Lidando com Componentes Resistentes a Mudanças no Contexto de Internet das Coisas

Adrilene Fonseca, Denis de Sousa, Matheus Chagas, Paulo Henrique M. Maia

[1]Grupo de Engenharia de Software Adaptativo e Distribuído (GESAD)
Universidade Estadual do Ceará - (UECE)
Fortaleza – CE – Brazil

{adrilene.fonseca, denis.sousa, matheus.chagas}@aluno.uece.br,

pauloh.maia@uece.br

Abstract. *In the context of smart homes, where devices in a home are connected, there may be situations in which a set of independent components must collaborate to achieve a common goal, forming a System-of-Systems (SoS). In this context, there may be components resistant to change that do not meet your local requirements and the global requirement of SoS. In this work we propose a cautious adaptation using an Observer component that adjusts the behavior of resistant components in the occurrence of exceptional scenarios in IoT. The proposal was evaluated through an experiment, with which it was possible to demonstrate that our solution contributes to reaching the global requirement.*

Resumo. *No contexto de smart homes, onde dispositivos de uma casa estão conectados, podem haver situações nas quais um conjunto de componentes independentes devem colaborar para atingir um objetivo em comum, formando um Sistema-de-Sistemas (SoS). Nesse contexto, podem haver componentes resistentes a mudanças que não atendem aos seus requisitos locais e ao requisito global do SoS. Neste trabalho propomos uma forma de adaptação cautelosa utilizando um componente Observer que ajusta o comportamento de componentes resistentes na ocorrência de cenários excepcionais em IoT. A proposta foi avaliada por meio de um experimento, com o qual foi possível demonstrar que a nossa solução contribui para atingir o requisito global.*

1. Introdução

Ross e Rachel são um casal que vive em uma casa inteligente, equipada com diversos dispositivos IoT, como luzes inteligentes, smartphones, smart TVs, assistentes inteligentes, dentre outros. Todas as noites, eles colocam a filha Emma, de apenas seis meses, para dormir em seu quarto. Enquanto Rachel canta para fazer Emma dormir, Ross configura a babá eletrônica para verificar a saúde e o sono da bebê durante a noite. Esses dados são enviados para um aplicativo de smartphone específico. Quando a aplicação da babá eletrônica detecta uma anormalidade e envia uma notificação para os celulares de Ross e Rachel, eles podem acabar não vendo por estarem distraídos, por exemplo, assistindo uma série na aplicação de streaming na smart TV. Nesse caso, o aplicativo então encaminha a notificação para a smart TV, interrompendo a transmissão e alertando os pais que Emma necessita de cuidados naquele momento.

A situação descrita anteriormente pode se tornar cada vez mais comum. De acordo com um estudo feito pela MarketSource[1] em 2017, espera-se que mais da metade das casas dos Estados Unidos sejam *smart homes* até 2021, visto que os consumidores estão cada vez mais inclinados a comprar produtos inteligentes. Esses produtos fazem parte do conceito de Internet das Coisas (*Internet of Things - IoT*), onde há coisas que, através de sensores, unem o mundo físico ao mundo da informação e utilizam interfaces inteligentes para se comunicarem entre si e com os usuários [Lu Tan and Neng Wang 2010]. Lu Tan e Neng Wang (2010) atestam que as novas "*smart* coisas" devem ser capazes de processar informações, realizar auto-configuração, auto-manutenção, auto-reparo e tomar decisões independentes, o que permite que os sistemas IoT se tornem sistemas autoadaptativos, ou seja, capazes de ajustar seu comportamento em resposta a alguma variação do ambiente[Cheng et al. 2009].

Uma vez que o sistema da babá eletrônica, composto pelo dispositivo de monitorar a criança e o aplicativo de *smartphone* (*Baby Monitor App*), esteja presente em um ambiente de Internet das Coisas, é possível considerar a situação de se conectar a demais dispositivos da casa. Assim, presume-se que, ao instalar o aplicativo, o usuário fornece a permissão de acesso à *smart* TV, por exemplo, permitindo que o mesmo envie a ela mensagens que poderão ser exibidas na sua tela. Esse conjunto de componentes composto pela babá eletrônica, aplicativo de *smartphone* e a TV forma um Sistema-de-Sistemas (*System-of-Systems* - SoS), que se define como um conjunto de sistemas independentes que possuem os seus requisitos locais, e que juntos, cooperam para o cumprimento de um requisito global comum [Maier 1998]. Nesse exemplo, o requisito global é aumentar as chances de uma notificação de emergência, como a não detecção de respiração da criança ou detecção de choro intenso, ser vista pelos pais.

