serviços (serviço de interpretação de contexto pode solicitar serviços do módulo de ordenação de eventos, por exemplo). Em situações de monitoramento que envolve o processamento de expressões lógicas, o serviço de interpretação de contexto é solicitado. A **Figura 5.8** mostra a descrição do esquema XML correspondente à subscrição de monitoramento para o serviço de interpretação.

```
<subscribe>
      <subscription_id>00057</subscription_id>
      <subscription_type>event/periodic/query</subscription_type>
      <subscription_value>CHECK_FIRE/any</subscription_value>
      <entity_type>environment/object/person</entity_type>
      <entity_value>environmentX/objectX/personX</entity_value>
      <expression>"FIRE AND TEMP>100 AND FLAME"/any</expression>
</subscribe>
```

Figura 5.8 - Exemplo XML para subscrição no SIC

A **Figura 5.8** apresenta os seguintes parâmetros:

- **subscription_id, subscription_type, entity_type** e **entity_value** já foram descritos nas seções acima;
- **subscription_value** – tipo de situação a ser monitorada (ex: CHECK_EXPLOSION, CHECK_FIRE, CHECK_REBELLION, CHECK_ GAS_LEAKING, etc);
- **expression** – expressão lógica correspondente que deve ser satisfeita para a emissão de notificação (ex.: "Vazamento de Gás Tóxico AND Presença de Fumaça AND Presença de Chama"). As expressões são no momento, definidas pelo usuário, através de uma interface de configuração implementada como parte de outro trabalho de dissertação.

O Serviço de Interpretação de Contextos é responsável por gerar, a partir das informações necessárias para o processamento da expressão, subscrições de interesse em eventos publicados pelos sensores (ou em serviços correspondentes). Cada uma dessas subscrições é identificada unicamente. Uma estrutura no serviço de interpretação é instanciada e identificada através de *subscribe_id*, que é ligado à condição sendo monitorada (por exemplo, "desabamento"). Nesta estrutura são colocados os valores publicados por

sensores, através de notificação de eventos. Cada evento notificado é munido de um identificador único que emparelha com o identificador da subscrição correspondente. O serviço de interpretação processa a expressão e quando todos os eventos correspondentes à expressão são entregues, o resultado do processamento é notificado para a entidade *subscriber*, através de descrição de notificação mostrada na **Figura 5.9**.

```
<notify>
      <subscription_id>00058</subscription_id>
      <value>returned_value</value>
</notify >
```

Figura 5.9 - Descrição XML para notificação.

A próxima seção descreve o serviço oferecido pelo módulo Serviço de Manutenção de Consistência.

5.3.5 Serviço de Manutenção de Consistência

O serviço de manutenção de consistência de eventos foi especificado com o objetivo de aumentar a confiabilidade das informações notificadas às entidades, atendendo parcialmente alguns **RNF's**. Problemas de inconsistência de eventos, tais como ambigüidade e redundância, são comuns em redes de sensores que produzem muitos eventos assíncronos conforme eles ocorrem no ambiente físico.

A introdução de mecanismos capazes de minimizar estes tipos de problemas diminui o número de mensagens transmitidas reduzindo, conseqüentemente, a propagação de mensagens falsas ou redundantes pela rede e para as aplicações. A manutenção da consistência de eventos

foi especificada como um serviço para que seja mais fácil uma eventual migração deste serviço para uma rede de sensores.

O serviço de manutenção de consistência de eventos é parametrizado de tal forma a atender diferentes políticas, como por exemplo: Média, MODA (valor a ser considerado é o valor da maioria), etc. Suponha que uma aplicação se subscreva para receber a média da temperatura da SALA-A. O serviço de manutenção de consistência subscreve-se no sensor ou nos sensores que "sentem" a temperatura da SALA-A, a partir do tópico de temperatura presente no sensor, para receber notificações sobre a variação da temperatura quando esta estiver acima de 40, como mostra o diagrama uml da figura 5.10.

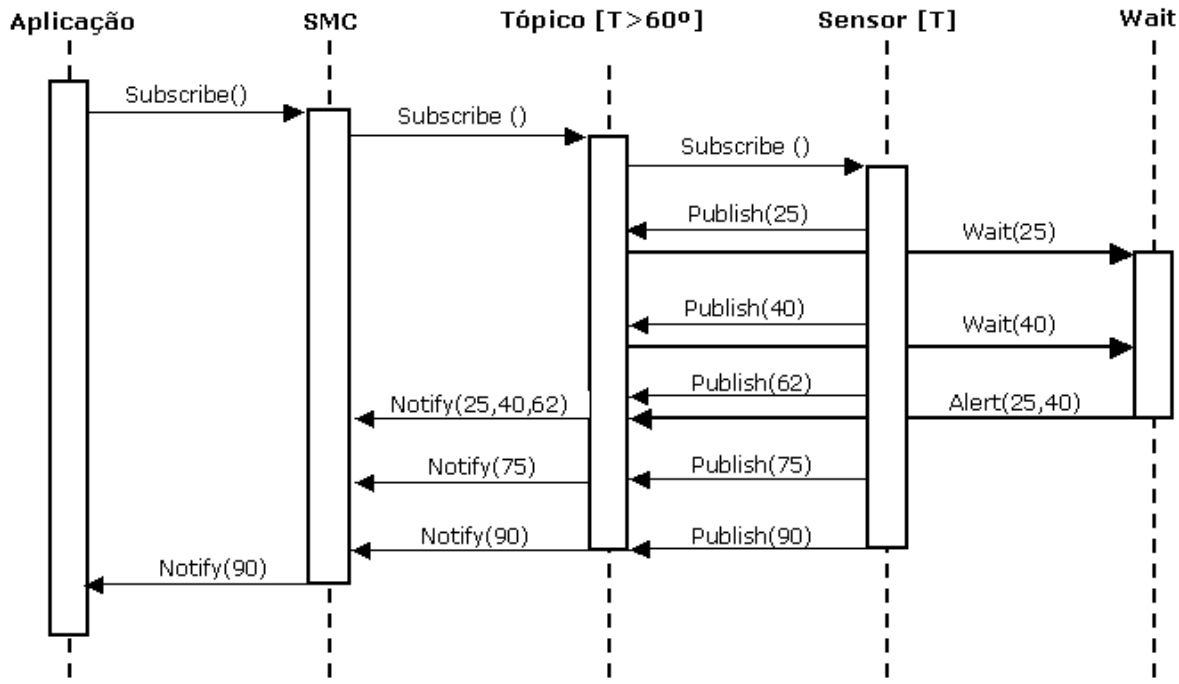


Figura 5.10 - Diagrama UML que representa monitoramento de Temperatura.

