UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

MARCOS PAULINO RORIZ JUNIOR

C3S: Uma Plataforma de *Middleware* de Compartilhamento de Conteúdo para Espaços Inteligentes

Goiânia
2013

MARCOS PAULINO RORIZ JUNIOR

C3S: Uma Plataforma de *Middleware* de Compartilhamento de Conteúdo para Espaços Inteligentes

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Mestrado em Ciência da Computação.

Área de concentração: Redes e Sistemas Distribuídos.

Orientador: Prof. Fábio Moreira Costa

Co-Orientador: Prof. Ricardo Couto Antunes da Rocha

Goiânia
2013

MARCOS PAULINO RORIZ JUNIOR

C3S: Uma Plataforma de *Middleware* de Compartilhamento de Conteúdo para Espaços Inteligentes

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Mestrado em Ciência da Computação, aprovada em 17 de Maio de 2013, pela Banca Examinadora constituída pelos professores:

Prof. Fábio Moreira Costa
Instituto de Informática – UFG
Presidente da Banca

Prof. Ricardo Couto Antunes da Rocha
Instituto de Informática – UFG

Prof. Renato de Freitas Bulcão Neto
Instituto de Informática – UFG

Prof. Francisco José da Silva e Silva
Departamento de Informática – UFMA

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

Marcos Paulino Roriz Junior

Graduou-se em Ciências da Computação pela Universidade Federal de Goiás. Durante a graduação, foi bolsista de pesquisa da RNP no projeto GingaCDN que visa especificar o padrão do *middleware* de TV digital brasileiro (Ginga). Nesse projeto, foram arquitetados componentes que realizam a intercomunicação entre as diferentes linguagens de programação suportadas pelo Ginga (Java, NCL e Lua). Também participou do Google *Summer of Code* pelo projeto GNU, no subprojeto GNU Classpath. Nesse projeto, estendeu e refatorou o *Escher*, uma implementação de X11 escrita em Java utilizada em ambientes embarcados. Durante o mestrado, foi bolsista da CAPES e participou do projeto Comunidades Virtuais de Prática em Salas de Aula Interativas, financiado pela Dell Computadores do Brasil, atuando no desenvolvimento do middleware de computação ubíqua UCLE para salas de aulas interativas.

À minha querida e amada família: minha vó Rosalha Maria Borges, minha mãe Luciane Franco Borges, minha tia Cristiane Franco Borges e meu irmão Diogo Borges Nery, que são o tripé da minha formação, educação e integridade.

Agradecimentos

Antes de tudo, gostaria de agradecer novamente minha família: minha vó Rosalha Borges, minha mãe Luciane Borges, minha tia Cristiane Borges e meu irmão Diogo Borges, que sempre estiveram me apoiando na vida, principalmente nos momentos difíceis como nos que tive de ausentar da presença deles para realiza este trabalho.

Ao meu orientador Prof. Dr. Fábio Moreira Costa, ao qual tenho uma imensa gratidão e admiração, por sua excelente orientação, disponibilidade, paciência e por ter me dado a honra de trabalhar ao seu lado. Nossa parceria iniciou-se com projetos de pesquisa durante a graduação e tenho certeza que irá se estender após o término do mestrado.

Ao meu coorientador Prof. Dr. Ricardo Couto Antunes da Rocha, ao qual também tenho uma imensa gratidão e admiração, por sua excelente orientação e paciência durante esse trabalho. Suas lições foram essenciais para me tornar um melhor pesquisador.

Ao Prof. Dr. Markus Endler pela paciência de me receber durante essa transição do mestrado para o doutorado e por me aceitar como aluno no doutorado na PUC-Rio.

Aos demais professores do INF/UFG, em especial ao Prof. Dr. Auri Marcelo Rizzo Vincenzi, Profa. Dra. Diane Castonguay, Prof. Dr. Humberto José Longo, Prof. Dr. Iwens Gervasio Sene Junior, Prof. Dr. Leandro Luís Galdino de Oliveira, Prof. Dr. Renato de Freitas Bulcão Neto, Profa. Dra. Telma Woerle de Lima Soares e Prof. Dr. Vagner José do Sacramento Rodrigues, pela imensa paciência e dicas durante a graduação e mestrado.

Aos amigos do INF/UFG: Adalberto Júnior, Guilherme Kenedy, Gustavo Nunes, Ivahy Santos, João Guilherme Araújo, João Neto, Júnio César, Leandro Alexandre, Marco Aurélio Lino Massarani, Mariana Soller e Roberto Vito pelo imenso apoio durante o curso e pelos momentos divertidos que passamos juntos! Nunca esquecerei vocês!

Aos novos amigos do DI/PUC-Rio: Álan Livio, Daniel Pires, Davi Rocha, Eduardo Araújo, Gustavo Baptista, Katia Cánepa Veja, Igor Vasconcelos, Lincoln David e Rafael Vasconcelos.

À equipe da secretaria do INF-UFG: Edir de Jesus Borges Pinto, Marlliny Oliveira Sales e Mirian Castro Portilho Dias. Um agradecimento especial a Mirian pela enorme paciência e disponibilidade durante todo o mestrado. Muito obrigado por me ajudar com todas as dúvidas e procedimentos burocráticos!

Por fim, à CAPES e CNPQ pelo auxílio financeiro.

"It's always good to take an orthogonal view of something. It develops ideas."

Ken Thompson,
"Unix and Beyond: An Interview with Ken Thompson", IEEE Computer, Volume 32, 5ª Edição, 1999.

Resumo

Roriz Junior, Marcos Paulino. **C3S: Uma Plataforma de *Middleware* de Compartilhamento de Conteúdo para Espaços Inteligentes**. Goiânia, 2013. 93p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

De acordo com Mark Weiser, a computação ubíqua se concentra na integração de maneira despercebida e sem rupturas (*seamlessy*) de tarefas da computação no cotidiano das pessoas. Por causa das atuais limitações tecnológicas, a realização dessa integração segue um ou mais aspectos da computação ubíqua, por exemplo, de mobilidade ou de contexto, que são baseados em serviços que integram o usuário em um ambiente ubíquo delimitado (como espaços inteligentes). Neste trabalho exploramos uma abordagem diferente, em que os serviços não são utilizados para integrar um usuário individual ao ambiente, mas são utilizados para integrar os usuários presentes no ambiente uns com os outros. Uma maneira de realizar esse aspecto é usando o compartilhamento de conteúdo, dados de primeira classe da aplicação que servem como meio de integrar os usuários. No entanto, devido à complexidade do ambiente de computação ubíqua e à falta de plataformas de *middleware*, aplicações que seguem esta abordagem são repetidamente construídas a partir "do zero", usando técnicas não convencionais. Com o objetivo de fornecer uma infraestrutura para o desenvolvimento deste tipo de aplicação, propomos o *Content Sharing for Smart Spaces* (C3S), um *middleware* que oferece um modelo de programação de alto nível, usando primitivas baseadas em um conjunto de semânticas de compartilhamento de conteúdo e em conceitos de aplicações ubíquas. As primitivas expressam um conjunto de comportamentos, tais como mover, clonar, e espelhar, que servem como blocos de construção para os desenvolvedores implementarem funcionalidades de compartilhamento, enquanto que os conceitos de ubiquidade permitem a manipulação de usuários, grupos e aplicações ubíquas. A proposta foi validada por meio de dois estudos de caso que exploram esses recursos. Os resultados permitiram concluir que o *middleware* fornece uma maneira mais fácil de desenvolver aplicativos baseados em compartilhamento em comparação com trabalhos semelhantes encontrados na literatura.

Palavras–chave

C3S, Middleware para Compartilhamento, Middleware Pervasivo, Espaço Inteligente, Computação Ubíqua, Sistemas Distribuídos

Abstract

Roriz Junior, Marcos Paulino. **C3S: A Content Sharing Middleware for Smart Spaces**. Goiânia, 2013. 93p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

According to Mark Weiser, ubiquitous computing focuses on seamlessly integrating computing tasks into people's daily lives. Because of current technology limitations, the realization of ubiquitous computing observes a limited set of aspects of ubiquitous computing, such as, mobility and context, which are based on services that integrate the users with the resources that are present on a delimited ubiquitous environment (such as in smart spaces). Instead, we explored a different approach, in which services are used not to integrate an individual user with the environment, but to integrate the users present in the environment with one another. One way to realize this aspect is by using content sharing, first-class application dat, that serve as integration medium between users. However, due to the environment complexity and lack of middleware platforms, applications that follow this approach are repeatedly built from scratch using raw techniques. Aiming to provide an infrastructure for the development of this kind of applications, we propose Content Sharing for Smart Spaces (C3S), a middleware that offers a high-level programming model using primitives that are based on a set of content sharing semantics and ubiquitous application concepts. The primitives express a small set of behaviors, such as move, clone, and mirror, which serve as building blocks for developers to implement sharing and content ubiquity features, while the ubiquitous concepts supported by the middleware allow the manipulation of users, groups and ubiquitous applications. We validated our proposal using two different case studies that allowed us to explore these features. Our results show that our middleware provides an easier way to develop sharing-based applications compared to related work found in the literature.

Keywords

C3S, Content Sharing Middleware, Sharing Middleware, Pervasive Middleware, Smart Spaces, Ubiquitous Computing, Distributed Systems

Sumário

Lista de Figuras

Lista de Definições

Lista de Tabelas

CAPÍTULO 1

Introdução

A popularização de dispositivos computacionais móveis [44, 57], tais como *tablets*, *smartphones* e *ultrabooks*, combinados com a infraestrutura de espaços inteligentes (*smart spaces*) [13, 42], facilita o compartilhamento transparente de conteúdo entre os usuários presentes no ambiente. Esse tipo de ubiquidade eleva os dados do usuário à categoria de entidades de primeira classe, denominadas conteúdo ubíquo, que podem ser transportadas naturalmente e sem interrupções pela infraestrutura do espaço inteligente. Essa visão de computação ubíqua serve de base para construção de aplicações ubíquas orientadas a compartilhamento, tais como aplicações que seguem o usuário (*follow-me*), aplicações colaborativas e telas públicas interativas. A definição de abstrações que habilitem esta modalidade de computação ubíqua e facilitem a construção dessas aplicações é a principal motivação desta dissertação.

Esta discussão está estrutura da seguinte forma: a Seção 1.1 descreve sucintamente o cenário de computação atual e como desenvolvedores podem combinar dispositivos atuais com espaços inteligentes para construir aplicações ubíquas centradas em compartilhamento de conteúdo. Na Seção 1.2 são descritos os principais problemas e limitações envolvidos no desenvolvimento de aplicações orientadas a conteúdo ubíquo, que leva a uma das contribuições deste trabalho: uma plataforma de *middleware* que fornece abstrações orientadas a conteúdo ubíquo para o suporte a esse aspecto da computação ubíqua. Na Seção 1.3 são apresentados os objetivos gerais e específicos do trabalho. A Seção 1.4 descreve a metodologia utilizada. Na Seção 1.5 são destacadas as contribuições do trabalho e, por fim, a Seção 1.6 descreve a estrutura da dissertação.

1.1 Motivação

Nos últimos anos observou-se um aumento expressivo na quantidade e uso de dispositivos computacionais móveis cada vez mais compactos e intuitivos, tais como *tablets*, *smartphones* e *ultrabooks* com *touchscreen* [44, 57]. A inserção desses dispositivos em espaços inteligentes, que podem ser definidos como ambientes físicos delimitados, tais como salas e prédios, permeados com uma infraestrutura de serviços computacionais

[13, 42], possibilita construir cenários que eram anteriormente apenas vislumbrados pela proposta de computação ubíqua de Mark Weiser [46, 60]. Por exemplo, usuários podem explorar a crescente conexão desses dispositivos com seu ambiente de uso, para realizar troca e sincronização de dados com outros usuários presentes de forma transparente e sem interrupções (*seamlessly*).

A tendência é que esses dispositivos estejam cada vez mais conectados e integrados aos espaços inteligentes [56], permitindo explorar e habilitar diversos aspectos da computação ubíqua nesses ambientes. A ubiquidade no nível de conteúdo vislumbra a possibilidade de permitir que conteúdos movam-se livremente entres esses dispositivos utilizando a infraestrutura do espaço inteligente. Esse grau de ubiquidade eleva os dados do usuário à categoria de entidades de primeira classe denominadas conteúdo ubíquo, que são transportadas naturalmente e sem interrupções externas entre os dispositivos dos usuários presentes no ambiente. Outra característica dessa visão encontra-se na possibilidade de que múltiplas cópias desses conteúdos coexistam, de maneira sincronizada em vários dispositivo associados aos usuários do espaço inteligente. O aspecto de conteúdo da computação ubíqua explora essa combinação de dispositivos com o espaço inteligente como forma de possibilitar a integração entre os usuários, como se vê no cenário da Figura 1.1, onde o conteúdo ubíquo funciona como intermediador para integrar os usuários do ambiente de computação ubíqua. Na ilustração verifica que as imagens estão sendo sincronizadas entre os usuários do ambiente, por exemplo, a imagem do coqueiro está visível para diversos usuários tanto em uma mesa interativa, quanto num *tablet*. A sincronização dessa imagem com vários usuários do ambiente, serve de meio comum para comunicação entre os usuários desse ambiente. O compartilhamento ubíquo desses conteúdos é a base para a construção de aplicações ubíquas que realizem a integração entre os usuários utilizando o ambiente, tais como aplicações colaborativas e telas públicas.

Figura 1.1: *Aspecto de conteúdo ubíquo da computação ubíqua visa integrar usuários de um espaço inteligente [56].*

Um cenário interessante dessa combinação seria uma aplicação para apresentação de *slides* em salas de aula interativas (um tipo de espaço inteligente). A infraestrutura computacional da sala de aula é composta por *tablets* PC, impressoras, lousa digital e servidores que orquestram a integração desses elementos com serviços de computação ubíqua. Por exemplo, a apresentação de *slide* do professor seria sincronizada nos *tablets* de cada aluno da sala de aula. As modificações na apresentação do professor são refletidas nos *tablets* de cada aluno, ou seja, quando o professor muda a página da apresentação o mesmo ocorre nos *tablets* dos alunos. As anotações criadas pelo professor nos *slides* também são também sincronizadas, ou seja, quando o professor adiciona uma nota em um determinado *slide* ela é sincronizada e aparece nas cópias dos alunos. Os alunos também podem criar anotações, que podem ser compartilhadas entre si e com o professor. Por exemplo, um aluno pode compartilhar uma dúvida com um colega, como ilustrado de forma genérica na Figura 1.2. Esse cenário é utilizado em diversas aplicações encontradas na literatura, como *iPH* [36], *Classroom Presenter* [4] e *Ubiquitous Presenter* [61].

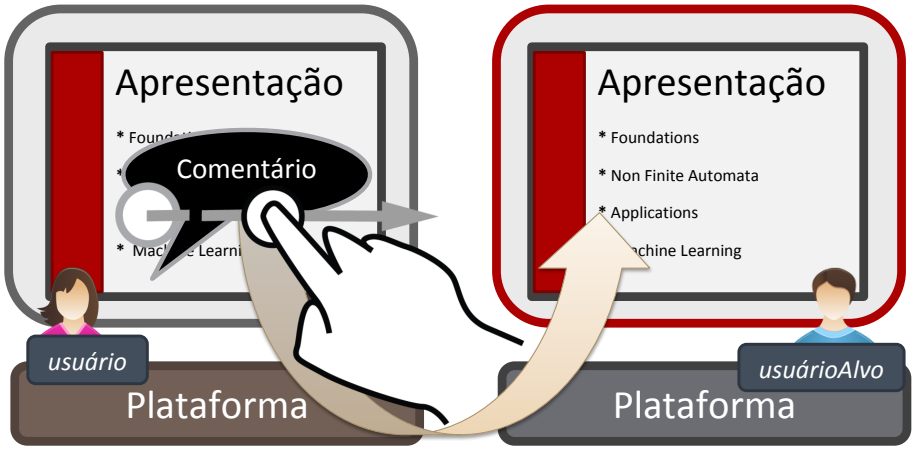

Figura 1.2: *Compartilhamento de um comentário com o usuário* `usuárioAlvo` *utilizando um gesto de mover.*

1.2 Justificativa

Diversas aplicações baseadas em compartilhamento de conteúdo são apresentadas na literatura relacionada [4, 25, 36, 52], as quais foram desenvolvidas com intuito de explorar o aspecto de conteúdo ubíquo em espaços inteligentes. Entretanto, devido à falta de plataformas de *middleware* que sustentem a modalidade de conteúdo da computação ubíqua, essas aplicações geralmente são desenvolvidas a partir "do zero" usando técnicas convencionais de sistemas distribuídos, tais como aquelas baseadas em RPC e objetos distribuídos [10, 11]. Essas plataformas não foram projetadas para esse domínio

de aplicações e expõem o programador a problemas não triviais do compartilhamento ubíquo em espaços inteligentes. Por exemplo, cada semântica de compartilhamento ou cada forma de compartilharmento, deve ser codificada manualmente pelo desenvolvedor. Durante uma operação de compartilhamento podem ocorrer problemas de inconsistência na transmissão ou devido ao acesso concorrente ao conteúdo. Por exemplo, após a conclusão da movimentação do conteúdo, as referências ligadas a ele ficarão quebradas, como retratado na Figura 1.3, que mostra a movimentação de um item de conteúdo do tipo comentário referenciado por um conteúdo *slide*. Nesse caso, quando o conteúdo *slide* tentar acessar o comentário transferido ocorrerá um erro devido a referência não o localizar.

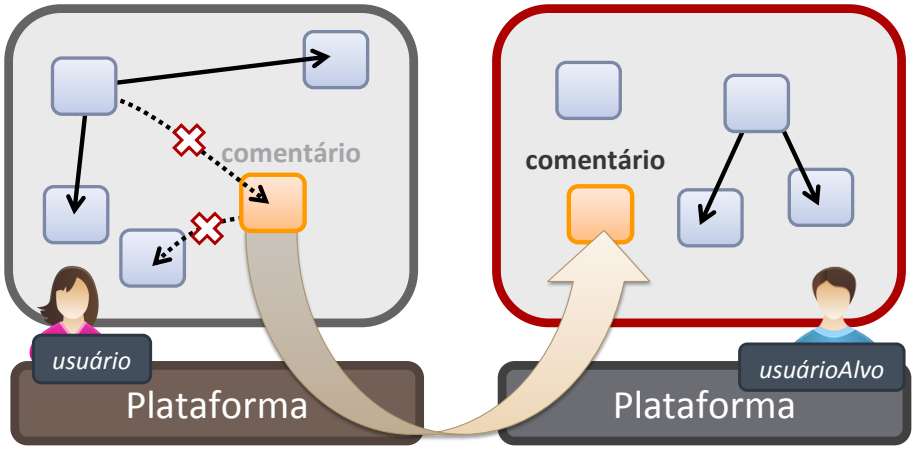

Figura 1.3: *Problema de integridade referencial ao mover um conteúdo para outro usuário.*

O tratamento desses problemas por parte do programador é oneroso e desvia o foco das questões específicas da aplicação. Visando prover uma infraestrutura para facilitar o desenvolvimento de aplicações orientadas ao aspecto de conteúdo da computação ubíqua em espaços inteligentes, propomos o ***C**ontent **S**haring for **S**mart **S**paces* (**C3S**), um *middleware* que fornece um modelo, baseado em primitivas, para manipulação e compartilhamento de conteúdo em ambientes de computação ubíqua. O modelo é composto por um conjunto de serviços que sustentam as operações de manipulação de conteúdo ubíquo no espaço inteligente, incluindo sua sincronização e movimentação transparente e sem interrupções entre os usuários presentes no ambiente. As abstrações de manipulação de conteúdo são fornecidas na forma de primitivas de compartilhamento que definem funções para mover, clonar e espelhar itens de conteúdo no ambiente. Por exemplo, no cenário descrito anteriormente, para compartilhar o conteúdo (*comentário*), o programador executaria a chamada da primitiva de mover, *mover*(*comentário, usuárioAlvo*) do C3S, que iria encarregar-se de mover o conteúdo e tratar automaticamente sua consistência e integridade referencial, durante e após a transferência, conforme ilustrado na Figura 1.4.

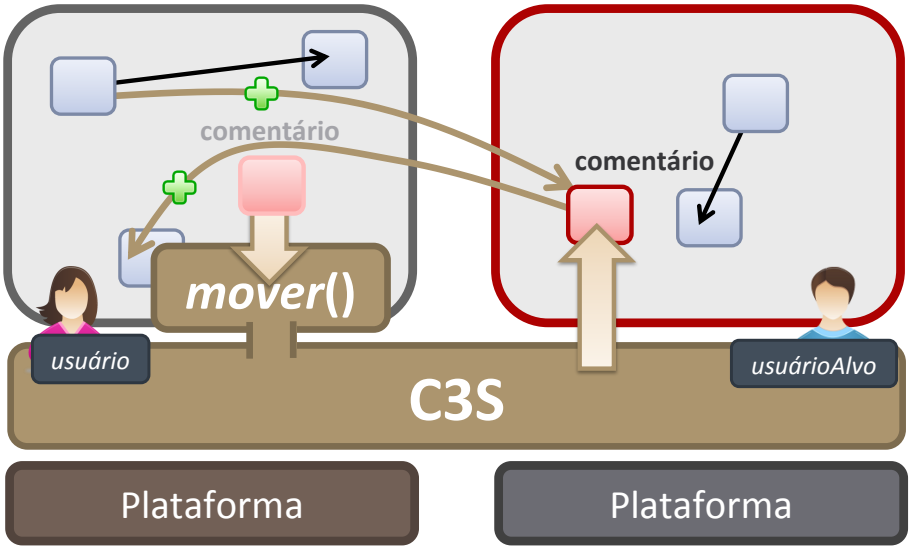

Figura 1.4: *Manipulação de um conteúdo ubíquo (comentário) usando a primitiva de mover do middleware C3S.*

1.3 Objetivos

Esta dissertação tem como objetivo investigar e propor soluções orientadas à manipulação de conteúdo ubíquos em ambientes de computação ubíqua para facilitar o desenvolvimento de aplicações orientadas a esse aspecto (como as aplicações descritas nas seções anteriores). A solução proposta visa explorar e fornecer serviços e abstrações que permitam trocar e compartilhar conteúdo transparentemente e sem rupturas entre os participantes presentes no ambiente. A manipulação de conteúdo utiliza um conjunto de primitivas para dar suporte ao compartilhamento em espaços inteligentes. O objetivo geral se desdobra nos seguintes objetivos específicos:

- Definir a estrutura e comportamento do conteúdo ubíquo, que deve ser flexível e independente de um tipo específico de aplicação;
- Investigar e implementar diferentes semânticas de compartilhamento de conteúdo ubíquo na forma de primitivas do *middleware* C3S, com o intuito de fornecer mecanismos de alto-nível para a manipulação de conteúdo em aplicações baseadas em compartilhamento em ambientes ubíquos;
- Definir um conjunto de serviços que compõem o modelo do *middleware* C3S para permitir particionar e gerenciar diversas aplicações no ambiente; e
- Implementar o *middleware* C3S e demonstrar seu uso por meio da construção de uma aplicação baseada em compartilhamento de conteúdo ubíquo.

1.4 Metodologia

Inicialmente o trabalho explorou, na literatura e por meio de protótipos, aplicações ubíquas orientadas a conteúdo com o intuito de identificar características comuns em seu funcionamento. Este estudo permitiu identificar os requisitos e comportamentos do compartilhamento de conteúdo em ambientes com características de computação ubíqua, os quais foram então capturados em um modelo de aplicações baseadas em conteúdo ubíquo. Esse modelo organiza os componentes das aplicações de maneira a permitir a movimentação e troca transparente e sem interrupções do conteúdo no ambiente.

O centro desse modelo de computação ubíqua é o conceito de conteúdo ubíquo, sendo necessário encontrar uma definição genérica deste conceito para que o *middleware* possa manipulá-lo de maneira opaca, ou seja, sem conhecimento de sua estrutura interna. Para isto, exploramos diversas definições pré-existentes de estruturas de compartilhamento utilizadas por aplicações nesse tipo de ambiente [7, 23, 27, 28, 34, 58, 63]. Entretanto, as definições encontradas eram construídas estaticamente em torno de uma única maneira de compartilhar conteúdo, tornando-se necessária a definição de uma unidade de compartilhamento ubíqua mais flexível e mais adequada ambientes ubíquos de propósito geral, aqui chamada de conteúdo ubíquo. As características comuns das aplicações analisadas também permitiram identificar diferentes modos de manipulação do conteúdo ubíquo, os quais foram capturados em três primitivas de compartilhamento: mover, clonar e espelhar. Esse conjunto de primitivas de compartilhamento forma a base para a manipulação de conteúdo ubíquo no modelo.

Os conceitos definidos anteriormente desdobraram-se na arquitetura do C3S, que materializa o aspecto de conteúdo da computação ubíqua. Foram especificados conceitos que particionam e organizam os componentes das aplicações no ambiente de computação ubíqua, além disso, também são fornecidas funções para criar e manipular conteúdo ubíquo com base nas semânticas de compartilhamento das primitivas. Os componentes da arquitetura são independentes da tecnologia de comunicação utilizada, podendo ser implementados em diferentes sistemas de *middleware* de comunicação. Construímos uma implementação do C3S utilizando a linguagem de programação Java e o *middleware* de comunicação Java RMI.

Por fim, para validar o *middleware* C3S, foram desenvolvidas duas aplicações e uma comparação com outros trabalhos na literatura. As aplicações desenvolvidas foram programadas na linguagem Java e utilizam o arcabouço MT4j [31], que permite construir aplicações que fazem uso de gestos multitoque para interação com o usuário, permitindo tornar ainda mais natural a movimentação de conteúdo entre as aplicações. A comparação deste trabalho com outros foi baseada nas características e requisitos de aplicações baseadas no aspecto de conteúdo da computação ubíquo. A análise compara cada um dos

trabalhos relacionados com o C3S, conforme essas características e requisitos e conclui com um posicionamento do C3S em relação aos trabalhos analisados.

1.5 Contribuições

As contribuições deste trabalho são derivadas dos objetivos apresentados na Seção 1.3 e podem ser resumidas nos seguintes pontos:

- Um conjunto de serviços que formam um modelo de computação ubíqua baseado no conceito de conteúdo ubíquo;
- Sistematização do conceito de conteúdo ubíquo, que consiste em elementos de dados de primeira-classe, que podem ser compartilhados de diferentes formas em diferentes dispositivos de usuários no ambiente ubíquo. A manipulação e transferência do conteúdo são implementadas em primitivas fornecidas pela infraestrutura do ambiente;
- Um conjunto de primitivas de compartilhamento de conteúdo ubíquo, que expressam diferentes semânticas de manipulação de conteúdo no ambiente ubíquo;
- O modelo arquitetural do *middleware* C3S, incluindo a descrição, interação e disposição dos componentes que permitem concretizar os conceitos apresentados; e
- Um protótipo do *middleware* C3S, utilizando a linguagem de programação Java e o *middleware* de objetos distribuídos Java RMI.

