Framework de Suporte ao Desenvolvimento e Evolução de Aplicações Auto-Adaptativas em IoT

Belmondo R. A. Junior[1,*], **Tales P. Nogueira** [1,†],
Marcio E. F. Maia[2], **Rossana M. C. Andrade**[1,‡]

[1]Grupo de Redes, Engenharia de Software e Sistemas (GREat)
Mestrado e Doutorado em Ciência da Computação (MDCC)
Universidade Federal do Ceará (UFC) – Fortaleza – Brasil

[2]Universidade Federal do Ceará (UFC)
Campus Quixadá - Quixadá - Brasil

{belmondorodrigues, tales, marcio, rossana}@great.ufc.br

Abstract. *The traditional development approach based on design-time decisions is inadequate for IoT applications considering (i) the heterogeneity of devices and technologies, and (ii) the inability to handle unforeseen scenarios at development time. Therefore, the use of practices that allow adapting the behavior and structure of applications in a more flexible way is required. In this context, we propose a decoupled support framework that allows IoT applications to achieve self-adaptation. The proposed framework was evaluated in relation to its performance and the results show that this approach can perform adaptations with a large number of devices in a timely manner.*

Resumo. *O desenvolvimento baseado em decisões em tempo de design é inadequada para aplicações IoT considerando (i) a heterogeneidade de dispositivos e tecnologias e (ii) a incapacidade de lidar com cenários imprevistos em tempo de design. Portanto, práticas que permitam adaptar o comportamento e a estrutura da aplicação de modo flexível são necessárias. Nesse contexto, este trabalho propõe um framework de suporte desacoplado para permitir que as aplicações IoT atinjam a auto-adaptação. O framework proposto foi avaliado em relação ao seu desempenho e os resultados mostram que esta abordagem pode realizar adaptações considerando um grande número de dispositivos em tempo hábil.*

1. Introdução

A crescente miniaturização das tecnologias de computação e pesquisas em áreas como a computação ubíqua permitiram o surgimento da Internet das Coisas (do inglês *Internet of Things* – IoT). Este paradigma consiste em um cenário no qual dispositivos dotados de

[*]Aluno de mestrado do MDCC/UFC, financiado pela CAPES até 03/2018 e agora financiado pela Fundação Cearense de Pesquisa e Cultura - FCPC

[†]Bolsista de Pós-Doutorado CAPES

[‡]Bolsista do CNPq de Produtividade em Desenvolvimento Tecnológico e Extensão Inovadora (DT) com o número de processo 314021/2009-4

sensores funcionam em colaboração com atuadores de forma a atingir um determinado objetivo de modo transparente aos usuários [Pires et al. 2015].

Na IoT, os objetivos são atingidos através da execução de ações no ambiente utilizando sensores e atuadores. Para isso, os desenvolvedores utilizam informações de contexto sobre os usuários e sistemas buscando adaptar o comportamento da aplicação para otimizar seu funcionamento ou tratar condições adversas. Nas abordagens tradicionais de desenvolvimento, as ações executadas no ambiente e os estados da aplicação que disparam adaptações (regras de adaptação) são definidos em tempo de desenvolvimento [Lee 2015].

Essa abordagem baseada em decisões em tempo de desenvolvimento é inadequada para a IoT por dois motivos [Baresi and Ghezzi 2010]: i) grande quantidade de dispositivos, tecnologias de suporte e estados da aplicação; e ii) incapacidade de tratar, em tempo de execução, os cenários não previstos em tempo de desenvolvimento. Assim, a criação de aplicações para a IoT requer o uso de práticas que consigam adaptar o comportamento e a estrutura da aplicação de forma mais flexível.

Para tratar cenários com nível de complexidade elevado, abordagens baseadas em auto-adaptação têm sido utilizadas a fim de permitir a flexibilidade do comportamento e estrutura com um mínimo de intervenção do usuário ou desenvolvedor [Kramer and Magee 2007]. Deste modo, a auto-adaptação é baseada na criação de modelos que descrevem o comportamento e a estrutura do sistema e que são traduzidos em módulos de execução que implementam os requisitos de aplicação. Por exemplo, para alterar o comportamento da aplicação, os modelos que descrevem o estado do sistema são analisados, regras de adaptação geradas em tempo de execução e ações de otimização ou correção são implantadas através da substituição de módulos inadequados.

