UNIVERSIDADE FEDERAL DA BAHIA

TRABALHO DE GRADUAÇÃO

Detecção inteligente de efeitos colaterais indesejáveis na Internet das coisas - um estudo de caso no Home Network System

Heron Sanches Gonçalves Pires Ferreira

Programa de Graduação em Ciência da Computação

Salvador
31 de outubro de 2016

HERON SANCHES GONÇALVES PIRES FERREIRA

DETECÇÃO INTELIGENTE DE EFEITOS COLATERAIS INDESEJÁVEIS NA INTERNET DAS COISAS - UM ESTUDO DE CASO NO HOME NETWORK SYSTEM

Este Trabalho de Graduação foi apresentado ao Programa de Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Daniela Barreiro Claro
Co-orientador: Roberto Cerqueira Figueiredo

Salvador
31 de outubro de 2016

Ficha catalográfica.

Ferreira, Heron Sanches Gonçalves Pires

Detecção inteligente de efeitos colaterais indesejáveis na Internet das coisas - um estudo de caso no Home Network System/ Heron Sanches Gonçalves Pires Ferreira– Salvador, 31 de outubro de 2016.

27p.: il.

Orientadora: Profa. Dra. Daniela Barreiro Claro.
Co-orientador: Roberto Cerqueira Figueiredo.
Monografia (Graduação)– UNIVERSIDADE FEDERAL DA BAHIA, INSTITUTO DE MATEMÁTICA, 31 de outubro de 2016.

"1. Efeitos Colaterais Indesejáveis. 2. Internet das Coisas. 3. Home Network System. 4.Dispositivos como serviços Web RESTFul. 5.DECORATE. 6.Cross-validation".
I. Claro, Daniela Barreiro. II. .
III. UNIVERSIDADE FEDERAL DA BAHIA. INSTITUTO DE MATEMÁTICA. IV. Título.

CDD 005.13307

À minha mãe.

AGRADECIMENTOS

Obrigado minha mãe, por sua dedicação durante todos estes anos. Obrigado minha tia Suzi, por ter me dado oportunidade de estudo e sempre está disposta a me ajudar em qualquer situação. Obrigado as forças boas que vêem me guiando e me ajudando. Obrigado Daniela e Roberto pela orientação prestada para este trabalho.

''Fale para ele não desistir''.

—MENSAGEM ESPÍRITA.

RESUMO

Diversos dispositivos têm sido incorporados em ambientes domésticos através dos *Home Server* (HS) com o intuito de automatizar as tarefas cotidianas de uma casa inteligente. Porém, muitos destes dispositivos independentes não conseguem atender aos requisitos dos usuários. Assim, há a necessidade de composição destes com outros dispositivos para prover funcionalidades que atendam a esses requisitos. Quando existe uma composição de dispositivos, o comportamento de pelo menos um destes pode provocar interação de características com efeitos colaterais indesejáveis. Assim, o presente trabalho propõe detectar efeitos colaterais indesejáveis entre dispositivos utilizando-se do *ensemble* DECORATE. Para isto, observou-se o comportamento físico e as mensagens trocadas entre os dispositivos, os quais foram disponibilizados como serviços Web RESTFul. Através do conjunto de dados obtidos realizou-se um processo classificatório e como resultado final foi obtido um modelo com mais de 90% de precisão, o qual foi incorporado no FIM (*Feature Interaction Manager*) do *Home Network System* (HNS), mostrando que e possıvel detectar efeitos colaterais indesejaveis entre dispositivos.

Palavras-chave: dispositivos, serviços Web RESTFul, *Feature Interaction Manager*, *Home Network System*, efeitos colaterais indesejáveis, *ensenble* DECORATE

ABSTRACT

Many devices have been put on domestics environments through the Home Servers with the aim of automate the daily tasks of a smart home. But, some devices when alone do not attend the user specifications. Then, there is the needed to compose them to provide features that are accord to them specifications. When there is a device's composition, the behave of at least one of them can cause feature interactions with undesirable side effects. So, this work proposes detect undesirable side effects between devices using the ensemble DECORATE. For it propose was realized observations of the physical devices behave and the messages changed by these devices, which were made available how Web RESTFul services. Through the data collected was made a classification process and as the final result was obtained a model with more than 90% of precision. It was put into FIM (Feature Interaction Manager) of the Home Network System (HNS), showing that is possible detect undesirable side effect between these devices.

Keywords: devices, Web RESTFul services, Home Network System, undesirable side effect, Feature Interaction Manager, ensemble DECORATE

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução sobre do que se trata esta monografia e a maneira como o texto está organizado.

INTRODUÇÃO

A Internet nesta última década tem contribuído de forma significativa na economia e sociedade, deixando como legado uma notável infraestrutura de rede de comunicação. O seu maior disseminador nesse período vem sendo *World Wide Web*(WWW), o qual permite o compartilhamento de informação e mídia de forma global (Chandrakanth et al., 2014).

A Internet está se tornando cada vez mais persistente no cotidiano, devido, por exemplo, ao crescente número de usuários de dispositivos móveis, os quais possuem tecnologias de conexão com a Internet, as quais cada dia tornam-se mais acessíveis (Chandrakanth et al., 2014).

Em 2010 havia aproximadamente 1,5 bilhão de PCs conectados a Internet e mais que 1 bilhão de telefones móveis (Sundmaeker et al., 2010). Segundo Gartner[1], 6,4 bilhões de coisas estarão conectadas até o final de 2016 e, em 2020 esse número atingirá cerca de 20,8 bilhões. A previsão de (Sundmaeker et al., 2010), a qual dizia que a denominada Internet dos PCs seria movida para o que se chama de Internet das Coisas fica então mais evidente neste atual cenário.

A ideia básica da *Internet of things (IoT)*, traduzido para o português como Internet das Coisas é a presença pervasiva de uma variedade de "coisas ou objetos", tais como RFID tags, sensores, telefones móveis, dentre outros. Os quais, através de esquemas de endereçamento único são capazes de interagir com os outros e cooperar com seus vizinhos para alcançar um objetivo em comum (Atzori et al., 2010).

Apesar da grande potencialidade da Internet das Coisas, a qual gerou em 2015 um faturamento em torno de 130,33 bilhões de dólares americanos e tem prospecção de chegar até 883.55 bilhões em 2020[2], ainda existem muitos desafios a serem vencidos, tais como segurança da informação, armazenamento e processamento de grande quantidade de dados, dentre outros. Adentrando um pouco em um dos desafios da IoT, o da disponibilidade

[1] ⟨http://www.gartner.com/newsroom/id/3165317⟩

[2] http://www.marketsandmarkets.com/Market-Reports/iot-application-technology-market-258239167.html

de uma interface de comunicação (acesso aos serviços e informações dos dispositivos) e programação comum aos objetos, pode-se dizer que a falta desta padronização faz com que se torne oneroso o desenvolvimento de aplicações para o objeto. Mais difícil ainda é prover uma única funcionalidade ou serviço com a composição dos diversos objetos. Para diminuir a dificuldade deste cenário, pode-se disponibilizar os dispositivos como serviços Web, desta forma pode-se utilizar os protocolos Web como linguagem comum de integração dos dispositivos a Internet (Franca et al., 2011).

Diante deste cenário de crescente adição de dispositivos a IoT, muitos dispositivos quando de forma isolada, não conseguem atender aos requisitos dos usuários finais (humano ou outro dispositivo). Assim, há a necessidade de composição destes com outros dispositivos para prover funcionalidades que atendam a esses requisitos. Quando existe uma composição de dispositivos, o comportamento de pelo menos um destes pode provocar interação de características com efeitos colaterais indesejáveis, que é quando esta composição leva a um estado inconsistente do sistema, um sistema instavel ou dados imprecisos.

Assim, o presente trabalho propõe detectar efeitos colaterais indesejáveis entre dispositivos através da meta-classificação, metodologia que utiliza um conjunto de classificadores para construir diversas hipóteses (modelos). Então, é utilizado o conhecimento de cada modelo afim de se chegar em uma decisão final mais confiável (Melville e Mooney, 2004).

Os dados de treino foram gerados manualmente a partir de um conjunto de objetos do cenário "Levar compras", o qual foi utilizado no estudo de caso do *Home Network System*.

Os resultados obtidos demonstraram que através da meta-classificação foi possível detectar os efeitos colaterais indesejáveis com mais de 90% de precisão.

O restante deste trabalho está organizado da seguinte maneira:

- Capítulo 5: Traz todo embasamento teórico necessário para compreender o estudo realizado neste trabalho.
- Capítulo 3: Este capítulo descreve a proposta para detecção de efeitos colaterais indesejáveis no cenário "Levar compras".
- Capítulo 4: Trata dos procedimentos realizados para validar a proposta de pesquisa deste TCC.
- Capítulo 6: Este capítulo é uma síntese da investigação e dos experimentos realizados neste TCC.

Capítulo 2

Este capítulo tem como objetivo fundamentar as bases necessárias dos campos de estudos utilizados nesta monografia.