Quando os eventos são entregues dos sensores, estes podem apresentar valores ambíguos e redundantes, dependendo das variações capturadas pelos diversos sensores e das próprias

interferências durante a transmissão da mensagem. O serviço de manutenção de consistência de eventos deverá então registrar cada valor recebido dos sensores e realizar um processamento que atenda a política solicitada pelo *subscriber.* As políticas podem ser estabelecidas no momento da configuração do sistema, através de interface de configuração. Exemplos de políticas são: **média aritmética** que informará o quociente entre a soma dos valores e o número total dos valores. Por exemplo, num total de 6 sensores, 4 deles informam que a temperatura da SALA-A é igual a 60 ºC e os outros 2 informam que a temperatura sentida é 48 ºC. Se a subscrição requisitar a média aritmética, a notificação informará que a média de temperatura é 56 ºC; a **moda** informa o valor que ocorre com maior freqüência entre todos os valores recebidos. Outro exemplo de política pode solicitar que se deseja receber a **média** dos valores descartando aqueles mais discrepantes (ex. descarta o maior e o menor valores). Outras políticas podem ser solicitadas pela entidade que deseja o serviço como, por exemplo, ela pode informar que deseja receber o resultado final com aproximação para valor inteiro. Neste caso, o valor que, eventualmente, seria fracionário é notificado como valor inteiro aproximado.

O esquema XML para as subscrições no Serviço de Consistência de Eventos é mostrado na **Figura 5.11.**

```
<subscribe>
      <subscription_id>00059</subscription_id>
      <entity_type>entity</entity_type>
      <entity_value>entityX</entity_value>
      <policy_type>CONSISTENCY</policy_type>
      <policy_value>ARITH_AVRG/MODA/etc</policy_value>
</subscribe>
```

Figura 5.11 - XML para subscrição no serviço de manutenção de consistência.

Onde:

- **subscription_id** e **entity_type** já foram descritos nas seções acima;
- **entity_value** – representa a instância de um tipo de entidade, por exemplo, “Pedro” (pessoa); “maca” (objeto), etc.
- **policy_type** – indica política, no caso, de manutenção de consistência (CONSISTENCY);
- **policy_value -** política a ser atendida (ARITH_AVRG, MODA, APROX_INT, HIGHEST, LOWEST, etc.).

O serviço de manutenção de consistência descrito acima deverá ser implementado como parte da camada de aplicação de uma rede de sensores, a ser usado em conjunto com protocolos de roteamento de rede que agregam dados com o objetivo de minimizar o tráfego da rede e economizar energia dos sensores.

A descrição da notificação de entrega segue a estrutura descrita na **seção 5.3.2.3.** A próxima seção descreve o serviço de localização de pessoas e objetos baseando-se no rastreamento destas entidades no ambiente físico que está sendo monitorado.

5.3.6 Serviço de Localização

O Serviço de Localização provê a localização de pessoas e/ou objetos em ambientes físicos, um serviço fundamental principalmente em situações de emergência. Em uma condição de incêndio, por exemplo, informações sobre a localização de pessoas podem ser utilizadas pela equipe de resgate para minimizar o número de vítimas durante a definição de estratégias de salvamento, além de auxiliar a tomada de decisões no combate ao fogo. A

localização de objetos também é um serviço importante em situações de emergência, por exemplo: localização de "macas" em uma determinada ala de um hospital, ou a localização de extintores de incêndio, numa área particular (extintores podem deslocar-se de suas posições padrão numa situação de sinistro).

Os métodos mais comuns de localização usam medir a distância de uma fonte emissora (por exemplo, uma pessoa usando um crachá ou etiqueta RFID) móvel ou não, aos sensores fixos. Essas medidas podem ser baseadas na **Potência do Sinal** (através de leituras de RSSI - *Received Signal Strength);* no **Tempo de Chegada** (tri-lateração hiperbólica, ToA ou Diferença de tempo de chegada - TDoA); ou ainda no **Ângulo de Chegada** (AoA, triangulação). A informação de localização (por exemplo, a posição x, y e z mais provável da fonte emissora) é publicada como evento nos tópicos correspondentes. Assim que estes eventos são publicados, o serviço de localização pode mapear esta informação em uma localidade conhecida (SALA-A, SETOR-1, ALA-B, etc). A aplicação que se subscreve para receber esta informação é então notificada.

Basicamente, três tipos de consulta são suportados pelo serviço de localização:

- **Onde está pessoa/objeto X** – consulta que requer o local onde está uma pessoa ou objeto sendo procurado;

- **Quem/O que está no ambiente Z** – tipo de consulta utilizada quando a aplicação requer a lista de pessoas (Quem) ou de objetos (O que) presentes em determinada localidade;

- **Tem alguém/algo no ambiente Y** - Neste caso, deseja-se receber notificação (resposta booleana S/N) sobre a existência ou não de pessoas ou objetos numa determinada localidade (ex. SETOR-1).

As **Figuras 5.12, 5.13 e 5.14** ilustram os esquemas XML para cada um dos tipos de subscrição descritos acima.

```
<subscribe>
      <subscription_id>00510</subscription_id>
      <subscription_type>query</subscription_type>
      <subscription_value>LOCATE</subscription_value>
      <entity_type>person/object</entity_type>
      <entity_value>personX/objectY</entity_value>
</subscribe>
```

Figura 5.12 - XML para localização de pessoa ou objeto.

```
<subscribe>
      <subscription_id>00511</subscription_id>
      <subscription_type>query</subscription_type>
      <subscription_value>LIST_PEOPLE/LIST_OBJS</subscription_value>
      <entity_type>environment</entity_type>
      <entity_value>environmentX</entity_value>
</subscribe>
```

Figura 5.13 - XML para determinar quais pessoas ou objetos estão em um particular ambiente.

```
<subscribe>
      <subscription_id>00512</subscription_id>
      <subscription_type>query</subscription_type>
      <subscription_value>PERSON/OBJECT</subscription_value>
      <entity_type>environment</entity_type>
      <entity_value>environmentX</entity_value>
</subscribe>
```

Figura 5.14 - XML para determinar a presença de pessoas ou objetos em um particular ambiente.

Onde:

- **subscription_id, subscription_type** e **entity_type** já foram descritos nas seções acima;
- **subscription_value** – tipo de subscrição (locate, list_people; list_objs; isthere_person; isthere_obj);

5.3.7 Serviço de Ordenação de Eventos

O comportamento correto de um sistema não depende só da integridade dos resultados obtidos, mas também dos valores de tempo em que são produzidos. Um evento notificado fora de tempo pode ser sem utilidade ou até representar risco em ambientes de monitoramento de condições de emergência uma vez que os usuários do sistema confiam que ele estará operando de forma correta.

