UCLE: Um *middleware* de computação ubíqua para compartilhamento de conteúdo em salas de aula inteligentes

Marcos Paulino Roriz Junior, Leandro Alexandre Freitas, Marco Aurélio Lino Massarani, Ricardo Couto Rocha, Fábio Moreira Costa

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – CEP 74001-970 – Goiânia – GO – Brasil

{marcosjunior,leandroaf,lino,ricardo,fmc}@inf.ufg.br

Abstract. *Collaborative applications are being used in the integration of new computing devices into the classroom. However, their implementation exposes the programmer to system level issues, such as communication and concurrency. Due to the complexity of these problems, most academic work in the area, deals entirely with their construction. Aiming to provide an infrastructure for the development of these applications, we propose* ***U****biquitous* ***C****omputing for* ***L****earning* ***E****nvironments (UCLE), a middleware which makes these problems transparent to the developer through a set of content sharing primitives. They define operations for content movement such as move, clone and mirror, which serves as a high-level communication model for building these applications.*

Resumo. *Aplicações colaborativas vêm sendo utilizadas na integração de novos dispositivos computacionais à sala de aula. Entretanto, sua implementação expõem o programador a problemas de nível de sistema, tais como comunicação e concorrência. Devido à complexidade destes problemas, grande parte dos trabalhos acadêmicos na área, lidam inteiramente com a sua construção. Visando prover uma infraestrutura para o desenvolvimento dessas aplicações, propomos o* ***U****biquitous* ***C****omputing for* ***L****earning* ***E****nvironments (UCLE), um middleware que torna esses problemas transparentes ao desenvolvedor através de primitivas de compartilhamento de conteúdo. Elas definem operações de movimentação de conteúdo, tais como, mover, clonar e espelhar, que servem como um modelo de comunicação de alto nível para a construção dessas aplicações.*

1. Introdução

Nos últimos anos observamos um expressivo aumento na quantidade e uso de novos dispositivos computacionais cada vez mais compactos (*tablets*, *smartphones*) e intuitivos (*touchscreen*, sensores de movimento), aproximando-se da visão de computação ubíqua proposta por Mark Weiser [Weiser 1999, Schmidt 2010]. A utilização desses dispositivos no aprendizado na sala de aula tem a capacidade de tornar o aprendizado interativo e intuitivo aos alunos, tornando a sala de aula mais "inteligente" [Skipton et al. 2011]. Porém, o uso desses dispositivos e da computação em geral como ferramenta efetiva de ensino ainda é um grande desafio [Yau et al. 2003], principalmente devido à complexidade de sua integração na sala de aula. Esta complexidade, geralmente, está relacionada às dificuldades pedagógicas e técnicas no desenvolvimento de aplicações que permitem explorar as possibilidades desse ambiente [Pettersson 2011].

Aplicações colaborativas, tais como anotações interativas em *slides*, jogos educacionais e desenho colaborativo, tendem a integrar e explorar melhor as funcionalidades desses dispositivos na sala de aula inteligente do que aplicações de outras categorias [Patten et al. 2006]. Essas aplicações auxiliam de maneira positiva o ensino através do aumento de interação e compartilhamento de conhecimento entre alunos e professores [Hurford and Hamilton 2008]. Entretanto, sua implementação expõe o programador a diversos problemas não triviais de sistemas distribuídos, tais como mobilidade, concorrência e modelo de comunicação dos dispositivos da sala de aula [Byrne 2011].

O tratamento desses problemas por parte do programador é oneroso e desvia o foco das questões específicas da aplicação. De fato, devido à complexidade desses problemas, grande parte dos trabalhos acadêmicos na área lidam inteiramente com o projeto e construção de aplicações colaborativas no ensino [Hourcade et al. 2004, Anderson et al. 2007, Steimle et al. 2009]. Estes trabalhos tratam repetidamente dos problemas mencionas na construção destas aplicações. Visando prover uma infraestrutura para o desenvolvimento de aplicações colaborativas educacionais, propomos o ***U**biquitous **C**omputing for **L**earning **E**nvironments* (UCLE), um *middleware* que torna esses problemas transparentes para o desenvolvedor através de um conjunto de primitivas de compartilhamento de conteúdo. Essas primitivas definem operações de movimentação de conteúdo, tais como, mover, clonar e espelhar, que servem como um modelo de comunicação de alto nível para a construção dessas aplicações.