A TV possui um requisito local de bloquear a exibição de notificações quando a mesma estiver executando aplicações de terceiros. Com isso, um conflito emerge diante da situação em que há a tentativa de encaminhamento, por parte do *Baby Monitor App*, e a mesma está ocupada com algum serviço de *streaming*, por exemplo. Dessa forma, considerando o SoS da *smart home*, a TV pode ser vista como um componente resistente a mudanças (*defiant component* [Maia et al. 2019]), ou seja, um componente cujos requisitos locais conflitam com os requisitos globais do SoS, diante de uma situação excepcional (neste caso, a exibição da mensagem do *smartphone* durante a execução de uma outra aplicação). Maia *et al.* (2019) propõem uma adaptação cautelosa para lidar com esse tipo de componente que consiste em um criar um *wrapper*, implementado utilizando Programação Orientada a Aspectos, que muda o comportamento do componente apenas durante o cenário excepcional, preservando assim seu comportamento original nas demais situações.

Na proposta original de Maia *et al.* (2019), o uso de *wrappers* foi adequado, pois considerou-se que o código-fonte do componente resistente era inacessível. Contudo, em outros domínios em que existe algum nível de acesso ao código-fonte, como sistemas IoT que geralmente fornecem uma API (*Application Programming Interface*) de serviços, e como os componentes se comunicam de outra forma, um *wrapper* pode não ser a solução mais apropriada. Assim, este artigo traz uma nova estratégia de adaptação cautelosa em ambientes de *smart home*, no qual os dispositivos se comunicam via *bro-*

[1]https://www.marketsource.com/

ker, componente responsável pela comunicação entre os dispositivos. Nossa proposta baseia-se na inclusão de um agente observador, que conecta-se ao *broker*, verificando a ocorrência de cenários excepcionais e adaptando o componente resistente por meio de alteração nas suas configurações via chamadas de API.

O restante deste artigo está estruturado da seguinte forma: na Seção 2 os principais trabalhos relacionados são discutidos, enquanto a Seção 3 detalha a solução proposta. A Seção 4 explica o simulador desenvolvido para os testes da solução, que são apresentados na Seção 5. A Seção 6 traz discussões acerca das limitações da abordagem. A Seção 7 apresenta conclusões e trabalhos futuros. Por fim, a Seção 8 traz agradecimentos.

2. Trabalhos Relacionados

Devido ao crescimento da utilização de dispositivos inteligentes, autoadaptação torna-se uma área de estudo em ascensão nesse contexto. Nesta seção, discutimos alguns artigos que abordam adaptação em Internet das Coisas de diferentes formas.

Focando em adaptação baseada na arquitetura, Weyns *et al.* (2018a) propõem o MARTAS, um modelo que permite o uso de modelos probabilísticos no *loop* de *feedback*, cuja decisão de adaptação é tomada com base na estimativa da qualidade de cada configuração de adaptação disponível. Já o estudo de Arcaini *et al.* (2020) apresenta uma investigação sobre uma adaptação baseada em arquitetura no domínio *smart home*, que faz uso de vários *loops* MAPE-K (Monitorar, Analisar, Planejar e Executar utilizando uma base de conhecimento - *Knowledge*) para prover uma adaptação descentralizada.

Ambas as abordagens propõem uma maneira de usar *loop* de *feedback*, baseadas em MAPE-K, para realizar uma melhor adaptação dos dispositivos IoT. Nessas obras, as configurações de dispositivos são o alvo de sua adaptação. Essa escolha é semelhante à abordagem deste estudo, divergindo no uso da arquitetura para aplicar a adaptação. Neste trabalho, um componente observador é usado para assistir e enviar comandos de adaptação, diferentemente do que se encontra no estudo de Nascimento e Lucena (2016), que utilizam um agente, também chamado de observador, mas que é responsável por gerenciar e controlar cada dispositivo.