FUNDAMENTAÇÃO TEÓRICA

2.1 INTERNET DAS COISAS

Há anos atrás, por volta de 2009, a informação gerada na Internet era quase toda dependente da ação humana. Os aproximadamente 50 PB *petabytes* gerados na Internet até este ano de 2009 foram gerados por humanos. Como, por exemplo, através de um clique numa câmera digital, digitalização de um código de barras, digitando texto, pressionando um botão para gravar. O problema é que o ser humando tem capacidade finita de atenção e acurácia, o que não é bom para capturar dados das coisas do mundo real. Sendo assim, há a necessidade de prover aos computadores seus próprios meios de coletar informação para que estes possam sentir o ambiente em que se encontram. RFID e sensores permitem que estes observem, identifiquem e compreendam o ambiente em que estão imersos sem a necessidade de dados que são introduzidos por humanos (Ashton, 2009). Assim sendo, *Internet of Things* (IoT), traduzido para o português como Internet das Coisas, pode ser definida como uma rede de coisas que têm a capacidade de sentir, identificar e compreender o ambiente em que estão imersas.

"Coisas ou objetos", tais como, RFID tags, sensores, telefones móveis, dentre outros. Estes, atraves de esquemas de enderecamento unico sao capazes de interagir com os outros e cooperar com seus vizinhos para alcancar um objetivo em comum (Atzori et al., 2010). Outros exemplos de "coisas ou objetos" podem ser pessoas, geladeiras, televisores, veículos, roupas, medicações, livros, passaportes, contanto que possam ser identificadas unicamente e possam se comunicar com as outras coisas e/ou possam ser acessados remotamente por humanos ou por máquinas.

Uma definição tecnológica para IoT pode ser como em (Sundmaeker et al., 2010), o qual diz que IoT é parte integrante da futura Internet e pode ser definida como uma infraestrutura de rede global dinâmica com capacidades de autoconfiguração baseada nos padrões e interoperabilidade dos protocolos de comunicação onde coisas físicas e virtuais têm identidade, atributos físicos, personalidade virtual, usam interfaces inteligentes e, são integradas dentro da rede de informações.

Diante deste cenário pode-se citar exemplos de aplicações na IoT para diversos domínios, tais como, logística e transporte; cuidados com a saúde; ambiente inteligente (casa (Seção **??**), escritório); (Atzori et al., 2010) verificação de procedência alimentícia; dentre outros. Seguem dois exemplos de aplicações, uma na área de cuidados com a saúde e outra na verificação de procedência de alimentos, respectivamente.

- Dispositivos implantáveis com capacidade de comunicação sem fio podem ser utilizados para armazenar registros sobre a saúde de um paciente em situações de risco e podem ser decisivos para salvar a vida do paciente. A capacidade de ter acesso a essas informações nessas circunstâncias fazem com que hospitais possam saber de imediato como tratar um paciente que esta a caminho. Esta possibilidade é especialmente útil para pessoas com diabetes, câncer, problemas de coração na artéria coronária, doenças do pulmão, assim como as pessoas com implantes médicos complexos, tais como, marcapassos, tubos, transplantes de órgãos e aqueles que podem ficar inconscientes e incapazes de comunicar-se durante uma operação (Weber e Weber, 2010).

- Rastreabilidade de produtos alimentícios ajudam os usuários a verificar a origem de um produto, assim como informações de composição química, dentre outros. Mas também previne de doenças não desejadas. Por exemplo, avisos atuais sobre a mercadoria em questão podem ser disponibilizados à medida que o produto sai da origem e vai passando para os outros níveis de consumo, desta forma os consumidores podem evitar o contágio de doenças como gripe aviária e, encefalopatia espongiforme bovina (EEB), mais conhecida como doença da vaca louca (Weber e Weber, 2010).

2.1.1 Dispositivos como serviços Web RESTFul

Serviços Web surgiram tendo como principal foco o reúso de aplicações existentes (dentre as quais incluíam código fonte legado) para que pudessem se integrar de forma leve com outras aplicações. Geralmente essa integração tinha como referência o desejo de novas formas de compartilhamento dos serviços ao longo das diversas linhas do negócio ou entre parceiros (Papazoglou, 2008).

Um dos desafios referentes a visão da IoT é sua interoperabilidade e, como possível solução se pode disponibilizar os dispositivos como serviços Web, por exemplo, seguindo os príncipios REST. Desta maneira os dispositivos podem interagir entre si e com outros sistemas na Web.

Segundo (Heffelfinger, 2014), Representational State Transfer (REST) é um estilo de arquitetura no qual serviços Web são vistos como recursos e podem ser identificados por Uniform Resources Identifiers (URIs). Serviços Web que são desenvolvidos usando REST são conhecidos como RESTful web services.

Os principais princípios do estilo arquitetural REST são: endereçamento global através de identificação dos recursos, interface uniforme compartilhada por todos os recursos, interações *stateless* entre os serviços e, mensagens auto-descritivas e *hypermedia* como um

mecanismo para descoberta descentralizada de recursos por referência (Pautasso, 2014), conforme abaixo.

1. Identificação dos recursos - todos os recursos que são publicados por um serviço Web deveriam ser disponibilizados por um único e estável identificador global (Pautasso, 2014). A exemplo das URIs (Franca et al., 2011).

2. Interface uniforme - todos os recursos interagem através de uma interface uniforme, a qual prover um conjunto de métodos pequeno, genérico e funcionalmente suficiente para suportar todas as possíveis interações entre os serviços. Cada método tem uma semântica bem definida em relação aos efeitos que causará no estado do recurso. No contexto da Web e do protocolo HTTP que é utilizado, "Interface uniforme" pode ser alcançado com os seus métodos (GET, PUT, DELETE, POST, HEAD, OPTIONS, dentre outros) os quais podem ser aplicados para todos os identificadores dos recursos Web (URIs) (Pautasso, 2014).

3. Interações *stateless* - os serviços não podem estabelecer sessões permanentes entre eles. Isto assegura que as requisições para um recurso sejam independentes entre si. No final de cada interação, não há estados compartilhados entre clientes e servidores. Requisições podem resultar em uma mudança de estado do recurso, mas o novo estado é imediatamente visível para todos clientes (Pautasso, 2014).

4. Mensagens auto-descritivas - Serviços interagem através de requisição e mensagem de resposta, que contem tanto os dados (representações dos recursos) e correspondente *meta-data*. As representações podem variar de acordo com o contexto do cliente, interesses e habilidades. Por exemplo, um cliente *mobile* pode obter uma representação do recurso que exige pouco consumo de banda de dados. Da mesma forma, um navegador pode requisitar a representação de uma página Web em uma linguagem específica, de acordo com suas preferências. Esta característica aumenta de maneira significativa o grau de interoperabilidade, pois um cliente pode dinamicamente negociar o mais apropriado formato de representação com o recurso (também conhecido como media type), em vez de ser forçado a sempre trabalhar com uma determinada representação do recurso. As requisições e mensagens de respostas também devem conter explicitamente *meta-data* sobre sua representação, desta maneira os serviços não precisam assumir algum tipo de acordo de sobrecarga no sentido de como tal representação seria analisada, processada e entendida (Pautasso, 2014).

5. *Hypermedia* - Recursos podem ser relacionados entre si. *Hypermedia* embute referências a outros recursos relacionados ou dentro de suas representações ou em sua correspondente *metadata*. Clientes então podem descobrir os *hyper-links* dos recursos relacionados ao processar suas representações e escolher seguir o link. Como exemplificado em (Franca et al., 2011), um sistema de uma instituição que possui um recurso lista_cursos (lista todos os cursos da instituição), este pode oferecer os *hyper-links* que representam cada recurso que representa um determinado curso.

> *Hypermedia* auxilia na descoberta descentralizada de recursos e também pode ser utilizada para descoberta e descrição de protocolos de interação entre os serviços. Pelo fato deste ser o princípio menos utilizado pelas APIs que se dizem ser RESTFul, algumas vezes as APIs disponibilizadas por serviços Web que além das outras restrições contemplam esta, são também chamadas de *Hypermedia APIs* (Pautasso, 2014).

A disponibilização dos dispositivos como serviços Web RESTFul vêm sendo empregada através de duas abordagens. Na primeira, quando os dispositivos têm recursos suficientes (memória, processamento e largura de banda de rede) servidores Web são embarcados nos próprios dispositivos e estes são disponibilizados como recursos REST. Enquanto que na segunda, quando uma coisa não possui recursos suficientes para tal propósito, utiliza-se de outro dispositivo, por exemplo, um [1]Raspberry Pi ou qualquer outro controlador com recursos suficientes para instalação e execução de um servidor Web. Este então atua como um intermediador entre o objeto e a Internet (Franca et al., 2011).

