CMFRAME: a Framework for Managing Dynamic and Hierarchical Context Histories

Felipe Lauermann Vielitz
Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo - RS, Brasil
felipevielitz@gmail.com

Márcio Garcia Martins
Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo - RS, Brasil
marciog@unisinos.br

Jorge Luis Victória Barbosa
Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo - RS, Brasil
jbarbosa@unisinos.br

Kleinner Farias de Oliveira
Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo - RS, Brasil
kleinnerfarias@unisinos.br

Lucas Pfeiffer Salomão Dias
Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo - RS, Brasil
lucaspfsd@gmail.com

Alexandre Wolf
Programa de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo - RS, Brasil
wolfalexandre@gmail.com

***Abstract*—With the growing availability of devices capable of capturing information about their surroundings and the expansion of mobile connectivity, Internet of Things (IoT) solutions are increasingly been integrated into society. For IoT solutions to emerge successfully on the market, they will employ more than traditional mobile computing, but they will also require the use of everyday objects in an interconnected way. This interconnected world will support the intelligence in environments. In order to provide this infrastructure for environments, it will be necessary to propose platforms for software development based on context awareness and context processing. In this sense, this article proposes CMFrame, a framework for managing contextual information captured from physical environments using hierarchical and dynamic entities. CMFrame allows that entities to modify their hierarchical organization to manage environments and their related contexts. Contexts linked to each entity are also dynamic and can store different amounts of values at any time. The article presents the proposed framework and its evaluation through two applications focused on intelligent enviroments. The fisrt is dedicated to monitor the movements of entities in an environment, and the second addresses energy monitoring. The scientific contribution of CMFrame is the proposal to abstract the management of dynamic and hierarchical context histories through a framework.**

Keywords— framework, context awareness, context histories

I. INTRODUÇÃO

Em 2001, Satyanarayanan [1] expandiu a visão de Weiser [2] ao definir que a computação ubíqua, ou computação pervasiva, seria uma evolução dos sistemas distribuídos e computação móvel. A aplicação da computação ubíqua vem sendo pesquisada em diferentes áreas do conhecimento, tais como saúde [3][4], bem-estar [5], gerenciamento de competências [6], educação [7][8], comércio [9], acessibilidade [10] e agricultura [11].

Através da computação ubíqua surgem os ambientes inteligentes que contemplam a fusão de dois mundos, o físico e o virtual, permitindo a captação de dados de sensores e o controle de um mundo a partir de outro. Essa integração permite o desenvolvimento de ambientes capazes de proatividade, onde sensores se mesclam ao ambiente, tornando-se transparentes à percepção do usuário e eliminando a necessidade de interagir explicitamente com eles [12].

Ambientes proativos requerem uma análise robusta das condições que os cercam, tornando necessário a existência de uma coleção de informações de contexto [13] e ações do usuário. Quando essas informações registram o passado, sendo denominadas Históricos de Contextos [9][10][14]. Os históricos, por sua vez, refinam a forma como o ambiente se apresenta, devido a sua capacidade de permitir a extração de informações relevantes, as quais registram o comportamento daquele ambiente ao longo do tempo.

De modo a integrar os elementos que compõem um ambiente, serão empregados, nesse artigo, os conceitos de entidades reais e virtuais, que designam que alguns elementos de um ambiente são representados através de uma entidade real, ou física, enquanto que outros por uma entidade virtual.

Esse artigo propõe o *framework* CMFrame, um gerenciador de históricos de contextos que suporta entidades reais e virtuais e introduz uma hierarquia para as mesmas. Tal hierarquia tem por objetivo definir a relação que cada entidade tem com as demais em um sistema ou aplicação que gerencie um ambiente. Ao longo do ciclo de vida do sistema, as entidades poderão modificar sua hierarquia, de forma dinâmica, e assim melhor representar um ambiente mutável, com entidades móveis que não se encontram restritas a apenas um ambiente. A esse conceito é atribuído o nome de Entidades Hierárquicas e Dinâmicas.

O texto se encontra organizado em 5 seções, onde a seção 2 apresenta os trabalhos que possuem relação com o *framework* e uma comparação entre eles. O framework proposto, com a descrição de sua estrutura e funcionalidades é exposto na seção 3. A seção 4 descreve o cenário e avaliação do *framework*. Por fim, a seção 5 apresenta conclusões e trabalhos futuros.

TABELA 1. COMPARATIVO DE TRABALHOS RELACIONADOS

Critérios	**CPA** Lee et al. [15]	**MDA** Coninck et al. [18]	**DTLC** Younis e Moayeri [19]	**ezContext** Martin, Lamsfus, e Alzua [20]	**CLET** Kotevska, Lbath e Bouzeframe [21]	**FrameTrail** Martins [22]
Mapeia informações de contexto	Sim	Sim	Sim	Sim	Sim	Sim
Oferta funcionalidades de *framework*	Sim	Não	Sim	Sim	Sim	Sim
Mantém histórico de contexto	Sim	Não	Não	Sim	Sim	Sim
Ambientes com entidades	Sim	Sim	Sim	Sim	Sim	Sim
Local para armazenamento de dados	Remoto	Não descrito	Não descrito	Não descrito	Remoto	Local/ Remoto
Estrutura de contextos com hierarquia	Não	Não	Não	Não	Não	Não
Arquitetura do modelo	Cliente-Servidor	Cliente-Servidor	*Peer-to-Peer*	Não descrita	Cliente-Servidor	Não descrita

II. TRABALHOS RELACIONADOS

O trabalho de Lee et al. [15] oferece um *framework* que captura diferentes tipos de contextos em dispositivos móveis, guarda-os em um repositório de contextos e realiza inferências sobre suas informações para gerar dados de predição acerca de um usuário como, por exemplo, seu possível comportamento futuro. A proposta dos autores é torná-lo reutilizável entre diversas aplicações móveis que lidam com Sistemas Ciber-físicos (CPS) [16][17], definidos como sistemas de alta conexão entre elementos físicos do mundo e sistemas de computação, que interagem entre si com base nas percepções das informações de contexto. As funcionalidades presentes são oferecidas na forma de serviços, e seus consumidores podem utilizá-las a qualquer momento e em qualquer local. Através da distribuição de funcionalidades complexas, diminui-se a carga nos clientes, e se torna possível substituir dinamicamente um determinado serviço, se necessário.