Alkhabbas *et al.* (2018) utilizam o conceito de Configuração Emergente (*Emergent Configuration* - EC) e, a partir disso, propõem uma ferramenta que permite uma formação e adaptação automatizadas de ECs, chamada de ECo-IoT. A ação do ECo-IoT requer uma interferência humana na realização da adaptação, pois sua abordagem aguarda aprovação humana. Em vez disso, neste trabalho, a adaptação entra em ação de forma automática.

Ao tratar do enfileiramento de mensagens e processos em tempo real, Chindanonda *et al.* (2019) abordam o problema de gargalos na fila, que podem diminuir o desempenho do sistema. Eles também usam as mensagens como um fator a ser observado para realizar a adaptação. No entanto, nesse caso, os destinos de adaptação são as mensagens, enquanto na abordagem deste trabalho, as mensagens são usadas para verificar a ocorrência dos cenários excepcionais.

No artigo de Weyns *et al.* (2018b) são propostas três abordagens iniciais que demonstram o potencial de sistemas IoT auto-adaptativos. Tal motivação é fundamentada nas dificuldades dos engenheiros de IoT em identificar problemas excepcionais que

possam surgir no tempo de execução de dispositivos IoT. As ideias levantadas nesse artigo estão inteiramente ligadas a aspectos de infraestrutura como: interferência, largura de banda e outros. Em nossa abordagem, realizamos uma adaptação comportamental que não está ligada aos recursos de *hardware* do sistema.

3. Apresentação da Solução

Nesta seção é explicada a proposta de solução. Primeiramente, damos uma visão geral do sistema de exemplo que utilizamos. Em seguida, detalhamos o funcionamento da solução proposta e como ela se aplica no exemplo descrito.

3.1. Visão geral do sistema

Como no exemplo citado da bebê Emma, consideramos uma babá eletrônica inteligente para aplicação da nossa solução. A Figura 1 apresenta tanto os cenários normais da aplicação (retângulos cheios) quanto os cenários excepcionais (retângulos pontilhados) utilizando a notação High Message Sequence Chart (hMSC), conforme apresentado por Maia *et al.* (2019). A babá eletrônica tem, como funcionalidade principal, o monitoramento do bem estar do bebê. Dentre as características monitoradas, estão respiração, movimentação, sons e sono. Este monitoramento é feito através do uso de câmera, sensores de áudio e movimento. Esse dispositivo está conectado a um aplicativo *mobile*, *Baby Monitor App*, para o qual envia as informações em tempo real. Os pais dão permissão ao aplicativo para conectar-se à *smart* TV da residência.

As mensagens enviadas são classificadas como normais, significando que está tudo bem com a criança, e críticas, que significam que algo não está bem, como choro ou ausência de respiração por 5 segundos. Toda notificação de mensagem crítica requer uma resposta de confirmação de visualização dentro de um tempo estipulado (neste caso, 10 segundos), pois do contrário a babá eletrônica continuará a enviar tais mensagens. O aplicativo aguarda um tempo pela confirmação de visualização e, caso o tempo de espera seja excedido, encaminha a mensagem para a *smart* TV da casa. O celular aguarda então a garantia de que o alerta foi exibido e envia a confirmação para a babá eletrônica.

Considera-se que o SoS citado anteriormente, o qual chamamos de *Baby Monitor*, formado pela babá eletrônica, *Baby Monitor App* e *smart* TV, está inserido em um ambiente de *smart home*, onde existe uma comunicação entre os dispositivos via *broker*. Considera-se, ainda, que a *smart* TV possui uma configuração que desabilita a exibição de alertas, caso esteja executando uma aplicação de terceiros (por exemplo, um aplicativo de *streaming* de vídeo).

Supondo a seguinte situação: o *smartphone* recebe uma alerta crítico, mas os pais não respondem à notificação por estarem usando uma aplicação de *streaming* na televisão naquele momento. O *Baby Monitor App* encaminha a mensagem para a TV que, por estar bloqueada, não exibe o alerta. Assim, detecta-se um conflito entre o requisito global do SoS e o requisito local da babá eletrônica, uma vez que TV não foi projetada para atender a esse cenário excepcional. Portanto, segundo a definição de Maia *et al.* (2019), nessa circunstância, a TV é um componente resistente. A solução de adaptação, neste cenário excepcional, consiste em, ao detectar que a TV está bloqueada devido à execução da aplicação, forçar a parada dessa aplicação na TV e, em seguida, reencaminhar a mensagem crítica para que seja exibida.