O restante deste artigo está organizado da seguinte forma: a Seção 2 discute o problema e a motivação para o compartilhamento de conteúdo em salas de aula inteligentes. As Seções 3 e 4 definem os conceitos e as primitivas que o *middleware* fornece respectivamente. A Seção 5 mostra como esses conceitos e primitivas são decompostos em componentes para formar a arquitetura do UCLE. A Seção 6 descreve uma aplicação desenvolvida com o fim de avaliar o UCLE. A Seção 7 compara este trabalho com outros sistemas de *middleware* e *frameworks* para compartilhamento em salas de aula inteligente e, por fim, a Seção 8 apresenta a conclusão e trabalhos futuros.

2. Compartilhamento de Conteúdo

As aplicações colaborativas em ambiente de sala de aula inteligente possuem várias funcionalidades em comum [Byrne 2011], dentre elas, o compartilhamento de conteúdo, como se vê na Figura 1. Essa funcionalidade compõe o núcleo dessas aplicações, provendo um mecanismo de comunicação que permite compartilhar conteúdos entre suas instâncias.

Figura 1. Intersecção entre algumas aplicações colaborativas.

É comum que aplicações de anotações interativas educacionais, particularmente em *slides*, permita que alunos e professores compartilhem anotações. Os alunos podem enviar (mover) estas anotações (conteúdo) ao professor durante a aula, por exemplo, em caso de dúvida. As anotações do professor também podem ser sincronizadas (espelhadas) para os dispositivos dos alunos. Em jogos educacionais existe uma constante interação entre alunos e professores. Por exemplo, em um jogo de formação de palavras (conteúdo) utilizando sílabas o aluno pode copiar (clonar) sua sílaba (conteúdo) para formar diferen-

tes palavras com outros alunos. Ao formar a palavra os alunos podem enviá-la (movê-la) ao professor. Aplicações de desenho colaborativo educativas geralmente utilizam uma tela compartilhada, onde os alunos enviam (movem) figuras geométricas (conteúdo) que são sincronizadas (espelhadas) ao restante da turma.

É possível identificar as semânticas de mover, clonar e espelhar itens de conteúdo nessas aplicações. Mover um conteúdo significa transportá-lo de um local para outro, clonar tem uma semântica parecida com exceção que retém o item na origem. O espelhamento fornece um compartilhamento contínuo através de cópias sincronizadas do conteúdo. Devido à falta de infraestrutura para o ambiente de salas de aula inteligentes [Springer et al. 2008] essas operações têm que ser implementadas pelo programador utilizando em uma série de operações de baixo nível, como serviço de nomes, representação externa dos conteúdos, primitivas de rede, tratamento de falhas e problemas de concorrência. Lidar com esses problemas requer não só um conhecimento adicional do programador, mas o desvia do foco principal da aplicação.

3. Conceitos Básicos

O UCLE é um *middleware* de computação ubíqua que trata estes problemas, fornecendo um conjunto de primitivas de compartilhamento de alto nível para o desenvolvedor. Estas primitivas englobam as semânticas de mover, clonar e espelhar conteúdo. Uma série de conceitos básicos é utilizadas na descrição geral do UCLE e de suas primitivas.

3.1. Aplicação, Ativação e Aplicação Ubíqua

Uma **aplicação** é formada por um executável e seus metadados. Nos metadados, além de nome, versão e plataforma, é especificado se a aplicação provê suporte para compartilhamento. Ela é instalada em um servidor e os dispositivos dos usuários (alunos, professores) podem requisitar sua instalação/atualização. A execução de uma aplicação é gerenciada pelo UCLE e obedece um ciclo de vida bem definido, que inclui estados para iniciar, pausar e resumir a aplicação.

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 UCLE, *i.e.*, ela é formada pelas instâncias espalhadas da aplicação. Várias aplicações ubíquas podem ser executadas ao mesmo tempo em uma mesma sala de aula. A Figura 2 mostra um esquema que exemplifica a relação entre estes conceitos. A aplicação ubíqua de colaboração em *slides* (Y) possui ativações nas três instâncias do UCLE dos usuários A, B e C, enquanto que o jogo infantil (X) possui ativações apenas nas instâncias de A e C. O servidor contém um repositório com as aplicações utilizadas neste ambiente.