1.6 Estrutura da Dissertação

O restante desta dissertação está organizado em seis capítulos que adotam o seguinte roteiro. O Capítulo 2 apresenta a fundamentação teórica que sustenta este trabalho e permite compreender a motivação e os problemas envolvidos na construção de aplicações e infraestruturas orientadas a conteúdo em ambiente de computação ubíqua.

No Capítulo 3 são definidos os principais conceitos deste trabalho, incluindo conteúdo ubíquo e aplicação ubíqua. O conceito de conteúdo ubíquo é definido de forma flexível, permitindo que ele seja compartilhado de diferentes formas no ambiente ubíquo. As diferentes formas de compartilhamento são materializadas em primitivas, funções que manipulam e transportam o conteúdo entre os dispositivos presente no ambiente. Ainda nesse capítulo, são descritos os conceitos que constituem o modelo de computação ubíqua baseado em conteúdo do C3S. Esses conceitos são definidos com o intuito de organizar a estrutura e o comportamento fundamentais dos elementos das aplicações nesse ambiente para permitir a navegação e troca (compartilhamento) transparente e sem rupturas do

conteúdo no ambiente. O Capítulo 4 desdobra os conceitos apresentados na arquitetura do *middleware* C3S e apresenta sua implementação.

Para avaliar o trabalho, o Capítulo 5 utiliza um estudo de caso composto de duas aplicações cuja camada de comunicação foi construída inteiramente sobre as primitivas do *middleware* C3S. O Capítulo 6 realiza uma comparação do trabalho com outras plataformas de *middleware* e *frameworks* que fornecem abstrações similares à modalidade de conteúdo da computação ubíqua. Por fim, o Capítulo 7 conclui esta dissertação destacando seus resultados e contribuições, além de uma discussão sobre possíveis trabalhos futuros.

CAPÍTULO 2

Computação Ubíqua Orientada a Conteúdo

A computação ubíqua visa tornar invisível a utilização da computação para o usuário por meio da sua integração com elementos físicos do dia-a-dia [60]. A complexidade de oferecer uma experiência de integração completa nos leva a delimitar a área física do ambiente e focar em ambientes delimitados, como espaços inteligentes, e em um aspecto específico da computação ubíqua [6, 15]. Na literatura, podem ser encontradas diversos sistemas orientados a um ou mais aspectos da computação ubíqua [42, 49, 55], como mobilidade e contexto. Os sistemas que implementam esses aspectos utilizam a infraestrutura como meio de imersão do usuário no ambiente físico. Diferentemente desses sistemas, o aspecto de conteúdo da computação ubíqua utiliza a infraestrutura como meio de integrar e promover a interação entre os usuários presentes no ambiente [30, 43]. Entretanto, as plataformas de *middleware* para computação ubíqua que fornecem construções orientadas ao aspecto de conteúdo ubíquo são escassas e limitadas, o que dificulta a construção de aplicações nesse tipo de ambiente. Entre as limitações podemos ressaltar a ausência de operações para manipulação de conteúdo ubíquo. Geralmente as plataformas fornecem apenas a sincronização de conteúdo, além de definirem a priori como o conteúdo é compartilhado, impossibilitando sua manipulação de outra forma. Devido a essas limitações, essas aplicações geralmente são construídas a partir "do zero", utilizando técnicas não convencionais de sistemas distribuídos, como RPC e objetos distribuídos. Essas técnicas não foram projetadas para esse domínio de aplicações e expõem o programador a problemas não triviais do compartilhamento ubíquo de conteúdo em espaços inteligentes.

O restante deste capítulo está estruturado da seguinte forma: a Seção 2.1 descreve o conceito geral de computação ubíqua e as dificuldades da integração dos dispositivos com o ambiente. A Seção 2.2 apresenta o conceito de espaço inteligente como um conjunto de serviços que orquestram a infraestrutura do ambiente de computação ubíqua para habilitar essa integração. Esses serviços são fornecidos por plataformas que destacam um ou mais aspectos da computação ubíqua. A Seção 2.3 expõe as limitações das plataformas orientadas a conteúdo e como elas afetam o desenvolvimento de aplicações baseadas em compartilhamento nesse ambiente. Por fim, a Seção 2.4 apresenta um resumo e considerações finais do capítulo.

2.1 Computação Ubíqua

A formalização da idéia de computação ubíqua teve origem na visão de Mark Weiser em 1988 [60], que vislumbrou a possibilidade de integrar e tornar a utilização da computação invisível para o usuário, fundindo-a com elementos do seu dia-a-dia. Segundo este conceito, a computação estaria permeada nos objetos do ambiente físico do usuário, ao contrário de requerer dispositivos computacionais tradicionais para interação, tais como teclado e *mouse*. Na visão de Weiser, o foco do usuário sai do dispositivo computacional que ele manipula e passa para a tarefa ou a ação a ser realizada. Na computação ubíqua, as informações utilizadas pelos usuários presentes no ambiente seriam disponibilizadas e processadas "invisivelmente" pela infraestrutura computacional embutida no ambiente. Os objetos físicos nesse local, como cadeiras, geladeira, carros e mesas, teriam capacidade computacional embutida e seriam interconectados ao ambiente e entre si [21]. Essa interconexão dos objetos com a infraestrutura computacional permite sobrepor o ambiente físico com uma camada computacional. A interação natural com os elementos físicos do ambiente permite estabelecer uma conexão com o mundo computacional e tornar a computação "invisível".

Devido ao enorme potencial da computação ubíqua, pesquisadores de várias áreas distintas da computação desenvolveram teorias e tecnologias para amadurecer e materializar essa visão [2]. Sob a perspectiva de *hardware*, pesquisadores e indústria vêm desenvolvendo dispositivos computacionais cada vez mais compactos e intuitivos, tais como *smartphones*, *tablets* e *ultrabooks*, que permitem aos usuários manterem conectividade e capacidade computacional ao se moverem pelo ambiente. No âmbito de *software*, esse problema tornou-se demasiadamente complicado e desdobrou-se em vários subproblemas, tais como mobilidade, ciência de contexto e heterogeneidade, devido às limitações desses dispositivos e à complexidade de fornecer uma integração completa do usuário com os elementos do ambiente, como vislumbrada pela visão de computação ubíqua [16, 29]. Essa divisão possibilitou tratar a computação ubíqua com base em pontos de vistas específicos, por exemplo, mobilidade, que lida com problemas de manter a conectividade dos dispositivos ao moverem-se pelo ambiente, o que geralmente é feito por meio de soluções pontuais para a transferência suave entre redes [32, 38].

Ambas as frentes de pesquisa concordam que a integração e intermediação desses dispositivos com o restante do ambiente físico requer a existência de meios de comunicação e interação com os elementos presentes [41]. Alguns trabalhos [51, 59], buscaram embutir essa infraestrutura sob a perspectiva individual de cada usuário. Nessa visão, cada indivíduo "usa" um *hardware* específico para integração com o restante do ambiente físico. O usuário desse cenário carrega consigo um dispositivo que implementa um conjunto de serviços para interagir com os outros elementos do ambiente. Na Figura

2.1, o usuário “veste” um servidor pessoal que permite interagir com outros elementos do ambiente. Ao entrar em uma sala com um quadro com capacidade computacional, o servidor realiza a mediação entre os dispositivos e permite ao *tablet* receber a pintura do quadro. Entretanto, essa visão pode tornar-se complexa na presença de outros usuários no ambiente, particularmente, devido à dificuldade de coordenar a sobreposição dos serviços dos servidores pessoais de cada usuário no ambiente.

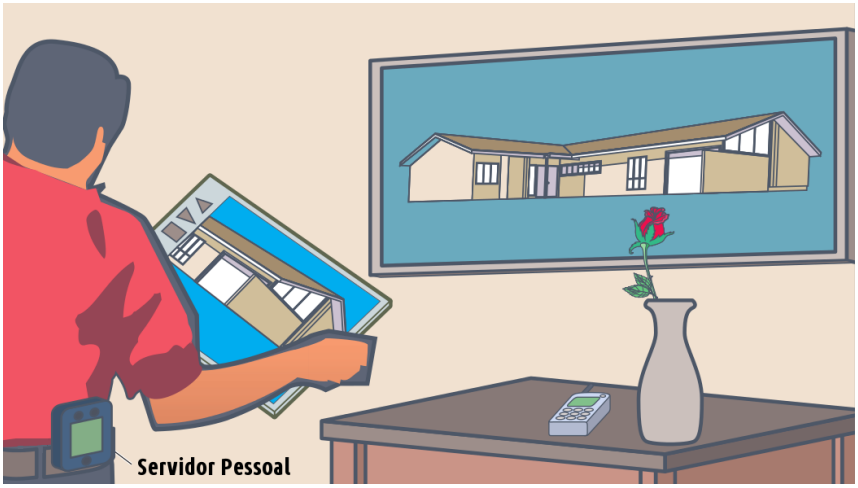

Figura 2.1: *Perspectiva de hardware como infraestrutura para computação ubíqua. [59].*

2.2 Espaço Inteligente

A diminuição do escopo do ambiente de computação ubíqua mitiga os problemas de integração, particularmente, devido à possibilidade de prever alguns comportamentos do usuário no local [41]. O espaço inteligente utiliza essa premissa para instrumentar e projetar a infraestrutura do ambiente como meio de estabelecer serviços para habilitar a computação ubíqua naquele local, como definido abaixo.

Espaço Inteligente. *É uma área física delimitada, como uma sala, prédio ou bloco, permeada com serviços computacionais que orquestram a infraestrutura existente do ambiente orientado a um ou mais aspectos da computação ubíqua [13, 42, 47, 62].*

A delimitação da área física do espaço inteligente facilita a construção de serviços de computação ubíqua sobre a infraestrutura do ambiente devido à diminuição do escopo e ao conhecimento prévio de elementos computacionais existentes. Utilizando essas premissas é possível projetar e preparar o local instalando e embutindo dispositivos computacionais como servidores, sensores e redes sem fio. Os serviços combinam e orquestram a infraestrutura do ambiente com o intuito de integrar os usuários presentes e

tornar o espaço “inteligente”. Um exemplo de ambiente comumente utilizado e reconfigurado para tornarem-se “inteligentes” são salas de aulas [48, 65], devido aos benefícios de aumentar a interação entre alunos e professores [3]. Um exemplo de aplicação nesse tipo de ambiente pode ser obtido equipando a sala de aula com quadros ou lousas interativas e construindo um serviço que sincronize a apresentação e suas anotações automaticamente entre a lousa e os dispositivos dos alunos presentes [61]. Um exemplo desse tipo de ambiente é mostrado na Figura 2.2. Nesse cenário a professora apresenta uma atividade na lousa digital aos alunos da sala de aula inteligente. Essa atividade é divida em partes, onde cada parte é representada por uma forma geométrica, como triângulo e retângulo. Cada aluno escolhe uma parte do problema e a sincroniza com seu computador. Os alunos podem trocar partes entre si com intuito de solucionar o problema de forma colaborativa.

Figura 2.2: *Exemplo de espaço inteligente, sala de aula permeada com serviços computacionais que visa integrar os dispositivos presentes, como a lousa e notebooks [54].*

2.2.1 Aspectos da Computação Ubíqua no Espaço Inteligente

Na literatura podemos encontrar vários argumentos para que os serviços ubíquos no espaço inteligente sejam fornecidos por plataformas de *middleware* [12, 47, 62]. Essas plataformas definem serviços que orquestram e combinam os elementos da infraestrutura para implementar um ou mais aspectos da computação ubíqua, tais como mobilidade, ciência de contexto, heterogeneidade dos dispositivos e conteúdo ubíquo. Esses aspectos

representam características que expressam uma ou mais visões que sustentam o conceito de computação ubíqua [15].

Devido à complexidade de fornecer uma integração completa com o ambiente, essas plataformas orientam e organizam os seus serviços aos aspectos da computação ubíqua que se deseja implementar. Por exemplo, o foco do aspecto de mobilidade da computação ubíqua é a movimentação do usuário e dispositivos pelo espaço inteligente. Nesse sentido, as plataformas de *middleware* orientadas a mobilidade geralmente especificam e organizam seus serviços em torno do conceito de migração do usuário pelo ambiente. No cenário de sala de aula inteligente e da aplicação de apresentação, descrito em 1.1, esse aspecto representaria a possibilidade do aluno poder trocar de computador e continuar lendo o *slide* na mesma página que estava sendo acessada no computador anterior. Sob esse mesmo aspecto ainda pode-se construir mecanismos para manter a conectividade e sincronização dos *slides* do aluno quando ele se movimenta pelo ambiente.

O aspecto de mobilidade define serviços do ambiente centrados na movimentação do usuário, usualmente realizados com abstrações do estado e sessão da aplicação. Por exemplo, o projeto Aura [20, 49], da Universidade Carnegie Mellon, propõe um conjunto de serviços utilizando o conceito de "aura", uma entidade que age como intermediário da localização do usuário. A aura abstrai a localização atual do usuário além de conter metadados sobre a tarefa em execução do usuário, tais como tipo de aplicação, dados, e dispositivos utilizados pelo usuário. Aplicações em Aura utilizam um ciclo de vida bem definido que contempla estados necessários para migração, como iniciar, pausar e resumir. Utilizando as informações da aura é possível pausar a aplicação em um local e migrar seu estado para uma instância compatível da tarefa no novo local do usuário.

Plataformas de *middleware* podem especificar novos serviços ou rearranjar/recombinar os serviços existentes para implementar outros aspectos da computação ubíqua. Por exemplo, o aspecto de ciência de contexto visa fornecer mecanismos para caracterizar a situação de uma entidade, geralmente do usuário no ambiente [18]. Esse aspecto de computação ubíqua possibilita que, no cenário da aplicação de apresentação de *slides* da sala de aula interativa, a apresentação utilize as informações do ambiente e do dispositivo atual do usuário para adaptar o formato da apresentação (*pdf, ppt, ps*, etc) para um formato compatível com o seu dispositivo. As plataformas de *middleware* que implementam o aspecto de ciência de contexto comumente especificam e organizam seus serviços na forma de abstrações que permitem recuperar e fornecer dados sobre o ambiente do usuário. Essas informações são fornecidas por sensores lógicos ou físicos espalhados pelo ambiente. Por exemplo, a extensão da plataforma de *middleware* MoCA, detalhada em [17], fornece um mecanismo uniforme para obtenção dos dados dos sensores do ambiente com o intuito de abstrair e resolver a heterogeneidade entre os sensores presentes no ambiente.

Foram desenvolvidas diversas plataformas de *middleware* que definem e organizam a configuração dos serviços que permeiam o espaço inteligente, principalmente plataformas orientadas aos aspectos de mobilidade e contexto da computação ubíqua [42, 49, 53, 64]. O foco principal dessas plataformas é fornecer serviços que integrem o usuário ao ambiente, isto é, essas infraestruturas lidam principalmente com a construção de mecanismos que permitam imergir um específico usuário com o ambiente, ou seja, essas plataformas especificam e distribuem serviços pela infraestrutura do espaço inteligente com objetivo de realizar a imersão do usuário com o ambiente físico.

2.2.2 Aspecto de Conteúdo no Espaços Inteligente

O aspecto de conteúdo da computação ubíqua vislumbra a possibilidade de permitir que conteúdos sejam compartilhados livre e naturalmente entres os usuários presentes no ambiente. Nesse aspecto, os principais dados das aplicações são elevados à categoria de entidades de primeira classe e passam a ser denominados *conteúdo ubíquo*, podendo ser trocados e compartilhados naturalmente entre os usuários do ambiente. Conteúdos ubíquos também podem existir simultaneamente em diversos pontos do espaço inteligente, habilitando mecanismos de sincronização entre os usuários. As aplicações neste domínio de computação ubíqua são centradas na manipulação de conteúdo no ambiente. Definições mais precisas de conteúdo e de conteúdo ubíquo conforme adotado neste trabalho são dadas a seguir.

Conteúdo. *São dados das aplicações elevados à categoria de entidades de primeira classe que tipicamente expressam as principais estruturas do modelo da aplicação.*

Conteúdo Ubíquo. *É um conteúdo opaco ao middleware que pode ser manipulado em um ambiente de computação ubíqua pelos usuários presentes utilizando diferentes formas de compartilhamento.*

As construções do aspecto de conteúdo ubíqua da computação estão centradas na manipulação de conteúdo ubíquo, permitindo compartilhar itens de conteúdo no ambiente. Os conteúdos ubíquos devem ser opacos ao *middleware*, possibilitando que eles sejam manipulados de diferentes formas pela plataforma de maneira independente da sua estrutura. O compartilhamento descreve uma ação de dar, dividir ou emprestar o conteúdo ubíquo a um ou mais usuários no ambiente, realizada através da transferência do conteúdo no espaço inteligente. Pode-se derivar diretamente dessa definição mecanismos mais elaborados de compartilhamento. Por exemplo, para representar a ação de dar definitivamente, isto é, mover o conteúdo a outro usuário, basta tornar o período do empréstimo demasiadamente longo (infinito). Como computadores podem copiar dados, também é possível emprestar o conteúdo sem perdê-lo, criando assim mecanismos de

clonagem e sincronização de conteúdo entre os usuários. Pela combinação dessas ações é possível construir outros mecanismos mais elaborados para compartilhar conteúdo no espaço inteligente.

O compartilhamento de conteúdo ubíquo compõe o núcleo de comunicação encontrados em diversas aplicações que suportam compartilhamento no espaço inteligente, como aplicações que seguem o usuário, aplicações colaborativas e telas públicas, como ilustrado na Figura 2.3. Por exemplo, no cenário da aplicação de apresentação de *slides* em sala de aulas interativas, descrita em 1.1, utiliza-se o compartilhamento de slides e anotações como meio de aumentar a interatividade entre alunos e professores. Nessa aplicação, o desenvolvedor define como conceitos principais os conteúdos de apresentação (*slides*) e anotações dos alunos, de modo que a aplicação seja centrada na manipulação desses conteúdos, particularmente, na sua sincronização e troca transparente e natural entre os usuários presentes no ambiente para proporcionar a interatividade desejada. A aplicação é centrada na manipulação desses conteúdos, tanto na sua criação e edição quanto na sua movimentação pelo ambiente. A movimentação dos conteúdos é um dos ingredientes necessários para que se construa um ambiente colaborativo de desenvolvimento.

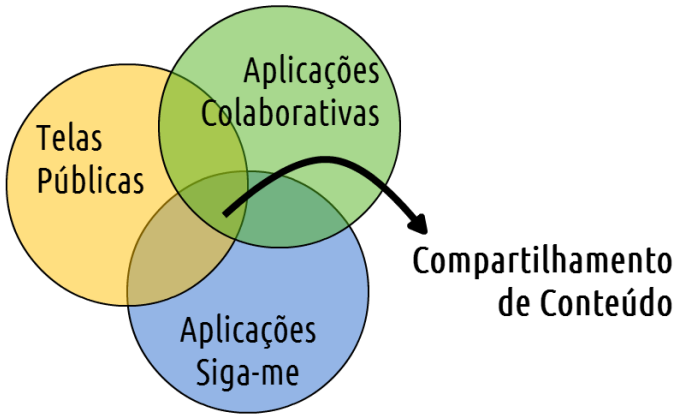

Figura 2.3: *O compartilhamento de conteúdo visto como intersecção entre aplicações orientadas a conteúdo ubíquo.*

Como outro exemplo, considere uma aplicação colaborativa para o ensino de programação em salas de aula interativas. Os alunos poderiam utilizar gestos na tela de um *tablet* como mecanismo para ativar o compartilhamento ubíquo de dados no ambiente. Por exemplo, para entregar o programa construído na aula para o professor, o aluno arrastaria o seu programa em direção a um avatar que representa o professor. Em um trabalho em grupo, o ambiente de programação poderia associar gestos para sincronizar o código fonte com o restante do grupo. Outra opção para trabalhar em grupo seria dividir o programa em múltiplos módulos, que seriam desenvolvidos separadamente. Um aluno poderia enviar sua parte do código para um colega com o intuito de reunir as partes do programa feitas pelo grupo.

Ainda no domínio educacional, [37] descreve uma aplicação para engajar alunos no aprendizado de química orgânica. Nessa aplicação, o professor envia perguntas (conteúdos) a grupos de alunos contendo uma descrição literal da estrutura química. Os alunos devem interagir entre si e responder a pergunta com um desenho (conteúdo) da composição química de um composto. Aplicações que seguem o usuário (*follow-me*) também podem ser orientadas a conteúdo. Ao contrário de mover a aplicação, como feito pelo projeto Aura, o foco seria transferir conteúdos entre as aplicações que o usuário utiliza à medida que ele(a) se move pelo ambiente.

Outro exemplo de conteúdo em espaço inteligente encontra-se na aplicação de telas públicas *MobiToss* [45], que define fotos e vídeos como conteúdos que podem ser manipulados pelo ambiente. Nessa aplicação os usuários podem utilizar um gesto de "tacar" (*toss*) para mover dados para uma tela pública. Similarmente, a aplicação *Dynamo* [26] permite mover e copiar imagens (conteúdos) para as telas ou usuários presentes do ambiente, possibilitando inserir e retirar conteúdos da tela pública do ambiente.

Diferentemente dos aspectos de mobilidade e contexto, que visam imergir o usuário no ambiente, o aspecto de conteúdo da computação ubíqua utiliza e organiza os serviços como meio para integrar os usuários presentes no ambiente de computação ubíqua [27, 30, 43]. Os aspectos de mobilidade e contexto especificam e organizam seus serviços para oferecer abstrações focadas na imersão dos usuários, tais como na migração de tarefa ou informações contextuais ao seu redor. Diferentemente, o aspecto de conteúdo orquestra os serviços do ambiente de computação ubíqua para possibilitar a integração entre os usuários presentes. O conteúdo funciona como meio intermediário para realizar a integração dos usuários presentes no ambiente. Por exemplo, na aplicação de apresentação de *slides*, a anotação de um *slide* compartilhada com outro aluno no espaço inteligente serve como meio de integrar os alunos. O mesmo raciocínio se aplica ao espelhamento e sincronização dos *slide* do professor nos *tablets* dos alunos. A Figura 2.4 mostra uma ilustração simplificada dos aspectos de computação ubíqua discutidos.

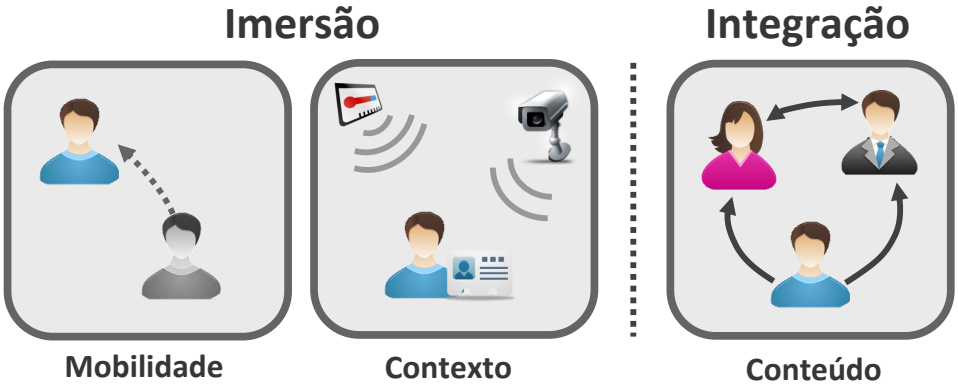

Figura 2.4: *Aspectos da Computação Ubíqua*

2.3 Plataformas de *Middleware* para o Aspecto de Conteúdo Ubíquo em Espaços Inteligentes

Existem poucas plataformas de *middleware* construídas para dar suporte ao aspecto de conteúdo da computação ubíqua. Esta pesquisa encontrou apenas plataformas de *middleware* que tratam em segundo plano [7, 27] ou parcialmente [58] esse aspecto. As plataformas identificadas tipicamente fornecem apenas uma maneira de compartilhar conteúdo no espaço inteligente, geralmente, a sincronização de dados com *todos* usuários do ambiente. Isto implica que os conteúdos ubíquos são manipulados de uma única maneira apenas pelas plataformas de *middleware* existentes. Por exemplo, na aplicação de apresentação *slides* em uma sala de aula interativa, uma anotação só poderia ser sincronizada ou transferida, não ambas.

A maioria das plataformas de *middleware* [7, 34, 23, 27] focadas no aspecto de conteúdo da computação ubíqua não oferecem mecanismos de compartilhamento direto entre usuários, ou seja, elas não fornecem abstrações que permitem enviar ou sincronizar um item de conteúdo ubíquo com um usuário específico. Essa limitação impacta diretamente as construções de compartilhamento utilizadas pelas aplicações, visto que limita a aplicação a uma determinada forma de compartilhamento. Por exemplo, na aplicação de apresentação de *slides*, as anotações seriam enviadas a todos os alunos, não sendo possível compartilhar uma anotação com um único usuário específico do ambiente. Essa ação de compartilhamento teria que ser escrita separadamente utilizando técnicas não convencionais, como RPC e objetos distribuídos, visto que não há suporte para essa forma de compartilhamento.

Outra limitação das plataformas de *middleware* orientadas a conteúdo ubíquo no espaço inteligente encontra-se na definição da operação do compartilhamento, isto é, como as abstrações de compartilhamento são fornecidas. As plataformas descritas requerem que a forma de compartilhar seja definida a priori para o tipo de conteúdo em questão, isto é, o modo de compartilhar o conteúdo faz parte da própria definição de seu tipo. Essa visão limita um tipo de conteúdo a uma semântica definida estaticamente, de maneira que qualquer conteúdo ubíquo de um determinado tipo terá a mesma semântica de compartilhamento definida pelo desenvolvedor. Isto limita o grau de abstração das aplicações orientadas a conteúdo. Por exemplo, anotações na aplicação educacional de sala de aula só poderão ser compartilhadas de uma única maneira, o que inviabiliza sua troca e sincronização entre os alunos.