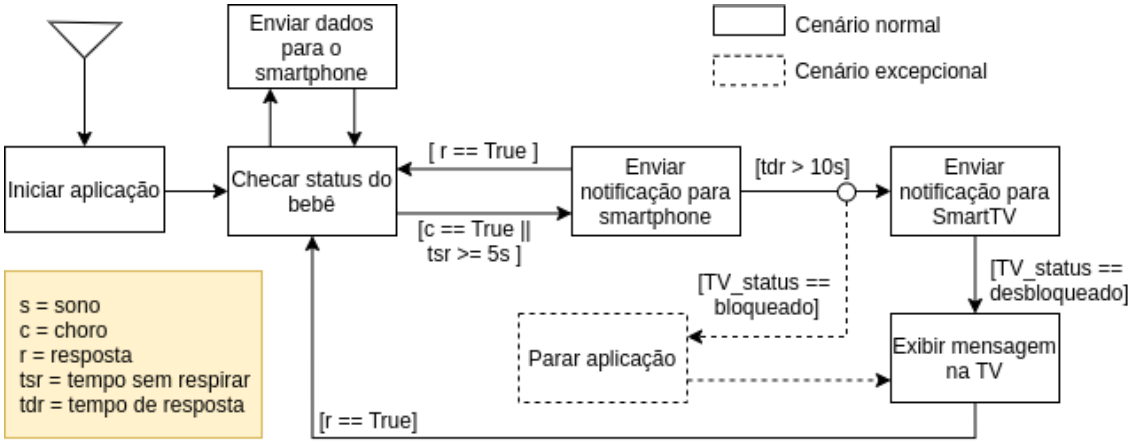

Figura 1. Diagrama de funcionamento do sistema *Baby Monitor*.

3.2. Proposta de solução

Nossa proposta consiste em adicionar um agente, nomeado de *Observer*, responsável por adaptar o componente resistente em sistemas IoT baseados em troca de mensagens entre os dispositivos. Para tanto, é necessário configurar o *Observer* com os dados de conexão do *broker* e com a definição das mensagens, fazê-lo identificar os cenários excepcionais do SoS e implementar os comportamentos adaptativos para tais cenários. Nesse contexto, assume-se que os dispositivos que serão adaptados disponibilizam uma API para ajustar configurações internas e permitir a chamada de métodos. Após esta etapa de modelagem do *Observer*, a adaptação em tempo de execução ocorrerá de acordo com os passos descritos na Figura 2.

Após o *Observer* ser configurado e conectado ao *broker*, o mesmo irá se inscrever nos tópicos relacionados ao sistema que ele irá adaptar. A partir das configurações, ele deve ser capaz de identificar as mensagens que serão observadas e as que indicam o cenário excepcional e, portanto, o momento em que ele deverá agir, de acordo com o conjunto de ações mapeadas. Após a execução de sua ação de adaptação, o componente faz a checagem do êxito da mesma, onde, caso tenha falhado, irá tentar adaptar novamente.