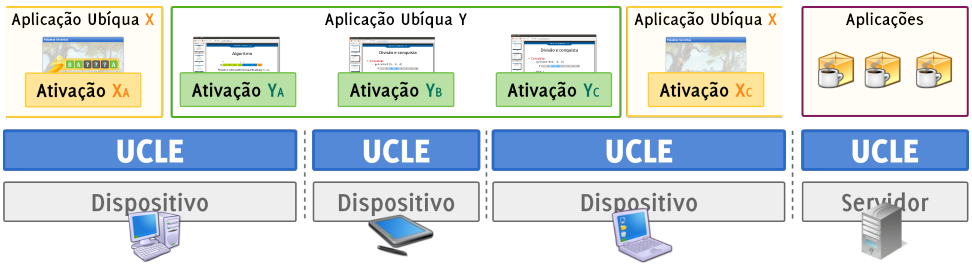

Figura 2. Relacionamento entre aplicação, ativação e aplicação ubíqua.

3.2. Conteúdo Ubíquo

Conteúdo Ubíquo é a estrutura lógica de compartilhamento de aplicações do UCLE. O corpo de um conteúdo ubíquo segue um formato definido pelo *middleware* que permite sua manipulação usando as primitivas do UCLE. Sua criação é feita em tempo de execução, em uma ativação, utilizando a operação de criação de conteúdo ubíquo. Esta operação vincula o conteúdo à ativação em execução.

O formato base de conteúdo ubíquo é a classe abstrata *ConteúdoUbíquo*, ver Figura 3. Esta classe define seus metadados, como ID, ativação, dono e estado de sincronização, que são utilizados pelas primitivas de compartilhamento. Os conteúdos ubíquos são construídos através da operação *criarConteúdoUbíquo(contUbiq : ConteúdoUbíquo) : ReferênciaUbíqua*. A chamada desta operação adiciona e vincula o conteúdo ubíquo ao espaço de armazenamento da ativação e retorna uma referência "ubíqua". Ele é acessado a partir desta referência utilizando a operação *get()*. Esta referência controla o acesso ao conteúdo, garantindo que este esteja em um estado consistente ao ser transferido.

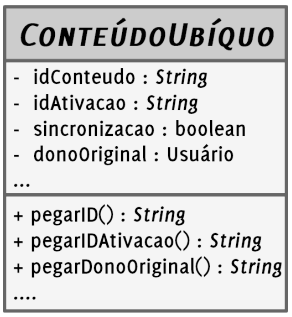

Figura 3. Classe Conteúdo Ubíquo

4. Primitivas de compartilhamento

As primitivas de compartilhamento no UCLE estão focadas no aspecto de mobilidade de conteúdo, visando fornecer um modelo de comunicação de alto nível para o desenvolvedor. Mais especificamente, elas suportam um conjunto de abstrações de compartilhamento que movimentam itens de conteúdo ubíquo entre ativações de uma mesma aplicação ubíqua. Essas abstrações são baseadas nas semânticas de mover, clonar e espelhar. Todas elas fornecem transparência de localização para o programador, ou seja, não é necessário especificar a localização ou dispositivo exato do usuário alvo para executá-las.

A primitiva **mover** provê suporte para a movimentação de itens de conteúdo ubíquo entre ativações de uma aplicação ubíqua. Ao mover um conteúdo ubíquo de um usuário para outro, ele deixa de existir na ativação de origem e passa existir na ativação do usuário alvo. Sua assinatura é: *mover(refUbi : ReferênciaUbíqua, usuário: Usuário)*.

A primitiva de **clonar** fornece a cópia de itens de conteúdo ubíquo entre usuários. Ela é interessante do ponto de vista da aplicação porque, ao copiar um conteúdo ubíquo, ele não é eliminado na origem, permitindo assim o compartilhamento de conteúdo seguido de seu desenvolvimento individual por parte de outro usuário (em outra ativação) da aplicação. Sua assinatura é: *clonar(refUbi : ReferênciaUbíqua, usuário : Usuário)*.