Em nosso entendimento algo aconteceu às 03:00h se ela ocorreu depois que nosso relógio marcou 02:59h e antes dele marcar 03:01h. Em sistemas que consistem de módulos e serviços espacialmente distribuídos que se comunicam através de mensagens, a afirmação anterior não é tão simples. É praticamente impossível dizer que um de dois eventos aconteceu primeiro. A relação “aconteceu antes de” é apenas uma ordenação parcial de eventos em sistemas desta natureza.

Diversos mecanismos são propostos na literatura para solucionar problemas de ordenação de eventos. Não é objetivo deste trabalho entrar em detalhes sobre soluções para ordenar eventos e sim justificar a importância deste serviço estar presente no *middleware* para sistemas de monitoramento de condições de emergência em ambientes físicos.

Sistemas de monitoramento baseados em sensores com comunicação assíncrona, como é o caso do sistema proposto, onde a comunicação entre as entidades envolvidas acontece independente uma das outras, os eventos podem chegar às partes interessadas em desordem de ocorrência. Isto pode comprometer todo um sistema, uma vez que eventos que aconteceram depois podem ser entregues primeiro e eventos que aconteceram antes podem chegar atrasados devido às falhas na rede que podem acontecer por diversos fatores, como por exemplo, a capacidade limitada de comunicação que pode ser compartilhada pelos diversos nós da rede.

A ordenação de eventos é especificada em um módulo separado cujo serviço pode ser solicitado por qualquer um dos outros serviços do *middleware* ou pelas aplicações. A ordenação pode inclusive migrar para a rede de sensores quando a rede emprega, por exemplo, agregação de dados (vários sensores enviam eventos para um nó da rede que então notifica o *subscriber* para aqueles eventos – antes da notificação este nó pode submeter os eventos recebidos de nós vizinhos ao serviço de manutenção de consistência e também ao de ordenação de eventos).

Um serviço que recebe eventos vindos diretamente da rede de sensores, por exemplo, pode solicitar ao Serviço de Ordenação que os eventos sejam ordenados temporalmente antes de serem entregues. Uma subscrição feita para o Serviço de Ordenação de Eventos é descrita como mostra a **Figura 5.15**:

```
<subscribe>
      <subscription_id>00013</subscription_id>
      <entity_type>environment</entity_type>
      <entity_value>environmentX</entity_value>
      <policy_type>order</policy_type>
      <policy_value>FIFO/LCFS/any</policy_value>
</subscribe>
```

Figura 5.15 - XML para subscrição de solicitação de ordenação de eventos.

Onde:

- **subscription_id, subscription_type** e **entity_type** já foram descritos nas seções anteriores;
- **policy_type** – indica política, no caso, de ordenação de eventos (ORDER);
- **policy_value -** indica política a ser atendida para ordenar os eventos. Alguns exemplos são: FIFO (*First In – First Out*) e LCFS (*Last Come First Served*);

A relação existente entre a rede de sensores e o *Middleware* para o sistema de monitoramento de condições de emergência é descrita na seção seguinte.

5.3.8 A Rede de Sensores e o *Middleware*

Redes de sensores podem ser beneficamente utilizadas em sistemas que monitoram condições críticas em ambientes, uma vez que avanços recentes tanto nas tecnologias de comunicação como no sensoriamento e processamento possibilitaram o desenvolvimento de nós sensores multifuncionais de baixo custo e consumo, que são pequenos em tamanho e que se comunicam a curtas distâncias. Além disso, os sensores podem detectar precisamente as mudanças ocorridas no ambiente como: agentes químicos no ar e na água, ajudando a identificação, concentração, e localização dos poluentes; a variação de nível de temperatura; a presença de gases tóxicos no ar ou o nível de fumaça.

Uma rede de sensores é composta por um grande número de nós sensores, os quais são densamente instalados no ambiente que se quer monitorar, como mostrado na **Figura 5.16**. Os nós sensores geralmente são espalhados por um campo de sensores. Cada um desses nós possui a capacidade de coletar e encaminhar dados a uma estação base (*sink*).

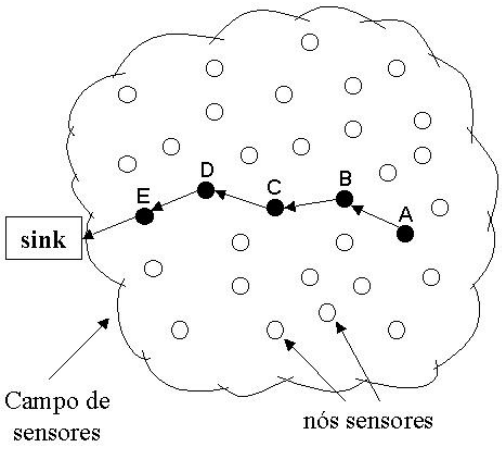


Figura 5.16 - Nós sensores espalhados em um campo de sensores.

A rede de sensores se comunica com o *middleware* para enviar pacotes de notificações que contem os dados requisitados para o monitoramento, como por exemplo, a concentração de gás tóxico em uma sala durante uma situação de emergência. Para realizar esta operação, o *middleware* envia subscrições para a rede de sensores informando sobre quais eventos deseja ser notificado: entidade Tal, nível de concentração acima de X, etc. Quando estas coincidirem com os critérios descritos nas subscrições, são publicadas nos tópicos pelos sensores e estes tópicos decidem, através do identificador da subscrição, quem deve ser notificado. Neste caso, o módulo, ou serviço, do *middleware* que fez a subscrição é quem será notificado.

A próxima seção apresenta um estudo de caso para exemplificar o mecanismo de subscrição-publicação-notificação no sistema de monitoramento de condição de emergência.

5.4 Estudo de caso de subscrição-publicação-notificação no monitoramento de situações críticas

O monitoramento de cada situação do ambiente físico é solicitado pelas aplicações, através de subscrições. Essas subscrições podem são geradas a partir de configurações realizadas, por exemplo, por um profissional responsável pela segurança (engenheiro ou

técnico de segurança). É este profissional quem define as expressões lógicas que correspondem às situações a serem monitoradas. Exemplos dessas situações foram dados nas seções anteriores e esta seção apresenta o monitoramento de uma dessas situações (incêndio), conforme ilustra a **Figura 5.17**.