Coninck et al. [18] propõem um *middleware* para sistemas CPS baseados em serviços, em um cenário utilizando-se robôs, atuadores e sensores em conjunto com a nuvem. O *middleware* ainda provê técnicas para desenvolver componentes com base em redes neurais, cuja popularidade cresceu recentemente na análise de dados sensoriais, bem como controle de robôs.

Younis e Moayeri [19] apresentam um *framework* voltado à otimização do fluxo de tráfego veicular em interseções, empregando luzes dinâmicas em semáforos. Sua proposta envolve a instalação de dispositivos computacionais ao longo das estradas, com o auxílio de controladores de luz de tráfego (TLC) em cada semáforo para controlar seu estado.

O ezContext [20] propõe um *framework* para o gerenciamento automático do ciclo de vida de dados de contexto, bem como uma extensão para dispositivos móveis e a disponibilização de ferramentas para auxiliar os desenvolvedores no estágio de prototipação e testes.

O trabalho de Kotevska, Lbath e Bouzeframe [21] apresenta um *framework* com arquitetura de *cloudlets*, definidos como nodos locais que influenciam um determinado território dividindo as requisições dos usuários em quadrantes, para modelos de predição de requerimentos de usuários para dispositivos móveis, em tempo real.

O FrameTrail [22] propõe um *framework* para auxiliar o desenvolvimento de aplicações que empregam históricos de contextos, de forma genérica, através de seus provedores de contexto, dados e comunicação.

A Tabela 1 apresenta uma comparação dos trabalhos relacionados. Após a análise dos trabalhos foi constatado que, embora eles ofertem soluções voltadas a históricos de contextos, nenhum apresentou uma forma de tratar os contextos como entidades que possuam características dinâmicas, no sentido de oferecer recursos que permitam seu uso em ambientes cujos elementos internos se modifiquem constantemente ao longo do tempo. A partir desses fatores, surge a necessidade de exploração desse conceito, oferecendo formas de gerenciar as informações vindas dos diversos elementos que compõem um ambiente altamente mutável. O *framework* proposto nesse artigo oferta essas características através do uso de entidades hierárquicas e dinâmicas, bem como o emprego de contextos dinâmicos durante todo o ciclo de vida da aplicação.

III. FRAMEWORK CMFRAME

O CMFrame suporta o gerenciamento de históricos de contextos em ambientes compostos por objetos de Internet das Coisas (IoT) [23][24], como sensores e atuadores, através de entidades que podem ser representadas fisicamente, na forma de entidades reais, ou virtualmente, como entidades virtuais.

O *framework* faz uso de entidades hierárquicas e dinâmicas, e suas características designam que cada entidade se encontra presente em um nível diferente dentro de uma hierarquia definida, permitindo que sejam consideradas como individuais dentro da hierarquia (entidades base) ou que possuam outras entidades dentro de si (entidades *container*), assim gerando um agrupamento de entidades. Considerando que contextos estão atrelados a uma entidade, é possível que uma determina entidade *container* tenha dentro de si os

contextos pertencentes à outra entidade. O seu fator dinâmico define que, durante a execução das aplicações que venham a usufruir do *framework*, a hierarquia pode sofrer modificações, mais especificamente, as entidades podem mudar de nível hierárquico. Dessa forma, é permitido as entidades: se tornarem entidades individuais caso façam parte de um grupo; aderirem a um grupo de entidades, na forma de nodos filhos; ou ainda apenas mudarem de grupo.

A mudança de hierarquia deve ser solicitada ao CMFrame, por parte de uma aplicação que faça uso de suas funcionalidades, informando qual entidade pertencerá a outra, ficando a cargo da aplicação designar o momento oportuno para essa mudança.

Esse dinamismo permite a melhor representação de ambientes altamente mutáveis, que para seu melhor funcionamento realizam trocas constantes de elementos internos, como no caso de diversas entidades móveis que transitam entre múltiplos ambientes, como robôs, ou até mesmo fábricas altamente automatizadas, cujas linhas de montagem necessitem trocar elementos específicos para atender a produção de um novo produto.

As entidades hierárquicas e dinâmicas que o *framework* propõe podem ser empregadas em conjunto com o conceito de entidades reais e virtuais, como uma forma de organizar os elementos de um ambiente em uma hierarquia, definindo a relação que cada entidade tem com as demais. Todos esses objetos se encontram interligados em uma rede local, normalmente *wireless*, coletando dados constantemente e trocando informações entre si. Dessa forma, torna-se possível aplicar o CMFrame em casos que contemplem diversas configurações de ambientes inteligentes, bem como os aplicados a sistemas CPS.

Dentro da estrutura do CMFrame, uma entidade representa um elemento do sistema, que pode ser tanto de caráter físico quanto virtual, e que realize uma função mais complexa que apenas um sensor ou atuador individualmente posicionado em um ambiente. A entidade completamente virtual não existe fisicamente no ambiente, e normalmente representa um nodo de processamento dentro do sistema, como uma aplicação cuja lógica e processamento afete o ambiente em que se encontra.

É obrigatório que a entidade gere dados a partir de um sensor ou conjunto de sensores, ou que modifique o sistema através de seus atuadores, seja de forma física através de um nodo que represente uma entidade física, ou por um nodo que represente um elemento completamente virtual. Uma entidade é capaz de representar diversos elementos de um ambiente, e até mesmo o próprio ambiente. Dentre alguns desses elementos pode-se citar: pessoas; um robô, sendo ele uma entidade móvel ou estática; uma unidade de processamento e que agregue sensores (nodo físico); uma sala com diversos componentes inteligentes em suas dependências, com seus sensores e atuadores; um andar, contendo diferentes salas e até mesmo um prédio, contendo uma coleção de salas.