Adotando esse padrão os dispositivos podem ter suas propriedades disponíveis através de qualquer navegador, sem a necessidade de instalação de nenhum programa ou driver adicional, como pode ser exemplificado na Figura 2.1. Nesta, o dispositivo tem suas informações acessadas através de uma requisição GET, que é realizada passando o URI (endereço que identifica a funcionalidade do serviço de forma única) do dispositivo ao navegador, e obtêm-se um retorno no formato JSON[2], o qual tem as informações do dispositivo: ligado (informa se o dispositivo está em operação); obstáculo (informa se existe algum obstáculo na linha do sensor); anguloDetecção (informa o ângulo de detecção); faixa de detecção (informa qual a faixa de detecção ajustada no momento); status (4 - informa que a requisição ao dispositivo foi processada sem problemas).

Figura 2.1: Dispositivo sensor de obstáculo do elevador disponibilizado como serviço. Faixa de detecção compatível com a largura do elevador utilizado na maquete do cenário (Figura 3.3)

[1]https://www.raspberrypi.org/products/
[2]⟨http://www.json.org/⟩

Além disso, mashups físicos[3] podem ser construídos com muito menos esforço do que as existentes abordagens, minimizando a barreira para o desenvolvimento de aplicações com dispositivos (Guinard e Trifa, 2009). Pelo fato das coisas serem disponibilizadas como serviços Web, ainda pode-se utilizar-se dos outros recursos disponíveis na Web, a exemplo de sistemas de cache, balanceamento de carga, indexação e pesquisa (Franca et al., 2011). Assim então promovendo a visão da Internet das Coisas.

Diversas comparações

[3]Aplicativos criados a partir da composição de dados e serviços de dispositivos físicos com outros recursos Web.

Capítulo 3

Este capítulo descreve a proposta para detecção de efeitos colaterais indesejáveis no cenário "Levar compras"

PROPOSTA

. Diante da visão da IoT propõe-se detectar efeitos colaterais indesejáveis de uma maneira inteligente em um HNS (Figura 3.1), mais especificamente no cenário "Levar compras" (seção 3.1). O cenário "Levar compras" foi desenvolvido com o intuito de criar um ambiente no HNS que tivesse efeitos colaterais indesejáveis. Uma vez que há efeitos colaterais indesejáveis, surge a necessidade de se detectar estes efeitos. O método de detecção destes efeitos utilizou meta-classificação com o intuito de gerar um modelo satisfatório para esta detecção (seção 4). Uma vez que o modelo foi validado, este foi implantado no cenário "Levar compras" do FIM.

3.1 EFEITOS COLATERAIS INDESEJÁVEIS NO HNS (CENÁRIO LEVAR COMPRAS)

O cenário "Levar compras" refere-se a incorporação de efeitos colaterais indesejáveis no HS. Esta funcionalidade é provida através da composição dos serviços do elevador, veículo e garra. O cenário em questão pode ser visualizado na Figura 3.2 e é descrito a seguir.

1. Um usuário da casa (morador ou visitante), organiza as compras numa cesta (que tem formato expansível) e a coloca sobre o veículo terrestre.

2. O usuário então aciona o serviço "Levar compras" (disponibilizado pelo HS) através do seu *smartphone*. O serviço então começa sua execução.

3. O FIM solicita informações de restrições do elevador e veículo terrestre, assim como quais objetos compõem a cesta. Este então verifica se a cesta colocada sobre o carro quebrou alguma restrição do mesmo e se a massa total dos objetos excedem a carga máxima do elevador.

4. Se nenhuma destas restrições foram quebradas o FIM solicita ao veículo terrestre que leve a as compras até as proximidades do elevador.

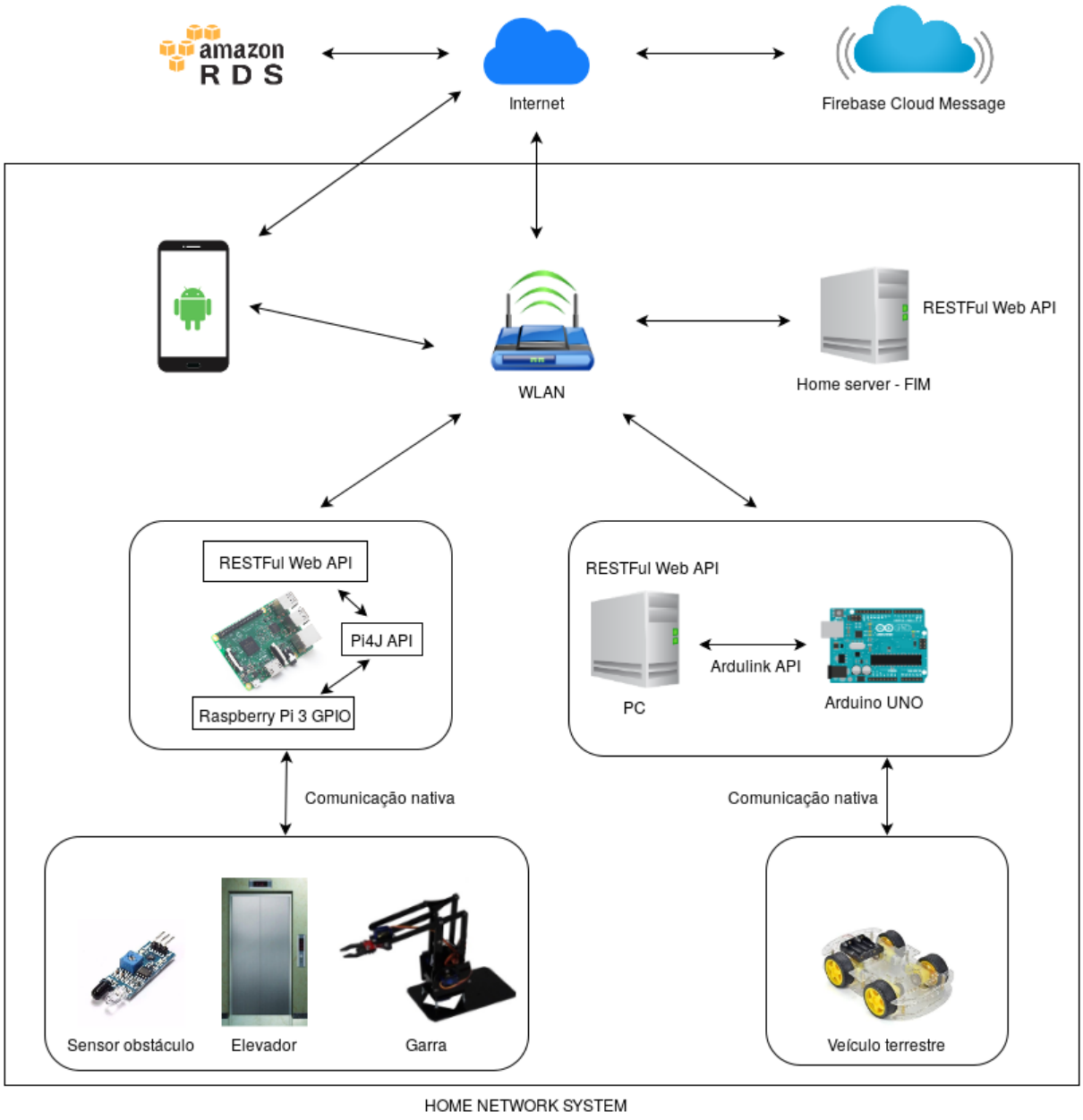

Figura 3.1: Modelo do HNS utilizado neste trabalho.

5. Assim que o veículo chega no seu destino, este envia uma mensagem ao FIM informando de sua chegada.

6. O FIM então requisita o elevador para levar as compras.

7. O elevador agora solicita a garra mecânica que pegue o cesto e coloque-o dentro do elevador.

8. Quando a garra completa a execução do seu serviço, esta avisa ao elevador.

9. Caso não haja nenhuma restrição, como por exemplo, alguma coisa que possa impedir a porta do elevador de fechar, então o elevador inicia o transporte das compras ao seu destino final e manda uma notificação ao HS.

10. HS Captura token referente ao *smartphone* com Android[1] do usuário da casa. Por fim, o HS solicita ao serviço de notificação *Firebase Cloud Message*[2] que envie uma notificação para o usuário da casa, que a recebe no seu *smartphone*. Assim completa-se a execução do serviço "Levar compras".