Com o objetivo de utilizar em IoT os conceitos propostos para os sistemas auto-adaptativos, este trabalho propõe um *framework* que permite o desenvolvimento de aplicações para cenários IoT considerando a heterogeneidade e o dinamismo, bem como as possibilidades de adaptação. Para atingir este objetivo, o *framework* orquestra as possíveis adaptações nos níveis de ambiente e de aplicação através de sequências de ações, chamadas de *workflows* neste trabalho.

2. Fundamentação Teórica

2.1. Internet das Coisas

Internet das Coisas é um cenário no qual dispositivos com capacidade de sensoriamento e atuação – *smart-objects* – produzem e consomem informação e comunicam-se através de redes sem fio com o propósito de atingir determinado objetivo. A literatura define uma série de requisitos para aplicações IoT que serão descritas a seguir.

A **descrição** dos dispositivos consiste em informações que representam as características funcionais e não funcionais de um dispositivo, e os serviços provindos destes [Li et al. 2014]. A **descoberta** prevê que os objetos inteligentes e serviços se tornem disponíveis em tempo de execução [Barreto et al. 2017].

A **segurança** está associada à privacidade, autenticação e controle de acesso. A **privacidade** trata de como os dados presentes no ambiente estarão dispostos para as aplicações. A **autenticação** está relacionada à comunicação segura entre objetos e serviços. Por fim, o **controle de acesso** é um requisito que visa garantir permissões sobre

dispositivos, sensores e serviços [Ng and Wakenshaw 2017]. O **sensoriamento** diz respeito às capacidades dos *smart-objects* de perceberem as variáveis presentes, internas ou externas. Já a **atuação** diz respeito à capacidade dos *smart-objects* exercerem ações no mundo físico [Ng and Wakenshaw 2017]. O **tratamento de exceção** é um requisito para garantir o equilíbrio do sistema de maneira que a execução seja contínua, mesmo diante da indisponibilidade dos *smart-objects* ou serviços envolvidos [Ng and Wakenshaw 2017].

2.2. Sistemas Auto-Adaptativos

Os sistemas auto-adaptativos são aqueles capazes de modificar seu comportamento, atributos ou artefatos em resposta a mudanças internas ou externas. O processo de adaptação, conhecido como MAPE-K Loop, consiste de cinco fases: Monitoria (M), Análise (A), Planejamento (P) e Execução (E) e o Conhecimento (K) [Kephart and Chess 2003].

A primeira fase (M), é responsável por perceber e coletar os estímulos e mudanças dos recursos do sistema. Os estímulos são agregados, correlacionados e filtrados até que sejam caracterizados como um fenômeno que necessite ser analisado. A segunda fase (A), é responsável por analisar e inferir a partir das informações fornecidas pela fase anterior, sendo influenciado por conhecimento prévio armazenado. Caso seja percebido que, para o fenômeno detectado, demanda-se alterações, uma solicitação de mudança é passada para a camada seguinte. A terceira fase (P), tem como função estruturar as ações necessárias a fim de atingir um determinado objetivo. Essas ações são criadas ou selecionadas a partir de um conhecimento prévio para iniciar uma alteração desejada no recurso onde o fenômeno foi detectado. Essas alterações podem variar desde uma única ação até um fluxo de execuções. A quarta fase (E) altera o comportamento do recurso com base nas ações recomendadas pela fase anterior. O Conhecimento (K) diz respeito ao conhecimento compartilhado entre as outras fases, como *logs* históricos e sintomas de possíveis problemas, por exemplo.