A. Contextos Dinâmicos

O CMFRAME propõe o conceito de Contextos Dinâmicos. Contexto para ambientes inteligentes, refere-se a toda informação passível de captação por um objeto inteligente [25] dentro de um ambiente físico que seja considerada relevante para que uma determinada aplicação possa desempenhar suas funções. No CMFrame, contextos se encontram atrelados a uma entidade, e essa por sua vez obedece a uma hierarquia. Cada entidade é hierárquica e dinâmica, enquanto que os contextos atrelados a cada entidade são apenas dinâmicos, pois podem armazenar diferentes quantidades de dados daquelas que foram armazenadas previamente.

Um exemplo da utilização de contextos dinâmicos é que em um momento, o microcontrolador (entidade) atrelado ao sensor de umidade pode realizar inferências adicionais e assim prover informações extras a base de dados, como em um caso ser calculado que o valor 320, para um sensor, corresponde a 68% de umidade no ambiente. Essa adição de informações não traz detrimento algum aos valores anteriormente inseridos pela mesma entidade, e não determina que no futuro esse valor adicional se torne obrigatório.

Além de informações de contexto adicionais, o uso desse fator dinâmico permite que a aplicação armazene dados completamente distintos uns dos outros, para a mesma entidade, sem a necessidade de uma reestruturação do *framework* ou da definição de um padrão de mensagens para cada microcontrolador. Nesse sentido, torna-se necessário somente que exista um identificador conhecido pela aplicação para que essa informação seja acessível posteriormente, como pode ser visto no caso do campo *CurrentAmbient*, que descreve o ambiente na qual aquela entidade móvel se encontra fisicamente. Outra vantagem do fator dinâmico está na possibilidade de empregar sensores e atuadores que ainda não foram desenvolvidos no momento que este *framework* foi proposto, visto que ele não impõe uma limitação acerca de quais informações podem ser salvas.

A cada entidade é atrelado um conjunto de informações de contexto. No CMFrame, uma informação de contexto obrigatória para todas as entidades é o tempo em que os demais dados de contexto de uma entidade foram gerados. Isso ocorre devido ao framework empregar, como mecanismo de indexação de contexto, as informações referentes ao tempo em que esses dados foram gerados e a que entidade eles pertencem, sendo que quaisquer mecanismos de indexação adicionais ficam a cargo do utilizador do *framework*.

O desenvolvedor utiliza classes que representam as entidades base e *container* para modelar o comportamento das entidades. Cada uma das entidades é inicializada e configurada com informações básicas, que dizem respeito aos sensores e atuadores acoplados a si, dentro de um ambiente. O desenvolvedor pode especializar tais classes a fim de prover um comportamento customizado a seus objetos, dessa forma, quando as informações de contexto forem recebidas, as funções especializadas serão acionadas, provendo o comportamento customizado.

O *framework* considera aspectos voltados ao contexto de IoT e seus objetos interconectados, trazendo à tona preocupações relacionadas a geração de um grande conjunto de dados, bem como sua grande variedade. O CMFrame inclui um módulo de banco de dados em suas definições, garantindo o gerenciamento de suas funcionalidades de históricos de contextos. Os requerimentos referentes à escalabilidade e disponibilidade em sistemas que armazenam dados de sensores resultaram em uma busca por bases de dados NoSQL, que são capazes de adicionar dinamicamente novos atributos aos seus registros, além de distribuir eficientemente os dados entre múltiplos servidores.

B. Estrutura do CMFrame

Optou-se pela organização do *framework* em duas partes, devido a questões técnicas e de forma a facilitar a implementação de suas funcionalidades. Dessa forma, o CMFrame consiste em um conjunto de componentes distribuídos em dois módulos. Para melhor representa-los, a parcela do *framework* que implementa a conexão com a base de dados é denominada CMFDatabase, e o módulo que foi incorporado a todos os dispositivos que desejam se comunicar com a base de dados é chamado de módulo CMFrame, apesar de ambos fazerem parte do *framework* e serem necessários para o seu funcionamento.

A Fig. 1 mostra o diagrama de classes do módulo CMFDatabase. Dentre as principais classes, destacam-se CFMController, DataManager, CommunicationManager, e o MessageInterpreter. O CFMController integra alguns dos comportamentos internos do *framework* e fornece os métodos de acesso as funcionalidades presentes no mesmo. Nesse módulo, o DataManager é a classe responsável por estruturar os dados recebidos e realizar a integração com o banco de dados NoSQL, além de traduzir os dados vindos do banco para um conjunto de informações que possa trafegar pela rede e ser interpretado por um aplicativo que utiliza o CMFrame.

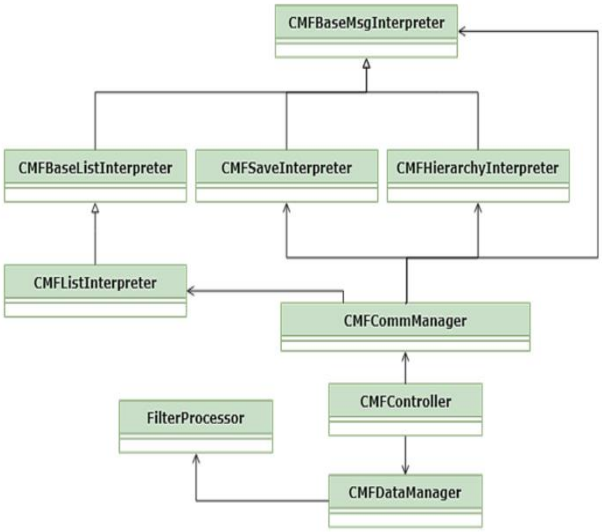

Fig. 1. Diagrama de classes CMFDatabase.