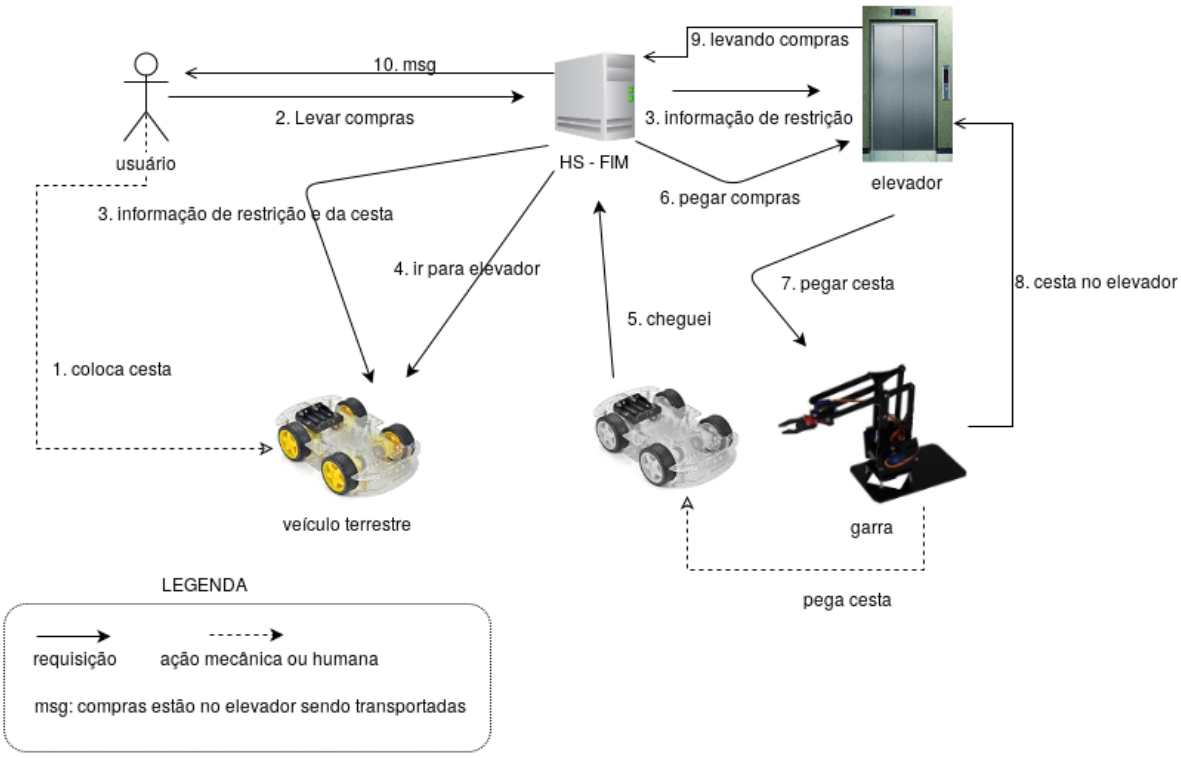

Figura 3.2: Cenário Levar compras.

No cenário descrito da Figura 3.2 o elevador não contém motores nem algum outro tipo de sistema elétrico ou mecânico, este é representado por uma caixa de papelão e um sensor de obstáculo acoplado. O veículo terrestre é representado por um programa carregado no Arduino UNO[3], o qual escuta através de sua porta serial mensagens provindas do PC (Figura 3.2) e as processa simulando as funcionalidades "Ir para elevador" e "cheguei". Pela mesma porta serial o veículo terrestre encaminha suas mensagens ao PC. Os dispositivos veículo, elevador e garra podem ser visualizados na Figura 3.3.

[1] ⟨https://www.android.com/intl/pt-BR_br/⟩
[2] ⟨https://firebase.google.com/docs/cloud-messaging/?hl=pt-br⟩
[3] ⟨https://www.arduino.cc/en/Main/ArduinoBoardUno⟩

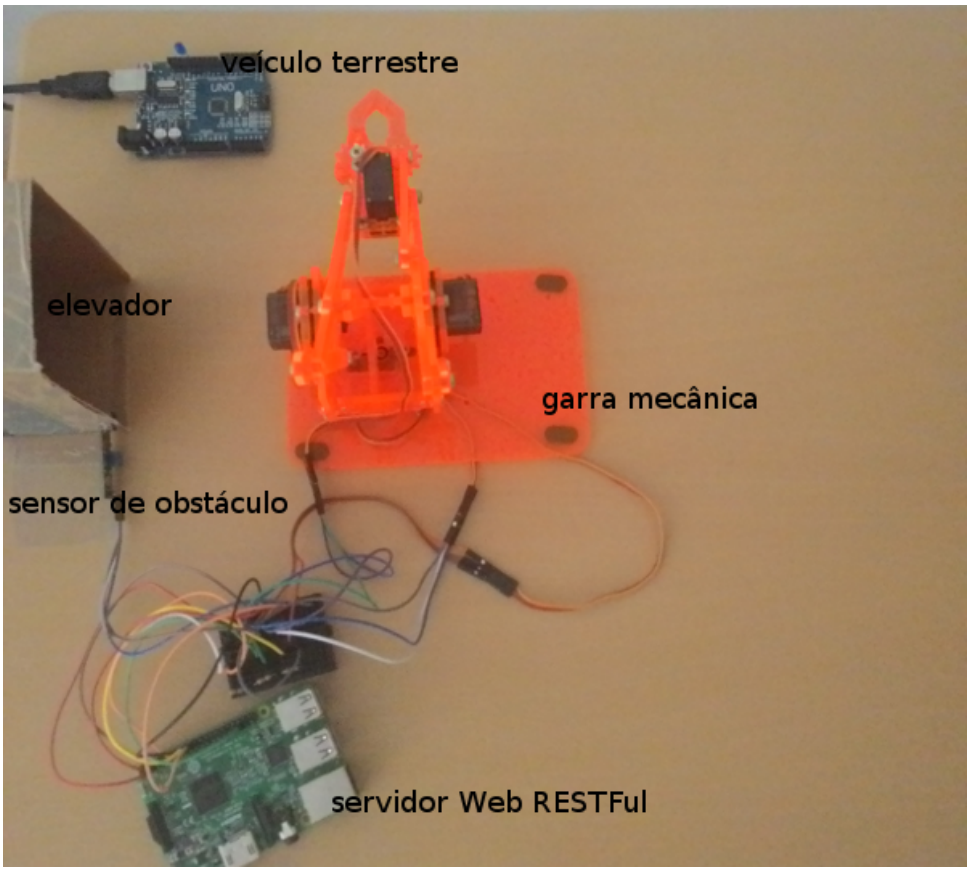

Figura 3.3: Dispositivos reais do cenário "Levar compras".

3.1.1 Execução do cenário Levar compras

De acordo com a Figura 3.2 o primeiro passo para inicializar a execução do cenário é colocar a cesta no veículo terrestre. Apesar do veículo ser simulado através de um software embarcado no Arduino, este necessita das informações dos objetos que se deseja transportar. Sendo assim, as informações da cesta são passadas quando o usuário executa o passo 2 (Levar compras). O usuário digita o "id da cesta" (no banco de dados) que se deseja transportar e clica em um botão para inicializar o serviço. Todas as cestas são compostas com miniaturas de objetos que normalmente são encontrados em supermercados, exemplos de cestas podem ser visualizados na Figura 3.4. Todos os objetos e cestas estão cadastradas no banco de dados na nuvem (amazon RDS[4]), o qual foi apresentado na Figura 3.1. A lista de objetos que podem fazer parte de uma cesta é demonstrada na Figura 3.5, observe que para cada objeto é cadastrado suas características de massa, largura, comprimento, altura e diâmetro.

Os próximos passos (3, 4, 5, e 6) do cenário já têm a informação necessária e não necessita de nenhuma intervenção humana. Isto já não acontece com o passo 7 (pegar cesta), pois este necessita de intervenção humana para a garra poder pegar a cesta e

[4] ⟨https://aws.amazon.com/pt/rds/postgresql/⟩

Figura 3.4: Exemplo de cestas do cenário “Levar compras“.

largura	altura	comprimento	massa	diâmetro	nome
0	0.031	0	0.003	0.009	Refrigerante nº 4 - Soda
0.01	0.014	0.03	0.001	0	Caixa Pasta de dente - vermelha.
0.004	0.02	0.014	0.001	0	Batata
0.004	0.02	0.014	0.001	0	Café
0	0.015	0	0.003	0.008	Café Classic
0.005	0.02	0.011	0.001	0	Cerveja em lata III
0	0.028	0	0.001	0.007	Champanhe - Azul
0.004	0.02	0.014	0.001	0	Farinha de Trigo
0	0.03	0	0.001	0.008	Garrafa de Rum
0.012	0.016	0.004	0.004	0	Goiabada
0	0.03	0	0.001	0.008	Garrafa de Vodka Special
0.027	0.014	0.018	0.001	0	Pote de sorvete
0	0.032	0	0.002	0.007	Vinho branco
0.012	0.03	0.006	0.002	0	Limpa vidros
0.021	0.015	0.007	0.002	0	Sabonete - II
0.015	0.024	0.008	0.001	0	Sabão em pó - II
0.019	0.03	0.039	0.018	0	Baú simples
0.046	0.061	0.017	0.006	0	Prateleira de acrílico pequena
0.016	0.025	0.004	0.001	0	Farinha de trigo
0	0.03	0	0.003	0.022	Barril em madeira
0.013	0.017	0.0063	0.002	0	Fermento em pó
0.077	0.0197	0.038	0.001	0	Colchão p/ berço azul
0.01	0.0432	0	0.001	0	Jogo de esfregões
0.0181	0.0258	0.0017	0.001	0	Pão de Queijo
0.022	0.0412	0.0135	0.001	0	Rastelo

Figura 3.5: Lista dos objetos que podem ser utilizados para compor uma cesta.

continuar a realizar o seu serviço, é preciso posicionar a cesta no lugar correto para a garra prender a cesta, como pode ser visto na Figura 3.6. Então, após a garra executar seu serviço esta manda uma mensagem para o elevador e, caso não tenha nada obstruindo

a porta do elevador este inicia o transporte da cesta até o local desejado e manda uma mensagem para o FIM. Daí em diante a execução do cenário continua da mesma forma com já explanado.