As primitivas descritas até agora não possuem efeito contínuo no compartilhamento, ou seja, após a operação, não é criado um vínculo entre o conteúdo ubíquo de origem e alvo. A primitiva **espelhar**, por outro lado, 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 ativação de origem quanto na ativação do usuário alvo, sendo que modificações no conteúdo ubíquo realizadas em qualquer ativação serão repassada as réplicas. Sua assinatura é: *espelhar(refUbi : ReferênciaUbíqua, usuário : Usuário)*.

5. Arquitetura

As primitivas de compartilhamento são implementadas por uma série de componentes específicos que colaboram entre si e que formam a arquitetura do UCLE. A arquitetura é híbrida, composta de duas partes, par-a-par (*peer-to-peer*) e cliente-servidor. A primeira é destinada aos dispositivos e a segunda para administração do ambiente de sala de aula. Os dispositivos dos usuários, que possuem ativações executando sobre o UCLE, são ao mesmo tempo clientes e fornecedores de conteúdos ubíquos, portanto, utilizam uma arquitetura par-a-par. A parte servidora, por sua vez, é responsável por armazenar aplicações e informações pertinentes ao ambiente, tais como endereço lógico dos usuários e aplicações ubíquas em execução. A arquitetura é a mesma para as duas partes, isto torna possível substituir o servidor, em caso de falha, por qualquer um dos pares (*peers*). A recuperação seria feita trocando os componentes do par (*peer*) pelos da parte servidora.

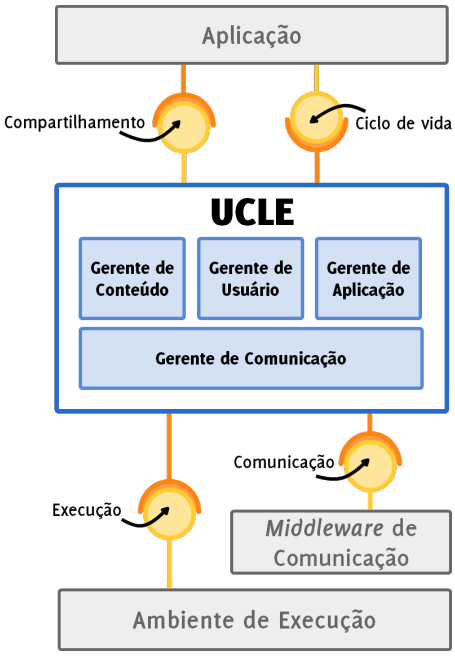

Figura 4. Arquitetura do UCLE

Os componentes do UCLE situam-se sobre o ambiente de execução da plataforma e um *middleware* de comunicação, como se vê na Figura 4. Este é utilizado para realizar troca de mensagens entre as diversas instâncias do sistema (pares e servidor). A semântica das primitivas é implementada através de interações entre os componentes sobre o gerente de comunicação. Interações remotas são enviadas a este, que as traduz para o *middleware* de comunicação subjacente. Desta forma, o UCLE pode ser implementado sobre diferentes sistemas de *middleware* de comunicação, incluindo, objetos distribuídos, *middleware* orientado a mensagens e espaço de tuplas, bastando, para isso, portar o gerente de comunicação. As primitivas oferecidas às aplicações são implementadas nos componentes de gerência de conteúdo, usuário, aplicação e comunicação. De maneira geral os gerentes realizam as seguintes tarefas:

- **Gerente de conteúdo**: Fornece uma interface de manipulação de conteúdo ubíquo para as ativações. Essa interface inclui métodos para criar, apagar, 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 usuário**: Gerencia a entrada e saída de usuários na rede. Fornece métodos convencionais, *i.e.*, cadastro, remoção, *login* de usuários e também específicos, como a tradução entre nomes de usuário e endereços dos dispositivos.
- **Gerente de aplicação**: Lida com aspectos de distribuição e gerência das aplicações. Na parte de distribuição, fornece operações para criar, apagar e listar aplicações disponíveis no ambiente. Gerencia o ambiente de execução através de operações para criar, pausar, retomar, entrar e sair de aplicações ubíquas.
- **Gerente de comunicação**: Traduz as requisições e respostas provenientes dos demais gerentes para o *middleware* de comunicação subjacente, é o barramento de comunicação do UCLE.