3. *Framework* Proposto

Como uma alternativa para a problemática apresentada, este trabalho propõe um *framework* que visa apoiar o desenvolvimento de aplicações considerando a dinamicidade de execução em ambientes auto-adaptativos, como ambientes IoT. Para atingir este objetivo, este *framework* foi desenvolvido levando em consideração os requisitos de descrição, descoberta, sensoriamento, atuação e tratamento de exceção detalhados na seção 2.1. Deste modo, este trabalho atua na fase de execução do MAPE-K Loop, especificando ações, estratégias e regras de adaptação.

3.1. Visão Geral

A principal característica deste *framework* é o gerenciamento da execução de *workflows* constituídos de ações e seus modos de execução (sequencial ou em paralelo). Ações especificam adaptações a serem executadas, tanto a nível de sistema quanto a nível de ambiente.

A instanciação dos *workflows* pode ocorrer de duas maneiras: (i) simples, quando o *workflow* é executado a partir de uma solicitação feita por um elemento do sistema; ou (ii) baseado em regras, que permite a execução a partir de um determinado estado da aplicação. Na segunda maneira, o desenvolvedor precisa especificar o estado da aplicação

e as ações a serem executadas quando esse estado for detectado. Assim, a execução baseada em regras é a maneira mais flexível de executar um *workflow*, permitindo a auto-adaptação do aplicativo em tempo de execução. Desse modo, um *workflow* pode ser facilmente trocado em tempo de execução, considerando uma mudança de estado ou uma situação não prevista. Essa troca de *workflows* é um passo do MAPE-K Loop descrito na Seção 2.2. Uma camada de inteligência ou um outro sistema devem estar constantemente analisando o estado do sistema e decidindo se o *workflow* atual vai levar o sistema para um próximo estado válido. Essa camada não faz parte do *framework* proposto.

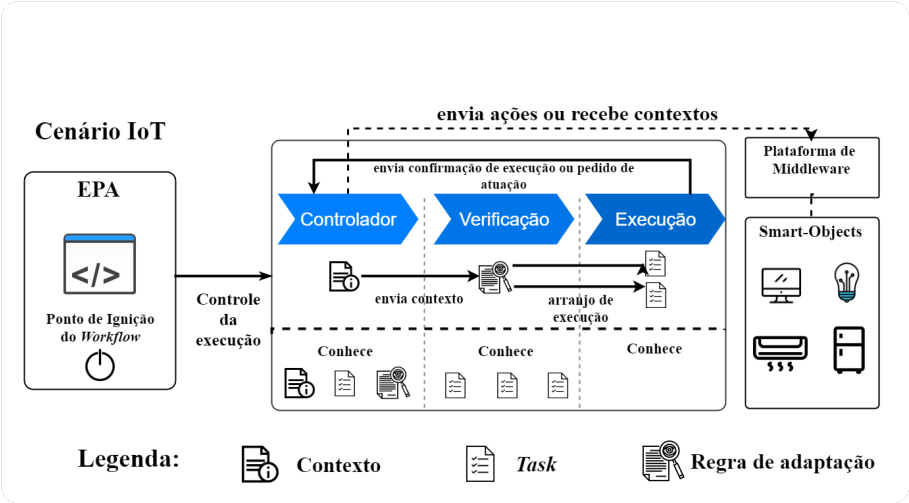


Figura 1. Visão geral do *framework* proposto por este trabalho

O *framework* proposto segue uma ordem que consiste em 3 etapas, conforme ilustrado na Figura 1: (1) Gerenciamento (feito pelo **Controlador**), (2) **Verificação de Regras**, e (3) **Execução**. O **Controlador** conhece os três componentes básicos de execução: estado da aplicação, regras de adaptação e ações. O Controlador monitora continuamente o estado da aplicação e o passa para o segundo módulo. Se qualquer regra for satisfeita, o controlador é responsável pela execução da adaptação associada. O segundo módulo é responsável por verificar se uma ou mais regras estão satisfeitas. Cada regra define um estado da aplicação e as ações que ela deve ativar. Por exemplo, se esta etapa detectar que o usuário está dirigindo, alguma adaptação específica deverá ser executada.