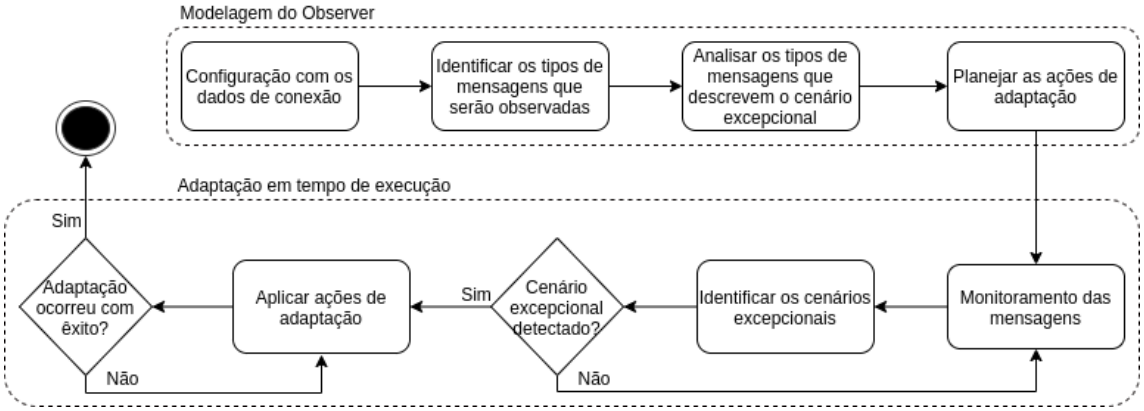

Figura 2. Diagrama de funcionamento do *Observer*

No sistema *Baby Monitor*, o *Observer* funciona da seguinte forma. Inicialmente, o componente é configurado para monitorar no *broker* as mensagens críticas (marcadas

como notificação) que não receberam confirmação. Quando não há confirmação de mensagem crítica dentro do tempo de espera, o *smartphone* encaminha essa mensagem para a *smart* TV que, se estiver bloqueada, não a exibe, configurando assim o cenário excepcional. Ao detectar esse cenário, o *Observer* envia uma requisição para a API da TV solicitando parar a execução da aplicação e, em seguida, encaminha novamente a mensagem crítica para que agora, com a TV desbloqueada, seja exibida.

4. Simulador Baby Monitor

Para testar a solução, foi desenvolvido um simulador *Web* com um servidor implementado em *Flask* [2] e comunicação utilizando RabbitMQ[3], um *broker open source* que implementa o protocolo AMQP para o envio de mensagens e que possui recursos que englobam uma baixa perda de mensagens. A ferramenta, nomeada *BabyMonitorSoS*, está disponível no repositório do GitHub [4], onde constam mais informações acerca da implementação. A aplicação permite a simulação da comunicação dos três dispositivos (babá eletrônica, aplicativo do *smartphone* e a *smart* Tv). Nesta seção, são descritas a arquitetura da ferramenta, configuração do *Observer* e a interface do simulador.

4.1. Arquitetura

A arquitetura do *BabyMonitorSoS* está estruturada de acordo com o padrão *Model-View-Controller* (MVC). A camada *Controller* possui os métodos correspondentes às possíveis ações do usuário para cada dispositivo. A funcionalidade da camada *View* é resumida em exibir a interface ao usuário e estabelecer uma conexão via *socket* com o servidor. A camada *Model* possui o modelo de dados e regras de negócios de cada dispositivo. Possui também os componentes *publisher* e *subscribers* que são responsáveis pelo processo de envio e recebimento de mensagens, respectivamente.

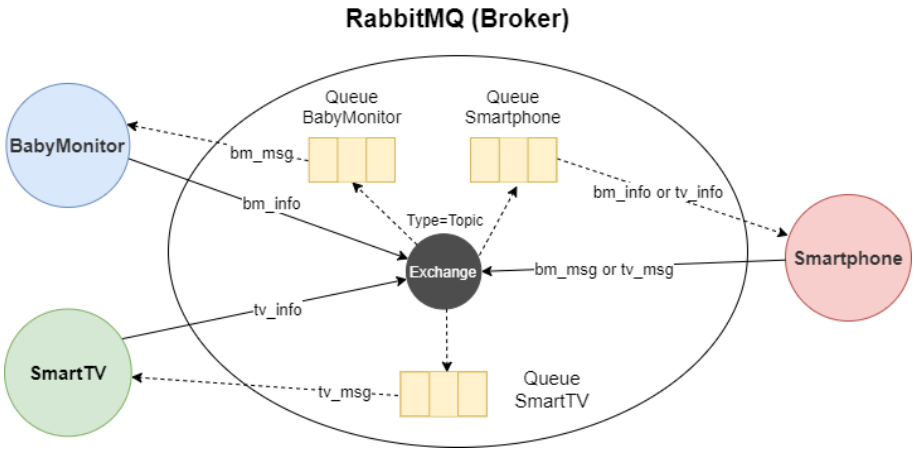

Figura 3. Configurações do *broker* utilizadas no simulador

Quanto à comunicação, as configurações utilizadas no *broker* são demonstradas na Figura 3. No momento em que um *publisher* envia uma mensagem para o *broker*, a mesma é enviada para uma *exchange* que, através de rotas previamente estabelecidas,

[2]https://flask.palletsprojects.com/en/1.1.x/
[3]https://www.rabbitmq.com/
[4]https://github.com/babymonitor-iot/babymonitor-iot

encaminha a mensagem para as filas que estão conectadas a essas rotas. As mensagens que chegam nas filas são entregues aos *subscribers* ligados a elas.