As classes relacionadas ao CMFMessageInterpreter realizam o tratamento de mensagens que são enviadas a esse módulo, contendo, portanto, todo o comportamento sobre as ações que devem ser realizadas para cada tipo de mensagem recebida com base no protocolo do *framework*. Podem ser citados entre os tratamentos possíveis, o tratamento de mensagens de armazenamento, de trocas hierárquicas, e dos interpretadores de listas. Sua expansão é possível e necessária para a criação de novas mensagens, em padrões novos e que incluam outros comportamentos não originalmente previstos na implementação do *framework*. O CommunicationManager realiza a comunicação externa ao *framework* como, por exemplo, a comunicação com o servidor de armazenamento, que pode ser realizada através de métodos REST (Representational State Transfer), que definem uma maneira uniforme de acesso a recursos de web services, através do protocolo HTTP. Esse componente também pode ser implementado pelo usuário desde que respeite o protocolo de comunicação definido internamente no *framework*.

O outro módulo pode ser observado na Fig. 2, que mostra a estrutura do CMFrame. Este módulo é inserido junto ao código do aplicativo que faz uso de suas funcionalidades. Dentre as estruturas presentes, pode-se mencionar CFMController, DataManager, CommunicationManager e o EventManager. É possível notar que alguns dos componentes possuem nomenclaturas comuns entre os dois módulos do *framework*, porém internamente conservam características distintas atreladas ao funcionamento particular de cada um. O CMFController e o CommunicationManager desempenham a mesmas funções gerais dentro do módulo CMFrame do que as vistas no CMFDatabase, porém o CommunicationManager apresenta uma implementação mais robusta nesse módulo, a fim de incorporar as diversas funcionalidades necessárias para a comunicação com o módulo da base de dados. As classes CMFFilter e CMFFilterData correspondem a filtros utilizados nas requisições por contextos, configurando seu comportamento conforme as necessidades do usuário.

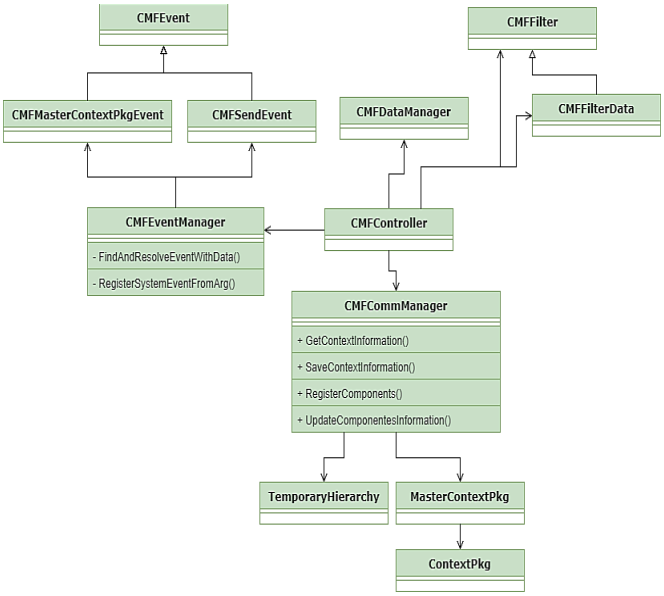

Fig. 2. Diagrama de classes CMFrame.

O DataManager, por sua vez, não possui a integração direta com nenhum tipo de banco de dados, mas desempenha outras funções de conversão de estruturas e criação de filtros para o *framework*. O EventManager é responsável pelas atividades relacionadas aos eventos suportados pelo *framework*, como os mecanismos de monitoramento de variáveis de contextos relevantes para uma determinada aplicação e aqueles relacionados ao envio de mensagens do sistema que retornem informações ao aplicativo. A expansão do DataManager permite também a criação e modificação dos filtros básicos para buscas na base de dados. Em relação ao CommunicationManager é possível a criação de novas estruturas de envio de mensagens do sistema, expandindo a funcionalidade do *framework*, bem como a implementação de outros protocolos de transmissão de dados.

O CMFrame armazena as informações de contexto tanto dos sensores quanto dos atuadores, visto que a informação sobre o estado dos atuadores em um ambiente, em certo período no tempo, é de suma importância para aplicações que desejam descobrir o último estado válido para um determinado ambiente, e pode ser aplicada a sistemas CPS, onde sempre é necessário ter uma configuração válida para onde o sistema possa retornar.

O processo de armazenamento tem início a partir de um nodo físico ou virtual, após um processo de configuração, como a classificação do nodo físico como uma entidade estática ou móvel, bem como sua informação de contexto de localização. Uma entidade estática classifica o nodo físico como um elemento que permanecerá sempre no mesmo ambiente, enquanto que móvel define que ele pode ser um controlador atrelado a um robô, por exemplo, caso este tenha sensores e atuadores que contribuam para o ambiente. Caso o dispositivo seja móvel, os valores de sua localização atual devem ser enviados em conjunto com as demais informações de contexto, caso contrário serão omitidos a fim de economizar na transmissão de dados. Posterior a essa etapa, o utilizador pode fazer envios de dados de contexto na cadência que for apropriada para seu uso.

Os dados mínimos de contexto exigem que seja informado a data em que a informação de contexto foi coletada, a fim de manter o registro de histórico, algum tipo de identificador utilizado pelo usuário para posterior consulta, bem como os dados provenientes do sensor em questão. Cada sensor ou atuador pode fornecer qualquer número de parâmetros adicionais, e não é necessário seguir o mesmo padrão entre envios. É possível, por exemplo, enviar um dado adicional a um sensor de temperatura em dado momento, e no próximo envio omiti-lo completamente, aproveitando então o fator dinâmico dos contextos.