3.2. Arquitetura

O *framework* proposto foi projetado em uma arquitetura extensível baseada em componentes, visando melhorar o desacoplamento e o reuso de código. A Figura 2 ilustra as camadas que fazem parte de sua arquitetura: Camada de Desenvolvimento e Camada de Integração e Execução.

A **Camada de Desenvolvimento** contém os módulos de Definição de *Workflow* e o *Workflow Builder*. O Módulo de Definição de *Workflow* é responsável por fornecer as estruturas necessárias para definir o fluxo de adaptações, que podem ser: regras de adaptação, adaptações e o modo de execução (e.g. paralelas, sequenciais ou exclusivas). O módulo *Builder* é responsável por armazenar as estruturas definidas na camada acima e associar as adaptações com suas regras e modos de execução.

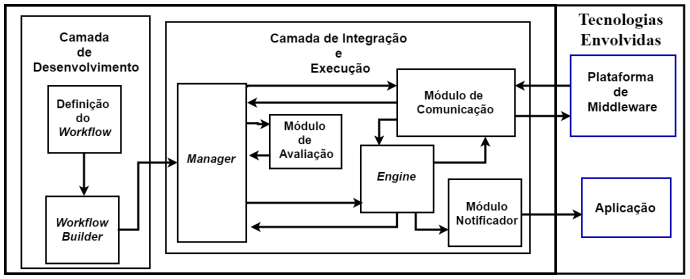


Figura 2. Arquitetura do *framework* proposto neste trabalho

A **Camada de Integração e Execução** é responsável por organizar a execução do *workflow* definido, verificando regras e iniciando adaptações. Ao atingir essa camada, o **Módulo *Manager*** organiza a execução e distribui as responsabilidades, uma vez que conhece todas as estruturas envolvidas na execução (e.g. regras de adaptação e *gateways*). Ele informa ao **Módulo de Avaliação** os estados da aplicação e ao **Módulo *Engine*** quais adaptações devem ser executadas e seus modos.

O **Módulo de Comunicação** foi desenvolvido para fornecer uma comunicação para quando for necessário sentir ou agir no ambiente. Portanto, é responsável pela comunicação com quaisquer soluções externas, como plataformas de *middleware*. O **Módulo *Engine*** é responsável pelo processamento das adaptações exigidas. Além de receber comandos do **Manager**, esta camada é responsável por informar ao **Módulo de Comunicação** sobre as adaptações no ambiente. É, ainda, responsável por enviar informações para o **Módulo Notificador**. A partir daí, o **Módulo Notificador** é responsável por passar as informações para a interface da aplicação ou esperar interação do usuário.

3.3. Prova de Conceito

Um Assistente Virtual (AV) Android foi implementado como prova de conceito (PoC) (Figura 3). No cenário proposto, ao chegar ao local de trabalho, o usuário recebe um e-mail de confirmação e o ar condicionado de sua sala é ligado. Quando o usuário entra no prédio e se aproxima da sala, as luzes recebem uma ordem para acendimento. O *framework* adapta a aplicação partindo de regras definidas, como mostra a Figura 4. Para esta PoC, foram definidas duas regras de adaptação: uma baseada na localização do usuário e uma baseada na proximidade da sala. A regra baseada em localização avalia a proximidade geográfica do local de trabalho do usuário, considerando os dados do GPS. A regra baseada na proximidade da sala avalia a distância do usuário e de sua sala, com base em um sinal de *beacon*.

Nesta PoC, foram definidas três ações: a ação responsável por acender as luzes, a ação que confirma a presença do usuário e a ação responsável por ligar o ar-condicionado. Essas ações fazem a transição para o estado de execução apenas quando as regras percebem que é necessário iniciar uma adaptação. Desse modo, as ações permanecem inativas enquanto a regra responsável pela adaptação não seja satisfeita. Os estímulos das regras são as informações contextuais, sejam estas provindas do ambiente (físico e lógico) ou do próprio sistema. Como resposta das regras, alguma adaptação pode ocorrer. Na implementação dessa PoC, o middleware LoCCAM [Maia et al. 2013] foi utilizado como plataforma de suporte à interação com os *smart-objects* dispostos no ambiente físico.