Muitas aplicações de compartilhamento em espaços inteligentes utilizam algum conceito para representar interações entre grupos de pessoas ou entidades. Entretanto, as plataformas de *middleware* existentes não dão suporte a operações com conteúdo ubíquo envolvendo grupos. Por exemplo, um usuário da aplicação de IDE colaborativa pode

sincronizar seu código fonte (conteúdo) com os membros de seu grupo. Outro exemplo, uma aplicação de tela pública, pode permitir que apenas usuários de um determinado grupo enviem ou sincronizem itens de conteúdo ubíquo com a tela.

No âmbito de implementação, uma das complicações de movimentar conteúdo entre os dispositivos do ambiente remete ao problema de deixar referências inconsistentes. Esse problema pode causar falhas na aplicação, como a terminação inesperada no caso de ponteiros nulos, ou levar a resultados errados, no caso de conteúdo desatualizado. Isto ocorre porque a estrutura de um conteúdo pode referenciar ou ser referenciada por outros conteúdos, seja para definir relacionamentos ou como meio de construir estruturas complexas de conteúdo; um exemplo disso seria um conteúdo composto, como um *slide* que contém uma anotação. Em alguns casos, por exemplo, na semântica de mover, como resultado do compartilhamento essas referências podem tornar-se inconsistentes.

Outra complicação do compartilhamento de conteúdo ocorre no controle de concorrência durante a movimentação ou sincronização do conteúdo no ambiente. Esse problema ocorre principalmente na transferência ou sincronização de conteúdos maiores, como desenhos, imagens e vídeos. Durante a navegação do conteúdo no meio ele pode sofrer alterações realizadas pela aplicação de origem, o que pode ocasionar inconsistência ao chegar no seu destino.

Em suma, as limitações encontradas nessas plataformas de *middleware* podem ser resumidas nos seguintes pontos:

- Poucas abstrações para compartilhamento de conteúdo (geralmente sincronização);
- Compartilhamento vinculado ao tipo de conteúdo ubíquo, impossibilitando que um conteúdo desse tipo seja compartilhado de outra forma;
- Pouca ou nenhuma abstração para representar compartilhamento entre usuários organizados em grupos; e
- No âmbito de implementação, problemas de integridade referencial e controle de concorrência no compartilhar conteúdo.

A escassez, juntamente com as limitações das plataformas de *middleware* orientadas ao aspecto de conteúdo ubíquo existentes, dificulta a construção dessas aplicações, fato confirmado pelas inúmeras aplicações [4, 25, 36, 37, 52] escritas a partir "do zero" (*from-scratch*) para espaços inteligentes. Esta dissertação tem como objetivo investigar e propor soluções que facilitem a construção de aplicações que exploram o aspecto de conteúdo da computação ubíqua em espaços inteligentes. Esses desafios, abstração e implementação de semânticas de compartilhamento, definição de estruturas de grupos, consistência e integridade no compartilhamento, estão intimamente ligados aos requisitos das aplicações centradas em compartilhamento e no aspecto de conteúdo no espaço inteligente. A definição de conceitos que atendam esses desafios serve de base para as

abstrações providas pela plataforma de *middleware* proposta nesta dissertação. Para isto, o próximo capítulo visa definir a estrutura, comportamento e abstrações para manipular conteúdos ubíquos no espaço inteligente. Esses conceitos servem de base para as construções do modelo da plataforma de *middleware* C3S, proposta nesta dissertação.

2.4 Sumário e Considerações Finais

Este capítulo apresentou o contexto conceitual geral desta dissertação, incluindo a motivação e desafios encontrados no desenvolvimento de aplicações ubíquas orientadas ao aspecto de conteúdo em espaços inteligentes. O capítulo inicialmente discorre sobre o estado atual da computação ubíqua, que busca tornar a computação invisível na visão do usuário ao integrá-la no dia-a-dia do seu ambiente físico [60]. Essa integração requer interação com os outros elementos computacionais presentes no ambiente [6]. Para realizar isso, faz-se necessário definir uma infraestrutura que facilita esse processo [22, 24]. Os espaços inteligentes realizam essa integração por meio de serviços computacionais que orquestram os dispositivos presentes em uma determinada área, tornando esse local "inteligente" [42, 47].

A organização e definição dos serviços que compõem o espaço inteligente permite habilitar um ou mais aspectos da computação ubíqua, tais como mobilidade e contexto. Existem diversos trabalhos que focam em prover esses aspectos [39, 40, 42, 49, 55]. Essas plataformas de *middlewares* lidam principalmente com o suporte a esses aspectos para integrar o usuário ao ambiente.

Espaços inteligentes orientadas a conteúdo fornecem mecanismos e abstrações centradas na manipulação ubíqua de conteúdo no ambiente. Essas abstrações servem de base para a construção de várias aplicações centradas em compartilhamento de conteúdo, como aplicações colaborativas e telas públicas. Entretanto, diferentemente dos aspectos de computação ubíqua mencionados anteriormente, o aspecto de conteúdo usa a infraestrutura do ambiente de computação ubíqua como meio para integrar com os usuários e facilitar a interação entre eles.

As plataformas de *middleware* orientadas a conteúdo ubíquo existentes são escassas e limitadas, principalmente por proverem suporte para apenas uma operação de compartilhamento, geralmente, a sincronização de dados com *todos* usuários do ambiente, assim como a necessidade de definir a priori e estaticamente o conjunto de dados que podem ser compartilhados. Dentro das limitações encontradas nessas plataformas de *middleware*, pode-se destacar:

- Poucas abstrações para compartilhamento de conteúdo (geralmente sincronização);
- Compartilhamento vinculado ao tipo de conteúdo ubíquo, impossibilitando que um conteúdo desse tipo seja compartilhado de outra forma;

- Pouca ou nenhuma abstração para representar compartilhamento entre usuários organizados em grupos; e
- No âmbito de implementação, problemas de integridade referencial e controle de concorrência no compartilhar conteúdo.

Para solucionar as limitações das plataformas de *middleware* existentes para construção de aplicações ubíquas orientadas ao aspecto de conteúdo, o próximo capítulo investiga a estrutura e comportamentos necessários para a manipulação de conteúdo ubíquo em espaços inteligentes. O resultado dessa investigação é materializado no modelo da plataforma de *middleware* C3S para espaços inteligentes. Esse modelo define os conceitos que sustentam a manipulação e compartilhamento de conteúdo no ambiente de computação ubíqua. Os conceitos desse modelo especificam as formas de compartilhamento de conteúdo ubíquo no ambiente e a organização dos componentes das aplicações que manipulam esse tipo de conteúdo.

CAPÍTULO 3

Modelo de Espaço Inteligente Orientado a Conteúdo

A manipulação de conteúdo ubíquo feita sobre a infraestrutura e serviços do espaço inteligente, possibilita o desenvolvimento de diversas aplicações baseadas em compartilhamento. Este capítulo explora as diferentes maneiras de compartilhar o conteúdo ubíquo no espaço inteligente com o intuito de definir o modelo da plataforma de *middleware* proposta nesta dissertação. Este modelo fornece serviços e abstrações para compartilhamento de conteúdo ubíquo. Inicialmente, exploramos o comportamento de algumas definições pré-existentes de unidade de compartilhamento, tais como objeto compartilhado (*shared object*) [7, 23, 27, 34] e objeto móvel (*mobile object*) [28, 58, 63]. Entretanto, a manipulação dessas unidades, por serem de outros domínios, é estática e simplicista quando aplicada a espaços inteligentes. Nesse sentido, buscamos compreender os aspectos comuns da manipulação de conteúdo em aplicações encontradas na literatura [4, 52]. Para dar suporte as diferentes formas de compartilhar conteúdo ubíquo entre usuários no ambiente de computação ubíqua foi necessário definir os conceitos e serviços aplicação, ativação de aplicação e aplicação ubíqua. Esses conceitos descrevem como elementos da aplicação são organizados no ambiente de computação ubíqua, como por exemplo, instâncias, grupos e servidores, para poderem utilizar os serviços e abstrações de compartilhamento de conteúdo fornecidas pelo *middleware* C3S.

O restante deste capítulo está estruturado da seguinte forma: a Seção 3.1 explora algumas formas de compartilhamento de conteúdo ubíquo identificadas por este trabalho no ambiente de computação ubíqua. A Seção 3.2 materializa essas manipulações de conteúdo ubíquo no conceito de conteúdo ubíquo e em primitivas da plataforma do *middleware* C3S. Para sustentar essas formas de compartilhamento, a Seção 3.3 descreve os conceitos de aplicação, ativação de aplicação e aplicação ubíqua do *middleware* C3S. Por fim, a Seção 3.4 apresenta o sumário do capítulo e as considerações finais.

3.1 Manipulação de Conteúdo em Espaços Inteligentes

A manipulação de conteúdo ubíquo em um espaço inteligente possibilita o compartilhamento transparente e intuitivo entre os usuários do ambiente. As plataformas de *middleware* para computação ubíqua orientadas a conteúdo ubíquo focam em habilitar o compartilhamento do conteúdo entre os usuários do ambiente. O compartilhamento faz com que o conteúdo seja o meio pelo o qual os usuários interagem e colaboram entre si. Por exemplo, na aplicação de apresentação de *slides* em sala de aula interativa, discutida na Seção 1.2, a troca do conteúdo "anotação" estimula a colaboração entre os alunos. A mesma ideia se aplica ao sincronizar o conteúdo de uma apresentação do professor nos dispositivos dos alunos. Para compreender e projetar mecanismos de manipulação de conteúdo ubíquo no espaço inteligente foi necessário identificar as semânticas de compartilhamento de conteúdo que são adequadas como base para implementar a colaboração entre usuários.

A migração de conteúdo ubíquo é uma das semânticas de compartilhamento mais comuns no espaço inteligente. Essa semântica de compartilhamento permite mover um item de conteúdo ubíquo entre os dispositivos de usuários diferentes, de maneira que o conteúdo deixa de existir no ambiente do usuário de origem e passa a existir apenas no ambiente de destino. No cenário da aplicação de apresentação de *slides*, esta semântica de compartilhamento poderia ser utilizada para mover anotações entre usuários. Por exemplo, o aluno com dúvida poderia criar uma anotação e enviá-la ao professor, que poderia respondê-la e, novamente, movê-la para o dispositivo do aluno. Para algumas aplicações, uma referência remota ao conteúdo precisa ser mantida, após a sua migração para outro local, como ilustra a Figura 3.1. A ilustração mostra a movimentação do conteúdo ubíquo do usuário *A* para o usuário *B*, seguida da extensão das referências do conteúdo para manter sua integridade referencial no local de origem.

A clonagem é uma semântica de compartilhamento utilizada para clonar itens de conteúdo ubíquo no espaço inteligente. Essa semântica possibilita que usuários copiem conteúdo ubíquos e que possam ser desenvolvidos independentemente. Por exemplo, na aplicação de apresentação de *slides*, a clonagem de uma anotação de um usuário *A* para um usuário *B* permite que o usuário *B* receba o conteúdo e modifique adicionando sem alterar o conteúdo de *A*. A execução interna dessa operação é semelhante a operação de mover, com exceção que o conteúdo não é removido na origem.

Um conteúdo ubíquo também pode ser manipulado ao mesmo tempo por um ou mais usuários presentes no espaço inteligente. Por exemplo, na aplicação de apresentação de *slides* na sala de aula interativa o professor pode compartilhar uma apresentação com cada aluno presente no ambiente de maneira que o *slide* correntemente apresentado seja parte do conteúdo compartilhado, ou seja, quando o professor avança um *slide* essa alte-

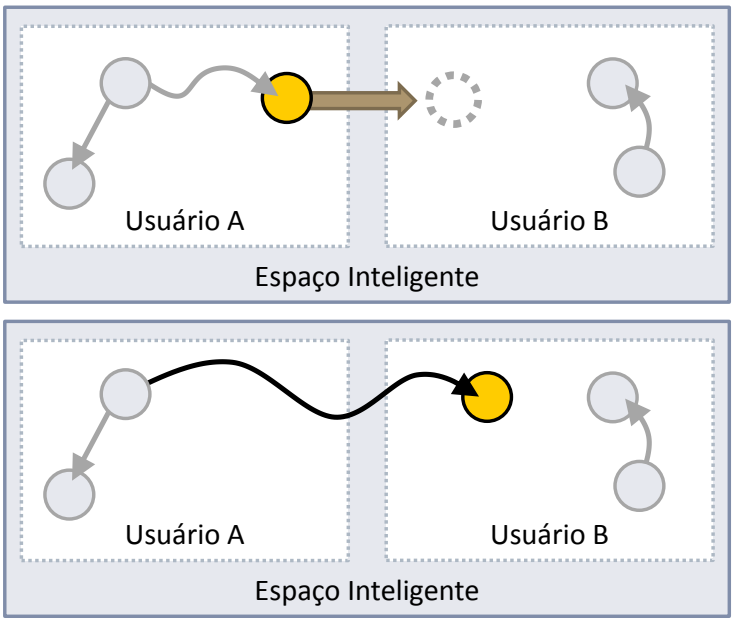

Figura 3.1: *Migração de um conteúdo referenciado entre dois usuários ($A \longrightarrow B$) no espaço inteligente.*

ração é efetuada nas cópias dos alunos. Essa semântica de compartilhamento dissemina atualizações do conteúdo no ambiente de computação ubíqua, ou seja, o conteúdo é apresentado igualmente para todos os usuários envolvidos de maneira que as modificações sofridas por ele são refletidas nas cópias de cada usuário. Esse comportamento possibilita que o conteúdo esteja logicamente disponível em diversos dispositivos e ao mesmo tempo proporcione uma visão compartilhada que promova a colaboração entre os usuários do ambiente, como ilustrado pela Figura 3.2. Por exemplo, as aplicações podem utilizar esta semântica para especificar algum mecanismo que coordene as ações dos usuários, como no caso da aplicação de apresentação que mantém o *slide*, exibido sincronizado entre os alunos. A implementação dessa semântica de compartilhamento na maior parte dos trabalhos [7, 23, 34] replica o conteúdo por todo o espaço inteligente, mesmo quando apenas pequenas parte do conteúdo foram alteradas, por exemplo, ao alterar um ou dois atributos o conteúdo inteiro é replicado.

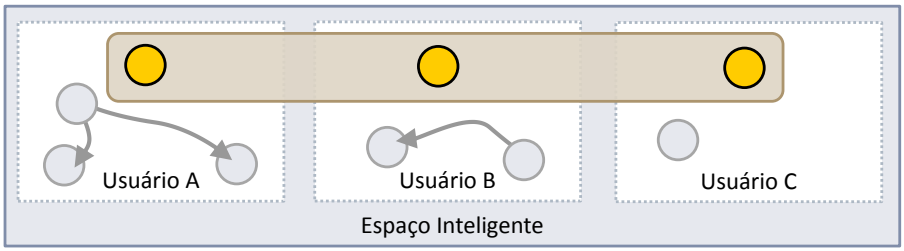

Figura 3.2: *Manipulação do mesmo conteúdo por diversos usuários (A, B, C) do espaço inteligente.*

Os serviços orientados a conteúdo ubíquo fornecidos pelas plataformas de *middleware* devem atender a um conjunto de aplicações diferentes. Portanto, é desejável que a estrutura do conteúdo ubíquo seja transparente e flexível no nível do *middleware*. Outra característica comum de conteúdo ubíquo e, consequentemente, de outras unidades de compartilhamento é a necessidade de ter uma representação externa de dados, como serialização de dados baseada em XML ou JSON [14].

Outras áreas de pesquisa já propuseram semânticas de compartilhamento de conteúdo, tais como objeto compartilhado [7, 8] e objeto móvel [28, 63], que apresentam algumas das características discutidas acima. Entretanto, a definição de conteúdo ubíquo nestes trabalhos são restritos a uma única semântica de compartilhamento, definida a priori pelo programador durante o desenvolvimento da aplicação. Por exemplo, ao especificar que um conteúdo do tipo anotação está associado à semântica de migração, todas as anotações ficarão restritas a essa modalidade de compartilhamento entre usuários, inviabilizando outras formas de compartilhamento, como a clonagem ou espelhamento. Além disso, tais semânticas são aplicadas a todas as unidades de compartilhamento do ambiente, mesmo àquelas para os quais o compartilhamento não é necessário [5, 33], o que restringe o comportamento das aplicações. Por exemplo, na aplicação de apresentação de *slides* em espaço inteligente, ao especificar que o conteúdo do tipo anotação é espelhado, ele só poderá ser compartilhado desta maneira e com todos os outros alunos do ambiente. Essa especificação torna o uso das anotações limitado e estático, inviabilizando outros tipos de colaboração entre os usuários, tais como aluno mover uma anotação privada para o professor ou copiá-la para um determinado colega.

3.2 Conteúdo Ubíquo no Espaço Inteligente

A seção anterior apontou as limitações de duas abstrações de compartilhamento de conteúdo existentes na literatura: *objeto compartilhado* e *objeto móvel*. Esta dissertação propõe um conceito mais amplo de *c*onteúdo ubíquo, como uma abstração de compartilhamento de conteúdo à qual podem ser aplicadas diversas semânticas de compartilhamento, inclusive aquelas mencionadas anteriormente. A estrutura do conteúdo ubíquo deve ser opaca ao *middleware*, para que possa ser manipulada no espaço inteligente utilizando diferentes semânticas de compartilhamento entre os usuários presentes no ambiente. Isto possibilita que um conteúdo ubíquo seja implementado de diferentes formas, por exemplo, por objeto, tuplas ou módulos, como representado na Figura 3.3.

Um conteúdo ubíquo também pode referenciar ou ser composto de outros conteúdos ubíquos. Por exemplo, um conteúdo de apresentação (*slide*) pode conter uma ou mais anotações. A interface de um conteúdo ubíquo também deve fornecer funções para criar, estabelecer e destruir referências e relacionamento com outros conteúdos,

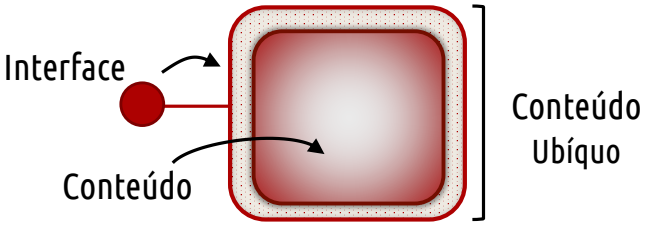

Figura 3.3: *Estrutura de um Conteúdo Ubíquo.*

denominados de ligações. Essas ligações servem de meio para estabelecer uma relação qualquer entre conteúdos, de modo que um dado item de conteúdo a use como forma de recuperar outro item de conteúdo ubíquo, ao invés de referenciá-lo diretamente. A ligação age como uma abstração do conceito de referência da linguagem de programação, permitindo que o conteúdo possa ser recuperado mesmo quando estiver em outro usuário. Quando uma ligação é criada ou destruída, tanto o conteúdo que é origem da ligação como o destino são notificados. Quando um conteúdo ligado migra para outro dispositivo, a referência da ligação é transformada em uma referência remota para o novo local do conteúdo. Ao executar uma operação nesse conteúdo, a chamada é encaminhada por meio de um *proxy* para o dispositivo onde o conteúdo se encontra.

Para materializar múltiplas formas de compartilhamento para um mesmo conteúdo ubíquo no espaço inteligente, identificamos três primitivas para compartilhamento de conteúdo que implementam as semânticas de mover, clonar e espelhar, como ilustrado na Figura 3.4. Os desenvolvedores podem combinar essas primitivas para construir novas semânticas de compartilhamento. Por exemplo, uma semântica de troca de conteúdo pode ser implementada a partir da primitiva de mover: um usuário move um conteúdo para outro usuário que, por sua vez, move outro conteúdo para o primeiro usuário.

A primitiva *mover* migra um conteúdo ubíquo do dispositivo de um usuário para outro, sendo que o conteúdo ubíquo deixa de existir na instância da aplicação de origem para ser recriado na instância de destino da aplicação. Depois da execução dessa primitiva, ligações contendo referências ao conteúdo podem tornar-se inconsistentes, visto que o conteúdo não se encontra mais presente localmente. Para lidar com isso é necessário transformar as referências locais para o conteúdo em referências para o novo local do conteúdo. A interface do conteúdo ubíquo remoto inspeciona a estrutura do conteúdo e redireciona as chamadas para o seu novo destino.

A primitiva *clonar* copia itens de conteúdo ubíquo entre usuários do espaço inteligente. Ela é interessante do ponto de vista da colaboração porque, ao copiar um conteúdo ubíquo, ele não é eliminado na origem. Por exemplo, um aluno pode compartilhar uma anotação realizada no *slide* com um colega sem perder o seu conteúdo de origem, como ocorre na primitiva mover. A utilização dessa primitiva também permite o desenvolvi-

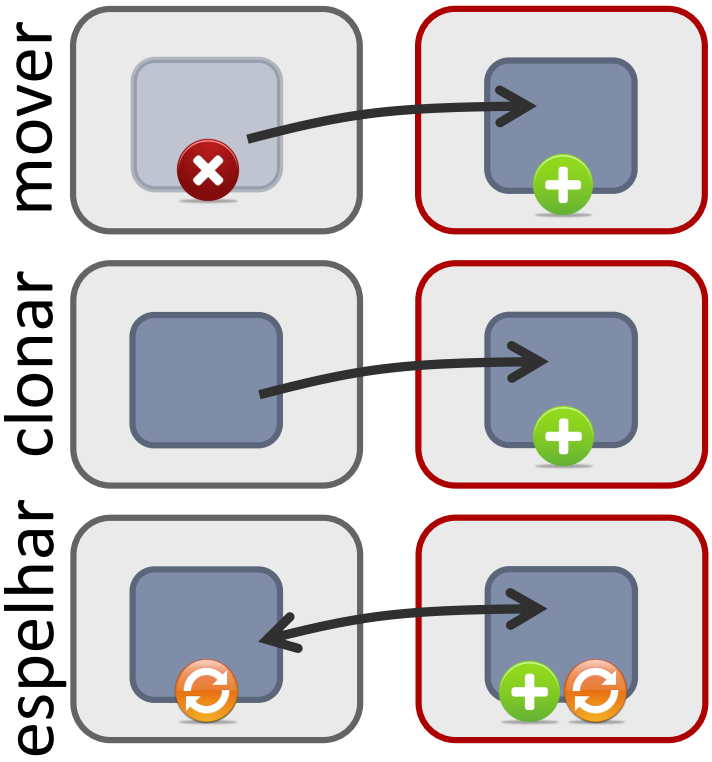

Figura 3.4: *As primitivas de compartilhamento de conteúdo ubíquo fornecidas pela plataforma de middleware C3S.*

mento de um item de conteúdo ubíquo individualmente; por exemplo, a anotação pode ser modificada na cópia clonada e isso não é refletido na cópia original. Como conteúdos clonados não são removidos na origem, não é necessário tratar ou lidar com questões de ligações inconsistentes.

A primitiva *espelhar* realiza a cópia de um conteúdo ubíquo e, em seguida, estabelece um vínculo de sincronismo entre a nova cópia e a cópia original. Ao espelhar um conteúdo ubíquo para um usuário qualquer, ele passa a existir tanto na instância da aplicação de origem quanto na instância da aplicação do usuário alvo, sendo que modificações no conteúdo ubíquo realizadas em qualquer instância serão repassadas as réplicas.

As primitivas também possibilitam manipular conteúdo ubíquo de maneira independente do dispositivo do usuário no espaço inteligente. Isto é, desenvolvedores podem utilizar apenas o nome de usuário como endereço de destino ao contrário da localização ou endereço real do dispositivo físico do usuário. Esse mecanismo permite que o compartilhamento seja executado mesmo quando o usuário troque de dispositivo, pois o dispositivo atual do usuário atualiza o ponteiro do endereço lógico para o endereço do dispositivo físico atual.

3.3 Aplicação Ubíqua

No modelo de espaço inteligente adotado para o C3S, uma aplicação colaborativa é representada usando os conceitos de *aplicação*, *ativação* e *aplicação ubíqua*. Uma aplicação é formada por seu código executável e seus metadados. Nos metadados, além de nome, versão e plataforma, é especificado se a aplicação provê suporte para compartilhamento. Todas as aplicações são mantidas em um servidor de aplicações colaborativas, a partir do qual os dispositivos podem requisitar a sua instalação ou atualização. A execução de uma aplicação é gerenciada pelo *middleware* e obedece um ciclo de vida simples bem definido, que inclui estados para iniciar, pausar e resumir a aplicação, além de estados para receber conteúdo. Por exemplo, no cenário de sala de aula interativa, o servidor de aplicações pode manter os binários da aplicação de apresentação de *slides* do professor para que ela possa ser então instalada nos *tablets* dos alunos.

Com intuito de organizar as aplicações no espaço inteligente e possibilitar o compartilhamento de conteúdo entre elas nós definimos os conceitos de ativação e aplicação ubíqua. Cada instância local da aplicação é chamada de *ativação da aplicação*. Uma aplicação ubíqua, por sua vez, é um conjunto de ativações de uma aplicação do C3S, isto é, ela é formada pelas instâncias da aplicação no ambiente de computação ubíqua. Várias aplicações ubíquas podem ser executadas ao mesmo tempo em um mesmo ambiente de computação ubíqua. Para exemplificar esses conceitos, a Figura 3.5 mostra a disposição de duas aplicações ubíquas em um ambiente de computação ubíqua. A aplicação ubíqua de apresentação de *slides* (*Y*) possui ativações nas três instâncias do C3S correspondente aos usuários *A*, *B* e *C*, enquanto que um jogo infantil (*X*) possui ativações apenas nas instâncias de *A* e *C*. O servidor contém um repositório com as aplicações utilizadas neste ambiente.

Figura 3.5: *Relacionamento entre ativação, aplicação e aplicação ubíqua.*

3.4 Sumário e Considerações Finais

Este capítulo apresentou o modelo de compartilhamento de conteúdo em espaços inteligentes adotado neste trabalho. Inicialmente destacamos a relevância desta funcionalidade ao mostrar que o conteúdo possibilita integrar os usuários presentes no ambiente de computação ubíqua. Por exemplo, no cenário da aplicação de apresentação de *slides*, discutido em 1.2 em uma sala de aula a movimentação de um conteúdo serve como canal para comunicação e integração entre os usuários presentes.

Foi discutido algumas formas de compartilhar (semânticas) conteúdo no espaço inteligente. Este trabalho identificou os comportamentos de migração e espelhamento como meios fundamentais para a realização do aspecto de conteúdo da computação ubíqua. O primeiro serve como meio de transferir e navegar conteúdo no ambiente, enquanto que o segundo possibilita que um conteúdo seja manipulado ao mesmo tempo por diversos usuários tornando-se onipresente. Nós exploramos algumas definições de conteúdo existentes, como objeto compartilhado (*shared object*) [7, 23, 27, 34] e objeto móvel (*mobile object*) [28, 58, 63], com intuito de reutilizá-las nesse ambiente para materializar esses aspectos. Entretanto, a semântica de compartilhamento é embutida no tipo dessas estruturas o que limita sua expressividade, isto é, um conteúdo desse tipo só poderia ser compartilhado usando a mesma semântica.