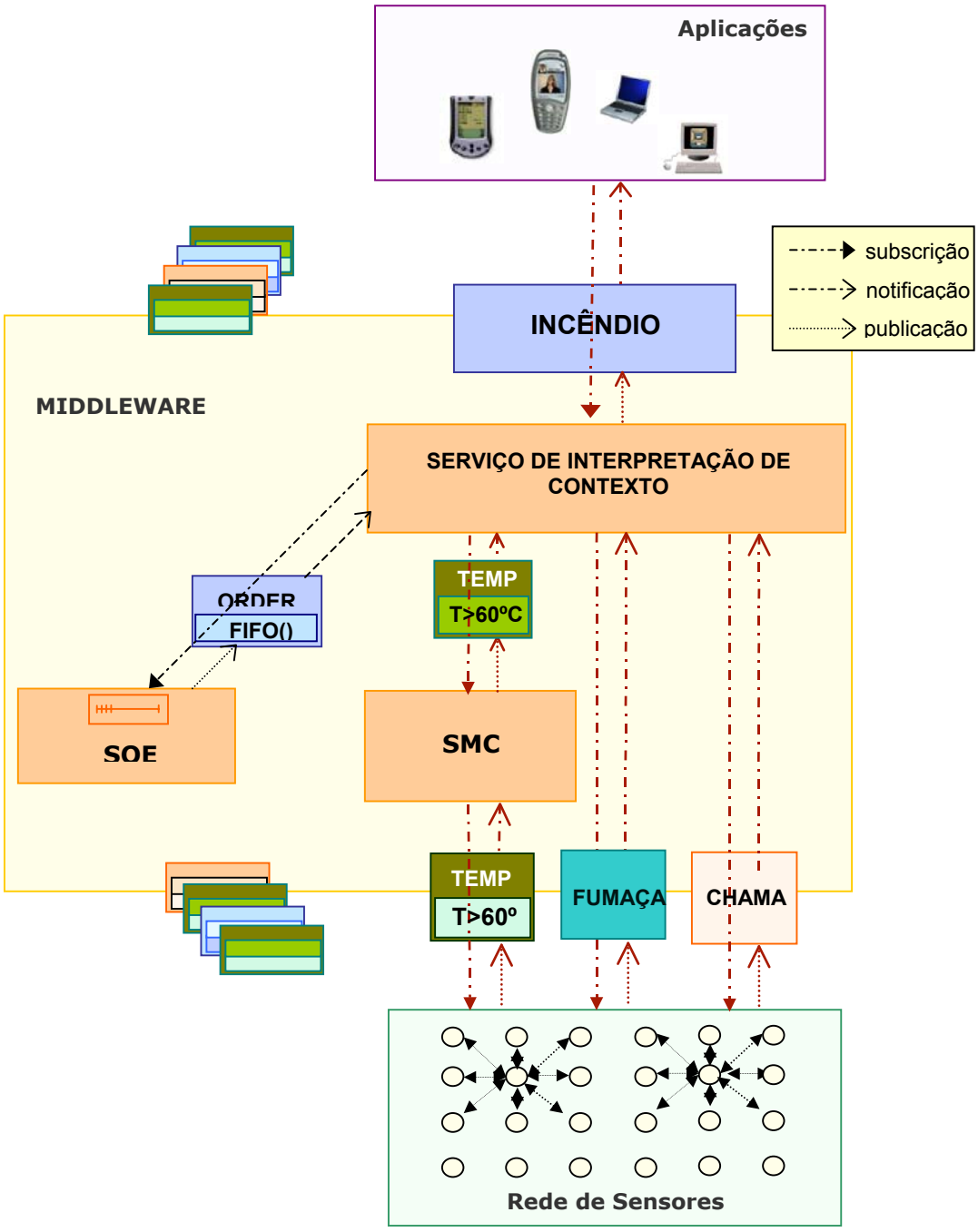


Figura 5.17 - Esquema de monitoramento de condição de incêndio.

Para o monitoramento da condição de incêndio, a aplicação subscreveu-se no Serviço de Interpretação de Contextos informando o tipo e o valor da subscrição (QUERY e FIRE, respectivamente), além da expressão lógica que indica incêndio neste ambiente (FIRE = "T>100 AND FUMAÇA AND CHAMA"), e das políticas, uma seqüência de tipos e valores que devem ser atendidas (tipo de política = CONSISTENCY, valor = MODA) e (tipo de política = ORDER e valor = FIFO). A **Figura 5.18** mostra como seria a descrição da subscrição para monitoramento da situação de incêndio:

```
<subscribe>
      <subscription_id>00515</subscription_id>
      <subscription_type>query</subscription_type>
      <subscription_value>CHECK_FIRE</subscription_value>
      <entity_type>environment</entity_type>
      <entity_value>PAVILHAO-4</entity_value>
      <expression>"T>60°C AND FUMAÇA AND CHAMA"</expression>
      <policy_type>order</policy_type>
      <policy_value>FIFO</policy_value>
      <policy_type>consistency</policy_type>
      <policy_value>MEDIA<policy_value>
</subscribe>
```

Figura 5.18 - XML para subscrição de monitoramento de situação de incêndio.

Para atender todas as requisições desta subscrição, o Serviço de Interpretação deve solicitar informações sobre temperatura, fumaça e chama. Além disso, a subscrição identifica políticas (FIFO e MODA) que devem ser atendidas para satisfazer aos resultados da situação monitorada. Os serviços responsáveis por atender estas políticas são o Serviço de Manutenção de Consistência de Eventos, e o Serviço de Ordenação de Eventos. O SIC subscreve-se então nos serviços SMC e SOE respectivamente, que deverão aplicar mecanismos de manutenção de consistência nos eventos de temperatura e o serviço de ordenação de eventos sobre todos os eventos de temperatura, fumaça e chama. Depois de concluídos estes serviços, notificações são enviadas ao SIC da média da temperatura, e da ordenação dos eventos. Desta forma, o

SIC não precisa se subscrever diretamente nos sensores de temperatura evitando receber todos os eventos puros publicados, somente se subscreve nos sensores de fumaça e chama. O SMC receberá todos os eventos ambíguos, redundantes e puros sobre temperatura que são notificados da rede de sensores. O SOE é solicitado pelo SMC para ordenar todos os eventos sobre média de temperatura e sobre eventos puros de fumaça e chama de acordo com a política FIFO. A **Figura 5.19** mostra como seria uma subscrição, feita pelo SIC, para o SMC solicitando média das ocorrências de temperatura.

```
<subscribe>
      <subscription_id>00516</subscription_id>
      <entity_type>environment</entity_type>
      <entity_value>PAVILHAO-4</entity_value>
      <policy_type>CONSISTENCY</policy_type>
      <policy_value>ARITH_AVRG</policy_value>
</subscribe>
```

Figura 5.19 - XML para subscrição no SMC de chama.

O SMC, a partir de subscrição de temperatura recebida, emite subscrições nos sensores responsáveis por publicar eventos de temperatura que ocorrerem no PAVILHAO-4. A **Figura 5.20** mostra um exemplo de subscrição para os sensores que monitoram a condição temperatura no PAVILHAO-4.

```
<subscribe>
      <subscription_id>00517</subscription_id>
      <entity_type>environment</entity_type>
      <entity_value>PAVILHAO-4</entity_value>
</subscribe>
```

Figura 5.20 - XML para subscrição no Sensor de temperatura localizado no PAVILHAO-4.