As requisições por dados de contexto podem ser feitas para uma entidade, especificando seu identificador, um filtro que define quais dados da entidade se deseja adquirir, e uma função a ser retornada com os valores de contexto coletados. A busca por contextos em hierarquia, por sua vez, também segue a mesma parametrização, porém nesse caso toda a hierarquia é percorrida a partir da entidade desejada, agregando os dados pertinentes com base no filtro definido pela aplicação. Também é possível utilizar suas funcionalidades de armazenamento de contexto sem a utilização do CMFrame, desde que o protocolo de comunicação do *framework* seja obedecido e se faça uso de uma implementação do protocolo MQTT.

IV. Avaliação

A etapa de avaliação tem como intuito verificar as contribuições e limitações do *framework* proposto. O *framework* suporta a implementação de aplicações que se utilizem de históricos de contextos para a realização de inferências, e que empreguem o conceito de entidades hierárquicas e dinâmicas. Para tanto, foi realizado o desenvolvimento de um cenário englobando diferentes ambientes, em uma representação de uma casa inteligente. Posteriormente o CMFrame foi aplicado no desenvolvimento de dois aplicativos, denominados AmbientMonitor e EnergyMonitor, que fazem uso das funcionalidades providas pelo *framework* no quesito de gerenciamento de históricos de contextos, empregando suas características particulares de entidades hierárquicas e dinâmicas, e averiguando as contribuições e limitações de seu uso em sistemas sensíveis a contexto.

A. Cenário de uma Casa Inteligente

O cenário proposto visa representar uma casa inteligente, que engloba três ambientes distintos: sala, cozinha e quarto, dispostos conforme Fig. 3. Todos os ambientes contêm microcontroladores responsáveis por agregar sensores estrategicamente distribuídos. Esse cenário possui dois usuários, X e Y, que podem se mover livremente entre qualquer um dos ambientes. Cada um dos usuários possui um cartão inteligente virtual, que se encontra em um aplicativo em seu *smartphone*, e possui um identificador único para o usuário. Vinculado a esse identificador se encontram informações de contexto armazenadas na base de dados do *framework*, representando um perfil próprio para aquele usuário.

Como um dos elementos utilizados para avaliar a funcionalidade de entidades hierárquicas e dinâmicas do CMFrame, foi empregado o uso de entidades móveis, representadas pelos usuários X e Y, e por um robô de limpeza, que reside no corredor, e pode se mover livremente entre qualquer um dos três ambientes. Periodicamente, o mesmo irá mudar de ambiente para desempenhar sua função.

Os dados armazenados na base de dados do CMFrame foram simulados, tendo em vista valores condizentes com os sensores propostos. A simulação foi realizada através de um aplicativo para realizar inserções na base de dados. Inicialmente, AmbientMonitor e o EnergyMonitor foram desenvolvidos com todas as regras necessárias para a execução dos passos estabelecidos no cenário proposto. Em seguida, o cenário foi montado com os dados iniciais, e ambos os aplicativos foram inicializados e conectados à mesma instância do CMFDatabase. Os dados necessários para simular as situações ocorridas no cenário foram então inseridos diretamente na base de dados. Dessa forma foi possível replicar o comportamento de um ambiente real, que recebe dados cronológicos de contexto, formando um histórico ao longo do tempo. Considerando que os aplicativos verificam pelos últimos dados de contexto presentes, e que não existem ações no fluxo do cenário próximas entre si que sejam dependentes umas das outras, se torna possível empregar o método de inserção de dados pelas etapas do cenário.

A avaliação realizada sobre o aplicativo AmbientMonitor visa averiguar os mecanismos de troca hierárquica, e se os mesmos apresentam contribuições para seus utilizadores.

O EnergyMonitor, por sua vez, tem sua avaliação mais direcionada ao gerenciamento de um aspecto específico (gerenciamento de energia), mas que diz respeito a sensores e atuadores que se encontram presentes em mais de um ambiente.

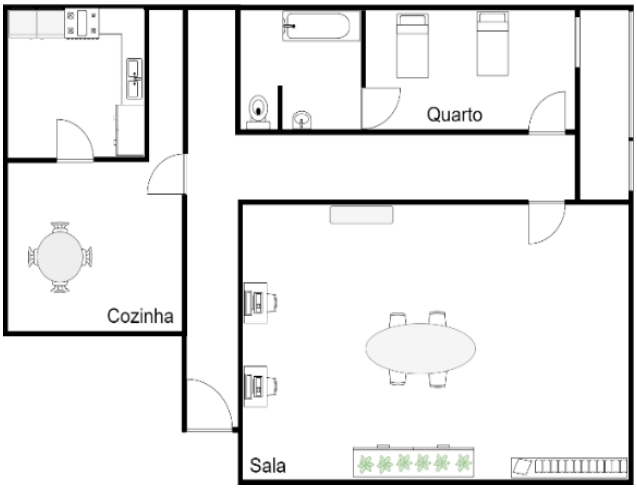

Fig. 3. Cenário Casa Inteligente.

B. *AmbientMonitor*

O AmbientMonitor tem como objetivo monitorar o estado atual de cada um dos ambientes da casa inteligente, bem como atuar sobre eles no momento em que é identificada uma mudança nos dados de contexto que implique em uma atuação. As atuações modificam o comportamento de um ou mais elementos do ambiente, através da mudança nos parâmetros de um atuador. A Fig. 4 ilustra a tela do aplicativo, que exibe cada um dos sensores relevantes ao propósito da aplicação, com base no ambiente em que se encontram.

As mudanças de hierarquia são sempre solicitadas ao CMFrame pelo aplicativo, a seu critério. No AmbientMonitor, esse critério é a análise dos dados de contexto vindos dos sensores de presença, sempre mantendo o local onde a entidade se encontra através da hierarquia. O aplicativo armazena cada uma das trocas de hierarquia dos usuários em seus respectivos históricos, através do uso de contextos dinâmicos fornecido pelo *framework*.