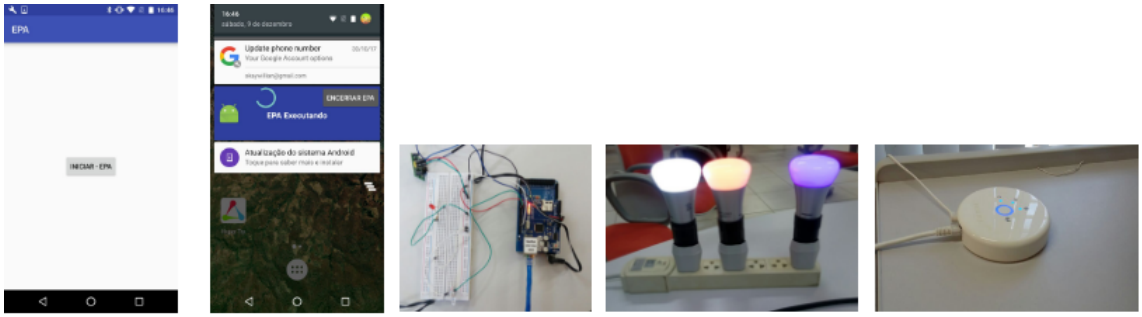
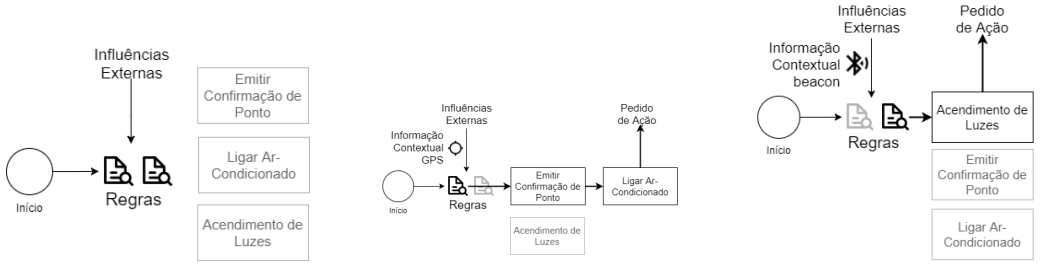


Figura 3. Aplicação e Hardware utilizados na prova de conceito

O fluxo de execução do AV é modificado em tempo de execução (Figura 4). No início, o fluxo permanece inativo, pois as informações contextuais recebidas não atendem às regras definidas (a). Quando uma das regras é satisfeita (b), o fluxo de adaptação relacionado é acionado. Após a sua conclusão, o fluxo retorna ao estado inativo (a). Quando a regra de proximidade é satisfeita (c), o fluxo de adaptações referente torna-se ativo.



(a) Sem fluxo definido

(b) execução de fluxo baseado na Regra de Localização

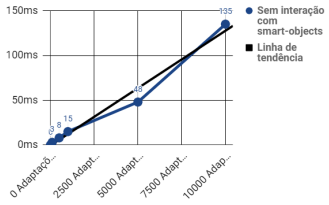
(c) execução de fluxo baseado na Regra de proximidade

Figura 4. Representação do *workflow* da PoC

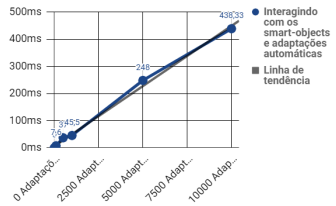
4. Avaliação de Desempenho

O *framework* foi avaliado de acordo com o desempenho, a fim de verificar se o mesmo é capaz de realizar adaptações em um cenário heterogêneo com um grande número de estados, regras e possíveis interações com objetos inteligentes. O objetivo desta avaliação foi verificar o comportamento do *framework* proposto em relação ao seu tempo de execução, adaptações instanciadas e completude das adaptações. Para isso, os experimentos foram definidos em três cenários distintos: (a) sem interação com objetos inteligentes, (b) com adaptações automáticas e sem interação com o usuário e (c) com interação e atualização da interface do usuário. Os três cenários foram elaborados para entender o comportamento do *framework* e avaliar se há algum cenário ou número de adaptações que inviabilize o uso desta solução.