Os parâmetros para subscrição nos sensores são reduzidos, visto que sensores "sentem" contextos de determinada condição (sensor de fumaça sente presença de fumaça, sensor de inundação sente o nível da água, sensor de temperatura sente a temperatura).

Desta forma, o SMC, receberá as notificações de eventos puros gerados pelos sensores que "sentem" temperatura no PAVILHAO-4, eliminará todas as redundâncias e ambigüidades de eventos recebidos, processará os eventos tirando a média e publicará o resultado no tópico correspondente. O tópico, por sua vez, notifica o SIC das variações de temperatura sentidas no ambiente.

O SIC aguarda até que todas as condições da expressão de incêndio são satisfeitas no mesmo instante de tempo e solicita os serviços de Ordenação de Eventos para ordenar estes temporalmente antes que sejam enviados para a aplicação.

O Serviço de Gerenciamento de Tópicos pode ser solicitado por qualquer um dos módulos para saber sobre a existência de tópicos e quais os sensores envolvidos no monitoramento da situação de incêndio.

A próxima seção apresenta as conclusões finais deste capítulo que introduziu uma proposta de *middleware* para suporte aos sistemas de monitoramento de condições de emergência em ambientes cientes de contexto.

5.5 Considerações Finais

A comunicação desempenha um papel importante nos tempos de resposta. Em sistemas convencionais o aumento de desempenho na forma de taxas de transferências

representa o ponto-chave, por outro lado, na comunicação em sistemas de monitoramento de condições de emergência, que pode ser requisitado em tempo real, o que se procura é a obtenção de altas probabilidades que um evento será entregue dentro de um prazo mínimo (muitas vezes específico) ou atendendo uma latência máxima **[FAR 00]**. A vantagem dos contextos serem fornecidos à aplicação no formato XML é o fato de não ficar atrelado a nenhum tipo de mídia, sendo que a aplicação pode utilizá-lo no formato que desejar (textual, 2D, 3D, etc), e também em qualquer tipo de dispositivo (PC, *notebook*, PDAs, celulares).

Os requisitos considerados os mais importantes neste trabalho, como já foi detalhado no capítulo 4, constituem: Confiabilidade, Eficiência, Modularidade e Segurança. Ainda outros requisitos desejáveis podem ser alcançados pela própria definição de arquitetura por adotar a plataforma *middleware* que além de grande flexibilidade, oferece vantagens como melhor portabilidade, interoperabilidade e facilidades na evolução do sistema. Mais uma vez é lembrado que este trabalho não especificou os requisitos não funcionais para o sistema de monitoramento de situações de emergência, uma vez que o foco principal está voltado para a proposta de um *middleware*, e para melhor exposição da idéia foram apresentados conceitos de RNF's e quais deles foram abordados para especificação da arquitetura.

Deve-se observar, porém, que a implementação, que não é detalhada neste trabalho, deve atender a outros requisitos mencionados no capítulo 4, como a portabilidade, por exemplo.

O requisito **eficiência** está ligado à velocidade e ao tempo de operação do sistema e ao **desempenho** que é um fator importante para obter resultados em projeto de sistemas e estão intimamente associados à arquitetura de *software*, assim os mecanismos de comunicação têm influência direta sobre o desempenho obtido. Neste trabalho, acredita-se que a especificação

de um Serviço de Manutenção de Consistência, responsável por receber o montante de eventos vindo da rede de sensores e tratar estes eventos de acordo com as políticas especificadas na subscrição eliminando ambigüidade e redundância de eventos, pode aumentar a **eficiência** e o **desempenho** do sistema. Da mesma forma o Serviço de Ordenação de Eventos é especificado com o intuito de aumentar o desempenho do sistema oferecendo um serviço específico para ordenar os eventos de acordo com *algoritmos* específicos para este fim.

A implementação do modelo de subscrição e publicação é parte do trabalho de outro aluno de mestrado. Vários mecanismos podem ser utilizados para que uma entidade participante possa se subscrever em um evento. Alternativas vão desde a utilização de plataformas do tipo CORBA até arquiteturas como JXTA da SUN.

6 Conclusões

Este trabalho apresentou um *middleware* de suporte à classe de aplicações de monitoramento de condições de emergência em ambientes físicos ou lógicos cientes de contexto que atende requisitos funcionais e não funcionais para garantir a qualidade dos serviços oferecidos a essas aplicações. As arquiteturas propostas na literatura, como mostra o capítulo 3 deste trabalho, não atendem a todas as necessidades destes sistemas de monitoramento, uma vez que focam sistemas como salas de reuniões e residências.

Middleware foi a estrutura escolhida neste trabalho por facilitar a utilização de componentes que podem ser desenvolvidos de forma independente, garantindo por sua própria natureza, requisitos como flexibilidade, interoperabilidade e facilidades de manutenção e extensão na evolução do sistema.

As principais contribuições deste trabalho são listadas a seguir.

6.1 Contribuições

A primeira contribuição é a proposta de uma camada independente de plataforma de software e hardware entre aplicações e a rede de sensores, que fornece ao usuário uma visão abstrata da rede como uma fornecedora de serviços e permite que diferentes aplicações de monitoramento de condições críticas utilizem redes de sensores. O *middleware* é proposto com a característica de ser usado por aplicações distintas que atendem a usuários com necessidades diferentes suprimindo algumas limitações de arquiteturas convencionais que são, geralmente, construídas para uma única aplicação alvo.

A segunda contribuição deste trabalho é uma melhor compreensão do mecanismo *publish/subscribe* baseado em tópicos para a comunicação entre a rede de sensores, os serviços do *middleware* e as aplicações. O mecanismo escolhido é simples conceitualmente podendo oferecer a troca de eventos entre os módulos do sistema de maneira mais rápida, diminuindo assim a latência, requisito importante em aplicações de monitoramento de granularidade fina. As mensagens são transmitidas entre a rede de sensores e o *middleware* e deste para as aplicações no formato XML, através do protocolo SOAP que foram escolhidos por oferecer maior interoperabilidade e flexibilidade para adicionar novos parâmetros. Além disso, sua utilização viabiliza a utilização dos serviços do *middleware* e a interação com a rede de sensores via *Internet*.

A terceira contribuição é a identificação e a especificação de mecanismos para garantir que requisitos de qualidade ou requisitos não funcionais das arquiteturas de computação ubíqua para o suporte aos sistemas de monitoramento de condições críticas sejam atendidos.