Os sensores de presença utilizam a mesma característica dinâmica, visto que seus dados somente são enviados pelo microcontrolador quando esse sensor é ativado. Com isso, a quantidade de informações armazenada naquele instante para a entidade (microcontrolador) difere dos demais. Através da hierarquia, o AmbientMonitor consegue detectar em qual ambiente o robô se encontra, bem como se está limpando o ambiente, qual a posição onde a limpeza ocorre, o número de limpezas diárias, o momento em que começa a limpeza, e o instante em que termina, através da verificação de contextos de um ambiente específico.

Fig. 4. Tela do AmbientMonitor.

No cenário, o aplicativo do *smartphone* de cada usuário, que os representa como entidades móveis, possui identificadores (ID) 4 e 5, para os usuários X e Y, respectivamente. Esse aplicativo também é responsável por informar quando os mesmos trocam de ambiente, e armazenar no perfil. Com base nessas informações, o AmbientMonitor verifica os dados de contexto apropriados para detectar o deslocamento do usuário, e solicita ao seu módulo do CMFrame pela troca na hierarquia e pelo armazenamento no histórico da entidade do ambiente em que ele se encontra. Dessa forma, o AmbientMonitor centraliza ambas as funções e permite a avaliação sob o ponto de vista de aplicativo único.

As informações de contexto são coletadas da base de dados de históricos a cada cinco segundos, atualizando a tela principal da aplicação com os últimos dados de contexto que foram armazenados pelos sensores e atuadores do ambiente, portanto refletindo as últimas informações coletadas no cenário e presentes no histórico. Esses dados por sua vez são analisados para verificar a necessidade de atuações por parte do aplicativo.

O AmbientMonitor faz uso das entidades hierárquicas e dinâmicas para manter a representação do ambiente constantemente atualizada, refletindo as condições físicas dos ambientes em sua representação virtual, através das entidades. Isso significa que conforme um elemento do mundo físico troca de ambiente, o aplicativo realiza uma ação correspondente de troca de hierarquia, informando o *framework* sobre o novo local (entidade) onde aquela entidade móvel se encontra.

A partir dessa hierarquia atualizada, o aplicativo pode solicitar os dados de uma entidade específica, que irá fielmente representar as condições reais do ambiente e proverá dados de contexto armazenados no histórico para todas as entidades presentes no ambiente.

No caso do AmbientMonitor, são realizadas solicitações por contextos das entidades “sala”, “cozinha” e “quarto”, com uma cadência constante. Como a hierarquia se encontra atualizada, os dados recebidos corresponderão a todas as entidades presentes nesses ambientes. A partir dos últimos valores de contexto recebidos para cada uma das entidades é verificado se existe a necessidade de atuar sobre o ambiente.

Considerando a implementação do AmbientMonitor, é possível afirmar que a utilização das consultas hierárquicas traz benefícios quando comparado ao método de consulta convencional. Sua principal vantagem está na redução de solicitações por contexto ao *framework* trafegando pela rede, bem como na organização e manutenção futura do código.

O cenário proposto é composto por 10 entidades estáticas e três móveis. A cada cinco segundos é solicitado o contexto hierárquico das entidades que representam os três ambientes, transmitindo também a qual aplicativo aquela hierarquia pertence, bem como um identificador para designar a estrutura hierárquica em questão, para quando houver mais de uma hierarquia.

Dessa forma, é somente realizada uma chamada por contexto, que será transmitida via rede até o módulo CMFDatabase e processada, retornando ao aplicativo todos os contextos solicitados através da hierarquia definida, em um único processo.

Os contextos dinâmicos foram empregados em duas situações distintas para o AmbientMonitor, primeiro para os sensores de presença distribuídos no ambiente, e posteriormente ao armazenar as hierarquias anteriores de cada entidade. No primeiro, somente ocorre o armazenamento por parte dos microcontroladores quando é detectado que alguém se encontra presente no ambiente, reduzindo a quantidades de informações armazenadas.

O armazenamento de hierarquias para uma entidade móvel, por sua vez, designa que, a partir da solicitação do aplicativo, será salvo o momento em que esta entidade realizou uma troca hierárquica, guardando a sua nova posição dentro da hierarquia. Como o armazenamento dessas informações é feito na base de dados específica da entidade,

emprega-se diretamente o uso dos contextos dinâmicos, visto que a base agora armazena simultaneamente dados normais de contexto, com seus valores específicos, e também dados relativos à hierarquia a que a entidade pertence.

Ao permitir que os tipos de informações armazenadas mudem livremente durante o tempo de execução, e sem definir previamente quais serão os formatos dos dados a serem armazenados, se torna possível afirmar, que para o AmbientMonitor, a possibilidade de armazenar contextos com tamanha dinamicidade traz maior praticidade e conveniência ao uso do *framework*, tanto em seu processo de armazenamento interno, quanto para o desenvolvedor de aplicativos.

Dentro do CMFrame, as solicitações por informações de contexto requerem um parâmetro para a criação das pesquisas, que é definido através das classes derivadas do CMFFilter e, a menos que seja explicitamente solicitado, emprega por padrão um filtro nulo (que retorna todos os resultados). Quando se considera a natureza do armazenamento de dados de contexto na forma de histórico, nota-se que essas informações chegarão a quantidades proibitivas para o transporte pela rede, trazendo limitações em sua transmissão e dificultando o uso desse volume de dados.

Para mitigar essa situação, é possível definir filtros mais específicos para o propósito do utilizador, limitando quais dados de contexto serão retornados, e até mesmo a quantidade máxima de elementos retornados. Para situações em que as informações de contexto são mais complexas ou muito concatenadas, recomenda-se que seu retorno seja feito com a utilização de estruturas de dados customizadas pelo desenvolvedor, que desmembrem os elementos a fim de facilitar a utilização dos contextos no aplicativo.