As limitações das "definições" das unidades de compartilhamento existentes ocorrem devido terem sido projetadas para outros domínios de aplicação como *groupware* e agentes móveis. Este trabalho buscou sistematizar um conteúdo ubíquo mais flexível e dinâmico. Um conteúdo ubíquo é a adição de uma camada de gerenciamento a um conteúdo "normal" que possibilita ele ser manipulado conforme diferentes semânticas no espaço inteligente sem lidar com detalhes de implementação do conteúdo. Essa camada envolve o conteúdo e permite inspecionar e refletir a sua estrutura interna. Ela inclui operações que fornecem metadados do conteúdo permitindo sua manipulação externa, por exemplo, funções para verificar se o conteúdo está sincronizado ou replicado, a qual ou-suário(s) pertence e obter sua representação externa (XML, JSON, etc). Essa camada também fornece funções para criar, estabelecer e destruir relacionamentos entre conteúdos, denominados ligações. Ligações servem como meio semântico para estabelecer relacionamentos entre conteúdos. Quando um conteúdo ligado migra para outro dispositivo, a referência da ligação é estendida remotamente para o novo local. Para realizar isso, a camada de conteúdo ubíquo inspeciona o conteúdo para redirecionar as chamadas de sua interface para o seu novo destino.

O comportamento de conteúdo ubíquo é separado de sua estrutura, isto é, um mesmo conteúdo pode ser manipulado de diferentes formas. Para materializar isso, essas semânticas são extraídas e encapsuladas em funções separadas. Identificaram-se

três primitivas básicas para troca de conteúdo, denominadas primitivas, mover, clonar e espelhar, como expressas na Figura 3.4. Essas primitivas podem ser combinadas para produzir novas semânticas de compartilhamento, por exemplo, uma semântica de troca pode ser construída usando as primitivas de mover. A primitiva de mover provê suporte para a movimentação de itens de conteúdo ubíquo entre instâncias da aplicação no ambiente. A primitiva clonar é parecida com a de mover, porém não elimina o conteúdo na origem. Por fim, a primitiva de espelhar permite múltiplos usuários manipular um dado conteúdo ao mesmo tempo.

Para materializar as primitivas de compartilhamento de conteúdo, por exemplo, como a instância recebe o conteúdo, encontrou-se necessário organizar os elementos da aplicação no espaço inteligente. A disposição desses elementos levou a definição dos conceitos de ativação, aplicação e aplicação ubíqua. Uma aplicação é formada por um executável junto de seus metadados. Ela é instalada em um servidor e os dispositivos dos usuários podem requisitar sua instalação/atualização. A execução de uma aplicação é gerenciada pelo C3S e obedece a um ciclo de vida bem definido, que inclui estados para iniciar, pausar e resumir a aplicação, além de estados para receber conteúdo. Por exemplo, pode pensar que no cenário de sala de aula interativa (um espaço inteligente) exista binários da aplicação de apresentação de *slide* do professor para que possa ser instalada nos *tablets* de cada aluno. Cada instância local da aplicação é chamada de ativação da aplicação. Uma aplicação ubíqua, por sua vez, é um conjunto de ativações de uma aplicação do C3S, isto é, ela é formada pelas instâncias da aplicação no ambiente de computação ubíqua. Várias aplicações ubíquas podem ser executadas ao mesmo tempo em um mesmo ambiente de computação ubíqua.

CAPÍTULO 4

Arquitetura e Implementação

Para materializar as ideias e construções que facilitam o compartilhamento de conteúdo em ambientes de computação ubíqua, este capítulo apresenta a plataforma de *middleware* proposta nesta dissertação, denominada **C**ontent **S**haring for **S**mart **S**paces (C3S). A plataforma C3S fornece abstrações de compartilhamento que visam sustentar as ideias descritas anteriormente, incluindo conteúdo ubíquo, primitivas de compartilhamento e aplicação ubíqua. Utilizamos os requisitos identificados na Seção 2.3, dentre eles a necessidade de possibilitar várias formas de compartilhamento para o mesmo conteúdo ubíquo, os quais guiam o projeto arquitetural e a implementação da plataforma. A arquitetura utiliza uma abordagem híbrida, sendo par-a-par (*peer-to-peer*) para interações entre usuários e cliente-servidor para o acesso a recursos, serviços e informações do ambiente. As primitivas são descritas como interações entre componentes do *middleware*, permitindo que o C3S possa ser implementado por diferentes plataformas de *middleware* de comunicação.

O restante deste capítulo está estruturado da seguinte forma: a Seção 4.1 enumera e descreve os requisitos que guiam o projeto arquitetural do C3S. A Seção 4.2 apresenta a sistematização do conceito de conteúdo ubíquo com o intuito de facilitar a descrição arquitetural do C3S. A Seção 4.3 apresenta a arquitetura do C3S, incluindo o funcionamento de cada componente e gerente do *middleware*. A Seção 4.4 descreve as interações necessárias entre os componentes do *middleware* para realizar as primitivas de compartilhamento de conteúdo. A Seção 4.5 apresenta os detalhes de uma implementação do C3S, baseada em Java RMI, seguida por uma discussão de suas limitações. A Seção 4.6 apresenta os conceitos e detalhes do modelo de programação do C3S. Por fim, a Seção 4.7 apresenta o sumário do capítulo e as considerações finais.

4.1 Requisitos do C3S

No capítulo anterior discutimos a manipulação e compartilhamento de conteúdo ubíquo em ambientes de computação ubíqua. Sistematizamos o comportamento e a estrutura do conteúdo ubíquo para que ele possa ser manipulado no espaço inteligente utili-

zando uma das primitivas (mover, clonar ou espelhar) de compartilhamento. Para enviar e receber os conteúdos foi necessário definirmos conceitos de ativação, aplicação e aplicação ubíqua, que permitem a disposição da aplicação no ambiente de computação ubíqua. A plataforma de *middleware* **C**ontent **S**haring for **S**mart **S**paces (C3S) visa implementar os conceitos de compartilhamento de conteúdo ubíquo descritos anteriormente. Os requisitos extraídos para materializar esses conceitos e guiar o projeto arquitetural da plataforma de *middleware* C3S são listados e resumidos nos seguintes pontos:

- **Conjunto de primitivas para mover, clonar e espelhar conteúdo ubíquo.** Essas três primitivas fornecem diferentes semânticas de compartilhamento do conteúdo ubíquo, como explicado no Capítulo 3. Essas funções devem ser fornecidas independentes do conteúdo ubíquo manipulado, ou seja, um mesmo conteúdo ubíquo pode ser manipulado por diferentes primitivas. Esse requisito possibilita que, no cenário da aplicação de apresentação de *slides* uma anotação em um *slide*, por exemplo possa ser tanto espelhada entre alunos como transferida (movida) para outros. Além disso, as primitivas podem ser combinadas para a construção de outras formas de compartilhamento. Nessa semântica, o desenvolvedor utiliza uma série de primitivas de mover conteúdo ubíquo como meio de movimentar e receber conteúdo ubíquo de um dado usuário.
- **Compartilhamento específico entre um ou mais usuários presentes no ambiente de computação ubíqua**. As plataformas de *middleware* existentes [7, 23, 27, 34] que oferecem suporte para o aspecto de conteúdo de computação ubíqua, fornecem mecanismos apenas para o compartilhamento de conteúdo ubíquo com todos os usuários presentes no ambiente de computação ubíqua, impossibilitando que um conteúdo ubíquo seja compartilhado entre usuários específicos do ambiente. A arquitetura do C3S implementa mecanismos que possibilitam que as primitivas de compartilhamento de conteúdo ubíquo sejam efetuados diretamente para um ou mais usuários específicos que estejam presente no ambientes, ou seja, cada uma das semânticas de compartilhamento das primitivas podem ser realizadas com um conjunto qualquer de usuários do ambiente.
- **Interdependência de conteúdo ubíquo.** Conteúdos ubíquos podem referenciar outros conteúdos ubíquos para construir estruturas complexas; por exemplo, pode-se construir um conteúdo ubíquo composto por outros conteúdos ubíquos. Para ilustrar esse cenário, pode-se compor um conteúdo ubíquo *slide*, que contém diversos conteúdos ubíquos representando anotações. Esse requisito especifica a necessidade da existência de mecanismos para relacionamento de conteúdos ubíquos na plataforma *C3S*.
- **Conceito de grupos**. Algumas aplicações ubíquas requerem a concepção de agrupamentos de usuários presentes em unidade denominadas grupos. Por exemplo,

pode-se agrupar os usuários da aplicação de apresentação de *slides* em grupos de alunos. Nesse sentido, esse requisito especifica a necessidade de mecanismos para criar, apagar, listar, entrar e sair de grupos de aplicações ubíquas existentes no ambiente de computação ubíqua. Por exemplo, na aplicação de apresentação de *slides* é interessante que o *middleware* C3S forneça mecanismos que possibilitem que o usuário entre ou saia de um determinado grupo de alunos da sala. O conceito de grupo pode ser utilizado para restringir o compartilhamento; por exemplo, a aplicação ubíqua pode utilizar o grupo como escopo de compartilhamento. Novamente, no cenário da aplicação de *slides*, pode-se utilizar o contexto de grupo para mover uma anotação em um *slide* para todos os usuários presentes naquele grupo.

4.2 Conteúdo Ubíquo

Para descrever e projetar a arquitetura do C3S, primeiramente foi necessário sistematizar e materializar o conceito de conteúdo ubíquo no espaço inteligente. Um conteúdo representa uma entidade de primeira classe da aplicação, tipicamente as principais estruturas do seu domínio. Um conteúdo ubíquo, por sua vez, possibilita disponibilizar o conteúdo para ser manipulado e compartilhado pelos usuários presentes no espaço inteligente. O fato de o conteúdo ser uma entidade de primeira classe da aplicação significa que conteúdos compõem a principal entidade do modelo de programação do C3S, ou seja, a aplicação é construída em torno da manipulação de conteúdo. Por exemplo, na aplicação de apresentação de *slides*, utilizada como cenário ao longo dessa dissertação, os *slides* do professor (apresentação) e as anotações são os conteúdos que formam as principais entidades da aplicação. A aplicação de *slides* é construída em torno da manipulação da apresentação de *slides* do professor e dos comentários dos alunos. No cenário descrito, a apresentação de *slides* do professor é replicada nos dispositivos dos alunos presentes.

Na plataforma C3S, o conteúdo ubíquo é sistematizado em uma classe definida pela aplicação que herda de outra classe bem-definida da plataforma (*UbiquitousContent*). Por estar associado às regras herdadas dessa classe do sistema, o conteúdo ubíquo é criado como um objeto que pode ser manipulado no ambiente de conteúdo ubíquo. Uma das propriedades herdadas é a capacidade do conteúdo ubíquo de ter uma representação externa de dados (usando serialização JSON ou XML) para ser utilizada no processo de compartilhamento. A classe *UbiquitousContent* especifica diversas propriedades de metadados sobre o conteúdo ubíquo, como usuário criador, estado de sincronização, estado de compartilhamento, permissões de compartilhamento por exemplo, (*onlyOwner*), que possibilitam ao *middleware* manipular o conteúdo ubíquo.

Para manipular conteúdos ubíquos, a plataforma de C3S cria automaticamente uma camada que envolve o conteúdo ubíquo. Essa camada reflete a interface do conteúdo

ubíquo a ser manipulado, agindo como um *proxy* ou interceptador. A utilização dessa camada intermediária, que é invisível ao programador devido à interface da camada ser idêntica à interface do conteúdo ubíquo, possibilita que o *middleware* intercepte e controle funcionalidades administrativas do conteúdo ubíquo, como o controle de acesso e consistência. A camada possui funções que possibilitam ao *middleware* checar o estado do conteúdo ubíquo, verificar os seus relacionamentos e replicar uma interação com o conteúdo, caso o conteúdo esteja espelhado. Por exemplo, as chamadas de método ao conteúdo ubíquo são interceptadas nessa camada. Ao perceber que o conteúdo ubíquo está espelhado, o C3S utiliza reflexão para serializar o método e seus argumentos e despachá-las as outras réplicas. A Figura 4.1 ilustra essa camada intermediária, que age como um *proxy* ao refletir a interface do conteúdo ubíquo. Além disso essa camada dispõem de funções administrativas do C3S, que permitem interceptar e facilitam o acesso ao conteúdo.

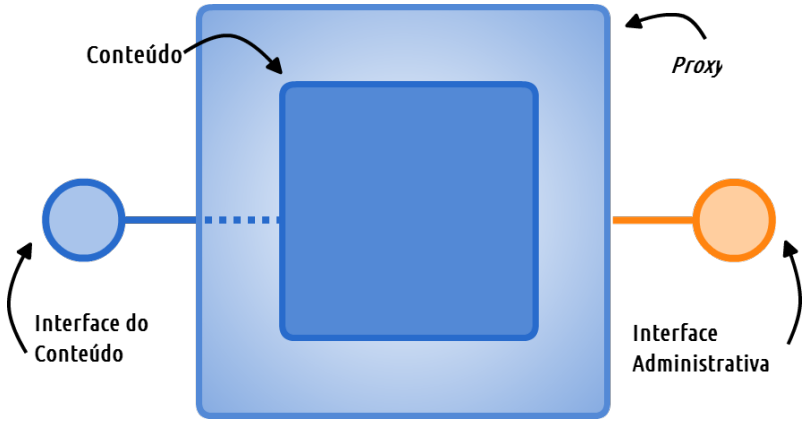

Figura 4.1: *Camada proxy que envolve um conteúdo ubíquo.*

As aplicações devem especificar seus conteúdo ubíquos como uma extensão da classe *UbiquitousContent*. Essa extensão permite adicionar as propriedades e comportamentos de cada entidade que forma um conteúdo ubíquo. Por exemplo, o conteúdo ubíquo de apresentação de *slides* teria como atributos um arquivo de PDF e um contador de página, que destaca a página atual da apresentação. Como comportamento pode-se definir as operações de passar uma página, voltar uma página, ou ir para uma página específica da apresentação. A Figura 4.2 ilustra uma representação da modelagem do conteúdo ubíquo de apresentação de *slides*. A ilustração também apresenta os atributos da apresentação, que incluem o arquivo em PDF e o número da página atual da apresentação e as operações para movimentar as páginas da apresentação. Na ilustração, percebe-se a distinção entre a representação gráfica e o conteúdo ubíquo.

O conteúdo ubíquo deve ser modelado independentemente de sua representação gráfica devido a dificuldades dessa representação ser serializada ou compartilhada de forma independente de aplicação. O desenvolvedor da aplicação é responsável por manter

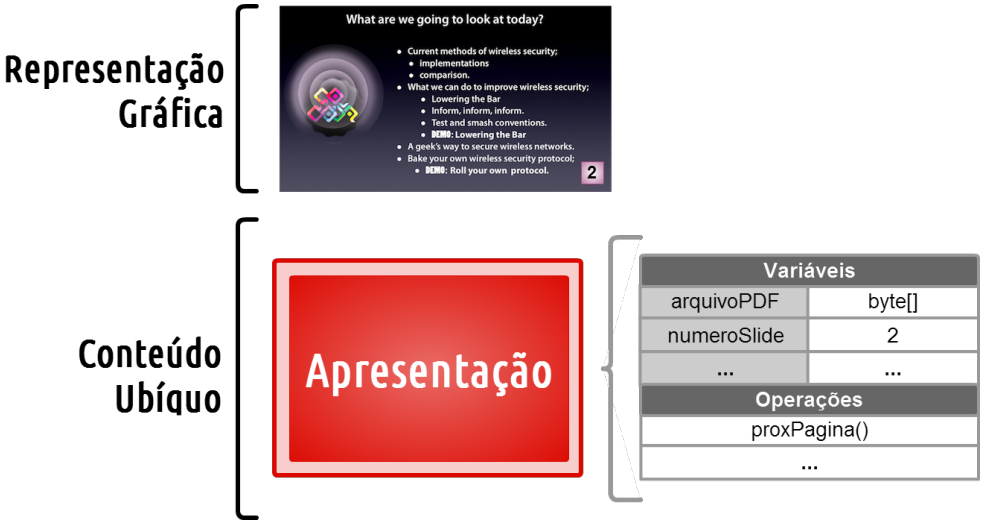

Figura 4.2: *Modelagem da apresentação como conteúdo ubíquo.*

o vínculo de causalidade entre a representação gráfica e o conteúdo ubíquo. Para isso, o *middleware* fornece métodos que são ativados quando ocorrem modificações na estrutura do conteúdo ubíquo. O conteúdo ubíquo implementa o padrão de projeto *Observer*, que permite que observadores sejam notificados em virtude de alterações na sua estrutura. Esses métodos são ativados por eventos gerados por modificações de origem externa, como por exemplo, por uma atualização em um conteúdo ubíquo replicado vinda de outras ativações. O C3S utiliza a camada intermediária ilustrada na Figura 4.1 para interceptar o método que interage com o conteúdo ubíquo, gerando um primeiro *checksum*. A execução do método é prosseguida, seguida de uma nova verificação de alteração na estrutura do conteúdo (*checksum*). Caso os *checksums* sejam diferentes, isto é, o método alterou a estrutura do conteúdo, o C3S gera um evento para os observadores indicando que a estrutura do conteúdo ubíquo foi modificada. Esses observadores, que geralmente correspondem à representação gráfica, são notificados dessa modificação e utilizam os atributos do conteúdo ubíquo para atualizar sua representação.

Conteúdos ubíquos não podem referenciar diretamente uns aos outros. Caso isso fosse permitido, ao mover ou apagar um conteúdo ubíquo essa referência estaria nula e poderia tornar o conteúdo inconsistente. Para contornar esse problema, a plataforma C3S utiliza o conceito de ligação (*link*) para vincular e acessar um conteúdo ubíquo. Um conteúdo ubíquo A ligado a outro B por uma ligação L pode utilizar operadores da ligação, como $L.getEndpoint()$ para acessar o conteúdo B. É importante observar que o acesso para recuperar o conteúdo ubíquo B na ligação é o mesmo quando o conteúdo ubíquo está no mesmo local de A ou quando foi transferido para outro usuário, ou seja, a ligação fornece transparência de acesso para o conteúdo ubíquo. Além disso, a ligação possui um ciclo de vida que representa os estados de estabelecimento e quebra da ligação,

chamados por *L.onConnect*(), *L.onDisconnect*(). Esses estados podem ser utilizados para identificar e resolver o problema de interconectar um conteúdo ubíquo envolvido em uma ligação. Por exemplo, ao apagar um conteúdo ubíquo B envolvido na ligação $A \rightarrow B$, a ligação L é notificada e o conteúdo A pode tomar uma decisão baseado com base em L. Um exemplo prático de ligação é o relacionamento do conteúdo ubíquo de apresentação e de comentário. A apresentação está ligada a diversos comentários e quando comentários são apagados, as ligações da apresentação são notificadas para que sejam removidas da apresentação.

4.3 Arquitetura

Utilizando a sistematização de conteúdo ubíquo, a plataforma de *middleware* C3S provê suporte para os serviços de compartilhamento utilizando uma série de gerentes específicos, que colaboram entre si e formam a arquitetura do C3S. A arquitetura proposta para o C3S é híbrida, onde os gerentes se comportam de duas maneiras, par-a-par (*peer-to-peer*) e cliente-servidor. O primeiro comportamento dos gerentes da arquitetura, par-a-par, é destinado à interação entre os usuários presentes, enquanto que o segundo comportamento é destinado à administração do ambiente de computação ubíqua. Os dispositivos dos usuários que possuem ativações de aplicações ubíquas executando sobre o C3S são ao mesmo tempo clientes e fornecedores de conteúdos ubíquos e, portanto, utilizam uma arquitetura par-a-par para interagirem entre si. Esses dispositivos interagem com os serviços do ambiente de computação ubíqua utilizando o estilo arquitetural cliente-servidor. A parte servidora, por sua vez, é responsável por fornecer serviços para armazenar aplicações e informações pertinentes ao ambiente, tais como endereço lógico dos usuários e um diretório das aplicações ubíquas em execução. Uma visão geral da disposição dos *hosts* e da infraestrutura do C3S no ambiente de computação ubíqua é ilustrada na Figura 4.3.

A arquitetura do C3S é divida em unidades lógicas representadas por gerentes, onde cada um implementa uma funcionalidade específica do compartilhamento de conteúdo ubíquo. A ideia de separar cada funcionalidade do compartilhamento em um gerente específico visa tornar a arquitetura do C3S modular e flexível ao isolar a implementação dos conceitos em um determinado gerente, ou seja, cada gerente fornece um serviço que sustenta as abstrações que a plataforma C3S fornece. Os gerentes situam-se entre a aplicação ubíqua e sobre o ambiente de execução da plataforma do dispositivo, bem como sobre um *middleware* de comunicação, como ilustrado na Figura 4.4. O *middleware* de comunicação é utilizado para a troca de mensagens entre as diversas instâncias da plataforma (entre os pares e com o servidor). As semânticas de compartilhamento de conteúdo ubíquo são implementadas inteiramente utilizando interações entre os gerentes do

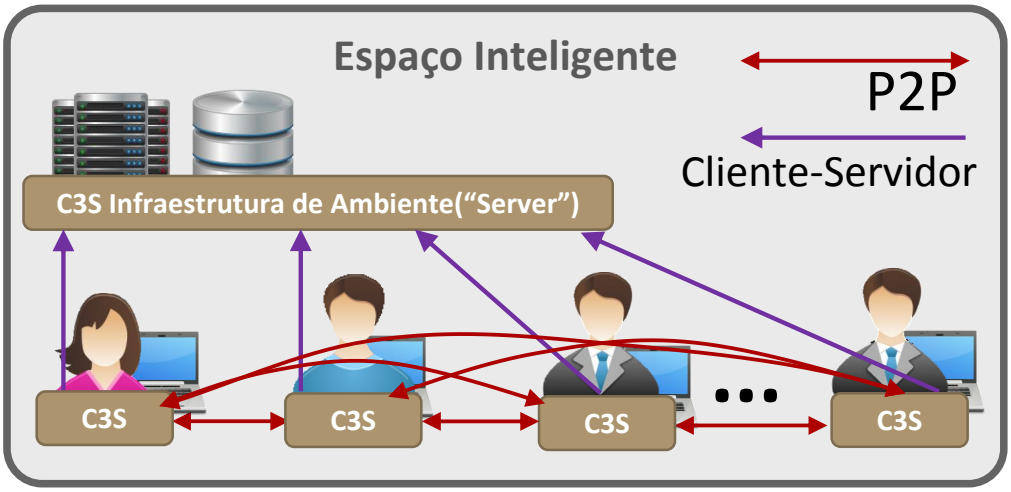

Figura 4.3: *Visão geral da disposição dos hosts e da arquitetura do C3S no espaço inteligente.*

middleware C3S, ou seja, elas não envolvem requisições específicas ao *middleware* de comunicação ou à plataforma de execução do usuário. Como as primitivas de manipulação de conteúdo ubíquo são expressas via interação de alto-nível de gerentes, o C3S pode ser implementado sobre diferentes sistemas de *middleware* de comunicação, incluindo, objetos distribuídos, *middleware* orientado a mensagens e espaço de tuplas. É interesse notar que a arquitetura do C3S é a mesma para os usuários e para a infraestrutura do ambiente. O que muda é apenas o comportamento e a funcionalidade provida pelos gerentes de em cada caso. Por exemplo, o gerente de conteúdo dos usuários lida com o armazenamento e relacionamento de conteúdos ubíquos do usuário, enquanto o servidor possui um papel administrativo ao lidar com o controle de conteúdos espelhados no ambiente.

As primitivas oferecidas às aplicações ubíquas são implementadas pelos gerentes do *middleware* (conteúdo, usuário, e aplicação). Essas primitivas realizam diversas interações entres os gerentes do C3S. De maneira geral, esses gerentes devem realizar as seguintes tarefas:

- **Gerente de conteúdo**: Fornece uma interface de manipulação de conteúdo ubíquo para as ativações. Essa interface inclui métodos para criar, apagar, ligar, mover, clonar e espelhar itens de conteúdo ubíquo. Coordena o processamento destas operações e gerencia a vinculação de itens de conteúdo ubíquo com ativações.
- **Gerente de usuários**: Gerencia a entrada e saída de usuários no ambiente. Fornece métodos convencionais, *i.e.*, cadastro, remoção, *login* de usuários, além de métodos específicos, tais como listar os usuários presentes no ambiente. Além das funções convencionais, esse componente fornece mecanismos para agrupamento dos usuários em grupos para cada aplicação ubíqua, permitindo criar e destruir grupos, bem como apagar, entrar e sair de grupos.

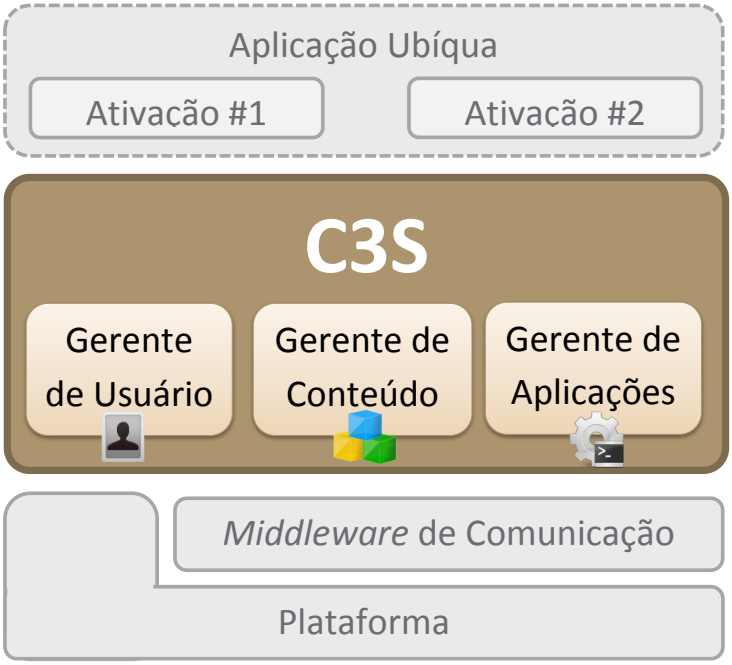

Figura 4.4: *Visão geral da arquitetura do C3S.*

- **Gerente de aplicações**: Lida com aspectos de distribuição e gerência das aplicações e ativações. Na parte de distribuição, fornece operações para criar, apagar e listar as aplicações ubíquas disponíveis no ambiente. Gerencia o ambiente de execução das aplicações através de operações para criar ativações e permitir que elas saiam e entrem de aplicações ubíquas.