6.2 Limitações e Trabalhos Futuros

O mecanismo baseado em tópicos possui uma limitação, que apesar de ser aceita para esta classe de sistemas de monitoramento, deve ser mencionada. A característica deste mecanismo é a criação de diversos tópicos, de acordo com a necessidade das aplicações e serviços, para cada situação a ser monitorada. Por exemplo, se a situação a ser monitorada é "*Pressão > 200 da caldeira BG4"* é criado um tópico "PRESSÃO" para monitorar todos os sensores que "sentem" a pressão da caldeira BG4, e um sub-tópico "PRESSÃO>200" para os *subscribers* interessados neste valor particular de pressão. Se existir um outro *subscriber* que deseja monitorar a pressão da mesma caldeira quando "*Pressão > 100"*, um novo sub-tópico

poderá ser criado sob o tópico “PRESSÃO” ou ainda pode-se criar um novo tópico “PRESSÃO>100”. Novas solicitações de variações de valores de pressão exigirão a criação de novos sub-tópicos ou ainda de tópicos adicionais, podendo levar a um número grande desses elementos, resultado de uma certa inflexibilidade do mecanismo de tópico. Por outro lado, é um mecanismo mais simples de implementar e de mais rápido processamento, o que pode vir a ser uma vantagem adicional na garantia de latência mínima na entrega de eventos.

Uma implementação futura do Serviço de Interpretação de Contexto através da utilização de Lógica *Fuzzy* está prevista. A lógica *Fuzzy* possibilita a interpretação de expressões como, por exemplo: *perigo de explosão* no SETOR-1, ou ainda *pressão alta* na caldeira BG4. A lógica *Fuzzy* pode oferecer resultados que seriam difíceis de obter com a lógica clássica.

Uma relação de dependência causal entre os vários eventos de um sistema é absolutamente desejável para os resultados de sistemas onde a transmissão de mensagens é feita de forma assíncrona, e a noção de tempo é básica para que seja possível estabelecer essa relação. Neste sentido, o Serviço de Ordenação de Eventos pode usar princípios baseados em políticas, que seriam ditadas pela aplicação, para tratar a ordenação de eventos. As políticas poderiam configurar qual é o mecanismo, ou a melhor política, a ser utilizada pelo SOE. Desta forma, pode-se flexibilizar o serviço de acordo com a necessidade.

Serviços do *middleware*, como a manutenção de consistência e ordenação de eventos, estão sendo implementados por outro aluno de mestrado como serviços da camada de aplicação da rede de sensores. A camada da aplicação utiliza protocolos de roteamento da camada de redes, que agregam dados, minimizando o tráfego na rede de sensores e economizando energia na transmissão de dados.

Também são trabalhos futuros a implementação e a avaliação, quanto à funcionalidade, latência, confiabilidade, dos serviços de localização, manutenção de consistência, gerenciamento de tópicos e interpretação de contexto, de acordo com, pelo menos, as seguintes métricas:

- Taxa de entrega do *middleware* – número de eventos que chegam da rede de sensores em relação ao número de eventos efetivamente entregues à aplicação;
- Atraso Médio – latência média do momento em que um evento é recebido da rede de sensores, em um dos serviços do *middleware*, até o momento em que é entregue à aplicação;
- Atraso de serviço – tempo decorrido do momento em que uma subscrição é recebida por um serviço, através do mecanismo baseado em tópicos, até o momento em que uma notificação é emitida para um destino (aplicação ou outro serviço); Esta avaliação será feita para cada serviço do *middleware*.

6.3 Conclusões Finais

Conclui-se ao final deste trabalho que, sistemas de captura e acesso, como os existentes para sistemas cientes de contexto, podem ser utilizados de forma inovadora em sistemas de monitoramento de condições de emergência em ambientes físicos e lógicos cientes de contexto. Contextos podem ser capturados dos ambientes de forma contínua e minuciosa, através de sensores, processados por serviços que aumentam a confiabilidade dos dados e interpretados de forma a detectar situações de perigo e auxiliando nas decisões que precisam ser tomadas de maneira rápida e eficaz quando situações anormais são detectadas.

A estrutura proposta poderia ser utilizada por aplicações cientes de contexto, para acesso em tempo real ou posterior, cujas experiências são mapeadas numa representação

virtual tridimensional com intuito de potencializar as ações humanas em situações que envolvem treinamento, perícia e monitoramento. Alguns exemplos dessas situações são: combate a incêndios, avaliação do desempenho de forças policiais em atividades de segurança (como bancos, prédios, penitenciárias, favelas) entre outras.

Este trabalho apresenta contribuições no sentido de explorar a utilização da computação ubíqua no monitoramento de condições de emergência destacando requisitos essenciais para esta classe de aplicação.

Referências Bibliográficas

[ABO 00] ABOWD, G.D., MYNATT, E. D., Charting Past, Present and Future Research in Ubiquitous Computing, **ACM Transactions on Computer-Human Interaction**, pp. 29-58, March 2000.

[ABO 97] ABOWD, G. D., ATKESON, C. G., HONG, J., LONG, S., KOOPER, R., PINKERTON, M. Cyberguide: A mobile context-aware tour guide. **Baltzer/ACM Wireless Networks**, p. 421-433, 1997.

[ABO 99] ABOWD, G.D. Classroom 2000: An experiment with the instrumentation of a living educational environment. **IBM Systems Journal**, v. 38, Outubro/1999.

[AIM 03] AIM (2003). Radio Frequency Identification (RFID) home page. Disponível em http://www.aimglobal.org/technologies/rfid/. *The* **Association for Automatic Identification and Data Capture Technologies.**

[BAL 03] BALDOCHI, L.A., CATTELAN, R.G., PIMENTEL, M.G., Building a Middleware Infrastructure for Capture and Access Applications, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, **s.n.t.,** May 2003.

[BAN, 2002] BANAVAR, G, BERNSTEIN, A. Software Infrastructure and Design Challenges. **Communications of the ACM**, v.45, n.12, pag.92-96, 2002.

[BOE 76] BOEHM, B., BROWN, J.R. and LIPOW, M. Quantitative Evaluation of Software Quality in Proc. of 2nd **International Conference on Soft. Eng**. San Francisco, oct 1976, pp: 592-605.

[BOE 78] BOEHM, B. Characteristics of Software Quality. North Holland Press, **s.n.t.,** 1978.

[BOW 92] BOWEN, J.; Stayridow, V. Formal methods and software safety. In: Safety of Computer Control Systems 1992 (SAFECOMP'92*)* of Proc. IFAC Symposium, Zürich, Switzerland. **Anais.** p. 28-30 Outubro 1992, Pergamon Press, 1992.

[BRO 97] BROWN, P.J., BOVEY, J.D, CHEN, X. Context-aware applications: From the laboratory to the marketplace. **IEEE Personal Communications**, p.58-64. Outubro/1997.