C. *EnergyMonitor*

Na implementação do EnergyMonitor, a hierarquia é realizada uma vez, no início da aplicação, e posteriormente é utilizada para consultar os contextos de toda a sua estrutura, filtrando em nível de aplicação os dados relevantes. Assim se torna possível que o EnergyMonitor e o AmbientMonitor tenham estruturas hierárquicas distintas, cada uma voltada à necessidade específica da aplicação, e que sejam executados simultaneamente.

A Fig. 5 mostra a tela do aplicativo, exibindo os dados de contexto considerados relevantes ao propósito da aplicação (gerenciamento de energia) dentre todos os sensores que se encontram presentes nos três ambientes. A estrutura hierárquica única e imutável serve como base para a coleta de todas as informações de contexto através de uma chamada de método única, em que os resultados são filtrados no método de retorno.

Tendo em vista que os contextos vindos das entidades móveis presentes nesse cenário específico não são relevantes para o propósito do gerenciamento de energia, se torna possível empregar a mesma hierarquia durante toda a execução do aplicativo, sem excluir nenhuma informação de contexto importante. Dessa forma, o emprego das entidades hierárquicas e dinâmicas com o EnergyMonitor foi mais voltado a sua capacidade de retornar contextos em cadeia, com base na hierarquia previamente definida.

Dentre as limitações encontradas no EnergyMonitor pode-se relatar que elas abrangem os mesmos pontos encontrados no aplicativo AmbientMonitor, visto que as dificuldades não estão atreladas diretamente aos conceitos de entidades hierárquicas e dinâmicas, mas sim a questões particulares da implementação realizada no *framework*.

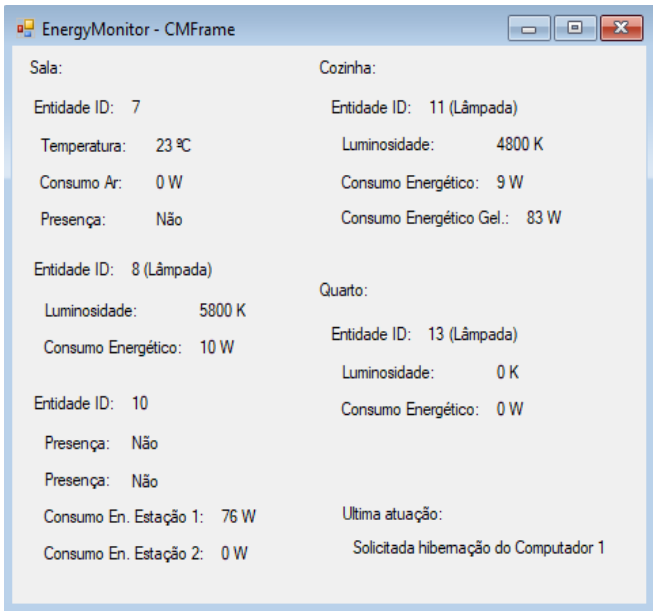

Fig. 5. Tela do EnergyMonitor.

V. Conclusão

Esse artigo propôs o CMFrame, um *framework* para a criação de aplicações que usem históricos de contextos. Seu propósito é oferecer uma forma de lidar com as informações de contexto armazenadas a fim de gerar um histórico, empregando o conceito de entidades hierárquicas e dinâmicas para melhor representar os ambientes inteligentes atuais, e prover suporte a informações de contexto variadas.

O *framework* oferece métodos para que dispositivos controladores possam interagir com uma base de dados, e assim armazenar os dados de contexto adquiridos em seu ambiente, bem como permitir que aplicações externas possam empregar esses dados de contexto para tomar decisões acerca do ambiente e reagir aos usuários presentes.

Com base no desenvolvimento de dois aplicativos e sua implementação no cenário de uma casa inteligente, verificou-se que o *framework* atende as necessidades como um gerenciador de históricos de contextos, para situações que utilizem ambientes físicos com elementos inteligentes e aplicações que fazem a gerência do ambiente através da atuação sobre os mesmos, como nos casos avaliados.

Tendo como mecanismo complementar a esse objetivo, o emprego de entidades hierárquicas e dinâmicas. Seu uso visa facilitar o processo organizacional dos elementos de um ambiente passíveis de representação através de entidades, sejam elas reais ou virtuais, permitindo que os contextos de maior relevância para a aplicação sejam disponibilizados no momento desejado através do seu relacionamento entre as entidades. Dessa forma, o CMFrame pode ser utilizado em situações onde existam ambientes físicos que necessitam de características inteligentes, armazenando e consultando informações, a fim de refinar seu comportamento através da análise de suas condições.

Como trabalhos futuros se propõe a simplificação da forma como o *framework* lida com as trocas de hierarquia, bem como a expansão da complexidade dos filtros de pesquisa

atualmente implementados, permitindo uma maior gama de resultados finais ao usuário. Também pode-se mencionar a expansão do componente de eventos, incorporando funcionalidades para a criação de tarefas predefinidas em intervalos de tempo constantes. Por fim, tem-se como intuito permitir que exista mais de uma instância do CMFDatabase em execução, permitindo sua operação sobre o mesmo conjunto de dados, e facilitando a redundância.

AGRADECIMENTOS

Os autores agradecem à Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Universidade do Vale do Rio dos Sinos (Unisinos) pelo apoio ao desenvolvimento desse trabalho. Os autores reconhecem especialmente o apoio do Programa de Pós-Graduação em Computação Aplicada (PPGCA) e do Laboratório de Computação Móvel (Mobilab) da Unisinos.

REFERÊNCIAS

[1] M. Satyanarayanan, "Pervasive computing: vision and challenges," IEEE Personal Communications, vol. 8, no. 4, pp. 10–17, 2001.

[2] M. Weiser, "The Computer for the 21st Century," Scientific American., vol. 265, no. 3, pp. 94–104, Sep 1991.