O restante desta seção descreve sucintamente as funções de cada um desses gerentes para, em seguida, descrever a interação entre eles que habilita a implementação das primitivas do C3S.

4.3.1 Gerente de conteúdo

O gerente de conteúdo fornece a principal interface de manipulação de conteúdo ubíquo do *middleware* C3S. Essa interface inclui operações para criar, apagar, ligar, mover, clonar e espelhar conteúdos ubíquos. A primitiva de criar conteúdo ubíquo possibilita que um conteúdo ubíquo seja instanciado e manipulado no espaço inteligente pelos usuários presentes. O gerente de conteúdo deve registrar o conteúdo ubíquo para que ele possa coordenar suas atividades e relacionamentos. Durante a execução dessa primitiva, o gerente utiliza mecanismos de reflexão para criar uma camada que envolve o conteúdo ubíquo e que age com um *proxy*. Além dessa camada apresentar todas as funções do conteúdo ubíquo alvo, ela também possui funções administrativas que possibilitam administrar o conteúdo ubíquo. Essa camada age como um interceptador do *middleware*,

que o possibilita controlar, entre outras coisas, o acesso e a consistência do conteúdo ubíquo.

As ligações (*link*) com outros conteúdos ubíquos também são gerenciadas por esse componente. Essas ligações criam um grafo de relacionamentos que permite compor conteúdos complexos. A utilização das ligações como estruturas intermediárias permite que o acesso ao conteúdo ubíquo se torne transparente ao desenvolvedor e que o problema de interconectar conteúdo ubíquo seja tratável pelo *middleware*. Por exemplo, uma ligação L de um conteúdo ubíquo A para outro conteúdo B, representada por $A \rightarrow B$, é mantida após mover B, visto que o acesso se dá por $L.getEndPoint()$. Sob essa perspectiva, o papel do gerente de conteúdo é auxiliar as ligações a tornar o acesso ao conteúdo ubíquo transparente. Para isso, o gerente cria entradas na ligação, do tipo $origem \rightarrow destino$, que mapeiam a origem de um conteúdo ubíquo ao seu determinado destino, de maneira que seja possível recuperar ou redirecionar as chamadas ao conteúdo ubíquo alvo (seja utilizando *proxies/stubs* de objetos distribuídos, eventos, ou outro mecanismo).

4.3.2 Gerente de usuários

O gerente de usuários é responsável por fornecer abstrações da localização e perfil dos usuários presentes no ambiente de computação ubíqua. Esse componente oferece diversas funções convencionais, como entrar, sair, listar, cadastrar e apagar usuários no ambiente de computação ubíqua. A infraestrutura do ambiente de computação ubíqua pode utilizar diferentes sistemas para validar as informações dos usuários, dentre eles: banco de dados, *OpenID*, *Facebook*, ou uma infra-estrutura local. Após o usuário entrar no ambiente e ser validado pelo sistema, o servidor deve criar um endereço lógico para ele. Esse endereço possibilita encontrar o usuário no espaço inteligente. O gerente também deve fornecer operações para encontrar os outros usuários presentes no ambiente com base em seu endereço lógico. Essa funcionalidade eleva o nível de abstração do *middleware* e permite que as aplicações compartilhem conteúdo tendo como alvo nomes de usuários e não com endereços específicos.

O gerente de usuários fornece mecanismos que permitem agrupar os usuários das aplicações em grupos. Para cada aplicação ubíqua executando no ambiente, o gerente de usuários permite criar, listar, apagar, entrar e sair de grupos. O conceito de grupo é dependente da aplicação, e varia de aplicação para aplicação, mas a utilização dessas funções permite que o desenvolvedor agrupe os usuários e somente acrescente a semântica necessária para especificar o agrupamento. Por exemplo, as funções para criar e entrar em um determinado grupo, apesar de genéricas, podem assumir um contexto e papel específicos na aplicação no agrupamento de alunos de uma apresentação de *slides* ou de uma reunião. Além disso, grupos podem ser utilizados para implementar escopos

de compartilhamento. Mais especificamente, o gerente de usuários possibilita que as primitivas de compartilhamento sejam realizadas considerando os usuários dos grupos. Nesse caso, a primitiva é realizada para cada usuário pertencente ao grupo. Por exemplo, ao mover um conteúdo ou ao clonar um conteúdo ubíquo em um grupo, todos os usuários do grupo receberão aquele conteúdo ubíquo.

4.3.3 Gerente de aplicações

O gerente de aplicações fornece abstrações para lidar com a execução, agrupamento e organização das aplicações ubíquas no espaço inteligente. No âmbito de execução, este componente fornece funções para criar, apagar, entrar, sair e listar aplicações ubíquas disponíveis no ambiente. Ao criar uma aplicação ubíqua, a infraestrutura do espaço inteligente deve alocar recursos para controlá-la no ambiente e disponibilizá-la aos usuários. Entre os recursos alocados pelo gerente, encontram-se, tipicamente, metadados tais como o identificador da aplicação ubíqua, o número de usuários máximo e as plataformas suportadas.

Após a aplicação ubíqua ser registrada no servidor, o gerente de usuários a lista no ambiente e possibilita que usuários do ambiente criem ativações da aplicação. As ativações são instâncias locais de cada usuário, que executam a mesma aplicação conforme visto na Seção 3.3. O gerente fornece mecanismos para que essas instâncias entrem e saíam da aplicação ubíqua. A aplicação ubíqua é a estrutura lógica que unifica as ativações e permitem que elas troquem conteúdo. Para receber conteúdos ubíquos, as ativações possuem métodos que as notificam, como $activation.onReceivedMoveContent()$, $activation.onReceivedCloneContent()$ e $activation.onReceivedMirrorContent()$. O gerente de conteúdo cria eventos que sinalizam e ativam cada um desses métodos. Além do conteúdo ubíquo, esse evento inclui como parâmetro o remetente da primitiva, de forma que permita identificar o usuário que iniciou a ação de colaboração. Essa informação pode ser utilizada pela aplicação para representar a origem do conteúdo ubíquo compartilhado.

4.4 Primitivas

As interações para implementação das primitivas de compartilhamento de conteúdo ubíquo são descritas utilizando uma série de chamadas de funções dos gerentes do C3S. Inicialmente, descrevemos o funcionamento das primitivas de compartilhamento de mover e clonar. Ambas são parecidas por não terem efeitos contínuos, isto é, elas não requerem o estabelecimento de um vínculo contínuo de compartilhamento entre os usuários envolvidos na operação. Por exemplo, após mover ou clonar um conteúdo ubíquo, existe pouca troca de informações entre os conteúdos resultantes, a qual se restringe a

informações para manter as ligações. Por outro lado, ao espelhar um conteúdo, é necessário manter o compartilhamento contínuo entre as réplicas do conteúdo envolvidas para mantê-las atualizadas. Outra semelhança é na semântica, visto que a primitiva de mover remove o conteúdo ubíquo da ativação enquanto a de clonar não, ou seja, a primitiva de clonar pode ser vista como uma versão simplificada de mover que não requer lidar com ligações inconsistentes. O processo de mover um conteúdo ubíquo para um grupo é idêntico ao de mover para cada usuário pertencente àquele grupo, ou seja, quando a primitiva recebe como parâmetro um grupo, ela executa o processo múltiplas vezes, com a diferença de que o conteúdo só é removido ao final da operação. A Figura 4.5 ilustra o processo necessário para mover um conteúdo ubíquo, que pode ser descrito por meio dos passos seguintes, aqui mostrados na forma de um exemplo:

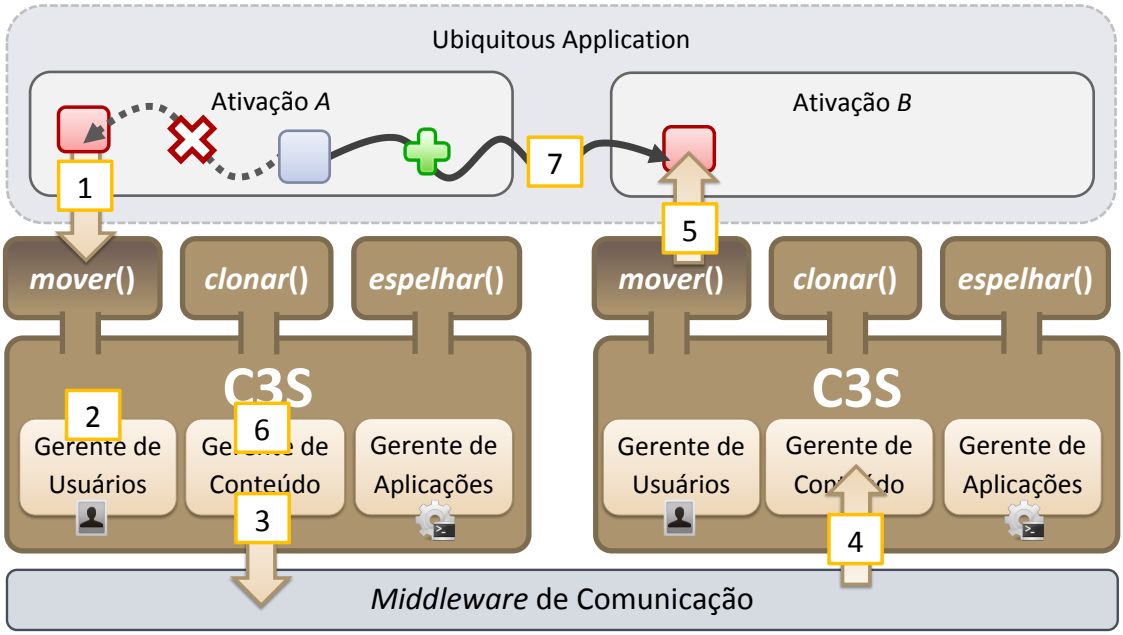

Figura 4.5: *Fluxo de interação dos componentes para a primitiva de mover.*

1. A ativação *A* da aplicação executa a primitiva *mover(ref, marcos)* do gerente de conteúdo, onde *ref* é a referência ao item de conteúdo ubíquo sendo transferido e *marcos* é o usuário alvo;
2. O gerente de conteúdo utiliza o serviço de tradução de nomes do gerente de usuários para referenciar o usuário *marcos*;
3. O gerente de conteúdo remove e desvincula o conteúdo ubíquo do espaço de armazenamento da ativação. Além disso, o gerente utiliza a camada intermediária do conteúdo ubíquo para extrair uma representação de dados externa e travar o seu acesso durante a operação. Ele então combina a representação externa com

metadados, que incluem informações sobre a ativação do usuário alvo (*marcos*) e o tamanho em *bytes* do conteúdo. Esses dados são encapsulados em uma mensagem, que é enviada pelo *middleware* de comunicação subjacente;

4. A instância do C3S do usuário alvo recebe a mensagem do *middleware* de comunicação que contém a representação do conteúdo ubíquo e seus metadados. Utilizando essa representação externa o conteúdo ubíquo é reconstruído e vinculado ao armazenamento da ativação local do usuário alvo;
5. Utilizando os metadados contidos na mensagem, o gerente de conteúdo interage com o gerente de aplicações para recuperar a ativação da aplicação correspondente ao usuário alvo. Ao recuperar a ativação, o gerente sinaliza um evento de notificação de recebimento de conteúdo ubíquo para a ativação do usuário alvo, esse evento gera uma chamada ao método $activation.onReceivedMovedContent(content, user)$ da ativação. Além do conteúdo ubíquo, o evento inclui o destinatário como parâmetro, possibilitando que a ativação identifique o usuário que iniciou a ação de compartilhamento.
6. Depois do conteúdo ubíquo ser transferido, o gerente de conteúdo de origem o remove de seu armazenamento;
7. Para lidar com possíveis referências inconsistentes, o gerente verifica o grafo de relacionamentos do conteúdo ubíquo manipulado. O conteúdo ubíquo no exemplo possuía uma ligação com outro conteúdo ubíquo. Portanto, o gerente de conteúdo ativa um intermediador para interceptar as futuras interações com esse conteúdo ubíquo e redirecioná-las ao seu novo destino.

A primitiva de espelhar, por sua vez, provê um efeito contínuo no compartilhamento de conteúdo ubíquo. Ela realiza a cópia do conteúdo ubíquo e, em seguida, estabelece um vínculo de sincronismo contínuo entre as cópias do mesmo conteúdo existentes no espaço inteligente. A primeira parte do funcionamento da primitiva realiza a clonagem do conteúdo ubíquo, acrescentando o fato de que o endereço da réplica é registrado no servidor. O C3S utiliza uma abordagem pessimista para controlar o acesso a conteúdos. Essa abordagem prevê que para evitar conflitos e diferença entre as réplicas o acesso ao conteúdo deve ser controlado. Para realizar isso, utilizamos o padrão de projeto "Atualização Mediada" (*Mediated Update*) [35] para realizar o processamento e sincronização que permitem que essas réplicas fiquem sincronizadas. Esse padrão de projeto utiliza um servidor para centralizar o acesso e controle do conteúdo e despachar suas modificações para cada réplica. A segunda parte da primitiva gerencia e executa esse padrão, ou seja, o servidor armazena filas com endereços de réplicas dos conteúdos ubíquos para poder sincronizar as modificações. As interações com conteúdos ubíquos sincronizados são interceptadas pelo C3S, na camada de acesso intermediário (*proxy*), com o objetivo de sincronizar as ações nas outras réplicas. Após a interceptação, a operação e seus respectivos argumentos são

empacotados em uma requisição ao servidor. O servidor então despacha uma requisição dessa operação para cada réplica, ou seja, apenas a ação é replicada, ao invés de replicar todo o estado. A escolha dessa estratégia se deve ao fato de que mover o conteúdo em si envolveria uma grande quantidade de dados. No padrão atualização Mediada, por outro lado as chamadas (interações) são replicadas e não o estado do conteúdo ubíquo. A Figura 4.6 mostra o fluxo de interações entre os gerentes do C3S para despachar e sincronizar um método em um conteúdo ubíquo espelhado. As etapas desse processo são:

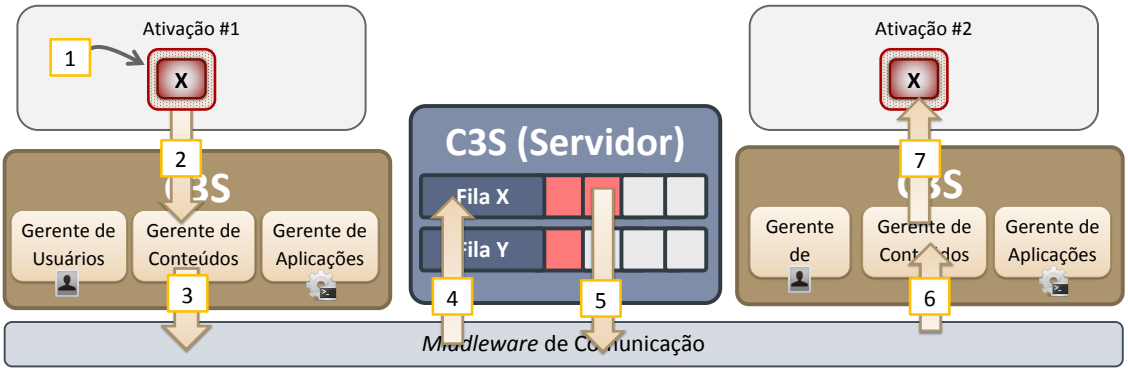

Figura 4.6: *Fluxo de interação dos componentes na execução de um método em um conteúdo ubíquo sincronizado.*

1. A ativação interage com o conteúdo ubíquo *X* através de uma chamada de método.
2. O C3S intercepta este método através do uso da camada intermediária do conteúdo ubíquo (*proxy*);
3. O gerente utiliza reflexão para identificar e serializar a chamada de método, junto com seus argumentos em uma requisição com destino ao servidor. Adiciona-se alguns metadados a essa mensagem, que incluem os identificadores da ativação, do usuário e do conteúdo ubíquo. Esses identificadores permitem realizar o *matching* nos servidores e nos usuários para processar os eventos;
4. O servidor recebe a requisição e utiliza o identificador do conteúdo ubíquo para recuperar a fila de réplicas onde essa chamada deve ser executada.
5. Para cada réplica cria-se uma requisição contendo a operação a ser executada, juntamente com os argumentos e metadados para cada réplica do conteúdo. Essas requisições são despachadas ao *middleware* de comunicação;
6. Cada réplica recebe essa requisição. O gerente de conteúdo desempacota a requisição e recupera a réplica local do conteúdo ubíquo utilizando o seu identificador;
7. Por fim, o gerente de conteúdo desempacota e reconstrói o método a ser executado e o executa sobre o conteúdo ubíquo;

Os conteúdos ubíquos possuem uma propriedade lógica (verdadeira ou falsa) denominada *onlyOwner*, que indica se o conteúdo pode ser compartilhado por outros

usuários. Essa propriedade permite especificar que um determinado conteúdo ubíquo seja manipulado apenas pelo seu dono. Utilizando essa propriedade, no caso do espelhamento, as alterações sofridas nas réplicas do conteúdo podem não ser propagadas para as outras cópias como exemplificado a seguir. O C3S verifica se essa propriedade está ativa ao realizar a interceptação da interação do conteúdo ubíquo na sua camada de acesso intermediário. Caso a propriedade seja verdadeira e o usuário que está manipulando o conteúdo não seja o dono a interação não será propagada para as demais réplicas.

Essa operação utiliza uma abordagem pessimista para lidar com o modelo de consistência dos conteúdos ubíquos. A abordagem da operação especifica a requisição de uma trava (*lock*) no conteúdo ubíquo antes dele ser modificado. A trava é controlada pelo servidor (controlador), que libera o acesso a apenas um cliente por vez (pessimista). Apesar dessa limitação, a abordagem garante a consistência das réplicas pois apenas um cliente modifica por vez.

4.5 Detalhes de Implementação

A plataforma de *middleware* C3S foi implementada utilizando a linguagem de programação Java e o *middleware* de comunicação Java RMI. A implementação contempla um subconjunto do modelo proposto nessa dissertação. Foram implementados os conceitos de conteúdo ubíquo, ativação, aplicação ubíqua e aplicação, além de todas as primitivas de compartilhamento do modelo. Entretanto, não foi implementado os mecanismos de ligação entre conteúdos ubíquos, de tal maneira que ao mover um conteúdo a referência da ligação não é atualizada remotamente para o destino do conteúdo. Apesar dessa limitação, o subconjunto implementado é autocontido e permite avaliar os principais conceitos e primitivas de compartilhamento do modelo proposto.

Para implementar a infraestrutura de nomes do C3S utilizamos um banco de dados local para autenticação de usuários no espaço inteligente. A inserção de um usuário no espaço inteligente, portanto, é feita utilizando a operação de *login* no gerente de usuários do servidor. Havendo êxito nesta operação, é registrado ou atualizado o endereço lógico do usuário. Esse endereço é usado para mapear o nome do usuário ao seu endereço físico. O endereço permite comunicar diretamente com os gerentes do usuário. Para realizar isso, decidimos criar uma entrada no servidor de registro do RMI com o nome do usuário, seguido de cada gerente para endereçar cada componente, como ilustrado na Figura 4.7. Dessa maneira, é possível interagir diretamente com o gerente de cada usuário, facilitando a construção e processamento das primitivas nesse cenário.

Ao iniciar uma aplicação é criada uma ativação utilizando o gerenciador de aplicações. Essa ativação é adicionada à sua respectiva aplicação ubíqua e caso esta não exista ela também é criada (no Servidor). Também é alocado um espaço no gerente de

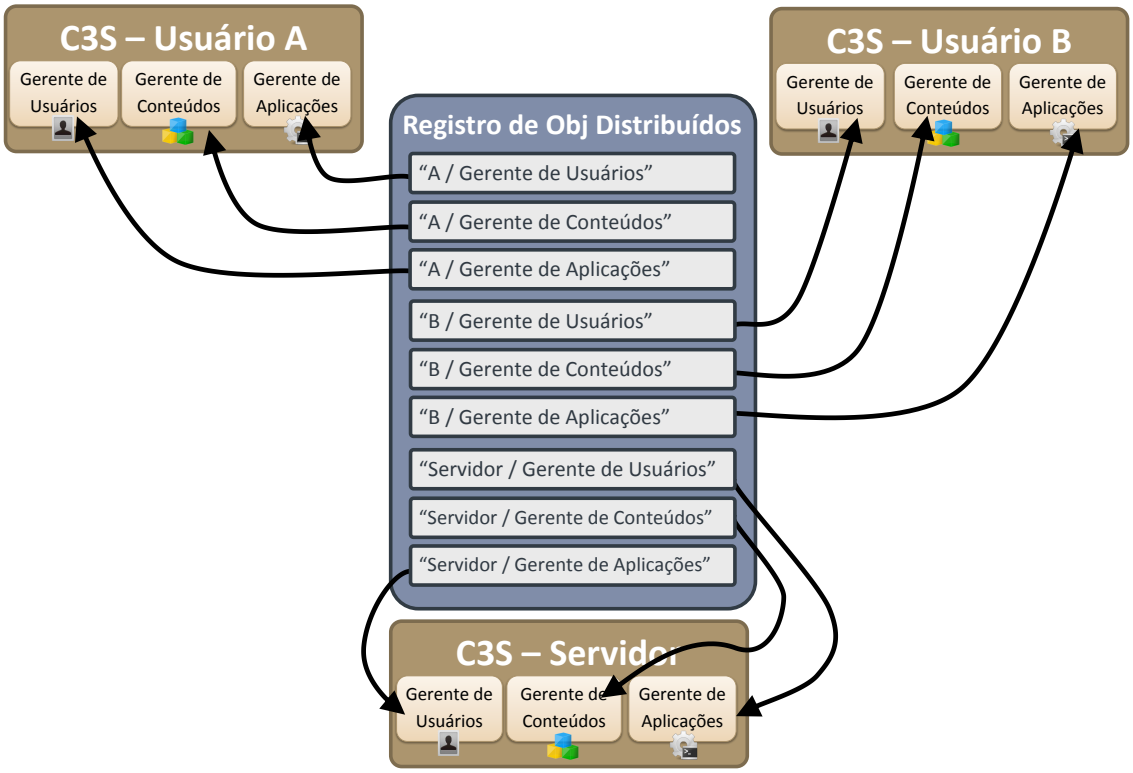

Figura 4.7: *Registro de componentes do usuário no ambiente.*

conteúdo para o armazenamento dos itens de conteúdo ubíquo criado na ativação. Na implementação, utilizamos uma tabela *hash* onde a chave é o identificador da ativação e a saída é o seu conjunto de conteúdos ubíquos armazenado.

O formato base de conteúdo ubíquo é definido pela classe abstrata *UbiquitousContent*, mostrada na Figura 4.8. Esta classe define seus metadados, como ID, ativação, dono e estado de sincronização, que são utilizados pelas primitivas de compartilhamento. Conteúdos ubíquos são construídos através da operação *criarConteúdoUbíquo(contUbiq : ConteúdoUbíquo) : ReferênciaUbíqua*. Uma chamada desta operação adiciona e vincula o conteúdo ubíquo ao espaço de armazenamento da ativação e retorna uma referência "ubíqua" do conteúdo. Essa referência encapsula o conteúdo ubíquo, agindo como um *proxy* e ponto de interceptação do *middleware*. O desenvolvedor utiliza a operação *get()* dessa referência para recuperar o conteúdo ubíquo. A referência ubíqua também controla o acesso ao conteúdo, de maneira a torná-lo consistente para ser compartilhado. Por exemplo, durante a execução de um método do item de conteúdo ubíquo, ele entra no estado inconsistente, isto é, está sendo modificado pela operação e, portanto, não pode ser compartilhado (*movido* etc). Este controle garante consistência e isolamento ao conteúdo. Após a execução da operação o intermediador reabilita o estado de compartilhamento do conteúdo.

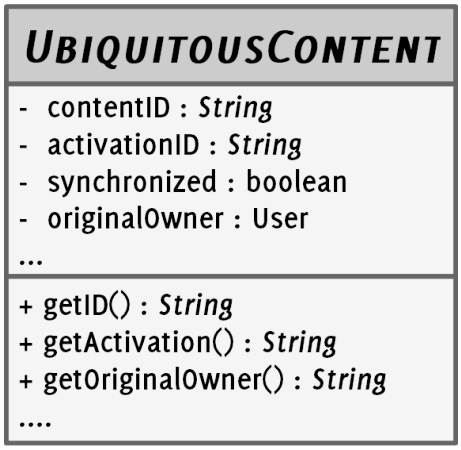

Figura 4.8: *Classe ConteúdoUbíquo.*

4.6 Modelo de Programação

Como mencionado ao longo desta dissertação, o modelo de programação do C3S fornece mecanismos para a manipulação de conteúdo ubíquo, ativações e aplicações ubíquas. Todos esses conceitos são representados como entidades de primeira classe no C3S, o que significa que a construção de aplicações gira em torno deles. Uma aplicação ubíqua é definida como um conjunto de instâncias, denominadas ativações. A aplicação ubíqua é criada no servidor por meio da definição da aplicação (binário), seu título e identificador. Após a aplicação ubíqua ser registrada no servidor, o gerente de aplicações possibilita que usuários do ambiente criem ativações para essa aplicação. Essas ativações podem ser organizadas em grupos para possibilitar que a aplicação ubíqua defina ou configure mecanismos de agrupamento de usuário. Por exemplo, pode ser interessante para a aplicação de apresentação *slides* utilizar a ideia de agrupamento para definir que o compartilhamento possa ser efetivado apenas entre usuários de um dado grupo. Observe que essas restrições devem ser especificadas pela própria aplicação por estarem relacionadas ao escopo de seu domínio. A Figura 4.9 ilustra uma aplicação ubíqua composta por diversas ativações organizadas em dois grupos.

Apesar das ativações de uma aplicação ubíqua serem instâncias da mesma aplicação, cada uma (ativação) possui um estado diferente, que é gerenciado pelo C3S. Cada ativação possui um ciclo de vida simples para denotar os estados de início, termino e execução, representados pelos métodos $activation.onStart()$, $activation.onDestroy()$ e $activation.onCreate()$ respectivamente. Esses estados são ativados e desativados pelo gerente de aplicações que administra a execução da ativação e aplicação ubíqua. No primeiro estado da ativação, $onStart()$, o desenvolvedor deve inicializar os recursos da aplicação, tipicamente os conteúdos ubíquos. Já no estado de término, $onDestroy()$, a ativação tem um último momento para destruir os seus recursos. A execução da ativação

Figura 4.9: *Agrupamento de ativações de uma aplicação ubíqua.*

se dá pelo estado *onCreate*(), onde o compartilhamento de conteúdo ubíquo é efetuado. Durante a execução da ativação são criados e compartilhados diversos conteúdos. A seguir discutimos o modelo de especificação e criação de conteúdo ubíquo no C3S.