[BRO 98] BROTHERTON, J.A., ABOWD, G.D., TRUONG, K.N. Supporting Capture and Access Interfaces for Informal and Opportunistic Meetings. **Georgia Institute of Technology Technical Report: GIT-GVU-99-06**, Atlanta, GA, 1998.

[BRU 00] BRUMITT, B., MEYERS, B., KRUMM, J., KERN, A., SHAFER, S. A. Easyliving: Technologies for Intelligent Environments. **Proceedings of the Handheld and Ubiquitous Computing Second International Symposium (HUC'2000),** p.12-29, 2000.

[BUR 02] BURRELL, J., GAY, G. K., KUBO, K., FARINA, N. Context-aware computing: A test case. **Proceedings of the International Conference on**

computing: A test case. **Proceedings of the International Conference on Ubiquitous Computing**, p.1-15, 2002.

[CAM 96] CAMARGO JUNIOR, J. B. **Estudo da segurança em sistemas de controle metro-ferroviários**. Escola Politécnica da Universidade de São Paulo. São Paulo, 1996. Tese (Doutorado).

[CHU 00] CHUNG, L.; NIXON, B.; YU, E.; MYLOPOULOS, J. Non-Functional Requirements in Software Engineering. **Kluwer Academic Publishers,** 2000.

[CHU 99] CHUNG, L.; NIXON, B. A.; YU, E.; MYLOPOULOS, J. Non-functional requirements in Software Engineering. **Kluwer Academic Publishers**, 1999.

[CYS 01] CYSNEIROS, L. M., Yu, E. Non-Functional Requirements Elicitation. University of Toronto, **s.n.t.**, 2001.

[CYS 97] CYSNEIROS, L. M. **Requisitos Não Funcionais: Da Elicitação ao Modelo Conceitual.** Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, 1997. 116p. Tese (Doutorado) - Departamento de Computação.

[DAR 04] DARPA Information Exploitation Office, Command Post of the Future. Disponível em: http://dtsn.darpa.mil/ixo/programdetail.asp?progid=18, último acesso: Março/2004.

[DEY 00] DEY, A.K., ABOWD, G.D. TOWARDS A BETTER UNDERSTANDING OF CONTEXT AND CONTEXT AWARENESS **In CHI 2000 Workshop on the What, Who, Where, When, and How of Context Awareness**, April 2000.

[DEY 01] DEY, A.K. Understanding and Using Context. **Personal and Ubiquitous Computing,** p.4-7, 2001.

[FAR 00] FARINES, J. M., FRAGA, J. S., OLIVEIRA, R. S. **Sistemas de Tempo Real**. 12ª Escola de Computação, IME-USP, São Paulo-SP, 24 a 28 de julho de 2000.

[FIN 96] FINKELSTEIN, A.; DOWELL, J. A. A comedy of errors: The London Ambulance Service Case Study. **International Workshop on Software Specification an Design**, **IEEE Computer Society Press**, vol. 8, p.2-4, 1996.

[FLE 02] FLECK, M., FRID, M., KINDBERG, T., O'BRIEN-STRAIN, E., RAJANI, R., SPASOJEVIC, M. From informing to remembering: Ubiquitous systems in interactive museums. **IEEE Pervasive Computing**, p.13-21, 2002.

[GÖR 89] GÖRKE, W. Fehlertolerante Rechensysteme. **s.n.t**, 1989.

[GRA 96] GRAHAM, M.; SMITH, D. Managing Information Technology Projects, **s.n.t**,, 1996.

[GTI 04] Site do Georgia Tech Broadband Institute. Disponível em: http://www.broadband.gatech.edu/, último acesso jul/2004

[HES 02] HESS, C.K., ROMAN, M., CAMPBELL, R.H., Building Applications for Ubiquitous Computing Environments. **Proceedings of the Pervasive Computing – First International Conference**, p. 16-29, Agosto/2002.

[HON 01] HONG, J.I. Context Fabric: Infrastructure Support for Context-Aware Systems. **Phd Thesis**, Universidade da Califórnia – Berkeley, 2001.

[HON 02] HONG, J., I.; LANDAY, J., A. *An Infraestructure Approach to Context-Aware Computing*, University of California at Berkeley, **s.n.t.**, 2002.

[HOP 78] HOPKINS, A. L; SMITH, T. B.; LALA, J. H. FTMP - a highly realible fault-tolerant multiprocessor for aircraft. **Procedings of the IEEE**, New York, p. 1221-1239, Outubro, 1978.

[HÜR 99] HÜRST, W., MÜLLER, R. A Synchronization Model for Recorded Presentations and its Relevance for Information Retrieval. **Proceedings of ACM Multimedia**, 1999.

[ICS 04] Site do **IEEE Computer Society. Di**sponível em: http://www.computer.org/computer/homepage/0403/invisible/figs.ht**, ultimo acesso jul/2004**

[ISO9126] ISO9126: Information Technology - Software Product Evaluation - Software Quality Characteristics and Metrics, **International Organization for Standardization**.

[JIA 02] JIANG, X., LANDAY, J. A. Modeling Privacy Control in Context-aware Systems Using Decentralized Information Spaces. **IEEE Pervasive**

Computing, v.1, n.3, 2002.

[JIA 04] JIANG, X., CHEN, N.Y., HONG, J., WANG K., TAKAYAMA, L., LANDAY, J.A. Siren: Context-aware Computing for Firefighting. **Pervasive' 2004**, 2004.

[JOH 02] JOHANSON, B., Fox, A., WINOGRAD, T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. **Pervasive Computing Magazine Special Issue on Systems**, 2002.

[KAT 78] KATZMAN, J. A. A fault-tolerant computing system. In: Hawaii International Conference of System Sciences, 1978, **Anais**. p. 85-102, 1978.

[KEL 90] KELLER, S. E. et al. Specifying Software Quality Requirements with Metrics. In: Tutorial System and Software Requirements Engineering **IEEE Computer Society Press**, p.145-163, 1990.

[KID 99] KIDD, C. D., ORR, R., ABOWD, G. D., ATKESON, C. G., ESSA, I. A., MACINTYRE, B., MYNATT, E., STARNER, T. E., NEWSLETTER, W. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. **Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild**, 1999.

[KIN 01] KINDBERG, T., BARTON, J. A Web-based Nomadic Computing System. **Computer Networks: The International Journal of Computer and**

Telecommunications Networking, v.35, nº 4, p. 443-456, Março/2001.

[KNI 02] KNIGHT, J. C. Safety Critical Systems: Challenges and Directions. **ACM**, Florida, 2002, p. 547 - 550 , 2002.

[KUD 04] KUDO, T. N. **Computação Ciente de Contexto aplicada ao Monitoramento de Aplicações Críticas em Ambientes Físicos.** Dissertação de mestrado, UFSCar, São Carlos-SP, Maio 2004.