5.1. Interação entre componentes

A inserção de um par na rede é feita usando a operação de *login* no gerente do usuário do servidor. No êxito desta operação é registrado ou atualizado o endereço lógico do usuário, que abstrai a localização do usuário. Ao iniciar uma aplicação, é criado uma ativação utilizando o gerenciador de aplicação. Ela é adicionada à sua respectiva aplicação ubíqua, se não existir ela é criada. Também é alocado um espaço no gerente de conteúdo para o armazenamento de seus itens de conteúdo ubíquo.

Ambas operações, mover e clonar, são parecidas por não ser contínuas, isto é, elas não estabelecem um vínculo contínuo de compartilhamento entre o conteúdo origem e destino. O que as difere é a semântica fornecida à aplicação. A operação de mover remove o conteúdo ubíquo da ativação enquanto a de clonar não. Durante a transferência do item é realizado diversas interações entre os componentes, tanto local como remoto. A Figura 5 descreve o fluxo de interação dos componentes do UCLE para mover um item de conteúdo ubíquo de uma ativação para outra. As etapas deste processamento são:

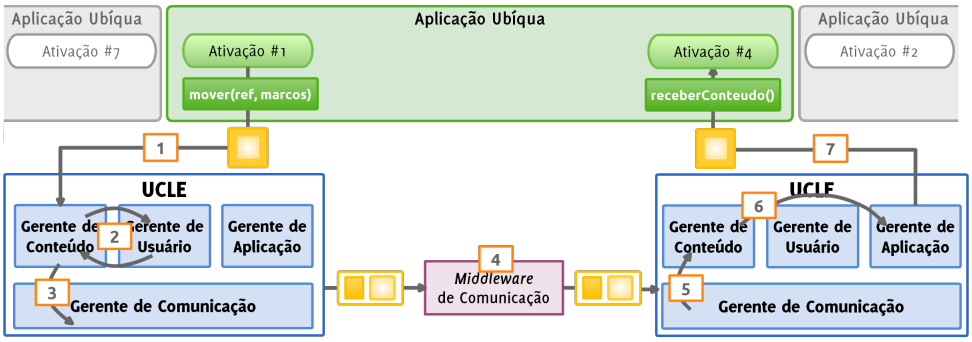

Figura 5. Fluxo de interação dos componentes para a operação mover

1. A ativação executa a primitiva *mover(ref, marcos)* do gerente de conteúdo, onde *ref* é a referência ao item de conteúdo ubíquo e *marcos* é o usuário alvo;
2. Utiliza-se a tradução de nomes para obter o endereço do usuário *marcos*;
3. Remove-se o conteúdo ubíquo do espaço de armazenamento da ativação, invalidando assim as referências a ele, ou seja, será gerada uma exceção ao tentar acessá-lo. Cria-se uma requisição para envio dele com destinatário *marcos*. Nesta requisição, é anexado metadados, como remetente e a aplicação ubíqua do item;
4. É feito a tradução seguida do envio da requisição para o *middleware* de comunicação subjacente. Ela é recebida no gerente de comunicação do *marcos*;
5. Repassa-se a requisição ao gerente de conteúdo que utilizando os metadados, vincula o item de conteúdo recebido com a ativação local;
6. É criado uma referência ao conteúdo recebido e feito um pedido de notificação de recebimento de conteúdo ao gerente de aplicação;
7. Utilizando-se o identificador de aplicação ubíqua é feito uma notificação à ativação local sobre o recebimento de conteúdo ubíquo.

A operação 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 entre as cópias. A primeira parte da operação é quase idêntica à clonar, acrescentando o fato de que o endereço da réplica é registrado no servidor. O servidor armazena filas com endereços de réplicas dos conteúdo ubíquo para poder

sincronizar as modificações. As interações com conteúdos ubíquos sincronizados são interceptadas pelo UCLE, com o objetivo de sincronizar as ações nas outras réplicas. Para isto, 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 contrário do estado inteiro. A escolha dessa estratégia é devido à réplica de estado ser pesada, visto que o estado inteiro seria replicado a cada interação com o conteúdo. A Figura 6 mostra o fluxo de interações entre os gerentes do UCLE para a sincronização de um conteúdo ubíquo. As etapas são:

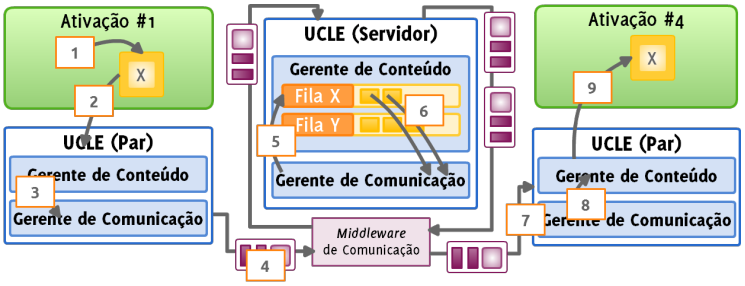

Figura 6. Fluxo de interação dos componentes na execução de um método em um conteúdo ubíquo sincronizado.

1. A ativação interage com o conteúdo ubíquo *X* através de uma chamada de método.
2. O UCLE intercepta este método através do uso de *proxies*;
3. Empacota-se o método, seus argumentos e metadados, em uma requisição com destino ao servidor.
4. O gerente de comunicação traduz e envia a requisição para o *middleware* de comunicação subjacente;
5. O servidor recebe a requisição e utilizando os metadados desta identifica a fila de réplicas onde esta operação deve ser executada.
6. Para cada réplica cria-se uma requisição contendo a operação a ser executada juntamente com os argumentos e metadados para cada réplica do conteúdo. Elas são despachadas ao *middleware* de comunicação;
7. Cada réplica recebe essa requisição. Desempacota-se a requisição e executa a operação com os respectivos argumentos sobre o conteúdo ubíquo;

6. Estudo de Caso

A aplicação *Palavras Secretas* foi desenvolvida para validar as primitivas e arquitetura do UCLE. Ela destina-se a crianças em fase de alfabetização e emprega uma metodologia simples que associa imagens com palavras. As palavras secretas são distribuídas entre os alunos e eles têm de acertar as letras para revelar as palavras. Estas palavras podem ser compartilhadas entre alunos, por exemplo, se algum aluno do grupo sabe uma letra, pode-se movê-la a ele. Os alunos interagem com essas palavras através de *touchscreen*. As Figuras 7 e 8 mostra a tela da aplicação e sua execução em *tablets* respectivamente.

A palavra secreta representa o item de compartilhamento nesta aplicação e, portanto, é modelada como um conteúdo ubíquo. Dentre seus atributos podemos listar a *string* que contém a palavra original, a descoberta, sua dica e imagem associada. Também foi modelado operações para inserção de letras, *porChar*(), efetuada pelos alunos. A Figura 9 mostra o modelo deste conteúdo ubíquo.

Figura 7. Tela da aplicação Palavras Secretas

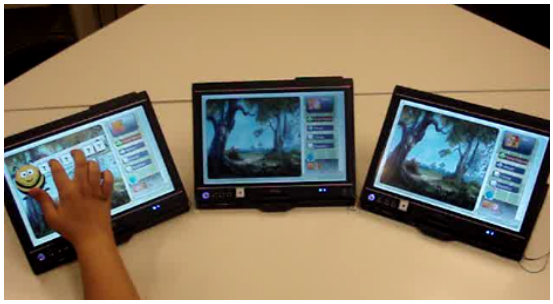

Figura 8. *Tablets* executando a aplicação Palavras Secretas.

A camada inteira de comunicação da aplicação foi mapeada para as primitivas do UCLE, ou seja, as interações entre os alunos foram decompostas à primitivas do UCLE. Entretanto, identificamos primitivas de compartilhamento não suportadas pelo UCLE. Particularmente, no suporte à mesclagem (*merge*) de conteúdos. Por exemplo, quando um aluno clona uma palavra secreta para outro, seria interessante que esse aluno pudesse aplicar modificações desenvolvidas pelo colega na cópia original.