As plataformas de *middleware* existentes para compartilhamento de conteúdo [7, 23, 27, 34] limitam o compartilhamento a uma única semântica de compartilhamento embutindo o tipo de compartilhamento na especificação do conteúdo ubíquo. A semântica de compartilhamento (mover, clonar, espelhar) é determinada no tipo do conteúdo ubíquo, ou seja, qualquer instância de um tipo de conteúdo ubíquo só poderá ser compartilhada conforme a semântica fixa do tipo. Por exemplo, ao codificar a classe de uma anotação de *slides* nessas plataformas e especificá-la com a semântica de espelhamento, todas as instâncias de anotações só poderão ser espelhadas, ou seja, as anotações não poderão ser simplesmente movidas ou clonadas.

O C3S sistematiza o conceito de conteúdo ubíquo como uma especificação de uma classe abstrata, *UbiquitousContent*. A classe do conteúdo ubíquo deve herdar dessa classe, que inclui propriedades, identificadores e funções administrativas que possibilitam que conteúdo seja ao mesmo tempo específico para o programador e opaco ao *middleware*. Utilizando as funções e propriedades dessa classe, juntamente com a camada que envolve o conteúdo ubíquo, o C3S pode compartilhar o mesmo conteúdo ubíquo utilizando diferentes semânticas. A operação de criar conteúdo ubíquo instancia uma dessas classes específicas. Durante a operação de instanciação a camada que envolve o conteúdo ubíquo é gerada e as propriedades da classe *UbiquitousContent* são preenchidas, como o identificador de usuário e da ativação, para que ele possa ser compartilhado. O compartilhamento das primitivas se dá utilizando as primitivas de *mover*(), *clonar*(), e *espelhar*(), fornecidas pelo gerente de conteúdo. Durante a execução da primitiva a camada intermediária bloqueia o acesso a outras operações e primitivas sobre o conteúdo ubíquo afim de manter a integridade conteúdo na operação de compartilhamento.

Para referenciar um conteúdo ubíquo utiliza-se o conceito de ligações (*links*). Uma ligação L de um conteúdo ubíquo A para outro B, representado por $A \rightarrow B$,

permite ao conteúdo ubíquo A acessar o conteúdo B. A ligação é criada utilizando referências aos conteúdos de origem e alvo. Por exemplo, o comando, *Link ligação* $= contentManager.createLink(A,B)$, ativa o gerente de conteúdo para criar a ligação $A \rightarrow B$. Utilizando esta ligação, o desenvolvedor pode recuperar o conteúdo B através do método $link.getEndpoint()$, que torna o acesso a B independente de sua localização. A estrutura de ligação pode utilizar métodos de notificação, como $L.onEndpointMove()$ e $L.onEndpointDisconnect()$ para facilitar sua implementação. O primeiro estado interno exemplo, $L.onEndpointMove()$, seria atingido ao perceber a movimentação do conteúdo ubíquo alvo. Isso poderia ocorrer utilizando inscrição e recebimento de eventos do conteúdo ubíquo. Esses eventos seriam gerador pela instância do conteúdo ubíquo alvo em sua camada de interceptação.

No próximo capítulo, é apresentada uma avaliação de como esta plataforma e o seu modelo de programação podem ser utilizados para construir aplicações a partir "do zero", assim como portar aplicações colaborativas existentes para o ambiente de computação ubíqua. O capítulo discute a implementação de duas implementações, de maneira a exemplificar o uso dos conceitos de primitivas de compartilhamento, ativações e aplicações ubíqua.

4.7 Sumário e Considerações Finais

Este capítulo apresentou a arquitetura de *middleware* proposta nesta dissertação, denominada **C**ontent **S**haring for **S**mart **S**paces (C3S), para implementar as ideias de compartilhamento de conteúdo ubíquo em ambientes de computação ubíqua (mais especificamente, espaço inteligentes) descritas nos capítulos anteriores. Essa arquitetura apresenta e materializa os conceitos e mecanismos de compartilhamento ubíquo vistos anteriormente, incluindo conteúdo ubíquo, primitivas e aplicações ubíquas.

O capítulo apresentou o conjunto de requisitos utilizados para guiar o projeto arquitetural da plataforma C3S. Esses requisitos incluem:

1. Fornecimento de um conjunto de primitivas para manipulação de conteúdo;
2. Múltiplas formas de compartilhamento para um mesmo conteúdo ubíquo;
3. Compartilhamento direto entre usuários;
4. Consistência e integridade no compartilhamento; e
5. Interação entre grupos;

O primeiro requisito visa fornecer várias semânticas de manipulação de conteúdo ubíquo no espaço inteligente. Mais especificamente, as semânticas de mover, clonar e espelhar. O segundo requisito enuncia que deve ser possível utilizar diferentes primitivas para manipular o mesmo conteúdo ubíquo, ou seja, possibilitando que um mesmo

conteúdo seja compartilhado de diferentes formas. O terceiro e quarto requisito visam possibilitar o compartilhamento direto e consistente entre os usuários presentes no ambiente. Por fim, o último requisito visa fornecer mecanismos para permitir interação e agrupamento dos usuários no ambiente de computação ubíqua.

Após enunciar esses requisitos, apresentamos a sistematização do conceito de conteúdo ubíquo. Discutimos a relevância de conteúdo ubíquo no modelo de programação do C3S e mais especificamente, o seu papel como entidade de primeira classe do modelo. Discutimos como as aplicações do C3S giram em torno da manipulação do conteúdo ubíquo. Em seguida descrevemos a realização do conceito de conteúdo ubíquo como uma extensão de uma classe abstrata *UbiquitousContent*, que inclui propriedades e comportamentos que permitem ao C3S processar conteúdo ubíquo de maneira opaca. O mecanismo de herança permite que o desenvolvedor especifique e desenvolva as propriedades e comportamentos específicos de cada tipo de conteúdo ubíquo. Uma das limitações dessa abordagem é que a vinculação causal entre a estrutura do conteúdo ubíquo e sua representação gráfica deve ser mantida explicitamente pelo desenvolvedor de aplicações. Durante a instanciação de um conteúdo ubíquo, é criada uma camada intermediária que o envolve. Essa camada reflete a interface do conteúdo ubíquo, agindo como um *proxy*, e adicionando funcionalidades administrativas que permitem ao *middleware* interceptar as interações com o conteúdo ubíquo. Por fim, apresentamos o conceito de ligação (*link*) como meio de um conteúdo ubíquo acessar outro utilizando a operação $link.getEndPoint()$. A ligação abstrai a localização do conteúdo ubíquo, permitindo que, mesmo após mover-se para outro local, o seu acesso seja idêntico.

Utilizando esses requisitos como modelo e a sistematização do conteúdo ubíquo, foi apresentada a arquitetura da plataforma de *middleware* C3S. A arquitetura utiliza uma abordagem híbrida, sendo par-a-par (*peer-to-peer*) para interações entre os usuários e cliente-servidor para interações com o ambiente. Os dispositivos dos usuários no ambiente fornecem e recebem conteúdo ubíquo. Portanto, eles utilizam a abordagem *peer-to-peer* primordialmente para interação, enquanto que, para utilizar os serviços de gerenciamento do ambiente, as aplicações utilizam a abordagem cliente-servidor. A arquitetura do C3S é composta por componentes que implementam e gerenciam uma determinada funcionalidade do compartilhamento ubíquo. Esses componentes situam-se sob a aplicação ubíqua e sobre um *middleware* de comunicação e a plataforma de execução dos dispositivos.

As semânticas de interação são implementadas por interações dos componentes do *middleware*, que incluem gerência de conteúdo, usuários e aplicações. Cada componente fornece abstrações específicas para lidar com compartilhamento de conteúdo. Por exemplo, o gerente de conteúdo oferece a interface principal para mover, clonar e espelhar conteúdo ubíquo, enquanto o gerente de aplicações possibilita criar e destruir ativações de

uma aplicação ubíqua. Como as primitivas são descritas por interações de componentes, é possível que elas sejam implementadas utilizando diferentes tecnologias de *middleware* de comunicação subjacentes.

As primitivas de compartilhamento do *middleware* são descritas utilizando interações entre componentes do C3S. As primitivas de mover e clonar são quase idênticas, com a diferença que a primitiva de clonar não envolve remover o conteúdo da origem. As interações e passos da primitiva de mover lidam diretamente com o trancamento e controle (*locking*) de acesso ao conteúdo ubíquo, a tradução do nome do usuário alvo para seu endereço, e a transferência e consistência do conteúdo ubíquo. Já a primitiva de espelhamento, provê um efeito contínuo no compartilhamento de conteúdo ubíquo. Ela realiza a cópia do conteúdo ubíquo e, em seguida, estabelece um vínculo de sincronismo entre as cópias no espaço inteligente. Essa primitiva utiliza um padrão de projeto, denominado "Atualização Mediada" (*Mediated Update*) [35], para realizar o processamento e sincronização das réplicas. Esse padrão de projeto centraliza o controle de acesso ao conteúdo ubíquo e o despacho das atualizações.

Como alternativa para manter a consistência entre as réplicas, [9] propõe uma solução otimista que utiliza algumas premissas sobre o compartilhamento para não trancar o acesso ao conteúdo. Entretanto, essas premissas requerem que o modelo de compartilhamento, isto é, como os conteúdos ubíquos são compartilhados, sejam definidos a priori no momento da especificação do conteúdo ubíquo. Apesar de aumentar a performance do compartilhamento em espelhamento a expressividade de como compartilhar o conteúdo é limitado por essa definição a priori.

Discutimos a implementação da plataforma de *middleware* C3S. Essa implementação utiliza a linguagem de programação Java e o *middleware* de comunicação Java RMI. Discutimos também como foi realizada a autenticação e a comunicação entre as instâncias do C3S. Também foi discutimos como é feita a criação de conteúdo ubíquo no C3S, além de apresentar o fato de que a implementação não suporta a funcionalidade de ligação entre conteúdos ubíquos. Por fim, discutimos o modelo de programação do C3S para construção de aplicações baseadas em compartilhamento de conteúdo ubíquo para ambientes de computação ubíqua. Apesar da escolha pelo Java RMI o C3S poderia ser implementado sobre outras plataformas de *middleware* de comunicação, como *middleware* orientado a eventos e espaço de tuplas.

No próximo capítulo, será avaliado como essa plataforma e o modelo de programação do C3S podem ser utilizados para construir aplicações "do zero" e portar aplicações colaborativas existentes para o ambiente de computação ubíqua. O capítulo demonstra a implementação de duas aplicações, de maneira a exemplificar o uso dos conceitos de primitivas de compartilhamento, ativações e aplicações ubíqua.

CAPÍTULO 5

Avaliação

O modelo de programação do C3S permite a construção de aplicações baseadas no conceito de conteúdo ubíquo. Este capítulo apresenta uma avaliação do *middleware* e do modelo proposto por meio de dois estudos de caso de desenvolvimento de aplicações. A primeira é uma aplicação que permite que palavras secretas sejam compartilhadas entre os alunos para que eles possam resolvê-las em conjunto. A segunda aplicação, *Apresentador de Slides Ubíquo*, é uma modificação de um leitor de *PDF* existente, que permite ao professor compartilhar os slides de uma aula entre os alunos, mantendo uma visão sincronizada com a apresentação em curso. A construção destas duas aplicações suscitou diversas questões e lições que são discutidas neste capítulo.

O restante deste capítulo está estruturado da seguinte forma: a Seção 5.1 descreve a metodologia utilizada para avaliar o modelo de programação do C3S. As seções 5.2 e 5.3 apresentam os dois estudos de caso utilizados na avaliação. A Seção 5.4 discute as limitações e lições aprendidas com este trabalho, enquanto, a Seção 5.5 apresenta o sumário do capítulo e as considerações finais.

5.1 Metodologia de Avaliação

As aplicações foram escolhidas com o intuito de explorar todas as primitivas de manipulação de conteúdo ubíquo, assim como o conceito de aplicação ubíqua e ativações fornecidos pelo *middleware*. A primeira aplicação, apresentada na Seção 5.2, explorou todas as primitivas de compartilhamento de conteúdo ubíquo apresentadas (mover, clonar, e espelhar) entre os usuários do ambiente. Esses usuários podem entrar e sair de grupos, o que afeta o compartilhamento, visto que a ação pode ser feita especificamente para um usuário ou com todos do grupo. A segunda aplicação, apresentada na Seção 5.3, foi baseada em uma aplicação pré-existente com intuito de avaliar o impacto de *portar* uma aplicação legada para o modelo de programação do C3S. Como o protótipo do C3S implementa um subconjunto do modelo de programação, ambas as aplicações não utilizam conteúdos ubíquos complexos, ou seja, conteúdos interligados com outros conteúdos.

Nessa avaliação queremos validar as seguintes características do modelo de programação proposto utilizando o protótipo do C3S:

- Modelagem das estruturas a ser compartilhadas em conteúdos ubíquos;
- Primitivas de compartilhamento (mover, clonar, espelhar); e
- Utilização de grupos (agrupamento de ativações) pela aplicação ubíqua.

Utilizamos um roteiro comum para avaliar essas características nas aplicações desenvolvidas no estudo de caso. Inicialmente apresentamos uma visão geral do funcionamento da aplicação, que inclui, objetivos, interações entre usuários, tipos de compartilhamento, e os conteúdos ubíquos que compõem a aplicação. Em seguida, as seções discutem a utilização do conceito de grupos na aplicação ubíqua, além das primitivas de compartilhamento que a aplicação realiza sobre os conteúdos ubíquos. Por fim, a seção avalia a construção das aplicações usando o *middleware*, em comparação com a implementação *ad hoc* do compartilhamento diretamente nas aplicações. Finalmente, são descritos e discutidos detalhes específicos de implementação de cada aplicação.

5.2 Palavras Secretas

A primeira aplicação desenvolvida no estudo de caso foi *Palavras Secretas*. Essa aplicação é destinada a crianças em fase de alfabetização e emprega uma metodologia simples que associa imagens com palavras. As letras da palavra devem ser descobertas pelas crianças utilizando a associação com a figura. Como as letras das palavras são inicialmente desconhecidas pelas crianças, elas são denominadas “palavras-desafios”. As palavras-desafio são geradas para cada aluno, que tem que acertar as letras para revelar a palavra por completo. Os alunos podem colaborar entre si para desvendá-las. A colaboração pode ser efetuada com o grupo do aluno ou com um aluno específico. A aplicação utiliza as primitivas de compartilhamento do *middleware* para dar suporte a colaboração. Por exemplo, se algum aluno do grupo sabe uma letra da sua palavra, ele pode mover a palavra-desafio para esse aluno. Um aluno também pode clonar uma palavra para tentar descobri-la em paralelo com outro aluno. Caso ele queira desvendar as letras da palavra ao mesmo tempo e colaborativamente, ele pode utilizar a primitiva de espelhamento. As Figuras 5.1 e 5.2 mostram, respectivamente, a tela da aplicação e sua execução em *tablet* PCs.

A palavra-desafio representa o item de compartilhamento nesta aplicação e, portanto, é modelada como um conteúdo ubíquo. Os atributos do conteúdo ubíquo *palavra-desafio* são a palavra original, a palavra correntemente exibida para o usuário, com alguns caracteres ocultos, uma dica e a imagem associada. A Figura 5.1 ilustra duas palavras-desafio, uma associada à abelha e outra a um balão. Os ícones delas

Figura 5.1: *Tela de Palavras Secretas.*

representam o atributo de imagem associada à palavra-desafio. Já os quadrados em verde representam as letras descobertas, enquanto que os quadrados amarelos são as dicas da palavra. Por exemplo, a palavra-desafio representada pelo ícone de abelha contém a palavra encoberta (`"A?EL?A"`) e a dica da letra `"L"` na quarta posição. Os conteúdos ubíquos também possuem operações específicas, que possibilitam alterar ou recuperar seu estado. No caso das palavras-desafios há operações para inserir uma letra (*putChar()*), obter uma dica (*getHint()*), recuperar a palavra-desafio (*getSecretWord()*) e recuperar a palavra descoberta até o momento (*getCurrentWord()*). Essas operações são utilizadas para manipular e alterar o estado dos conteúdos ubíquos. Por exemplo, a operação de inserção de letra (*putChar()*) verifica se a letra sugerida pelo usuário em uma determinada posição corresponde à letra da palavra-desafio. Em caso afirmativo, a letra é inserida na palavra descoberta.

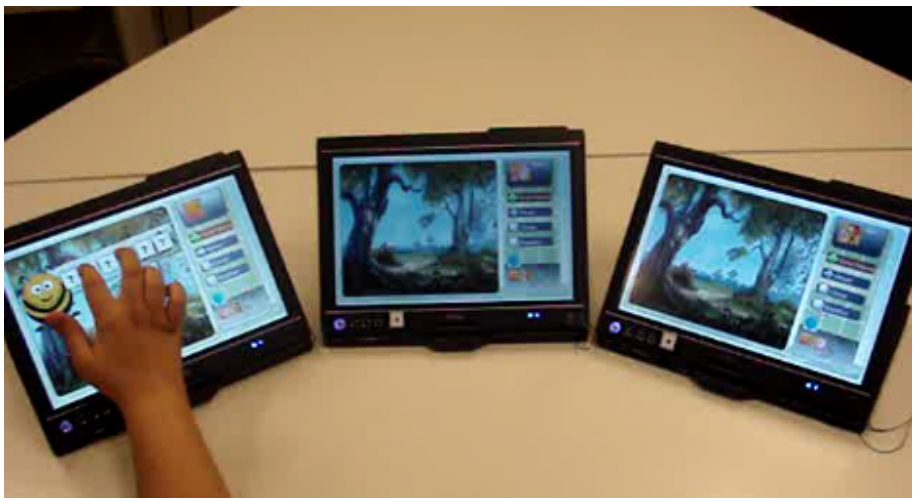

Figura 5.2: *Tablets PC executando Palavras Secretas.*

Um grande desafio encontra-se na separação da parte lógica da parte gráfica na modelagem dos conteúdos ubíquos. A separação é necessária porque a representação gráfica liga-se a componentes nativos do sistema (como *buffers* do OpenGL), que são vinculados à plataforma específica do usuário o que dificulta seu compartilhamento. Para evitar este problema, um conteúdo ubíquo deve conter apenas atributos que sejam independentes de sua representação gráfica. Cabe a uma ativação mapear o conteúdo para sua respectiva representação gráfica em uma plataforma em particular. Por exemplo, as palavras-desafio possuem o atributo *ícone*, que especifica a imagem a ser associada. A atribuição `ícone = "banana"` indica que o ícone a ser associada à palavra deve ser o de uma banana. A aplicação compreende o que é "banana" e isto permite a ela associar este conteúdo ubíquo à sua representação gráfica.

No C3S, o vínculo do conteúdo ubíquo para a representação gráfica é mantido utilizando o padrão de projeto Observador [19]. A aplicação cria observadores que se inscrevem em eventos de alteração da estrutura do conteúdo ubíquo. O processamento e geração desses eventos é realizado na camada intermediária do conteúdo ubíquo. Como descrito na Seção 4.2, o C3S intercepta o método que interage com o conteúdo ubíquo, gerando um primeiro *checksum*. Ele realiza a execução do método e em seguida executa outro *checksum* do conteúdo ubíquo. Ao comparar os *checksums*, se o C3S perceber que o conteúdo ubíquo foi modificado, ele cria um evento que é capturado por esses observadores (para atualização da representação gráfica). A Figura 5.3 mostra o modelo de um conteúdo ubíquo, representado pelo ícone *banana*, e o Código 5.1 mostra sua codificação.

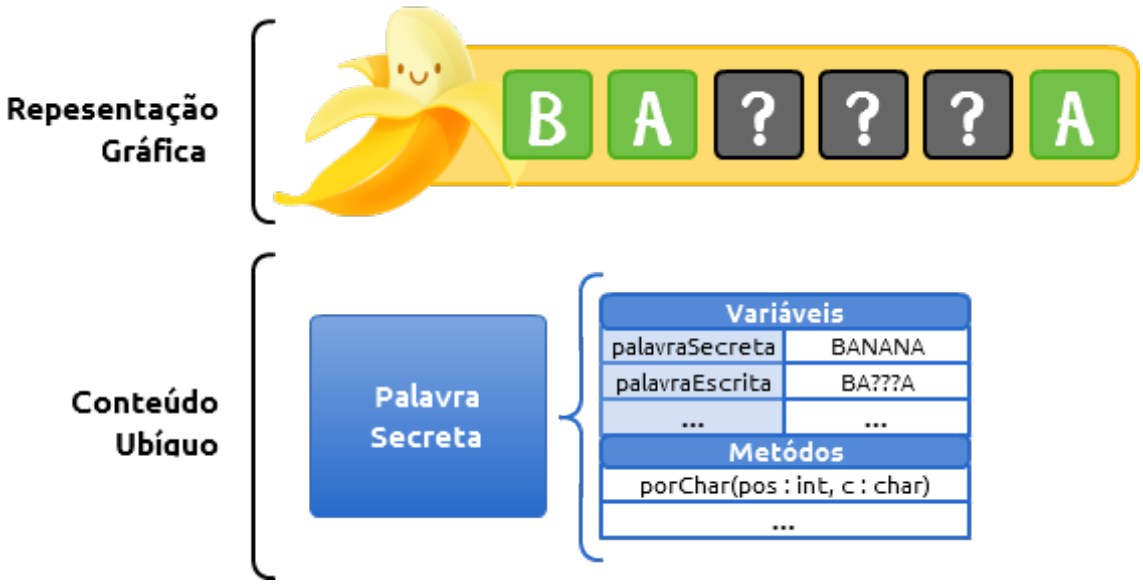

Figura 5.3: *Modelo do conteúdo ubíquo palavra-desafio.*

Código 5.1 Implementação do conteúdo palavra-desafio.

```
1  public class SecretWord extends UbiquitousContent {
2    // Domain
3    private String word;
4    private String spelledWord;
5    private String tip;
6
7    // GUI
8    private String iconName;
9
10   // sets and gets ...
11   public void setCharAt(char letter, int position) {
12     StringBuffer sb = new StringBuffer(spelledWord);
13     if (letter == sb.getChartAt(position))
14         sb.replace(position, position+1, letter + "");
15
16     this.spelledWord = sb.toString();
17   }
18 }
```

A aplicação foi construída utilizando o arcabouço MT4J [31], que permite que o usuário interaja com a aplicação por meio de gestos multitoque. Esses gestos permitem tornar não só a interação com a aplicação mais intuitiva ao permitir a inserção de uma letra utilizando um toque, mas também o compartilhamento de conteúdo utilizando gestos. Ao iniciar a aplicação, sua ativação no dispositivo de cada participante exibe uma tela que permite à criança selecionar seu usuário. Os usuários da aplicação são representados por avatares. A criança toca no seu respectivo desenho para entrar na aplicação, como ilustrado na Figura 5.4. A ativação traduz essa interação em uma mensagem de *login* com o usuário selecionado para o gerente de usuários. Após autenticar-se, o usuário pode entrar em um determinado grupo. Na aplicação *Palavras Secretas*, existem dois grupos, um representado pelo ícone de mel e outro pelo ícone do balão. O usuário novamente pode interagir com a tela e tocar no grupo em que deseja entrar, como ilustrado na Figura 5.5. Caso o grupo esteja cheio, o usuário deve escolher o outro grupo que, espera-se, esteja vazio. Essa interação é transformada na chamada de método para entrada no grupo, *activation.getGroup('balao').join(user)*, onde o parâmetro *'balao'* representa o identificador do grupo e *user* o usuário.

Após a ativação inserir o usuário em um grupo, é apresentada a tela principal da aplicação, ilustrada na Figura 5.1. O usuário pode gerar uma palavra-desafio e, em seguida, tentar identificá-la. O usuário também pode compartilhar sua palavra-desafio

Figura 5.4: *Tela de login da aplicação.*

Figura 5.5: *Grupos disponíveis.*

com outro usuário ou com seu grupo. A aplicação assume como comportamento padrão o compartilhamento com todos os usuários do grupo. O compartilhamento pode ser efetuado com os usuários do grupo, representado por seus avatares. Caso o usuário não queira compartilhar com um usuário do grupo, basta tocar em cima do seu avatar. Essa ação desativa o compartilhamento com aquele usuário, como ilustrado na Figura 5.6. Nessa figura, o usuário representado pelo desenho de um "burro" não receberá as palavras que forem compartilhadas. O usuário também pode limitar o compartilhamento a um ou outro usuário específico, bastando para isso tocar no seu avatar.

Figura 5.6: *Compartilhamento seletivo pelas Palavras Secretas.*

Em particular, nessa aplicação, as primitivas de compartilhamento do *middleware* são combinadas para realizar uma semântica de compartilhamento para mesclar (reunir em um só) conteúdos ubíquos clonados. Por exemplo, ao clonar uma palavra-desafio para um colega, o usuário do outro lado pode desenvolvê-la isoladamente (por exemplo, identificando algumas letras). Seria interessante que essas palavras, a do usuário de origem e a cópia compartilhada utilizando a primitiva de clonar pudessem ser mescladas ao retornar para o colega que clonou. Essa semântica de compartilhamento foi construída combinando as primitivas de clonar e mover. Ao mover o conteúdo de volta ao usuário que clonou, o identificador da palavra-desafio compartilhada é comparado com o de origem, se forem iguais, é realizanda uma mesclagem das letras das palavras. Por exemplo, considere um cenário onde um usuário X clona a palavra-desafio `"b?????"` (banana) para um usuário Y. O usuário Y identifica as letras `"bana??"`, enquanto que o

usuário *X* identifica `"b??ana"`. Quando o usuário *Y* move a palavra-desafio de volta para *X* as letras são mescladas, resultando na palavra `"banana"`.