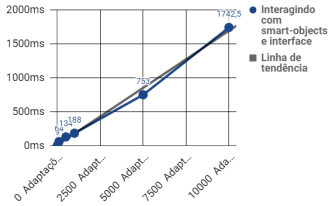
Para conduzir a avaliação, foi utilizado um dispositivo Android, versão 6.0 com 1854 MB de memória RAM total e processador Qualcomm 2,27 GHz com 4 núcleos. Como plataforma de *middleware*, foi escolhido o LoCCAM [Maia et al. 2013]. Ainda,



(a) Sem interação com dispositivos



(b) Interação com *smart-objects* e adaptações automáticas



(c) Interação com *smart-objects* e interação na interface

Figura 5. Avaliações do *framework* proposto neste trabalho

foi desenvolvida uma biblioteca Java [1] capaz de simular a existência de *smart-objects* e de instanciar adaptações a partir de um número desejado.

O desempenho foi analisado com base no tempo gasto no número de adaptações executadas. Para isso, a avaliação consistiu em cenários com 1, 10, 100, 500, 1.000, 5.000 e 10.000 adaptações. Cada cenário foi executado 10 vezes e foram calculadas as médias de execução em milissegundos. A Tabela 1 e a Figura 5 contêm os resultados desta avaliação.

Tabela 1. Tabela com os tempos comparativos de execução

Nº Adaptações	1A	10As	100As	500As	1.000As	5.000As	10.000As
Execução (a)	0ms	0ms	3ms	8ms	18ms	48ms	135ms
Execução (b)	0ms	0ms	7,6ms	37ms	65,33ms	248,33ms	438,33ms
Execução (c)	1ms	9ms	64ms	134ms	188ms	753ms	1742,5ms

O primeiro cenário (a) foi planejado para entender o tempo de execução basal da solução proposta. Para isso, a simulação consistiu na criação de adaptações ”vazias”que geravam mensagens em *logs* para verificação. O segundo cenário (b) foi desenvolvido considerando um ambiente onde a comunicação com sensores e atuadores ocorre de forma transparente para o usuário. O terceiro cenário (c) foi criado para representar um ambiente em que as adaptações interagem com o usuário por meio da interface do dispositivo, como mensagens de confirmação e escolha de opções. A Figura 5 mostra os gráficos para cada experimento. O comportamento do *framework* é representado pela linha azul e a linha preta representa a tendência de execução assintótica.

A principal disparidade ocorreu quando o número de adaptações com interações visuais aumentou. Enquanto o tempo máximo para 10.000 dispositivos foi de 135ms e 438ms nas duas primeiras avaliações, no terceiro cenário esse valor atingiu 1742ms. Isso se deve à arquitetura do sistema operacional Android, já que há apenas uma execução de *thread* principal (a *thread* da interface do usuário). Se é de interesse atualizar a interface do usuário, é necessário que o processo entre na fila de execução da interface do sistema operacional [Google 2017].

Apesar disso, os dados analisados mostram que o *framework* proposto tende a se comportar com uma complexidade linear quando executado em um cenário em que são

[1] https://github.com/Belmondo/GenerateObjects/

necessárias um grande número de adaptações. Essa tendência de comportamento sugere que a solução é eficiente se considerarmos que o processamento das adaptações depende do contexto ou do estado da aplicação, conhecendo-os apenas em tempo de execução.

5. Trabalhos Relacionados

Nesta seção, são discutidos trabalhos relacionados que propõem algum mecanismo ou infraestrutura de especificação e execução de adaptações que preveja, em sua arquitetura ou meios externos, as características levantadas na seção 2.1.