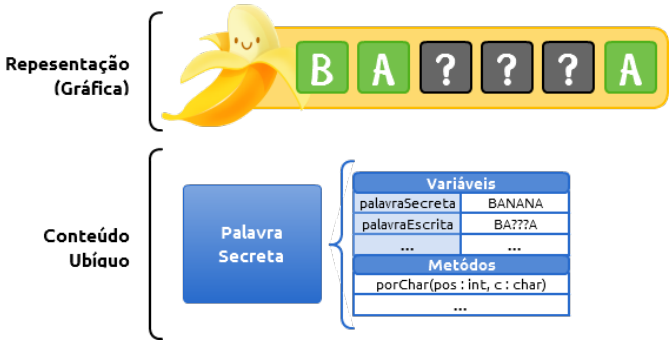

Figura 9. Modelo do conteúdo ubíquo palavra secreta.

7. Trabalhos Relacionados

Os trabalhos relacionados apresentados nessa seção visam apontar sistemas que propõem serviços similares àqueles desenvolvidos pelo UCLE. Dentre elas, identificamos o DOLCLAN, GOAL, OBIWAN, o Javanaise e a especificação *Life Cycle* de CORBA.

DOLCLAN [Bardram and Mogensen 2007] é um *framework* para suporte a compartilhamento de objetos distribuídos *peer-to-peer*, que realiza a distribuição física e sincronização dos objetos em todos os *hosts*, permitindo acesso local por qualquer aplicação. Essa abordagem pode gerar um alto custo causado pela excessiva troca de mensagens, uma vez que a atualização de um objeto implica na propagação desta operação em todos os *hosts*. O UCLE mantém as réplicas apenas nos hosts que utilizam os objetos, o que diminui a sobrecarga causada por alguma atualização. Além disso, o DOLCLAN não implementa as primitivas de clonar e mover objetos distribuídos.

O GOAL (**G**erenciamento de **O**bjetos de **A**prendizagem para o **L**OCAL) [Sonntag et al. 2010] provê um modelo de acesso a objetos de aprendizagem baseado em contexto em cima da plataforma LOCAL. O foco do trabalho é mais no modo de como os itens são acessados ao contrário de como eles são compartilhados entre os alunos.

O OBIWAN [Ferreira et al. 2003] é um *middleware* para desenvolvimento de aplicações distribuídas, que dá suporte ao compartilhamento de dados através de réplicas. Ele possui gerenciamento de memória distribuída (capaz de lidar com objetos replicados); suporte a recuperação automática de réplicas e; definição e suporte a políticas de segurança baseadas em histórico coletado pelos agentes. Entretanto, o OBIWAN não dá

suporte as operações de clonar e mover objetos, se restringindo apenas a replicação, o que também é oferecida pelo UCLE através da primitiva de espelhamento.

Javanaise [Hagimont and Boyer 2001] é um *middleware* para desenvolvimento de aplicações colaborativas, que baseia-se na replicação de objetos armazenados em um *cluster*. Ele provê uma primitiva de movimentação para estes. Ao mover um *cluster* ele cria um *proxy* na origem que referencia o destino da operação. O Javaneise não provê suporte a primitivas de clonar e espelhar.

A especificação de ciclo de vida (*Life Cycle*) de objetos distribuídos em CORBA [Object Management Group 2002] provê primitivas semelhantes (mover e clonar) a do UCLE para objetos distribuídos. Essas primitivas fazem o uso por meio de "fábricas" distribuídas. Estas fábricas devem ser construídas pelo programador, pois ele deve saber como recriar o objeto, esta etapa não é requerida pelo UCLE. A especificação também lida com movimentação de objetos interconectados, *i.e.*, *clusters* de objetos, o que não é suportado ainda pelo UCLE. Porém, ao contrário do UCLE não há primitivas contínuas de compartilhamento, *i.e.*, para espelhar objetos.

8. Trabalhos Futuros e Conclusão

O UCLE é uma plataforma de *middleware* que torna transparente para o programador de aplicações colaborativas diversos problemas envolvidos no compartilhamento de conteúdo. Ele abstrai estes problemas, modelos diferentes de comunicação, concorrência, representação externa, através de um conjunto de primitivas de compartilhamento de conteúdo ubíquo. Estas primitivas contemplam as semânticas de mover, clonar e espelhar.