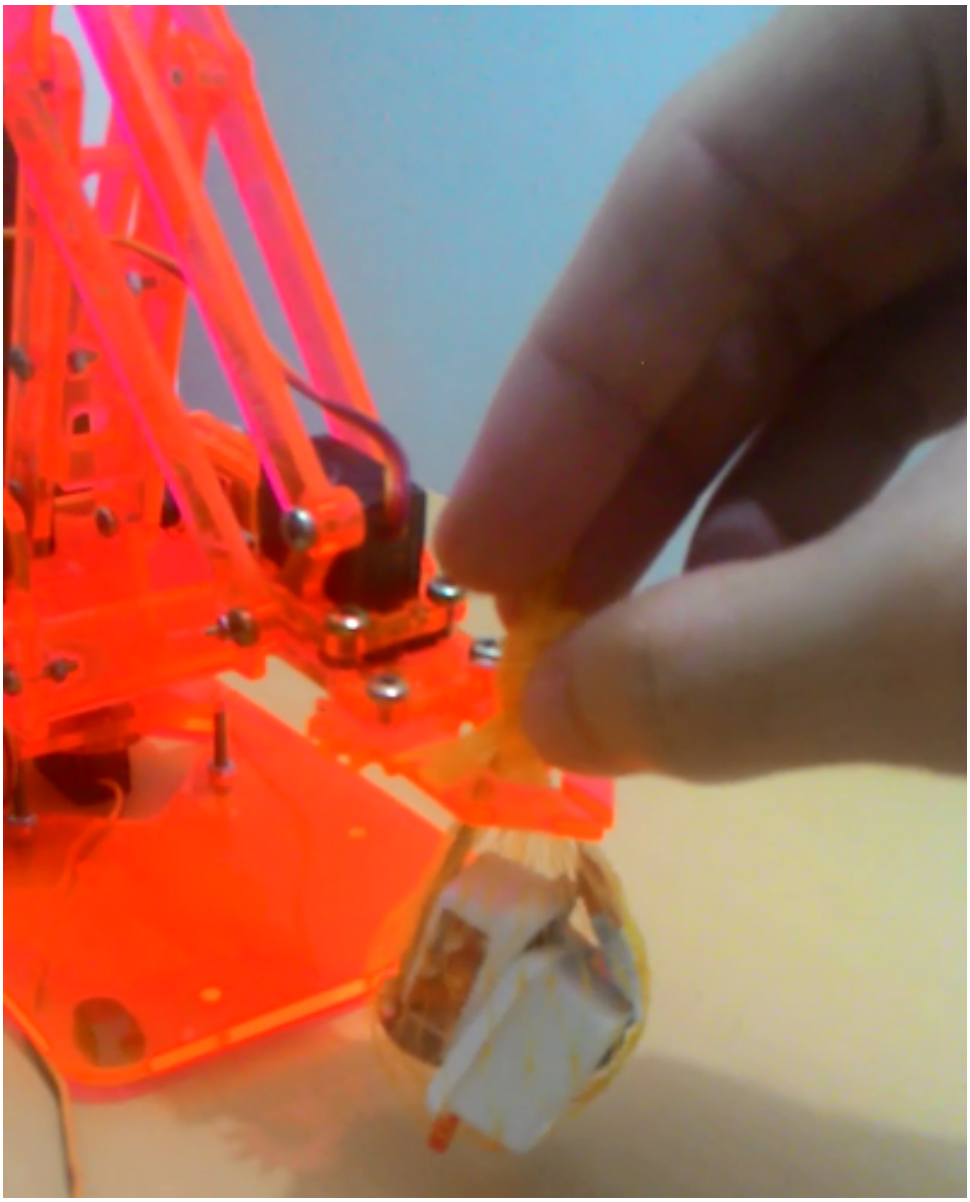

Figura 3.6: Cesta posicionada no local adequado por um humano, para que a garra possa prender a cesta adequadamente.

3.2 MÉTODO DE DETECÇÃO

3.2.1 Geração da massa de dados

A geração da massa de dados para o desenvolvimento do método proposto de detecção ocorreu de forma real. Os dados gerados foram obtidos através da execução do cenário "Levar compras". Para a execução deste cenário foram confeccionadas e cadastradas no banco de dados 59 cestas distintas. Antes de registrar as cestas no banco sempre era observado se a cesta cabia dentro do elevador quando colocadas de forma manual, e se também não ultrapassava os limites da porta, o limite de detecção do sensor de obstáculos. Então a cesta só era registrada caso coubesse no elevador e não estivesse em área que pudesse obstruir a porta. Diante das observações da execução do cenário, percebeu-se que as informações das cestas poderiam ser um indicativo ao problema (efeito colateral indesejável) provocado pelo comportamento da garra. Como todas as cestas estavam previamente cadastradas no banco de dados, a estas foram atribuídos valores de "é efeito colateral" ou "não é efeito colateral", à medida que se observava cada execução do cenário. Desta forma, para construção do modelo (seção 4.1) que classifique o problema em "é efeito colateral" ou "não é efeito colateral" utilizou-se as informações das cestas. Vale salientar que a informação da cesta é uma resposta do veículo terrestre referente a requisição "3" da Figura 3.2, ou seja, é uma das informações que os dispositivos e/ou serviços trocam entre si durante a excecução do cenário.

O cenário foi executado 59 vezes, uma para cada cesta, e pode-se observar que em algumas vezes o comportamento da garra provocava um efeito colateral indesejável (seção **??**) à funcionalidade "Levar compras", pois para todos esses casos se esperava sempre como resultado o elevador levar as compras sem problemas, já que era sabido que as cestas cabiam no elevador e, nenhuma delas feria as restrições de massa do carro nem do elevador.

A garra durante a execução da sua funcionalidade provocava a locomoção dos objetos. Esta locomoção ocorria em diferentes momentos da execução da funcionalidade: no momento em que a garra pegava o objeto, quando fazia os movimentos (direita, esquerda, cima, baixo, frente trás) ou no momento em que os objetos eram postos no compartimento do elevador. No momento em específico de por os objetos dentro do elevador, além dos objetos as vezes gerarem outras dimensões de cesta, esta também poderia se locomover, devido a natureza dos objetos e da forma como os objetos eram soltos pela garra. Por esse motivo utilizou-se a cesta como informação para construção do modelo.

3.2.2 Seleção dos atributos

Os dados coletados na etapa de execução do cenário "Levar compras", 59 cestas, ainda estão muito brutos. Uma cesta pode conter um objeto, como também pode conter muitos. Supomos que a cesta tenha 10 objetos, então o total de características da cesta seria "$10 \times 5 = 50$", pois cada objeto possui cinco características. Utilizar as características dos objetos desta forma terá um custo computacional muito elevado. Para diminuir a quantidade de atributos e melhor qualificá-los gerou-se características a partir de todos os atributos dos objetos da cesta: média das alturas, dos comprimentos, das larguras,

dos diâmetros e das massas, assim como o total de cada uma dessas medidas, e o total de itens. De ínicio então têm-se a seleção de 11 parâmetros.

Com a seleção prévia desses atributos, constriu-se um arquivo "arff", que é o tipo de arquivo padrão utilizado como *dataset* pelo Weka (Hall et al., 2009). Este *dataset* é composto de 59 instâncias (referente as 59 cestas), cada uma com 11 atributos mais um que identifica a qual classe pertence ("é efeito colateral" ou "não é efeito colateral"). Então têm-se 11 atributos númericos e um nominal (a classe a qual a instância pertence).

O próximo passo utilizado nesse processo de seleção de atributos foi o de visualizar estes novos parâmetros par a par, e verificar quais pares parecem melhor separar as classes "é efeito colateral" ou "não é efeito colateral". A Figura 3.7 passa a ideia de tal visualização.