4.2. Observer

Para esse estudo, as configurações do *Observer* foram aplicadas no próprio código-fonte. As informações consistem nos dados de conexão e descrição dos passos de adaptação e retorno ao estado normal, o que representa desbloquear a TV e bloqueá-la novamente. Ele passa a receber todas as mensagens que trafegam na rede e sempre que uma nova mensagem chegar em sua fila, o mesmo testará se ocorreu o cenário excepcional. Quando ocorrer, o *Observer* irá executar seus passos de adaptação e, logo após, verificará se a adaptação ocorreu com sucesso. Caso tenha realizado com sucesso, os componentes voltam ao comportamento normal; do contrário, o *Observer* tentará aplicar a adaptação novamente, até que um sucesso ocorra.

4.3. Interface

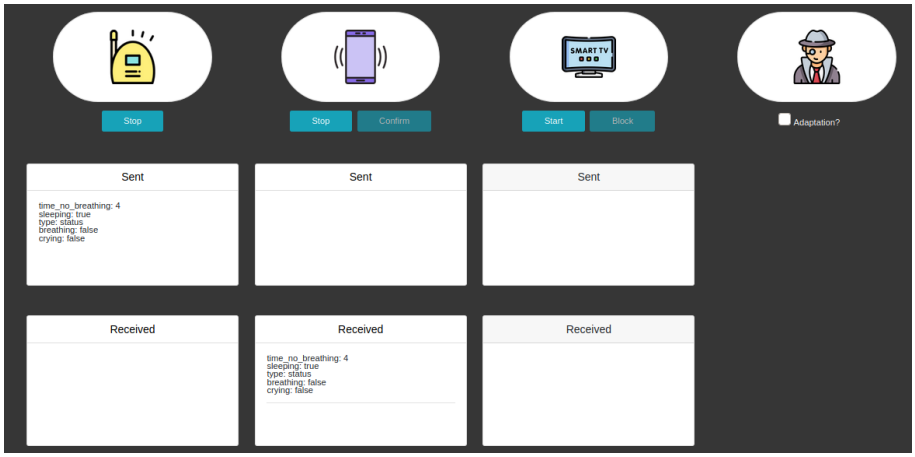

Figura 4. Interface *BabyMonitorSoS*

A interface *web* do simulador pode ser vista na Figura 4. Através dela é possível visualizar os três dispositivos que compõem o *SoS*, bem como painéis de *logs* de mensagens para cada componente. A ferramenta permite que o usuário veja o tráfego das mensagens que são enviadas entre os componentes e também consiga enxergar o estado em que as aplicações se encontram. Somente as funcionalidades necessárias para a ilustração do exemplo da babá eletrônica foram implementadas. Portanto, para cada sistema, é possível ligá-lo ou desligá-lo, e, para o *smartphone* e a *smart* TV, também é possível enviar confirmações e bloquear a exibição de alertas, respectivamente.

5. Validação

A validação foi feita por meio de uma prova de conceito, onde pretende-se mostrar que a abordagem proposta é viável no contexto de IoT. Para isso, foi utilizado o simulador *BabyMonitorSoS* considerando três possíveis cenários baseados na situação excepcional apresentada anteriormente. Assim, para diferenciar cada cenário, foram modificados os valores de probabilidade de bloqueio da TV. Definiu-se, então, o cenário *favorável* com probabilidade de 10% de a TV estar bloqueada; o *mediano* com probabilidade de 50%,

enquanto o *crítico* considera uma chance de 90% de bloqueio da TV. Para cada cenário, foram feitos três testes, denominados T1, T2, e T3, nos quais variou-se a porcentagem correspondente à chance de haver confirmação de recebimento de mensagem crítica por parte dos pais em 90%, 50% e 10%, respectivamente. Além disso, cada teste foi feito sem a solução e com a solução proposta, a fim de comparar os resultados obtidos.