O *framework* LateVa [Murguzur et al. 2015] trata a variabilidade dentro de linhas de produtos de software dinâmicas (LPSD). Para tal, um modelo base deve ser descrito com as comunalidades e variabilidades de cada família de LPSD, bem como de possíveis pontos de variação. Assim, dependendo da associação do contexto com a descrição da variação, o LateVa seleciona os fragmentos e os executa. Contudo, por seu escopo ser voltado para LSPD, esse *framework* não prevê em seu funcionamento a descoberta de *smart-objects*.

O Presto [Giner et al. 2010] é uma arquitetura desenvolvida para permitir aos programadores sistematizar o desenvolvimento de *workflows* implementados em dispositivos móveis considerando a existência de sensores. Contudo, é necessário descrever cada componente (e.g. dispositivo inteligente) e indicar suas capacidades.

O PROtEUS [Seiger et al. 2015] é um sistema integrado composto por um motor de execução, um mecanismo de processamento de eventos, uma plataforma de serviço, um *caller* de serviço e um servidor. O *core* do PROtEUS consiste de um meta-modelo que descreve a estrutura de um processo e uma *engine* responsável por instanciar esses meta-modelos. O sistema utiliza uma forma hierárquica de representar informações contextuais para capturar e perceber os objetos no ambiente. Contudo, é necessário que o objeto esteja instalado e atribuído no ambiente através da interface de controle e monitoramento.

O SitOPT [Wieland et al. 2015] é um sistema de gerenciamento de *workflows* para sistemas ubíquos. Sua arquitetura é dividida em 3 camadas: (i) *sensing*, (ii) *situation recognition* e (iii) *situation-aware workflow*. Contudo, sua arquitetura não prevê descoberta de objetos e, para possibilitar a modelagem das situações, é necessário descrever cada componente e indicar suas capacidades.

O *GET+UNICORN* [Baumgraßet al. 2015] é uma abordagem para conectar eventos de várias fontes heterogêneas e execução com foco em transportes. O projeto é dividido em dois sistemas básicos: (i) *Unicorn* e (ii) *GET Controller*. O *Unicorn* é responsável por coletar, processar e distribuir os eventos logísticos relevantes provindos de diversas fontes, enquanto que o *GET* executa e monitora os processos considerando os eventos provindos do *Unicorn*. Contudo, apenas atuação é prevista na abordagem dos autores.

No trabalho de [Montagut and Molva 2005], uma arquitetura de suporte à execução distribuída de *workflows* em cenários pervasivos é proposta: *Enabling Pervasive Workflow Execution* ou EPEW. Mesmo tendo os dispositivos como centro de cada execução, o trabalho não considera as capacidades de atuação.

O *framework* proposto é responsável por orquestrar as possibilidades de adaptação, de ambiente e de aplicação em tempo de execução, considerando os requi-

sitos descritos na seção 2.1. Para isso, tais requisitos são considerados dentro das camadas da arquitetura. O Módulo de Comunicação é responsável pela comunicação com a plataforma de *middleware* desejada pelos desenvolvedores. Portanto, os requisitos de descrição, descoberta, sensoriamento e atuação são considerados aqui. Deste modo, esses requisitos ficam transparentes para as demais camadas. Caso a adaptação esteja associada à aplicação, esta é feita pelo Módulo Notificador. Para o tratamento de exceções em tempo de execução, o Módulo Engine observa as possibilidades de adaptação e, caso uma exceção seja lançada, o fluxo de execução é alterado ou um tratamento pré-definido pelos desenvolvedores é executado. Para este trabalho, não foram considerados os requisitos relacionados à segurança. A Tabela 2 sumariza a discussão apresentada nesta seção.