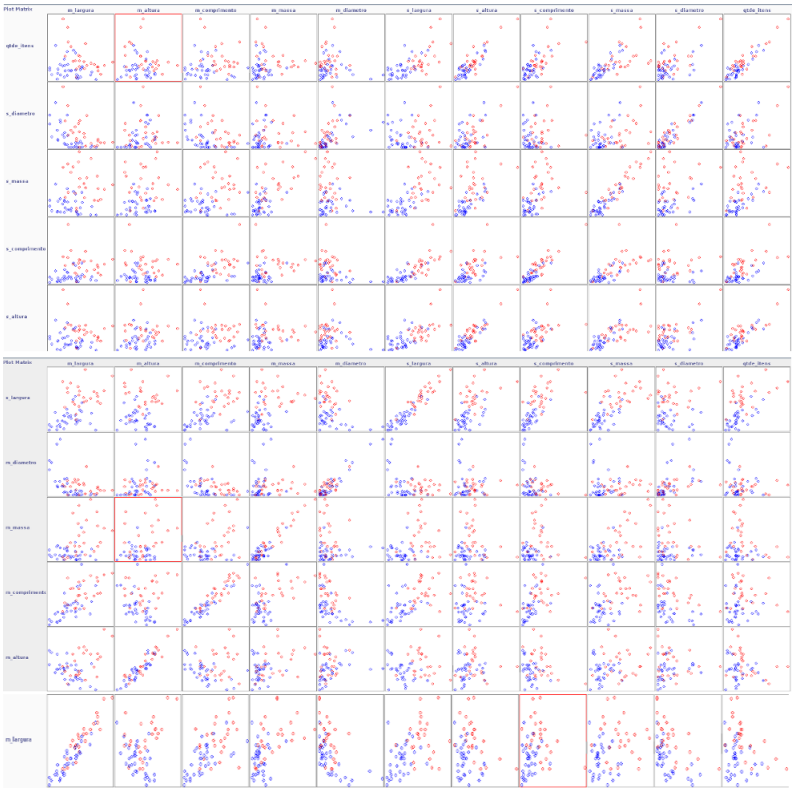

Figura 3.7: Plotagem do espaço de *features* par a par do dataset de 59 instâncias.

Dentre essas plotagens foram selecionadas três da Figura 3.8, as quais aparentam melhor separar as classes, são respectivamente os pares ordenados (somaLargura, somaMassa) - "sl x sm", (somaDiâmetro, somaLargura) - "sd x sl" e, (médiaAltura, somaLargura) - "ma x sl".

Então, nesta fase de seleção dos atributos, estes foram os pares selecionados para construção e validação do modelo (seção 4.1). Observe que o espaço de *features* foi reduzido a duas características.

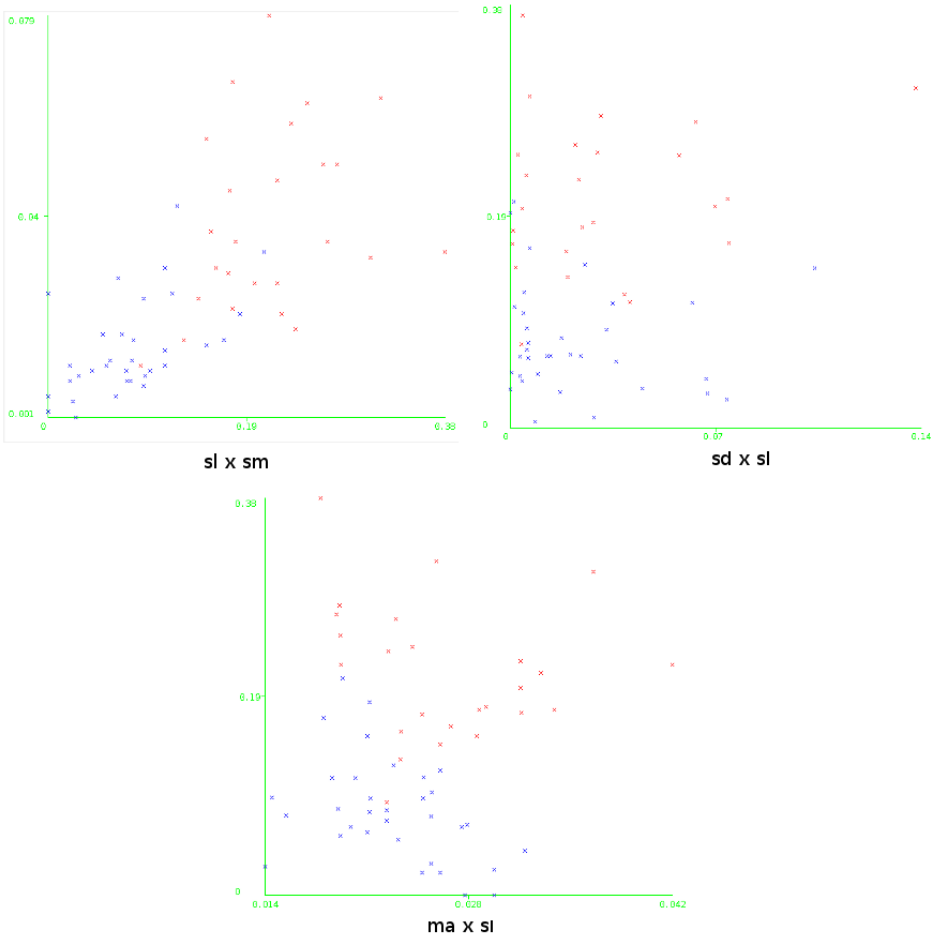

Figura 3.8: Separação das classes entre os atributos (somaLargura, somaMassa) - "sl x sm", (somaDiâmetro, somaLargura) - "sd x sl" e, (médiaAltura, somaLargura).

3.2.3 Meta-classificador DECORATE

Os parâmetros do DECORATE escolhidos neste trabalho foram semelhantes a um dos testes avaliativos realizados em (Melville e Mooney, 2004), nos quais comparam *DECORATE* com outros métodos classificatórios, em diferentes tamanhos de dataset e diferentes parâmetros do DECORATE. Para as avaliações neste trabalho o número de iterações é setado em 50, o número máximo de classificadores participantes no *ensemble* é setado em 14, o classificador base utilizado é o J48 e, a quantidade de exemplos artificiais gerados em cada iteração é setado para 100% (valores setados de 50% a 100% não faz variar muito o resultado (Melville e Mooney, 2004)) do tamanho do dataset de treino.

3.3 IMPLANTAÇÃO DO MÉTODO PROPOSTO

O classificador atua exatamente no momento em que o serviço veículo responde a requisição do FIM na etapa 3 da Figura (3.2). Nesta etapa o veículo primeiramente captura

o “id_cesta” (banco de dados) passado a ele pelo FIM. Então o veículo utiliza uma funcionalidade do serviço FIM que captura a cesta correspondente ao “id_cesta”, transforma-a em um arquivo “arff” contendo somente uma instância com os parâmetros correspondentes a “somaLargura, mediaAltura, classe”, sendo a classe com valor não rotulado, ou seja, não sabe-se a qual classe pertence (“é efeito colateral” ou “não é efeito colateral”). O FIM então salva este arquivo e retorna o caminho do arquivo como resposta para o dispositivo veículo. O veículo então responde a requisição 3, sendo um dos dados o caminho do arquivo “arff” referente a cesta. Neste momento o FIM utiliza o classificador (uma funcionalidade interna) para predizer se “é efeito colateral” ou “não é efeito colateral” passando como parâmetro ao classificador o caminho do arquivo “arff”. Caso o classificador classifique como “é efeito colateral” o FIM interrompe o serviço e manda uma mensagem para o usuário informando que ocorreu “efeito colateral indesejável. Caso contrário o fluxo do cenário (Figura 3.2) continua e no final da execução o usuário recebe uma mensagem de sucesso.

Capítulo

4

Procedimentos realizados para validar a proposta de pesquisa deste TCC.

VALIDAÇÃO

4.1 EXPERIMENTOS

Diante dos pares de atributos selecionados na etapa (seção 3.2.2), utilizou-se o método *stratified-10-fold-cross-validation* (repetido dez vezes) juntamente com o *ensemble DECORATE* a fim de se construir e verificar se é possível generalizar pelo menos um modelo.

Em cada verificação construi-se um modelo com diferentes frações do dataset, variando de 16 instâncias a 59 instâncias. Os resultados das avaliações podem ser vistos na Figura 4.1 e será discutido logo a seguir na seção 4.2.1.

4.2 RESULTADOS E DISCUSSÃO

4.2.1 Modelos

A Figura 4.1 demonstra os resultados obtidos através dos experimentos realizados (seção 4.1). Nesta, percebe-se um domínio quase absoluto da curva pontuada de aprendizado verde (médiaAltura, somaLargura - "ma x sl"). Por esso motivo essa será a curva analisada para escolher um modelo o mais adequado possível para implantar no FIM.

Como o principal motivo da implantação do classificador no FIM é detectar efeitos colaterais indesejáveis, então pretende-se que o modelo tenha um alto índice de *recall*, pois não se deseja que se tenha muitos falso negativos, ou seja, muitos casos classificados como não sendo efeito colateral, mas que na verdade era. Visualizando a curva percebe-se que esta têm um crescimento acentuado e depois torna-se estável a partir do dataset com 30 instâncias. Logo será realizada uma análise a partir deste ponto. Os maiores valores de *recall* partindo deste ponto são aqueles quando o *dataset* tem 40 ou mais instâncias, para todos estes pontos considera que se obteve um bom grau de generalização. Estes valores podem ser visualizados na Tabela 4.1.