Consideramos que a execução obteve sucesso quando a babá eletrônica recebe a confirmação da notificação de mensagem crítica. Logo, uma falha acontece quando a confirmação não é enviada e a TV encontra-se bloqueada para exibir alertas. Uma execução consiste em iniciar o sistema, ter uma mensagem crítica enviada e obter sucesso ou falha de notificação. Para cada teste, executamos a simulação 50 vezes com a solução e 50 vezes sem a solução, buscando avaliar a seguinte questão de pesquisa:

QP: A abordagem contribuiu para aumentar as chances da notificação ser vista?

Os resultados dos testes são apresentados na Tabela 1. Considerando os testes sem o *Observer*, nota-se que, para um cenário favorável houve falha apenas na situação em que os pais estão atentos às notificações apenas 10% das vezes, o que gerou 8% de mensagens críticas perdidas. Percebe-se que, como esperado, à medida que o cenário vai se tornando desfavorável, a taxa de sucesso (os pais confirmaram a notificação de mensagem crítica) vai caindo, se tornando mais evidente nas situações de desatenção dos pais. Por exemplo, nos cenários mediano e crítico, na pior situação, houve uma taxa de falha de 40% e 84%, respectivamente, o que representa um grande risco para a saúde do bebê.

				Sem Observer		Com Observer		
	Testes	Block	Confirmação	Sucesso%	Falha%	Sucesso%	Falha%	Observer%
Favorável	T1	10%	90%	100%	0%	100%	0%	0%
	T2	10%	50%	100%	0%	100%	0%	2%
	T3	10%	10%	92%	8%	100%	0%	10%
Mediano	T1	50%	90%	94%	6%	100%	0%	2%
	T2	50%	50%	82%	18%	100%	0%	22%
	T3	50%	10%	60%	40%	100%	0%	54%
Crítico	T1	90%	90%	94%	6%	100%	0%	4%
	T2	90%	50%	56%	44%	100%	0%	32%
	T3	90%	10%	16%	84%	100%	0%	82%

Tabela 1. Resultados dos experimentos

Analisando agora os testes que tiveram a intervenção do *Observer*, nota-se que, independentemente do cenário, não houve nenhuma perda de notificação de mensagem crítica, o que significa que em todas as vezes que a bebê necessitou ajuda, os pais conseguiram confirmar a notificação, o que deve ter minimizado problemas com a criança. É também possível ver que, quanto mais crítico o cenário, maior é o benefício do *Observer*, que foi acionado 10%, 54% e 82% das vezes na pior situação nos cenários favorável, mediano e crítico, respectivamente.

A partir dos testes realizados e dos resultados obtidos, é possível atestar que a solução proposta é viável no contexto de IoT. A abordagem conseguiu auxiliar o SoS apresentado a atender o seu requisito global, o que caracteriza uma resposta positiva à questão de pesquisa posta.

6. Discussão

A validação feita mostrou que a nossa proposta contribuiu para que o SoS atingisse seu requisito global, utilizando uma estratégia de solução diferente da que utiliza *wrappers*. No entanto, estamos cientes de limitações existentes e trazemos nesta seção uma discussão acerca das mesmas.

Inicialmente, a partir dos resultados apresentados, não é possível generalizar a solução, já que os testes foram baseados em um exemplo específico e executados por meio de simulações, e não em um ambiente real. Esses fatores podem afetar os resultados, visto que variáveis de um ambiente real podem não ter sido consideradas.

É importante reforçar que a nossa abordagem não se destina a revelar todos os conflitos de um ambiente de *smart home*, visto que o conflito é particular do cenário excepcional causado pela composição do SoS. Para ajudar na modelagem de tais cenários, outras abordagens podem ser utilizadas, como um processo de modelagem cenários de adaptação para SoS [Maciel et al. 2019].

Outra limitação é a especificidade da abordagem para uma arquitetura baseada em *broker*, i.e., para outras arquiteturas de IoT, outras propostas podem ser mais adequadas. Quanto à avaliação, foram testadas apenas variações do mesmo cenário do *Baby Monitor*. Portanto, para novos cenários envolvendo outros componentes, novas validações precisariam ser feitas para avaliar a abrangência da abordagem.