A instância em execução da aplicação ao receber um conteúdo compartilhado, é notificada pelos métodos *onReceivedMovedContent*(), *onReceivedClonedContent*(), e *onReceivedMirroredContent*(), conforme a semântica da ação de compartilhamento. Como dito, na Seção 4.6, essa notificação possui dois argumentos: o conteúdo ubíquo compartilhado e o usuário que executou a ação. Na aplicação *Palavra Secretas* esses métodos adicionam os respectivos conteúdos ubíquos na interface gráfica. As primitivas não fornecem mecanismos para ignorar a ação de compartilhamento. Entretanto, a aplicação pode rejeitar o compartilhamento ao descartar/inutilizar o conteúdo ubíquo vindo das chamadas de notificação. A notificação de recebimento da operação mover verifica se existe alguma palavra secreta na tela com o mesmo identificador. Em caso afirmativo, a ativação mescla as letras conhecidas da palavra recebida com as letras da instância local da palavra-desafio. O recebimento das notificações de clonar e espelhar apenas adicionam o conteúdo na tela.

O controle de acesso a palavras-desafios espelhadas é realizado pelo *middleware* C3S por meio de um servidor que centraliza o acesso e controle ao conteúdo ubíquo. O cliente C3S intercepta a interação com o conteúdo e requisita seu bloqueio no servidor. Por exemplo, quando dois usuários inserem a letra ao mesmo tempo, o servidor centraliza e enfileira o acesso ao conteúdo ubíquo, resolvendo, de forma básica, o problema de concorrência. Suponha que o usuário *X* insira a letra `"C"` na segunda posição da *string* `"B?NAN?"`, e outro usuário *Y* tente inserir em seguida a letra `"A"`. A instância do C3S de *X* irá requisitar o bloqueio ao conteúdo ubíquo, e enviará a notificação para executar o método $putChar(C,2)$. As réplicas do conteúdo irão executar o método, mas não irão inserir a letra visto estar incorreta. Em seguida é despachado o método de *Y* para inserção da letra `"A"`, que será aplicado por todas as réplicas.

5.3 Apresentador de Slides Ubíquo

A segunda aplicação desenvolvida para testar o C3S é um apresentador de slides *ubíquo*, que foi utilizado como exemplo ao longo desta dissertação. O programa permite que um professor espelhe sua apresentação de *slides* em PDF para um conjunto de alunos. A aplicação implementada é uma modificação da aplicação JPedal [1], um leitor de PDF implementado em Java. A modificação consistiu em adicionar uma camada de adaptação da aplicação para o C3S. A Figura 5.7 ilustra a arquitetura geral dessa adaptação. A adaptação alterou o código de execução inicial (*main*) do JPedal para inicializar os estados

[1] JPedal – http://www.jpedal.org

de uma ativação e para poder enviar e receber compartilhamentos de conteúdo. No estado inicial, acrescentamos uma tela de *login*, que permite ao usuário da aplicação entrar com suas informações. Nessa tela inicial, o professor ou aluno seleciona seu usuário no menu. O Código 5.2 também mostra esse procedimento. Essa aplicação contém apenas um grupo, denominado "sala", do qual todos os usuários participam. Tanto o professor como o aluno são inseridos nesse grupo.

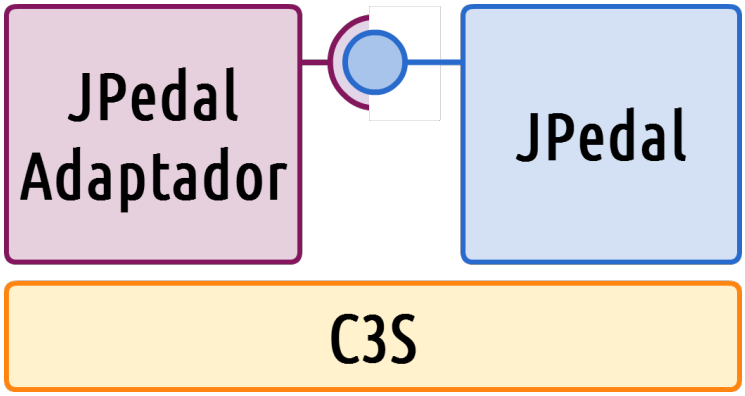

Figura 5.7: *Arquitetura da adaptação da aplicação JPedal para o C3S.*

Código 5.2 Estado de Criação da Apresentação Ubíqua.

```
1  boolean onBegin() {
2    C3S c3s = C3S.getInstance();
3    UserManager um = c3s.getUserManager();
4
5    // Login
6    User user = chooseUser(um.getAvailableUsers());
7    boolean authenticated = um.login(user);
8
9    if (authenticated) { // Authenticated
10     c3S.getGroupManager().getGroup("sala").join(user); // Join
11     return true; // App started correctly
12   } else { // Error
13     return false; // App not started correctly
14   }
15 }
```

O conteúdo ubíquo dessa aplicação é a apresentação em PDF ao usuário, quem foi utilizado como exemplo na Seção 4.2 do capítulo anterior. A apresentação possui como atributos o arquivo em PDF e o número da página atual. Além disso, o conteúdo

possui operações que possibilitam mover para frente, para trás ou ir a uma página específica da apresentação. Quando esse conteúdo ubíquo é espelhado, o arquivo inteiro da apresentação é transferido, ou seja, cada aluno recebe uma cópia dessa estrutura (composta por um PDF, número de páginas e página atual), possibilitando que a ativação crie uma representação gráfica para ela. A tela dessa aplicação é ilustrada na Figura 5.8. O professor abre sua apresentação, que é instanciada como um conteúdo ubíquo. Ao clicar no botão *Sync*, a ativação espelha o conteúdo para o grupo "sala", que representa todos os usuários do ambiente.

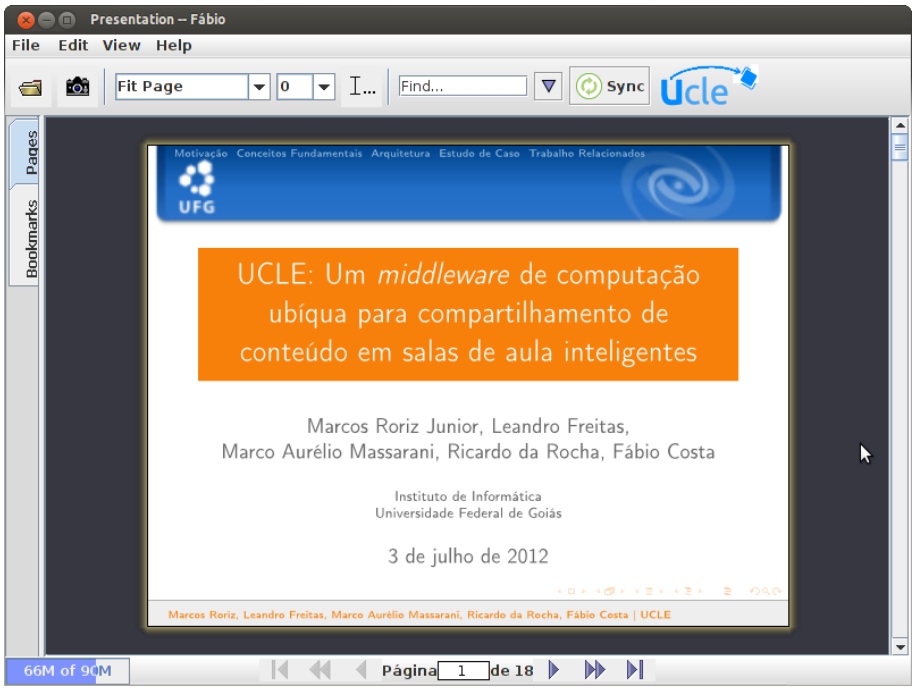

Figura 5.8: *Tela principal do apresentador de slides.*

Quando o professor passa a página da apresentação, o C3S intercepta a chamada de método no conteúdo ubíquo da apresentação e a replica em cada cópia. A chamada encapsulada inclui uma das operações do conteúdo ubíquo, como ir para a próxima página, para trás ou ir para uma página específica. Ao receber a chamada, o conteúdo notifica a representação gráfica (observador), que recupera o número da página atual e a renderiza. Entretanto, ao contrário da semântica tradicional, quando o aluno passa a página da apresentação, isso não é replicado para o restante da classe, ou seja, as alterações nas réplicas dos alunos não são submetidas aos outros usuários. Isto é feito utilizando a opção $content.onlyOwner(true)$ que limita as modificações ao dono do conteúdo ubíquo (nesse caso, o professor). Nessa configuração, quando um aluno passa ou volta uma página da apresentação isso ocorre apenas em sua cópia, ou seja, essas alterações não são refletidas nas cópias dos outros alunos. O C3S intercepta as interações com esses conteúdos, po-

rém ao inspecionar essa propriedade (*onlyOwner*) ele não despacha a alteração para ser enviada aos outros pares.

5.4 Discussão

O estudo de caso utilizou duas aplicações para validar o protótipo do C3S baseado em um subconjunto do modelo de compartilhamento em espaço inteligente desta dissertação. A implementação do aspecto de conteúdo ubíquo das aplicações apresentadas seria complexa de serem realizada no modelo de programação de plataformas de *middleware* de compartilhamento existentes [7, 23, 27, 34]. Isso porque a estrutura de compartilhamento dessas plataformas só permitem uma única forma de compartilhamento. No caso do primeiro conteúdo ubíquo, o mesmo item (instância) da palavra secreta pode ser compartilhado usando múltiplas semânticas diferentes. Por exemplo, a mesma palavra-desafio pode ser tanto movida, clonada ou espelhada entre usuários do ambiente de computação ubíqua. No segundo caso, a apresentação de *slides* está sendo compartilhada de apenas uma forma (espelhar). Entretanto, existe a possibilidade, em nível de *middleware*, desse mesmo conteúdo ser clonado ou movido entre os usuários. Uma generalização da aplicação permitira que usuários pudessem mover e clonar apresentações. Por exemplo, um aluno poderia clonar uma aula (em PDF) que o professor lecionou na semana passada. Como mencionado anteriormente, as plataformas relacionadas não apresentam essa funcionalidade, exigindo que o item de conteúdo ubíquo seja modelado e definido junto com a ação de compartilhamento desejada. De fato, não existem plataformas de *middleware* que forneçam as três semânticas do modelo C3S.

As aplicações do estudo de caso também requerem a noção de grupos, para permitir o agrupamento de usuários em ações de compartilhamento. O conceito de grupos é importante para essas aplicações ubíquas no ambiente de espaços inteligentes, pois permite agrupá-los em conjuntos, geralmente, com interesses em comum (conforme conhecimento da aplicação), o que facilita o compartilhamento e interconexão dos conteúdos ubíquos dos usuários presentes. A primeira aplicação utiliza esse conceito como mecanismo para permitir que alunos se organizem em grupos para resolver o problema em conjunto (identificar as letras das palavras-desafio). Além disso, a aplicação possibilita que o aluno utilize os mecanismos de grupo do *middleware* para realizar o compartilhamento apenas com usuários específicos do seu grupo. Na segunda aplicação, utilizamos o grupo “sala” para identificar os alunos no ambiente. Entretanto, poderiam ser definidos outros grupos para permitir o espelhamento de outras apresentações (em outros grupos).

Uma das lições aprendidas no estudo de caso foi a facilidade de alterar uma aplicação baseada em compartilhamento para portá-la para o modelo do C3S. Necessitou-se de pouco tempo e trabalho para identificar os elementos-chave de compartilhamento

dessa aplicação legada e transformá-los em conteúdos ubíquos. Por exemplo, na aplicação de apresentação de slides, a identificação e adaptação da estrutura do *slide* para conteúdo ubíquo foi feita com menos de 100 linhas de código. Outra lição aprendida, reflete o fato da necessidade da existência de compartilhamento direto entre usuários do ambiente. As plataformas existentes lidam exclusivamente com mecanismos de compartilhamento para todos (*"broadcast-sharing"*). Entretanto, em ambas as aplicações identificamos a necessidade de mecanismos que limitem o compartilhamento a um usuário ou a um determinado grupo.

Um aspecto limitante da avaliação realizada nos estudos de caso é a inexistência de conteúdos ubíquos complexos nas aplicações. Isso ocorreu devido ao fato do protótipo do C3S implementar apenas um subconjunto do modelo do C3S. Por fim, outro aspecto limitante do estudo de caso é o fato de não apresentar testes quantitativos sobre as primitivas do *middleware* C3S, como testes de desempenho e estabilidade. O problema de realizar esses testes surge da dificuldade de agrupar e equalizar o cenário de teste para as diversas plataformas de *middleware* relacionadas existentes. As plataformas relacionadas, não apresentam as mesmas características, muitas vezes implementando apenas algum pequena funcionalidade das características do modelo proposto nesta dissertação. Por exemplo, a única intersecção da plataforma Mica [27] com este trabalho é na primitiva de espelhamento. Entretanto, ela implementa essa primitiva utilizando um mecanismos centralizado utilizando espaço de tuplas. Já a plataforma NetMorph [58] implementa apenas a funcionalidade de mover "conteúdo ubíquo". A discrepância entre os trabalhos relacionados dificulta a equalização para realização de comparações quantitativos.

Futuramente, a avaliação quantitativa pode ser estendida para realizar testes com usuários. É interessante saber se de fato o *middleware* C3S facilita o desenvolvimento de aplicações baseadas em compartilhamento em espaços inteligentes. Essa avaliação poderia ser executada por desenvolvedores que devem construir aplicações com e sem o C3S. Para estender a avaliação para abordagens quantitativas pode-se considerar testar as seguintes propriedades e operações do *middleware*: escalabilidade em relação à quantidade de conteúdos ubíquos compartilhados, tempo para replicar mensagens de um conteúdo ubíquo espelhado e tempo para mover e clonar conteúdos ubíquos entre usuários. Essas métricas podem ser coletadas utilizando mensagens de depuração no servidor central (para conteúdos espelhados) e *timestamps* em cada cliente do espaço inteligente (ao enviar e receber o conteúdo).

5.5 Sumário e Considerações Finais

Este capítulo apresentou duas aplicações, *Palavras Secretas* e *Apresentador de Slides Ubíquo*, como estudo de caso de uso das construções da plataforma de *middleware*

C3S. A primeira aplicação, *Palavras Secretas*, foi construída a partir "do zero" e utiliza uma abordagem de associar palavras a imagens para crianças em fase de alfabetização. Utilizando a associação com a imagem, as crianças devem inserir as letras correspondentes que formam a palavra. A palavra-desafio representa o item de compartilhamento nesta aplicação, que possui atributos como uma *string* que contém a palavra original, a palavra atual, dicas de letras e imagem associada. Esse conteúdo ubíquo também possui métodos para realizar a inserção das letras, *putChar()*, e para recuperar a palavra secreta (*getSecretWord()*) e para obter a parte da palavra descoberta até o momento (*getCurrentWord()*). Em seguida, foi detalhada construção da representação gráfica desse conteúdo ubíquo, e detalhes de sua implementação.

Modificamos a aplicação JPedal, que é um leitor de PDF em Java, para construir o apresentador de slides ubíquo. O método inicial da aplicação foi isolado para ser executado no estado de inicialização da ativação, $onBegin()$. Na parte de construção da ativação, $onCreate()$, a ativação apresenta uma tela de *login*, que permite ao aluno ou o professor selecionar seu usuário correspondente. A apresentação foi modelada como conteúdo ubíquo, com atributos que incluem um vetor de bytes para representar o arquivo do *slide* e sua página atual. Esse conteúdo contém operações para passar, voltar ou ir a uma página específica. A apresentação é espelhada para todos os alunos da sala, entretanto, a apresentação ativa a propriedade $onlyOwner(true)$ para especificar que apenas as ações do professor serão refletidas, impedindo, que quando o aluno passar para o próximo *slide*, isso seja propagado para as outras cópias.

Por fim, foram apresentadas as principais limitações do estudo de caso. Que envolve o fato de não ter conteúdo complexo, pois a implementação não lida com a funcionalidade de ligação entre conteúdos ubíquos. Outras limitações envolvem o fato das aplicações não terem sido testadas por um grupo de controle, onde pudéssemos refletir se o modelo de colaboração baseado em conteúdo ubíquo facilita a colaboração dos usuários no ambiente de computação ubíqua.

No próximo capítulo, iremos comparar o modelo de conteúdo ubíquo apresentado nesta dissertação, assim como o C3S, com outros trabalhos relacionados. Entre os quesitos de comparação, incluímos as semânticas das primitivas de compartilhamento suportadas, a forma como o compartilhamento é realizada pela aplicação (por tipos ou por função), suporte ao conceito de grupos, e a consistência dos conteúdos ubíquos ao serem compartilhadas pelas plataformas de *middleware* correlatas.

CAPÍTULO 6

Trabalhos Relacionados

Existem poucas plataformas de *middleware* construídas para dar suporte ao aspecto de conteúdo da computação ubíqua. Esta pesquisa encontrou apenas plataformas que tratam em segundo plano [7, 23, 27, 34] ou parcialmente [28, 58, 63] esse aspecto. Essas plataformas atendem, em maior ou menor grau, os requisitos desse tipo de aplicação. Este capítulo apresenta uma análise dessas plataformas, comparando-as com o modelo do *middleware* C3S com o intuito de explicitar as principais contribuições deste trabalho.

A análise é feita com base nos requisitos de características do compartilhamento de conteúdo ubíquo em ambientes de computação ubíqua. Esses requisitos incluem:

1. Tipos e semânticas de compartilhamento fornecidas;
2. Como as semânticas são implementadas, isto é, se elas são implementadas como primitivas ou embutidas de forma fixa a um tipo de conteúdo ubíquo;
3. Existência ou não de abstrações para agrupamento de usuários; e
4. Tratamento de problemas de integridade referencial e controle de concorrência no compartilhamento.

A análise se estende ao longo deste capítulo. A Seção 6.1 utiliza os requisitos apontados acima para realizar uma comparação com os trabalhos relacionados. Em seguida, a Seção 6.2 apresenta uma visão geral da análise, seguida de sua conclusão.

6.1 Comparação

Apesar de existirem diversos trabalhos em computação ubíqua, poucos focam no aspecto de conteúdo ubíquo. Trabalhos como Gaia [1] e Aura [50] podem ser utilizados como base para a construção do C3S, isto é, eles servem como infraestruturas que abstraem a comunicação e noção de usuários para o C3S. Decidimos focar a nossa análise em trabalhos que apesar de estarem em domínios "diferentes" realizam semanticamente as mesmas atividades das primitivas propostas.

As características relevantes para o compartilhamento em aplicações ubíquas com foco em conteúdo são extraídas dos requisitos dessas aplicações, identificados no Capítulo 2. Esses requisitos são traduzidos em abstrações fornecidas pelas plataformas de middleware analisadas. Com base nas características apresentadas, esta seção compara cada trabalho relacionado com o modelo do C3S. A comparação visa posicionar o C3S em relação aos demais trabalhos.

6.1.1 Semântica de compartilhamento

Existem diversas formas de compartilhar conteúdo ubíquo, por exemplo, movendo, clonando e espelhando. Cada forma de compartilhar é vista como uma semântica de compartilhamento. Por exemplo, a semântica de mover, remove o conteúdo de um local e o move para outro. Já a semântica de espelhar, realiza a cópia do conteúdo, seguida da sua sincronização. Apesar de ter diferentes nomes em outras plataformas, as semânticas de manipulação de com teúdo ubíquo são as mesmas. Serão analisado neste quesito, as semânticas de compartilhamento que as plataformas de *middleware* fornecem ao desenvolvedor para compartilhar conteúdo ubíquo.

Grande parte dos trabalhos relacionados fornecem apenas uma semântica de compartilhamento de conteúdo ubíquo, esse é o caso do *middleware Mica* [27]. Essa plataforma utiliza um mecanismo centralizado, denominado quadro-negro, baseado em espaço de tuplas para dar suporte à semântica de espelhamento, nela denominada sincronização. Essa semântica de compartilhamento é realizada no quadro negro, onde as aplicações lêem e escrevem as tuplas, impossibilitando o compartilhamento direto entre usuários. De fato não existe o conceito de 'usuário' nesta plataforma, apenas *hosts* que escrevem e lêem no quadro negro.

A plataforma *DOLCLAN* [7] também apresenta somente a primitiva de espelhamento e utiliza uma abordagem parecida com o *Mica*, porém descentralizada. Nesse *middleware*, o quadro-negro é substituído pelo conjunto de dispositivos presentes, de forma que os dados são replicados em todos os elementos do ambiente. Outros trabalhos, como DreamObject [34], e Agilo [23], também são limitados à primitiva de espelhamento e realizam a replicação do conteúdo ubíquo com todos os usuários do ambiente. As plataformas DOLCLAN e DreamObject transferem o conteúdo ubíquo completo a cada modificação do conteúdo nas réplicas, deixando lento a semântica de espelhamento dessas plataformas.

As plataformas AOM [28] e ProActive [63] foram construídas em cima de *middleware* de objetos distribuídos e proveêm serviços que possuem características semelhantes às primitivas de mover e clonar. Entretanto, ambas requerem que o desenvolver especifique e implemente manualmente partes da operação (no caso, a parte de servidor,

para receber e enviar pacotes e objetos).

Uma abordagem mais interessante e diferente para tratar esse problema é apresentada no *middleware NetMorph* [58], que fornece a primitiva de mover. A plataforma também requer que os dados sejam especificados com a semântica de mover durante a sua definição. Essa plataforma particiona o ambiente em um espaço lógico 2D e possibilita mover dados a uma coordenada (x, y) do mapa, onde cada coordenada representa um usuário. Nessa abordagem, os dispositivos dos usuários que estiverem presentes nessas coordenadas receberão os dados transferidos.

Com base nessa comparação, podemos perceber que as plataformas de *middleware* relacionadas fornecem, com apenas duas exceções, uma única primitiva de compartilhamento, como ilustrado pela Tabela 6.1. Além disso, as primitivas fornecidas por essas plataformas em geral são de baixo nível de abstração, requerendo que o programador implemente parte da primitiva ou limitando seu uso ao compartilhamento com todos os usuários.

Plataforma de *Middleware*	Mover	Clonar	Espelhar
DOLCLAN			X
DreamObject			X
Mica			X
Agilo			X
ProActive	Parcialmente	Parcialmente	
AOM	Parcialmente	Parcialmente	
NetMorph	X		
C3S	X	X	X

Tabela 6.1: *Semânticas de compartilhamento fornecidas pelas plataformas de middleware.*

6.1.2 Formas de implementar o compartilhamento

O compartilhamento de conteúdo ubíquo pode ser realizado por meio de primitivas procedurais, ou pode ser embutido no tipo do conteúdo. No primeiro caso, o compartilhamento é feito com o uso de funções que expressam as diferentes semânticas de compartilhamento, enquanto que no segundo caso a semântica faz parte da própria definição do tipo de conteúdo.

As primitivas definem diversas semânticas de compartilhamento utilizando funções. Logo, o compartilhamento utilizando primitivas pode ser classificado como dinâmico, visto que se pode escolher em tempo de execução a função (semântica) de compartilhamento desejada. Por outro lado, utilizando tipagem, a semântica de compartilhamento é estática e embutida no próprio tipo de conteúdo, de maneira que qualquer conteúdo de

um determinado tipo terá sempre a mesma semântica de compartilhamento definida pelo desenvolvedor.

Nesse quesito, os trabalhos relacionados são agrupados de acordo com a utilização de tipagem ou de primitivas para definição da semântica de compartilhamento, como visto na Tabela 6.2. Os trabalhos DOLCLAN, DreamObject, Mica, e Agilo, embutem a forma de compartilhamento na definição do conteúdo ubíquo. Nesses trabalhos, toda instância do conteúdo ubíquo só pode ser compartilhada de acordo com a semântica embutida em sua definição de tipo. Por outro lado, os demais trabalhos relacionados, assim como o modelo definido nesta dissertação (C3S), desacoplam a semântica de compartilhamento do conteúdo ubíquo, de forma que um mesmo conteúdo possa ser compartilhado de diferentes formas.

6.1.3 Agrupamento de usuários

Diversas aplicações permitem a formação de grupos para facilitar o compartilhamento no ambiente de computação ubíqua. Essas aplicações requerem mecanismos para criar e apagar grupos, além de operações para inserir e remover usuários desses grupos. Esse critério busca analisar se existem operações que tratam essas abstrações e se elas podem ser usadas em conjunto com as primitivas de compartilhamento, por exemplo, para definir se um usuário pode compartilhar um item de conteúdo ubíquo com os membros do seu grupo.

Apenas duas plataformas de *middleware* relacionadas fornecem a abstração de grupos, como apresentado na Tabela 6.3. O *middleware* DOLCLAN permite delimitar a sincronização de um conteúdo ubíquo a um conjunto de hospedeiros. Nesse sentido, ele fornece a abstração de agrupamento de usuário e define o escopo da operação de compartilhamento a esse conjunto. Já a plataforma DreamObject eleva o nível de abstração do conjunto de hospedeiros do DOLCLAN para um agrupamento de usuários da aplicação. Ela também permite que sua primitiva de compartilhamento (sincronização)

Plataforma de *Middleware*	Tipagem	Primitiva
DOLCLAN	X	
DreamObject	X	
Mica	X	
Agilo	X	
ProActive		X
AOM		X
NetMorph		X
C3S		X

Tabela 6.2: *Formas de implementar o compartilhamento fornecidas pelas plataformas de middleware.*

também seja efetuado entre os membros do grupo. Ambas abordagens, DOLCLAN e DreamObject, utilizam a replicação do conteúdo ubíquo completo a cada alteração para manter as cópias sincronizadas, ao contrário da abordagem do C3S que utiliza um despachante que envia apenas a operação para ser executada em cada cópia. Pelas funcionalidades apontadas, podemos concluir que apenas duas plataformas relacionadas fornecem um mecanismos parecido com o do C3S. Entretanto, mesmo essas plataformas possuem um certo baixo nível de abstração e realizam replicações constantes do conteúdo ubíquo no grupo.

Plataforma de *Middleware*	Grupos
DOLCLAN	X
DreamObject	X
Mica	
Agilo	
ProActive	
AOM	
NetMorph	
C3S	X

Tabela 6.3: *Abstrações de grupos de usuários fornecidas pelas plataformas de middleware.*

6.1.4 Integridade referencial e consistência

A estrutura de um conteúdo ubíquo pode referenciar outros conteúdos. Essa referência permite não só relacionar conteúdos diferentes como também construir estruturas complexas de conteúdo como, por exemplo, um conteúdo composto por outros conteúdos. Em alguns casos, por exemplo, na semântica de mover, como resultado do compartilhamento estas referências podem tornar-se inconsistentes. As plataformas de *middleware* existentes não tratam este problema. Entretanto, existem algumas soluções manuais para lidar com elas: reconfigurar as referências automaticamente ou expor ao programador os detalhes do problema, como, a referência em uso que está para ser quebrada, para que este possa resolver o problema manualmente.