Finalizando a análise decidiu-se que o melhor ponto para poder se construir um classificador é o com $n^o i = 45$, pois além de se ter um alto grau de *recall* e *accuracy* têm-se também *precision* quase de 100%. Então, para se construir um classificador final com

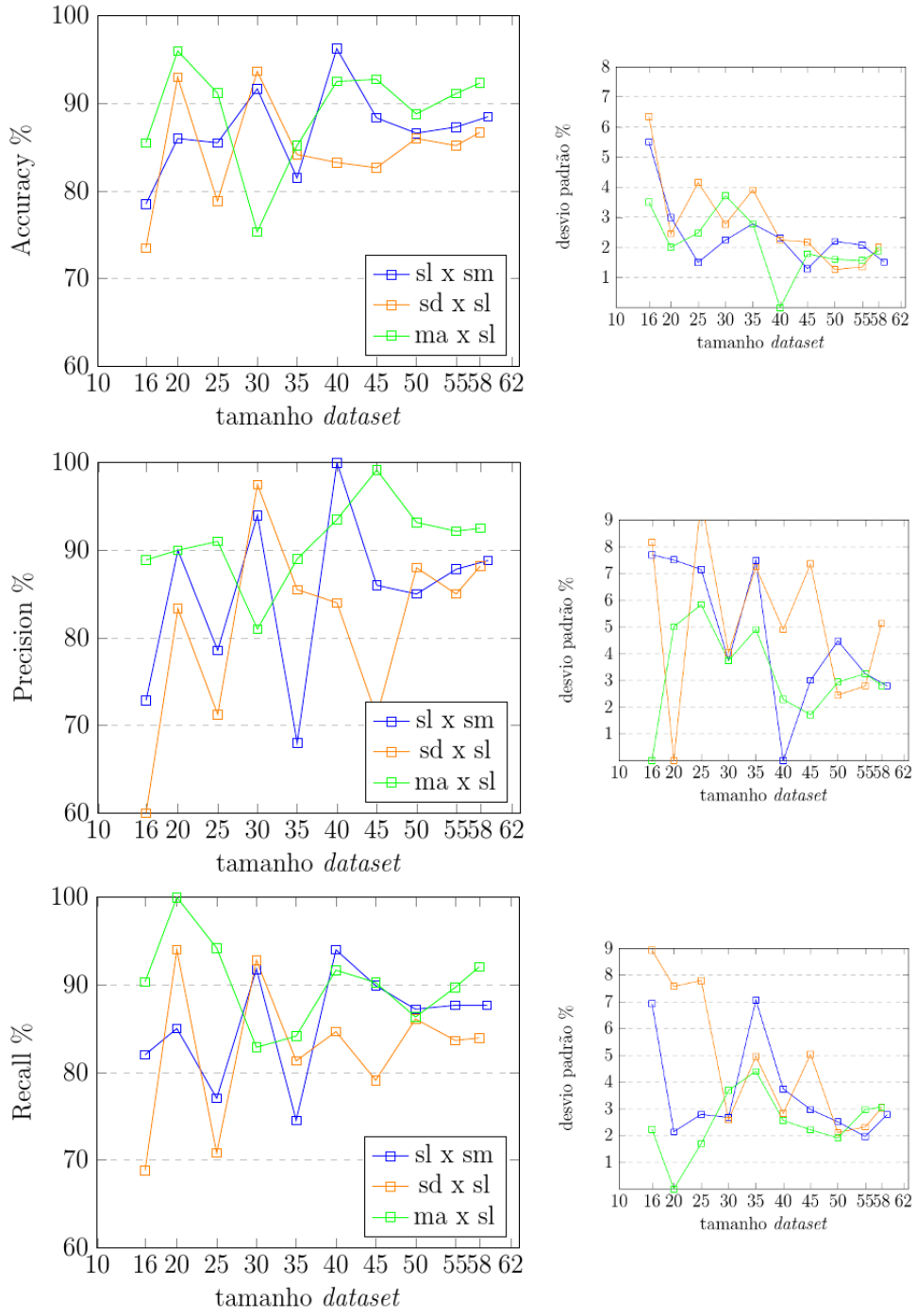

Figura 4.1: *Accuracy*, Precision e *Recall* obtidas utilizando stratified-10-fold-crossvalidation (repetido 10 vezes) e DECORATE sobre diferentes frações do *dataset* com os pares de parâmetros selecionados na seção 3.2.2.

base neste modelo escolheu-se randomicamente 45 instâncias do *dataset* de 59 instâncias e aplicou-se *stratified-10-fold-cross-validation* (repetido dez vezes). Este procedimento foi repetido diversas vezes (Figura 4.2) até que se obtivesse um modelo que satisfizesse a

$n^o i$	**40**	**45**	**50**	**55**	**58**
A, DP	92.5, 0	92.75, 1.78	88.8, 1.6	91.13, 1.56	92.33, 1.89
P, DP	93.5, 2.29	99.17, 1.71	93.17, 2.93	92.17, 3.25	92.5, 2.81
R, DP	91.69, 2.56	90.25, 2.21	86.33, 1.91	89.71, 2.97	92.05, 3.06

Tabela 4.1: Valores das curvas "ma x sl" da Figura 4.1 a partir do *dataset* com 40 instâncias. A (*Accuracy*), P (*Precision*, R (*Recall*), DP (Desvio Padrão), i (instâncias). Os valores A, P, R e DV estão em %.

expressão 4.1.

$$(accuracy \geq (92.75 - 1.78)) \wedge (precision \geq (99.17 - 1.71)) \wedge (recall \geq 90.25) \qquad (4.1)$$

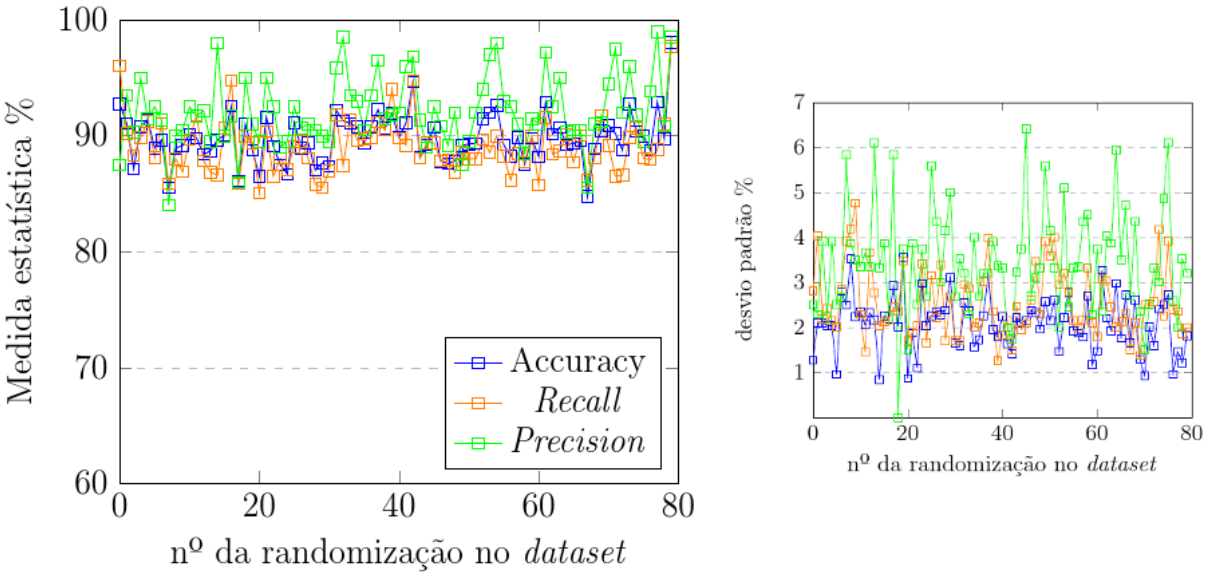

Figura 4.2: *stratified-10-fold-cross-validation* (repetido dez vezes). 45 instâncias foram escolhidas randomicamente em cima do *dataset* de 59 instâncias. Procedimento repetido até que satisfizesse a expressão 4.1

O modelo que satisfez essa expressão obteve resultados que podem ser vistos na Tabela 4.2 e este será o modelo final de classificador que será implantado no FIM.

		DP %
Accuracy %	98.05	1.81
Precision %	98.5	3.2
***Recall* %**	97.67	2

Tabela 4.2: Valores da validação do modelo final. DP (Desvio Padrão).

4.2.2 Implantação

O modelo final construído foi implantado no sistema de acordo com a proposta da seção 3.3. Na Figura 4.3 se tem a visão real do fluxo de execução em relação ao usuário do HNS quanto a funcionalidade "Levar compras". A imagem "a)" representa a execução quando não há efeitos colaterais indesejáveis. Enquanto que a imagem "b)" representa a execução quando se tem efeito colateral indesejável.