7. Conclusão

Neste artigo foi proposta uma abordagem de adaptação cautelosa de componentes resistentes a mudanças em sistemas-de-sistemas no contexto de Internet das Coisas, cuja comunicação entre os dispositivos é feita com troca de mensagens via *broker*. A proposta consiste em configurar um módulo extra, chamado de *Observer*, e inseri-lo no sistema, sendo ele responsável por monitorar as mensagens trocadas e realizar a ação de adaptação quando o cenário excepcional ocorrer.

A solução foi aplicada em um SoS constituído em um ambiente de *smart home*, formado por uma babá eletrônica, um aplicativo *smartphone* e uma *smart* TV, sendo esta última o componente resistente à mudança no cenário excepcional de estar bloqueada e não exibir mensagens críticas sobre saúde do bebê. Neste exemplo, o *Observer*, ao detectar a ocorrência do cenário excepcional, desbloqueia a TV, que então consegue exibir a mensagem. Foi implementado um simulador para esse SoS, com o qual foram realizados experimentos que demonstraram que nossa abordagem é eficaz, garantindo que o sistema execute sem haver falhas quanto ao cumprimento do requisito global do SoS.

Como trabalhos futuros, pretendemos estender o exemplo do *Baby Monitor* com outros dispositivos e aplicar a solução em outros cenários e arquiteturas de IoT. Também planejamos tornar o *Observer* uma ferramenta plugável e isolada do sistema, possibilitando sua configuração fora do código-fonte. E, ainda, pretende-se investigar soluções semelhantes em outros domínios e que possam ser implementadas como adaptação cautelosa.

8. Agradecimentos

Esse trabalho é parcialmente financiado pela FUNCAP por meio do projeto número UKA-0160-00005.01.00/19.

Referências

Alkhabbas, F., Spalazzese, R., and Davidsson, P. (2018). Eco-iot: An architectural approach for realizing emergent configurations in the internet of things. In Cuesta, C. E., Garlan, D., and Pérez, J., editors, *Software Architecture*, pages 86–102, Cham. Springer International Publishing.

Arcaini, P., Mirandola, R., Riccobene, E., Scandurra, P., Arrigoni, A., Bosc, D., Modica, F., and Pedercini, R. (2020). Smart home platform supporting decentralized adaptive automation control. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 1893–1900, New York, NY, USA. Association for Computing Machinery.

Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Di Marzo Serugendo, G., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G., Kienle, H. M., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H. A., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D., and Whittle, J. (2009). *Software Engineering for Self-Adaptive Systems: A Research Roadmap*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chindanonda, P., Podolskiy, V., and Gerndt, M. (2019). Metrics for self-adaptive queuing in middleware for internet of things. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pages 130–133.

do Nascimento, N. M. and de Lucena, C. J. P. (2017). Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things. *Information Sciences*, 378:161 – 176.

Lu Tan and Neng Wang (2010). Future internet: The internet of things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–376–V5–380.

Maciel, M., Maia, P. H., Oliveira, F. C. M. B., and Maciel, F. (2019). Adore: An adaptation-oriented requirement modeling approach for systems of systems. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, SBES 2019, page 166–171, New York, NY, USA. Association for Computing Machinery.

Maia, P. H., Vieira, L., Chagas, M., Yu, Y., Zisman, A., and Nuseibeh, B. (2019). Cautious adaptation of defiant components. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 974–985.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Weyns, D., Iftikhar, M. U., Hughes, D., and Matthys, N. (2018a). Applying architecture-based adaptation to automate the management of internet-of-things. In Cuesta, C. E., Garlan, D., and Pérez, J., editors, *Software Architecture*, pages 49–67, Cham. Springer International Publishing.

Weyns, D., Ramachandran, G. S., and Singh, R. K. (2018b). Self-managing internet of things. In Tjoa, A. M., Bellatreche, L., Biffl, S., van Leeuwen, J., and Wiedermann, J., editors, *SOFSEM 2018: Theory and Practice of Computer Science*, pages 67–84, Cham. Springer International Publishing.