[LAE 03] LAERHOVEN, K. V. **Adaptive Multi-Sensor Fusion for Awareness in Dynamic Environments**, Lancaster University 2003.
DISPONÍVEL EM: http://www.comp.lancs.ac.uk/~kristof/research/papers/phd/2003/

[LAM 04] LAMPE, M., STRASSNER, M., FLEISCH, E. A Ubiquitous Computing Environment for Aircraft Maintenance. **ACM - SAC'04**, p. 14-17, Nicosia, Cyprus, Março/2004.

[LEV 95] LEVESON, N. G. Safeware Systems Safety and Computers. **Addison Wesley Publishing Company.** United States, 1995.

[LYY 02] LYYTINEN, K., YOO, Y. Issues and Challenges in Ubiquitous Computing. **Communications of the ACM**, Vol. 45, No. 12, pp. 62-65, December 2002.

[MÉI 02] MÉIER, R.; CAHILL, V. Taxonomy of Distributed Event-Based Programming Systems. **ICDCS/DEBS**. Vienna, 2002.

[MIL 80] MILNER, R. A Calculus of Comunicating Systems, Lecture Notes in Computer Science, vol 92, 1980, Ed. Springer-Verlag, 1980

Computer Science, vol 92, 1980, Ed. Springer-Verlag, 1980.

[MOS 85] MOSTOW, J. Towards Better Models of Design Process. **AI Magazine**, Vol. 6 No. Janeiro 1989, p. 44-57, 1985.

[MOZ 98] MOZER, M. C. The Neural Network House: An Environment that Adapts to its Inhabitants. **In American Association for Artifcial Intelligence Spring Symposium on Intelligent Environments**, p.110-114, 1998.

[MUK 99] MUKHOPADHYAY, S., SMITH, B. Passive Capture and Structuring of Lectures. **Proceedings of ACM Multimedia**, 1999.

[MYL 92] MYLOPOULOS, J.; CHUNG, L.; NIXON, B. A. Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. **IEEE Transactions on Software Engineering,** vol. 18, n. 6, p. 483 - 497, 1992.

[OMG 00] OBJECT MANAGEMENT GROUP. CORBAservices: Common Object Services Specification – Notification Service Spec, **s.n.t**,, 2000.

[PRA 96] PRADHAN, D. K., Fault-Tolerant System Design. Prentice Hall, New Jersey, **s.n.t**., 1996.

[RIC 01] RICHTER, H., at el. Integrating Metting Capture within a Collaborative Team Environment. **Proceedings of the International Conference on Ubiquitous Computing (UbiComp-2001)***,* Atlanta, GA, 2001.

[ROM 00] ROMÁN, M., HESS, C.K., CAMPBELL, R., H. Gaia: Enabling Active Space*.,* **Science, University of Illinois at Urbana Champaign**, pp. 1-6,

2000.

[ROM 02] ROMÁN, M., HESS, C., CERQUEIRA, R., RAGANATHAN, A., CAMPBELL, R.H., NAHRSTEDT, K. A middleware infrastructure for Active Spaces. **IEEE Pervasive Computer**, v.1, n.4, p. 74-83, 2002.

[ROM 85] ROMAN, G. A Taxonomy of Current Issues in Requirements Engineering, **IEEE Computer,** Vol. 18, No. 4 Abril 1985, p. 14-23, 1985.

[SAL 99] SALBER, D., DEY, A.K., ABOWD, G.D. The Context Toolkit: Aidind the Development of Context-Enabled Applications. **CHI'99, ACM Press**, p.434-441, Pittsburgh, PA, Maio/1999.

[SCH 93] SCHILIT, B. N., ADAMS, N., GOLD, R., TSO, M., and WANT, R. The ParcTab mobile computing system. **Proceedings of the Workshop on Workstation Operating Systems**, p.34-39, 1993.

[SCH 94] SCHILIT, B. N., THEIMER, M. M. Disseminating active map information to mobile hosts. **IEEE Network**, p.22-32. Setembro-Outubro/1994.

[SCH 95] SCHILIT, B., ROY, W. The Xerox PARCTAB. Disponível em: http://sandbox.parc.xerox.com/parctab/, 1995.

[SCH 98] SCHILIT, B.N., GOLOVCHINSKY, G., PRICE, M.N. Beyond Paper: Supporting Active Reading with Free form Digital Ink Annotations. **In Proceedings of the Conference on Human Factors in Computing Systems**, p.249-256, 1998.

[SOM 98] SOMMERVILLE, I. e KOTONYA, G. Requirements Engineering, **s.n.t.**,, 1998.

[STA 02] STAJANO, F. Security for Ubiquitous Computing. **Hardcover**, pp. 267, February, 2002

[TUR 96] TURINE, M. A. S.; MASIERO, P. C. **Especificação de Requisitos: Uma introdução**. Relatório Técnico. ICMC/USP, 1996.

[VIL 02] VILLATE, Y., ILLARRAMENDI, A, PITOURA, E. KEEP YOUR DATA SAFE AND AVALABLE WHILE ROAMING **Mobile Networks and Applications**, pp. 315-329, 2002.

[W3C 00] W3C (World Wide Web Consortium) Note on Simple Object Access Protocol (SOAP) 1.1. Disponível em: http://www.w3.org/TR/SOAP/, Ago 2000.

[WAN 92] WANT, R., HOPPER, A., FALCÃO, V., GIBBONS, J. The Active Badge Location System. **ACM Transactions on Information Systems**, p.91-102, Janeiro/1992.

[WAN 92] WANT, R., HOPPER, A., FALCÃO, V., GIBBSONS, J. The Active Badge Location System. **ACM Transactions on Information Systems**, p.91-102, Janeiro/1992.

[WEI 91] WEISER, M. The computer for the 21st century, **Scientific American**. vol. 265, no. 3, pp. 94-104, September 1991.

[WEI 94] WEISER, M. The world is not a desktop. **ACM Interactions**, vol. 1 no. 1, pp. 7-8, January 1994.

[WEI 96] WEISER, M., BROWN, J. The Coming Age of Calm Technology. **Xerox PARC**, Outubro/1996.

[WEL 01] WELLS, A., T. Comercial Aviation Safety. McGraw-Hill, 3rd edition, 2001.

[WEN 78] WENSLEY, J. H. et al. SIFT: design and analysis of fault-tolerant computer for aircraft control. **Proceedings of the IEEE**, New York, p. 1240-1254, Outubro, 1978.

[WEN 83] WENSLEY, J. H. Industrial-control system does things in threes for safety. **Electronics**, New York, p. 98-102. Janeiro, 1983.

[WIT 98] WITBROCK, M.J., HAUPTMANN, A.G. Speech Recognition for a Digital Video Library. **Journal of the American Society of Information Science**, 49(7), p.619-632, 1998.