Tabela 2. Trabalhos relacionados. Legendas: X - Possui, (X) - Possui Parcialmente e O - Não Possui/Não Descreve

	Descrição	Descoberta	Relacionados à Segurança	Sensoriamento & Atuação	Exceção
LateVa	**X**	O	O	***(X)***	O
Presto	**X**	O	***(X)***	O	O
PROtEUS	**X**	***(X)***	O	**X**	**X**
SitOPT	**X**	O	O	**X**	**X**
Get+Unicorn	O	O	O	***(X)***	**X**
EPEW	O	**X**	***(X)***	***(X)***	O
Framework Proposto	**X**	**X**	**O**	**X**	**X**

6. Considerações Finais

As capacidades de adaptação e evolução em tempo de execução de uma aplicação IoT podem ser comprometidas se considerarmos o modo tradicional de desenvolvimento de aplicações. Visando tratar esse problema, este trabalho apresentou um *framework* para descrição e execução de adaptações baseadas em *workflows* para cenários IoT. O *framework* proposto fornece um conjunto de estruturas de modelagem para especificar ações, estratégias e regras de adaptação. Além disso, permite integrar estes componentes, provendo a auto-adaptação de aplicações em tempo de execução.

O *framework* foi avaliado através de uma PoC e medidas de desempenho. Nesta análise, os resultados mostraram que a solução é vantajosa e pode ser usada em um cenário real. Além disso, esta abordagem se mostrou capaz de realizar e suportar, em tempo hábil, um grande número de adaptações em tempo de execução. Como perspectiva futura, avaliações específicas para sistemas auto-adaptativos devem ser executadas.

Agradecimentos

Este trabalho foi parcialmente apoiado pelo INCT INES (CNPq/465614/2014-0).

Referências

Baresi, L. and Ghezzi, C. (2010). The disappearing boundary between development-time and run-time. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 17–22. ACM.

Barreto, F. M., Maia, M. E., Andrade, R. M., and Viana, W. (2017). Coap-ctx: Extensão sensível ao contexto para descoberta de objetos inteligentes em internet das coisas. *XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos-SBRC2017*.

Baumgraß, A., Ciccio, C. D., Dijkman, R. M., Hewelt, M., Mendling, J., Meyer, A., Pourmirza, S., Weske, M., and Wong, T. Y. (2015). Get controller and unicorn: Event-driven process execution and monitoring in logistics. In *BPM*.

Giner, P., Cetina, C., Fons, J., and Pelechano, V. (2010). Developing mobile workflow support in the internet of things. *IEEE Pervasive Computing*, 9(2):18–26.

Google (2017). Process and threads. `https://developer.android.com/guide/components/processes-and-threads.html`.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kramer, J. and Magee, J. (2007). Self-managed systems: an architectural challenge. In *Future of Software Engineering, 2007. FOSE '07*, pages 259–268.

Lee, E. A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15(3):4837–4869.

Li, Q.-s., Chu, H., Xue, B.-y., and Zhang, C. (2014). Semantic-based dynamic positioning mechanism for problem solving in multi-agent systems. *Journal of Central South University*, 21(2):618–628.

Maia, M. E. F., Fonteles, A., Neto, B., Gadelha, R., Viana, W., and Andrade, R. M. C. (2013). Loccam - loosely coupled context acquisition middleware. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 534–541. ACM.

Montagut, F. and Molva, R. (2005). Enabling pervasive execution of workflows. In *2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 10 pp.–.

Murguzur, A., Trujillo, S., Truong, H. L., Dustdar, S., Ortiz, , and Sagardui, G. (2015). Run-time variability for context-aware smart workflows. *IEEE Software*, 32(3):52–60.

Ng, I. C. and Wakenshaw, S. Y. (2017). The internet-of-things: Review and research directions. *International Journal of Research in Marketing*, 34(1):3 – 21.

Pires, P., Delicato, F., Batista, T., Avila, T., Cavalcante, E., and Pitanga, M. (2015). Plataformas para a internet das coisas. In *Minicursos do XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, chapter Multimodal Interfaces, pages 119–178. Sociedade Brasileira de Computação – SBC.

Seiger, R., Huber, S., and Schlegel, T. (2015). *PROtEUS: An Integrated System for Process Execution in Cyber-Physical Systems*, pages 265–280. Springer International Publishing.

Wieland, M., Schwarz, H., Breitenbücher, U., and Leymann, F. (2015). Towards situation-aware adaptive workflows: Sitopt x2014; a general purpose situation-aware workflow management system. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 32–37.