[3] H. Vianna and J. L. V. Barbosa, "A Model for Ubiquitous Care of Noncommunicable Diseases," IEEE Journal of Biomedical and Health Informatics, vol. 18, no.5, pp. 1597–1606, 2014. http://dx.doi.org/10.1109/JBHI.2013.2292860

[4] H. Vianna and J. L. V. Barbosa, "In the Pursuit of Hygge Software," IEEE Software, vol. 36, no.6, pp. 48-52, 2017. https://doi.org/10.1109/MS.2017.4121208

[5] H. Vianna and J. L. V. Barbosa, "A scalable model for building context-aware applications for noncommunicable diseases prevention," Information Processing Letters, vol. 148, pp.1-6, 2019. http://dx.doi.org/10.1016/j.ipl.2019.03.010

[6] J. L. V. Barbosa, M. R. Kich, D. N. F. Barbosa, A. Z. Klein and S. J. Rigo, "DeCom: A Model for context-aware competence management," Computers in Industry, vol. 72, pp. 27-35, 2015. https://doi.org/10.1016/j.compind.2015.03.012

[7] A. Wagner, J. L. V. Barbosa and D. Barbosa, "A Model for Profile Management Applied to Ubiquitous Learning Environments," Expert Systems with Applications, vol. 41, no. 4, pp. 2023–2034, 2014. http://dx.doi.org/10.1016/j.eswa.2013.08.098

[8] A. Larentis, J. L. V. Barbosa, D. Barbosa, C. Silva and J. Barbosa, "Applied Computing to Education on Noncommunicable Chronic Diseases: A Systematic Mapping Study," Telemedicine and e-Health, pp.1-10, 2019. https://dx.doi.org/10.1089/tmj.2018.0282

[9] J. L. V. Barbosa, C. Martins, L. Franco and D. Barbosa, "TrailTrade: a model for trail-aware commerce support," Computers in Industry, vol. 80, p.43-53, 2016. http://dx.doi.org/10.1016/j.compind.2016.04.006

[10] J. L. V. Barbosa, J. Tavares, I. Cardoso, B. Mota and B. Martini, "TrailCare: na Indoor and Outdoor Context-aware System to Assit Wheelchair Users," Internacional Journal of Human-Computer Studies. vol. 116, pp.1–14, 2018. https://doi.org/10.1016/j.ijhcs.2018.04.001

[11] R. Souza, J. Lopes, C. Geyer, L. João, A. Cardozo, A. Yamin, G. Gadotti and J. L. V. Barbosa, "Continuous Monitoring Seed Testing Equipaments Using Internet of Things," Computers and Electronics in Agriculture, vol. 158, pp. 122-132, 2019. http://dx.doi.org/10.1016/j.compag.2019.01.024

[12] E. Torunski, R. Othman, M. Orozco and A. E. Saddik, "A Review of Smart Environments for Energy Savings," Procedia Computer Science, vol. 10, pp. 205–214, 2012. http://dx.doi.org/10.1016/j.procs.2012.06.029

[13] A. Dey, D. Salber and G. Abowd, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware application," Human-Computer Interaction, vol. 16, no. 2, pp. 97–166, 2001. http://dx.doi.org/10.1207/S15327051HCI16234_02

[14] J. H. Rosa, J. L. V. Barbosa, M. Kich and L. Brito, "A Multi-Temporal Context-aware System for Competences Management," International Journal of Artificial Intelligence in Education, vol. 25, no. 4, p. 455–492, 2015. http://dx.doi.org/10.1007/s40593-015-0047-y

[15] J. Y. Lee, D. W. Cheun, and S. D. Kim, "A comprehensive framework for mobile cyber-physical applications," in 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–6, 2011.

[16] M. Törngren, F. Asplund, S. Bensalem, J. McDermid, R. Passerone, H. Pfeifer, A. Sangiovanni-Vincentelli and B. Schätz, "Chapter 1 - Characterization, Analysis, and Recommendations for Exploiting the Opportunities of Cyber-Physical Systems," in Cyber-Physical Systems, H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, Eds. Boston: Academic Press, pp. 3–14, 2017.

[17] E. Geisberger and M. Broy, "Living in a networked world: Integrated research agenda Cyber-Physical Systems (agenda CPS)," 1st ed., Herbert Utz Verlag, München, 2015.

[18] E. D. Coninck, S. Bohez, S. Leroux, T. Verbelen, B. Vankeirsbilck, B. Dhoedt and P. Simoens, "Middleware Platform for Distributed Applications Incorporating Robots, Sensors and the Cloud," in 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), pp. 218–223, 2016.

[19] O. Younis and N. Moayeri, "Cyber-physical systems: A framework for dynamic traffic light control at road intersections," in 2016 IEEE Wireless Communications and Networking Conference, pp. 1–6, 2016.

[20] D. Martin, C. Lamsfus, and A. Alzua, "Automatic context data life cycle management framework," in 5th International Conference on Pervasive Computing and Applications, pp. 330–335, 2010.

[21] O. Kotevska, A. Lbath, and S. Bouzefrane, "Toward a real-time framework in cloudlet-based architecture," Tsinghua Science Technology, vol. 21, no. 1, pp. 80–88, 2016. http://dx.doi.org/10.1109/TST.2016.7399285

[22] M. V. L. Martins, "FrameTrail: Um Framework para o Desenvolvimento de Aplicações Orientadas a Trilhas," Unisinos, São Leopoldo - RS. Dissertação, maio 2011.

[23] D. P. F. Möller, "Guide to Computing Fundamentals in Cyber-Physical Systems," Cham: Springer International Publishing, 2016.

[24] K. K. Patel and S. M. Patel, "Internet of Things - IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," International Journal of Engineering Science and Computing, vol. 6, no. 5, pp.6122-6131, 2016.

[25] N. E. Petroulakis, E. Z. Tragos, A. G. Fragkiadakis, and G. Spanoudakis, "A lightweight framework for secure life-logging in smart environments," Information Security Technical Report, vol. 17, no. 3, pp. 58-70, Feb. 2013. http://dx.doi.org/10.1016/j.istr.2012.10.005