A avaliação do UCLE visou validar as primitivas de compartilhamento desenvolvidas. Para isso, foi criada uma aplicação para ambientes de sala de aula inteligente, denominada *Palavras Secretas*. Ela tem por objetivo auxiliar crianças que estão em fase de alfabetização, por meio de associação de imagens à palavras. Grande parte das operações desta aplicação foram cobridas pelas primitivas do UCLE, entretanto identificamos algumas primitivas adicionais inexistentes.

Como trabalhos futuros pretendemos explorar novas primitivas de compartilhamento de conteúdo para sala de aula inteligente. Além disso, queremos estender estas primitivas para suportar conteúdos ubíquos complexos, *i.e.*, interconectados com outros conteúdos ubíquos e "normais".

9. Agradecimentos

Este trabalho foi parcialmente financiado pela DELL. Os autores gostariam de agradecer a todos do Grupo de Pesquisa Aplicada à Internet e Sistemas Distribuídos do INF/UFG.

Referências

[Anderson et al. 2007] Anderson, R. R., Davis, P., Linnell, N., Prince, C., Razmo, V., Videon, F., and Razmov, V. (2007). Classroom Presenter: Enhancing Interactive Education with Digital Ink. *Computer*, 40(9):56–61.

[Bardram and Mogensen 2007] Bardram, J. and Mogensen, M. (2007). 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*, pages 119–132. Springer-Verlag.

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

[Ferreira et al. 2003] Ferreira, P., Veiga, L., and Ribeiro, C. (2003). OBIWAN: design and implementation of a middleware platform. *Parallel and Distributed Systems, IEEE Transactions on*, 14(11):1086–1099.

[Hagimont and Boyer 2001] Hagimont, D. and Boyer, F. (2001). A configurable RMI mechanism for sharing distributed Java objects. *IEEE Internet Computing*, 5(1):36–43.

[Hourcade et al. 2004] Hourcade, J. P., Bederson, B. B., and Druin, A. (2004). Building KidPad: an application for children's collaborative storytelling. *Software: Practice and Experience*, 34(9):895–914.

[Hurford and Hamilton 2008] Hurford, A. and Hamilton, E. (2008). Effects of tablet computers and collaborative classroom software on student engagement and learning. In *38th Annual Frontiers in Education Conference*, pages S3J–15–S3J–20. IEEE.

[Object Management Group 2002] Object Management Group (2002). CORBA 1.2 Life Cycle Service Specification.

[Patten et al. 2006] Patten, B., Arnedillo Sánchez, I., and Tangney, B. (2006). Designing collaborative, constructionist and contextual applications for handheld devices. *Computers & Education*, 46(3):294–308.

[Pettersson 2011] Pettersson, O. (2011). *Towards a Mobile Learning Software Ecosystem*. PhD thesis, Linnaeus University.

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

[Skipton et al. 2011] Skipton, C., Matulich, E., Papp, R., and Stepro, J. (2011). Moving From "Dumb" To "Smart" Classrooms: Technology Options And Implementation Issues. *Journal of College Teaching & Learning (TLC)*, 3(6):19–27.

[Sonntag et al. 2010] Sonntag, N. L., Barbosa, D. N. F., Barbosa, J. L. V., and Pinto, S. C. C. d. S. (2010). Gerenciador de Objetos de Aprendizagem para um Ambiente de Educação Ubíqua. In *I Simpósio Brasileiro de Computação Ubíqua e Pervasiva*.

[Springer et al. 2008] Springer, T., Schuster, D., Braun, I., Janeiro, J., Endler, M., and Loureiro, A. a. F. (2008). A flexible architecture for mobile collaboration services. *Proceedings of the ACM/IFIP/USENIX international middleware conference companion on Middleware '08 Companion - Companion '08*, pages 118–120.

[Steimle et al. 2009] Steimle, J., Brdiczka, O., and Muhlhauser, M. (2009). CoScribe: Integrating Paper and Digital Documents for Collaborative Knowledge Work. *IEEE Transactions on Learning Technologies*, 2(3):174–188.

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

[Yau et al. 2003] Yau, S. S., Gupta, S. K. S., Karim, F., Ahamed, S. I., Wang, Y., Wang, B., and Science, C. (2003). Smart Classroom: Enhancing Collaborative Learning Using Pervasive Computing Technology. In *ASEE 2003 Annual Conference and Exposition*, pages 13633–13642.