Nenhuma das plataformas de *middleware* relacionadas tratam completamente esse problema. Como as plataformas DOLCLAN, DreamObject, Mica e Agilo, implementam apenas a semântica de espelhamento e não possuem conteúdos complexos, é esperado que ele não forneçam soluções para esse problema. Já as plataformas AOM e ProActive utilizam uma cópia rasa (*shallow copy*) do conteúdo ubíquo compartilhado e requerem que o desenvolvedor especifique manualmente essas referências. No modelo C3S, utilizamos uma estratégia de encadeamento de referências, de modo que, quando o conteúdo ubíquo transferido, ele é acessado e é gerado um evento de falha de acesso

(*miss*), que redireciona a chamada para a nova localização do conteúdo. Esse processo é realizado automaticamente pelo C3S, visto que ele intercepta automaticamente as chamadas e as redireciona para seu novo destino.

6.2 Análise e Conclusão

Com base nas características levantadas e analisadas, ao longo do texto e ilustrada na Tabela 6.4, pode-se perceber que o modelo do C3S apresenta uma abordagem mais completa para o aspecto de conteúdo da computação ubíqua. Isto ocorre devido ao fato de que o domínio do C3S possui foco na utilização do compartilhamento de conteúdos ubíquos como meio de interconectar os usuários presentes no ambiente. Apesar de possuírem um subconjunto de funcionalidades em comum, os trabalhos relacionados possuem outros focos. Em geral, percebemos que os trabalhos relacionados podem ser categorizados em dois tipos: *middleware* colaborativo (CSCW) e agentes móveis. Esses tipos são não exclusivos visto que um trabalho de CSCW pode ser implementado utilizando agente móveis e vice-versa. Os trabalhos possuem propriedades geralmente apresentadas por esses domínios, por isso essa categorização.

Os trabalhos baseado primariamente em vertentes de CSCW, como DOLCLAN, DreamObject, Mica e Agilo, apresentam as mesmas características, diferindo pouco nas funcionalidades. Todos os trabalhos pertencentes a esse conjunto fornecem apenas a semântica de espelhamento, vinculada fortemente ao tipo de conteúdo ubíquo, não sendo possível compartilhar de outra forma. Apesar de possuírem abstrações para definição de grupos, eles possibilitam apenas o compartilhamento com todos os usuários do ambiente ou grupo. Por fim, todos esses trabalhos não tratam integridade referencial ou consistência do conteúdo ubíquo.

Plataforma	Semântica Suportada	Implementação	Grupos	Integridade Ref.
DOLCLAN	Espelhar	Tipagem	Sim	Não
DreamObject	Espelhar	Tipagem	Sim	Não
Mica	Espelhar	Tipagem	Não	Não
Agilo	Espelhar	Tipagem	Não	Não
ProActive	Mover[1] e Clonar[1]	Primitiva	Não	Não
AOM	Mover[1] e Clonar[1]	Primitiva	Não	Não
NetMorph	Mover	Primitiva	Não	Não
C3S	Mover, Clonar e Espelhar	Primitiva	Sim	Parcial

Tabela 6.4: *Resumo da comparação das funcionalidades de compartilhamento das plataformas de middleware que implementam o aspecto de conteúdo ubíquo.*

[1] Parcialmente

Os outros trabalhos foram construídos utilizando estratégias para agentes móveis. Nesses trabalhos, os agentes assumem o papel de conteúdo ubíquo e podem mover-se entre os hospedeiros do sistema. As plataformas ProActive, AOM e NetMorph foram construídas utilizando esse conceito. Todas, com exceção do NetMorph, foram projetadas sobre *middleware* de objetos distribuídos. Apesar dos trabalhos em agentes possuírem abstrações de alto-nível, como objetivo, rota, etc, eles fornecem abstrações de baixo nível para o compartilhamento entre os agentes. Além disso, eles não fornecem mecanismos para agrupamento de usuários e nem semânticas de espelhamento.

CAPÍTULO 7

Conclusões

A computação ubíqua visa integrar a computação ao dia-a-dia do usuário. Devido a limitações das tecnologias computacionais atuais ou do escopo da aplicação essa integração é realizada do ponto de vista de um dado aspecto que se deseja ressaltar. Por exemplo, o aspecto de contexto visa fornecer informações contextuais, possivelmente do ambiente do usuário, para que as aplicações possam reagir a essas informações. O aspecto de conteúdo ubíquo, por sua vez, traz uma diferente perspectiva de como realizar a computação ubíqua. Ao contrário de outros aspectos, como mobilidade, contexto e interface, que visam especificar serviços que permitem integrar individualmente um dado usuário ao ambiente, esse aspecto visa combinar os serviços do ambiente de computação ubíqua para integrar os usuários presentes através de serviços de compartilhamento de conteúdo.

Nesta dissertação, foi apresentado um modelo formado por um conjunto de serviços e conceitos, como conteúdo ubíquo, ativação e aplicação ubíqua, que visam atender o aspecto de conteúdo da computação ubíqua. Para especificar esse modelo, foi necessário sistematizar a definição do conceito de conteúdo e mais especificamente, de conteúdo ubíquo. O primeiro se refere a entidades de primeira classe que representam estados da aplicação, enquanto que o segundo se refere à capacidade desse conteúdo de ser manipulado no ambiente de computação ubíqua.

O modelo projetado nesta dissertação fornece primitivas de compartilhamento de conteúdo ubíquo para ambientes de computação ubíqua do tipo espaço inteligente. As primitivas fornecidas possibilitam mover, clonar e espelhar conteúdos ubíquos entre usuários do espaço inteligente. O modelo possibilita a execução de qualquer uma dessas primitivas de compartilhamento para o mesmo tipo de conteúdo ubíquo, ou seja, um conteúdo ubíquo pode ser compartilhado de diferentes formas. Supreendentemente, essa funcionalidade não foi encontrada em trabalhos relacionados. Além disso, o modelo possibilita o compartilhamento com usuários específicos ou com grupos de usuários. Por fim, o modelo permite capturar e manipular os relacionamentos entre conteúdos ubíquos.

A partir da especificação deste modelo, projetamos uma arquitetura que realiza os conceitos propostos. A plataforma de *middleware*, denominada *Content Sharing for*

Smart Spaces (C3S), situa-se sobre um *middleware* de comunicação e materializa o modelo especificado utilizando uma série de gerentes. Cada gerente é responsável por uma funcionalidade, a saber: conteúdo, aplicação e usuários. As primitivas de compartilhamento e os conceitos de ativação e aplicação ubíqua, são descritos inteiramente na forma de interações entre esses gerentes. Isto torna a arquitetura independente do *middleware* de comunicação subjacente. Uma implementação do C3S foi construída na linguagem de programação Java e sobre o *middleware* de comunicação Java RMI. Essa implementação contempla um subconjunto do modelo proposto. Implementamos os conceitos de conteúdo ubíquo, ativação e aplicação ubíqua, além de todas as primitivas de compartilhamento do modelo. Entretanto, não foram implementado os mecanismos de ligação entre conteúdos ubíquos, de tal maneira que, ao mover um conteúdo, as referência de ligação não são atualizadas remotamente para o novo destino do conteúdo. Apesar dessa limitação, o subconjunto implementado é autocontido e permite avaliar os principais conceitos e primitivas de compartilhamento do modelo proposto.

Para avaliar o modelo, desenvolvemos duas aplicações como estudo de caso. A primeira aplicação foi desenvolvida a partir "do zero", enquanto que a segunda foi baseada em uma aplicação existente. As duas aplicações permitem testar as funcionalidades do modelo e de sua implementação, com exceção da funcionalidade de ligação. Durante o desenvolvimento dessas aplicações, ficou clara a necessidade de primitivas de compartilhamento de alto-nível, assim como o conceito de grupos. Ambas funcionalidades são limitadas ou inexistentes em trabalhos relacionados. Outra lição aprendida no estudo de caso foi a facilidade de portar uma aplicação baseada em compartilhamento para o modelo do C3S. Necessitou-se de pouco tempo e trabalho para identificar os elementos-chave de compartilhamento dessa aplicação legada e transformá-los em conteúdos ubíquos.

Por fim, um estudo sobre os trabalhos relacionados verificou que existem poucas plataformas de *middleware* construídas para dar suporte ao aspecto de conteúdo da computação ubíqua. As plataformas existentes [7, 23, 27, 34] tratam em segundo plano ou parcialmente os requisitos de compartilhamento levantados.

7.1 Contribuições

As contribuições deste trabalho são derivadas dos objetivos apresentados na Seção 1.3 e são resumidas nos seguintes pontos:

1. A sistematização do aspecto de conteúdo da computação ubíqua e do conceito de conteúdo ubíquo, além dos conceitos de ativação e aplicação ubíqua.
2. Um conjunto primitivas de compartilhamento de conteúdo ubíquo, que expressam diferentes semânticas de manipulação de conteúdo no ambiente ubíquo;

3. Um conjunto de serviços e conceitos baseados em conteúdo ubíquo, ativação, aplicação e primitivas de compartilhamento, que compõe um modelo de computação ubíqua baseado em conteúdo;
4. A arquitetura do *middleware* C3S, incluindo a descrição, interação e disposição dos componentes que permitem concretizar os conceitos apresentados; e
5. Um protótipo do *middleware* C3S, utilizando a linguagem de programação Java e o *middleware* de objetos distribuídos Java RMI.

Estas contribuições deram origem às seguintes publicações:

1. RORIZ JUNIOR, M. P.; FREITAS, L. A.; MASSARANI, M. A. L.; DA ROCHA, R. C. A.; COSTA, F. M. **UCLE: Um middleware de computação ubíqua para compartilhamento de conteúdo em salas de aula inteligentes**. In: *Anais do IV Simpósio Brasileiro de Computação Ubíqua e Pervasiva - SBCUP*, p. 10, Curitiba, 2012. SBC.
2. RORIZ JUNIOR, M. P.; FREITAS, L. A.; MASSARANI, M. A. L.; DA ROCHA, R. C. A.; COSTA, F. M. **C3S: A Content Sharing Middleware for Smart Spaces**. In: *Workshop on Pervasive Collaboration and Social Networking 2013*, San Diego, 2013. IEEE.

7.2 Trabalhos Futuros

Como trabalho futuro, pretendemos investigar mecanismos que possibilitem desfazer uma primitiva de compartilhamento. Por exemplo, ao desfazer a movimentação de um conteúdo ubíquo, ele deveria retornar ao usuário que originou a ação. Isto requer que a ação de desfazer seja distribuída, visto que o conteúdo pode ter sido modificado e, inclusive, compartilhado com outro usuário.

Outro tópico interessante a ser pesquisado seria a possibilidade de usuários negarem as ações de compartilhamento. Por exemplo, o usuário poderia negar a recepção de conteúdo ubíquo após um número específico de conteúdos na sua aplicação, negar o recebimento de um dado usuário ou negar uma ação específica (como espelhamento).

O conceito de ligações foi explorado utilizando a abstração da localização do conteúdo ubíquo seguido do encadeamento de referências. Como trabalho futuro, pretendemos explorar uma abordagem mais eficiente e outros mecanismos que possibilitem relacionar conteúdos ubíquos. A integridade referencial é crucial para garantir a consistência dos conteúdos ubíquos. Portanto, é interessante explorar outras alternativas, possivelmente, mais eficientes para essa questão.

Outro trabalho interessante surge da combinação do aspecto de contexto. Utilizando um mecanismo de provisionamento de contexto é possível tornar as operações

sensíveis ao estado dos usuários. Por exemplo, em um cenário educacional o efeito das primitivas seriam desativados durante uma prova. Outra possibilidade de modificar o comportamentos das primitivas surge caso o usuário esteja ocupado ou queira trabalhar isoladamente.

Por fim, uma digressão. Como trabalho futuro, seria interessante investigar a possibilidade de generalizar os diferentes aspectos de computação ubíqua de maneira a possibilitar a comunicação entre plataformas de *middleware* e aplicações que implementem esses aspectos. Por exemplo, um *middleware* orientado ao aspecto de mobilidade poderia ter seus serviços adaptados para comunicar com um *middleware* de outro aspecto de computação ubíqua.

Referências Bibliográficas

[1] **Gaia: A development infrastructure for active spaces.**, 2001.

[2] ABOWD, G.; MYNATT, E.; RODDEN, T. **The human experience [of ubiquitous computing]**. *IEEE Pervasive Computing*, 1(1):48–57, Jan. 2002.

[3] AMBRÓSIO, A. P. L.; COSTA, F. M. **Evaluating the impact of PBL and tablet PCs in an algorithms and computer programming course**. In: *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, p. 495, New York, New York, USA, 2010. ACM Press.

[4] ANDERSON, R. R.; DAVIS, P.; LINNELL, N.; PRINCE, C.; RAZMO, V.; VIDEON, F.; RAZMOV, V. **Classroom Presenter: Enhancing Interactive Education with Digital Ink**. *Computer*, 40(9):56–61, Sept. 2007.

[5] AVGERIOU, P.; TANDLER, P. **Architectural patterns for collaborative applications**. *International Journal of Computer Applications in Technology*, 25(2/3):86, 2006.

[6] BANAVAR, G.; BERNSTEIN, A. **Software infrastructure and design challenges for ubiquitous computing applications**. *Communications of the ACM*, 45(12), Dec. 2002.

[7] BARDRAM, J.; MOGENSEN, M. **DOLCLAN: middleware support for peer-to-peer distributed shared objects**. In: *Proceedings of the 7th IFIP WG 6.1 international conference on Distributed applications and interoperable systems*, p. 119–132. Springer-Verlag, 2007.

[8] BAUDE, F.; CAROMEL, D.; HUET, F.; VAYSSIÈRE, J. **Communicating Mobile Active Objects in Java**. In: Bubak, M.; Afsarmanesh, H.; Hertzberger, B.; Williams, R., editors, *High Performance Computing and Networking*, volume 1823 de **Lecture Notes in Computer Science**, p. 633–643. Springer Berlin / Heidelberg, 2000.

[9] BENMOUFFOK, L.; BUSCA, J.-M.; MANUEL MARQUÈS, J.; SHAPIRO, M.; SUTRA, P.; TSOUKALAS, G. **Telex: Principled System Support for Write-Sharing in Collaborative Applications**. Research Report RR-6546, INRIA, 2008.

[10] BRUMITT, B.; MEYERS, B.; KRUMM, J.; KERN, A.; SHAFER, S. A. **EasyLiving: Technologies for Intelligent Environments**. In: *Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, HUC '00, p. 12–29, London, UK, UK, 2000. Springer-Verlag.

[11] BYRNE, P. **MUSE - Platform For Mobile Computer Supported Collaborative Learning**. PhD thesis, University of Dublin, Trinity College, 2011.

[12] COOK, D. J.; DAS, S. K. **How smart are our environments? An updated look at the state of the art**. *Pervasive and Mobile Computing*, 3(2):53–73, 2007.

[13] COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. **Distributed Systems: Concepts and Design**. Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[14] CROCKFORD, D. **The application/json Media Type for JavaScript Object Notation (JSON)**. RFC 4627 (Informational), July 2006.

[15] DA COSTA, C. A.; YAMIN, A. C.; GEYER, C. F. R. **Toward a General Software Infrastructure for Ubiquitous Computing**. *IEEE Pervasive Computing*, 7(1):64–73, Jan. 2008.

[16] DAVIES, N.; GELLERSEN, H.-W. **Beyond prototypes: challenges in deploying ubiquitous systems**. *IEEE Pervasive Computing*, 1(1):26–35, Jan. 2002.

[17] DE ROCHA, R. C. A.; ENDLER, M. **Middleware: Context Management in Heterogeneous, Evolving Ubiquitous Environments**. *Distributed Systems Online, IEEE*, 7(4):1, 2006.

[18] DEY, A. K. **Understanding and Using Context**. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[19] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns: elements of reusable object-oriented software**. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[20] GARLAN, D.; SIEWIOREK, D. P.; SMAILAGIC, A.; STEENKISTE, P. **Project Aura: toward distraction-free pervasive computing**. *Pervasive Computing, IEEE*, 1(2):22–31, 2002.

[21] GREENFIELD, A. **Everyware: The Dawning Age of Ubiquitous Computing**. Peachpit Press, Berkeley, CA, USA, 2006.

[22] GRIMM, R.; DAVIS, J.; LEMAR, E.; MACBETH, A.; SWANSON, S.; GRIBBLE, S.; ANDERSON, T.; BERSHAD, B.; BORRIELLO, G.; WETHERALL, D. **Programming for Pervasive Computing Environments**. Technical report, University of Washington, 2001.

[23] GUICKING, A.; TANDLER, P.; AVGERIOU, P. **Agilo: a highly flexible groupware framework**. In: *Proceedings of the 11th international conference on Groupware: design, Implementation, and Use*, CRIWG'05, p. 49–56, Berlin, Heidelberg, 2005. Springer-Verlag.

[24] HELAL, S. **Programming pervasive spaces**. *Pervasive Computing, IEEE*, 4(1):84–87, 2005.

[25] HOURCADE, J. P.; BEDERSON, B. B.; DRUIN, A. **Building KidPad: an application for children's collaborative storytelling**. *Software: Practice and Experience*, 34(9):895–914, July 2004.

[26] IZADI, S.; BRIGNULL, H.; RODDEN, T.; ROGERS, Y.; UNDERWOOD, M. **Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media**. In: *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, p. 159–168, New York, NY, USA, 2003. ACM.

[27] KADOUS, M. W.; SAMMUT, C. **MICA: pervasive middleware for learning, sharing and talking**. In: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, p. 176–180, 2004.

[28] KAPITZA, R.; SCHMIDT, H.; SÖLDNER, G.; HAUCK, F. **A Framework for Adaptive Mobile Objects in Heterogeneous Environments**. In: Meersman, R.; Tari, Z., editors, *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, volume 4276 de **Lecture Notes in Computer Science**, p. 1739–1756. Springer Berlin / Heidelberg, 2006.

[29] KINDBERG, T.; FOX, A. **System software for ubiquitous computing**. *IEEE Pervasive Computing*, 1(1):70–81, Jan. 2002.

[30] KRUMM, J.; DAVIES, N.; NARAYANASWAMI, C. **Guest Editors' Introduction: Content Sharing**. *IEEE Pervasive Computing*, 8(4):33–34, Oct. 2009.

[31] LAUFS, U.; RUFF, C.; ZIBUSCHKA, J. **MT4j - A Cross-platform Multi-touch Development Framework**. *CoRR*, abs/1012.0, 2010.

[32] LEE, W.; KIM, E.; KIM, J.; LEE, I.; LEE, C. **Movement-Aware Vertical Handoff of WLAN and Mobile WiMAX for Seamless Ubiquitous Access**. *Consumer Electronics, IEEE Transactions on*, 53(4):1268–1275, 2007.

[33] LUKOSCH, S. **Adaptive and Transparent Data Distribution Support for Synchronous Groupware**. In: *Proceedings of the 8th International Workshop on Groupware: Design, Implementation and Use*, p. 255–274, 2002.

[34] LUKOSCH, S. **Transparent and Flexible Data Sharing for Synchronous Groupware**. PhD thesis, University of Hagen, 2003.

[35] LUKOSCH, S.; SCHUMMER, T. **Patterns for Managing Shared Objects in Groupware Systems**. In: Marquardt, K.; Schutz, D., editors, *Proceedings of the Ninth European Conference on Pattern Languages of Programs (EuroPLoP'04)*, p. 333–378, Konstanz, 2005. UVK Universitätsverlag.

[36] MALCHER, M. G.; ENDLER, M. **iPH: Uma Aplicação para Compartilhamento e Co-Edição de Apresentações em Sala de Aula**. In: *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2010.

[37] PARGAS, R.; COOPER, M.; WILLIAMS, C.; BRYFCZYNSKI, S. **OrganicPad: A Tablet PC Based Interactivity Tool for Organic Chemistry**. In: *Pen-Based Learning Technologies, 2007. PLT 2007. First International Workshop on*, p. 1–6, May 2007.

[38] PITOURA, E.; SAMARAS, G. **Data management for mobile computing**, volume 10 de **Advances in Database Systems**. Springer, 1998.

[39] QIN, W.; SUO, Y.; SHI, Y. **CAMPS: A Middleware for Providing Context-Aware Services for Smart Space**. In: Chung, Y.-C.; Moreira, J., editors, *Advances in Grid and Pervasive Computing*, volume 3947 de **Lecture Notes in Computer Science**, p. 644–653. Springer Berlin Heidelberg, 2006.

[40] RANGANATHAN, A.; CHETAN, S.; AL-MUHTADI, J.; CAMPBELL, R. H.; MICKUNAS, M. D. **Olympus: A High-Level Programming Model for Pervasive Computing Environments**. In: *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, p. 7–16, 2005.

[41] RAYCHOUDHURY, V.; CAO, J.; KUMAR, M.; ZHANG, D. **Middleware for pervasive computing: A survey**. *Pervasive and Mobile Computing*, -(-):–, 2012.

[42] ROMAN, M.; HESS, C.; CERQUEIRA, R.; RANGANATHAN, A.; CAMPBELL, R.; NAHRSTEDT, K. **A middleware infrastructure for active spaces**. *IEEE Pervasive Computing*, 1(4):74–83, Oct. 2002.

[43] RORIZ JUNIOR, M. P.; FREITAS, L. A.; MASSARANI, M. A. L.; DA ROCHA, R. C. A.; COSTA, F. M. **UCLE: Um middleware de computação ubíqua para compartilhamento de conteúdo em salas de aula inteligentes**. In: *Anais do IV*

Simpósio Brasileiro de Computação Ubíqua e Pervasiva - SBCUP, p. 10, Curitiba, 2012. SBC.

[44] ROTMAN, S. **The "post-pc" era: It's real, but it doesn't mean what you think it does**. http://blogs.forrester.com/sarah_rotman_epps/11-05-17-the_post_pc_era_its_real_but_it_doesnt_mean_what_you_think_it_does, 2012. Accessed: 07/10/2012.

[45] SCHEIBLE, J.; OJALA, T.; COULTON, P. **MobiToss: a novel gesture based interface for creating and sharing mobile multimedia art on large public displays**. In: *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, p. 957–960, New York, NY, USA, 2008. ACM.

[46] SCHMIDT, A. **Ubiquitous Computing: Are We There Yet?** *Computer*, 43(2):95–97, Feb. 2010.

[47] SINGH, R.; BHARGAVA, P.; KAIN, S. **State of the art smart spaces: application models and software infrastructure**. *Ubiquity*, 2006(September):7:2—-7:9, 2006.

[48] SKIPTON, C.; MATULICH, E.; PAPP, R.; STEPRO, J. **Moving From "Dumb" To "Smart" Classrooms: Technology Options And Implementation Issues**. *Journal of College Teaching & Learning (TLC)*, 3(6):19–27, 2011.

[49] SOUSA, J.; GARLAN, D. **Aura: an architectural framework for user mobility in ubiquitous computing environments**. In: *Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture*, número August, p. 29–43. Citeseer, 2002.

[50] SOUSA, J.; GARLAN, D.; OTHERS. **Aura: an architectural framework for user mobility in ubiquitous computing environments**. In: *Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture*, número August, p. 29–43. Citeseer, 2002.

[51] STARNER, T. **Wearable computers: no longer science fiction**. *IEEE Pervasive Computing*, 1(1):86–88, Jan. 2002.

[52] STEIMLE, J.; BRDICZKA, O.; MUHLHAUSER, M. **CoScribe: Integrating Paper and Digital Documents for Collaborative Knowledge Work**. *IEEE Transactions on Learning Technologies*, 2(3):174–188, July 2009.

[53] STREITZ, N. A.; GEI\SSLER, J.; HOLMER, T.; KONOMI, S.; MÜLLER-TOMFELDE, C.; REISCHL, W.; REXROTH, P.; SEITZ, P.; STEINMETZ, R. **i-LAND: an interactive landscape for creativity and innovation**. In: *Proceedings of the SIGCHI conference*

on Human Factors in Computing Systems, CHI '99, p. 120–127, New York, NY, USA, 1999. ACM.

[54] SYSTEMS, U. A. **Smart boards and collaborative classrooms**. http://www.unifiedav.com/blog/68/smart-boards-collaborative-classrooms/, 2012. Accessed: 10/10/2012.

[55] TANDLER, P. **The BEACH application model and software framework for synchronous collaboration in ubiquitous computing environments**. *Journal of Systems and Software*, 69(3):267–296, Jan. 2004.

[56] TANDLER, P. **Synchronous Collaboration in Ubiquitous Computing Environments**. PhD thesis, Technischen Universität Darmstadt, 2004.

[57] TAURION, C. **O que é o mundo pós-PC?** http://www.ibm.com/developerworks/mydeveloperworks/blogs/ctaurion/entry/o_que_e_o_mundo_pos-pc, 2011. Accessed: 10/05/2012.

[58] UMEZAWA, M.; ABE, K.; NISHIHARA, S.; KURIHARA, T. **NetMorph - an intuitive mobile object system**. In: *Creating, Connecting and Collaborating Through Computing, 2003. C5 2003. Proceedings. First Conference on*, p. 32–39, 2003.

[59] WANT, R.; PERING, T.; BORRIELLO, G.; FARKAS, K. **Disappearing hardware [ubiquitous computing]**. *IEEE Pervasive Computing*, 1(1):36–47, Jan. 2002.

[60] WEISER, M. **The computer for the 21st century**. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999.

[61] WILKERSON, M.; GRISWOLD, W. G.; SIMON, B. **Ubiquitous presenter: increasing student access and control in a digital lecturing environment**. *ACM SIGCSE Bulletin*, 37(1):116, Feb. 2005.

[62] XIE, W.; SHI, Y.; XU, G.; MAO, Y. **Smart Platform - a software infrastructure for Smart Space (SISS)**. *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, p. 429–434, 2002.

[63] XU, B.; LIAN, W.; GAO, Q. **Migration of active objects in proactive**. *Information and Software Technology*, 45(9):611–618, 2003.

[64] YAMIN, A.; AUGUSTIN, I.; DA SILVA, L. C.; REAL, R. A.; FILHO, A. E. S.; GEYER, C. F. R. **EXEHDA: adaptive middleware for building a pervasive grid environment**. In: *Proceedings of the 2005 conference on Self-Organization and Autonomic Informatics (I)*, p. 203–219, Amsterdam, The Netherlands, The Netherlands, 2005. IOS Press.

[65] YAU, S. S.; GUPTA, S. K. S.; KARIM, F.; AHAMED, S. I.; WANG, Y.; WANG, B.; SCIENCE, C. **Smart Classroom: Enhancing Collaborative Learning Using Pervasive Computing Technology**. In: *ASEE 2003 Annual Conference and Exposition*, p. 13633–13642, 2003.