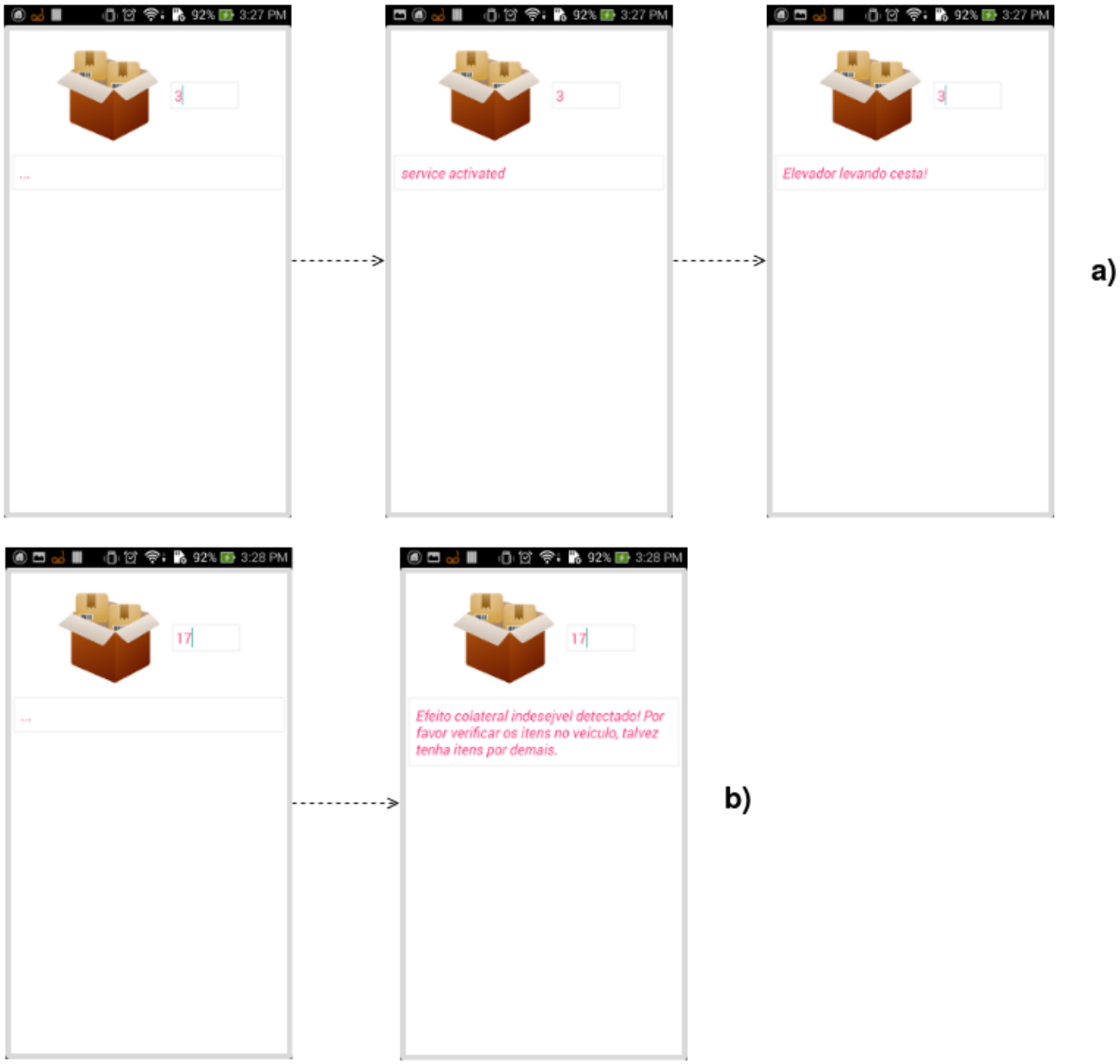

Figura 4.3: Modelo classificatório atuando no cenário "Levar compras" do FIM. a) Fluxo da aplicação sem efeito colateral indesejável. b) Fluxo da aplicação com efeito colateral indesejável.

Capítulo 5

Descreve de forma resumida os trabalhos relacionados ao desta monografia, assim como em que difere destes.

TRABALHOS RELACIONADOS

(Maternaghan e Turner, 2013) propõem uma abordagem *offline* (na fase de construção do sistema) para detecção de efeitos colaterais indesejáveis causados por conflitos de políticas. Nesta abordagem é verificado as ações de cada política de forma isolada e os pares de políticas que causam conflitos.

Em (Wilson et al., 2008) é apresentado um modelo de gerenciamento de Unidade de Funcionalidade de Software (UFS) ou *feature* em um *Home Automation System* para detecção de interação de características. Caso um efeito colateral indesejado ocorra, este deve ser evitado, mas se ocorre um efeito colateral desejado, este deve ser permitido. Este modelo consiste em três camadas de cima para baixo, como segue.

- Camada superior (serviços que automatizam a casa) - utiliza um ou mais dispositivos, os quais estão localizados na camada intermediária (dispositivos);
- Camada de dispositivos – contem dois tipos de dispositivos (entrada (ex.: termômetro, somente monitora um aspecto do ambiente e retorna para os serviços) e saída (ex.: aquecedor, o qual altera o ambiente))
- Camada de ambiente – contêm variáveis de ambiente, por exemplo: movimento do quarto, temperatura do quarto, luminosidade do quarto, umidade do quarto, fumaça do quarto.

O gerenciamento de UFS trabalha controlando o acesso à camada de ambiente utilizando de variáveis de bloqueio de acesso, especificando prioridades de serviços e além disso se utiliza de um banco de dados na nuvem para manter detalhes a respeito de cada dispositivo, com a finalidade de entender o efeito das ações de cada dispositivo no ambiente.

Os autores em (Nakamura et al., 2009) apresentam um método de detecção e resolução online para efeitos colaterais indesejáveis entre serviços integrados no *Home Network System*. Os autores definem dois tipos de efeitos colaterais indesejáveis, *appliance interaction* e *environment interaction*. *Appliance interaction* refere-se a uma situação onde dois

serviços compartilham um mesmo dispositivo de forma conflitante, enquanto que *environment interaction* ocorre quando dois serviços, os quais não necessariamente compartilham os mesmos dispositivos, mas conflitam-se por propriedades de ambientes, como, por exemplo, luminosidade. O método proposto utiliza mecanismos de suspender/resumir, prioridade e tempo de ativação dos serviços para detectar e resolver as interações. Em outro trabalho mais adiante (Nakamura et al., 2013), no qual Nakamura e Ikegami participaram, estes com Matsumoto propuseram uma extensão do trabalho anterior. Nesta, os autores definem um modelo (*environment impact model*) o qual define como cada dispositivo contribui para as variáveis de ambiente. Também é introduzido um *environment requirement* para definir o estado esperado de cada serviço. Então o conceito anterior de environment interaction é reformalizado introduzindo uma condição que uma certa quantidade de impactos acumulados viola os requerimentos dos serviços em questão. O autor realizou cinco casos de estudos os quais conseguiu detectar interações que não tinha conseguido com a definição anterior.

(Alfakeeh e Al-Bayatti, 2016) utilizam um método de detecção e resolução online o qual é um mecanismo de negociação entre os serviços ou as preferências do usuário da casa a fim de que seja possível estes serviços trabalharem em conjunto ao mesmo tempo. Este mecanismo de negociação é um *Agent-Based Negotiation System* (ABNS) de detecção e resolução de efeitos colaterais indesejáveis na casa inteligente.

A proposta apresentada neste TCC apresenta um método de detecção online, assim como em (Wilson et al., 2008; Nakamura et al., 2009; Nakamura et al., 2013; Alfakeeh e Al-Bayatti, 2016). Nos trabalhos citados, na arquitetura de HNS utilizada, os dispositivos são utilizados para prover serviços ao usuário da casa, e estes são acessados somente pelo HS através das APIs dos dispositivos. Os dispositivos nesta arquitetura não têm capacidade de se comunicar e oferecer serviços entre eles de forma independente. No trabalho desenvolvido neste TCC, os dispositivos são disponibilizados como serviços Web RESTFul e estes têm capacidade de se comunicar e oferecer serviços de forma independente, representando desta forma um cenário da IoT. Além disto, o método de detecção online proposto no Capítulo 3 utiliza aprendizagem de máquina para detecção dos efeitos colaterais indesejáveis, mais precisamente se utilizou do *ensemble* DECORATE para prover tal capacidade.

Capítulo

6

Síntese da investigação e dos experimentos realizados nesta monografia

CONCLUSÃO

O presente trabalho demonstrou através de um cenário ("Levar compras" - seção 3.1) no HNS que quando dispositivos são compostos para prover uma única funcionalidade, o comportamento de pelo menos um destes pode provocar interação de características com efeitos colaterais indesejáveis. Então, para o cenário em questão foi proposto detectar tais efeitos colaterais indesejáveis, de forma online, utilizando-se de um *Feature Interaction Manager* (FIM). Para que o FIM pudesse realizar a detecção dos efeitos colaterais indesejáveis foi implantado um modelo de meta-classificador (DECORATE) na funcionalidade "Levar compras".

O modelo implantado no FIM foi concebido através de um processo classificatório (Figura **??**), no qual as etapas podem ser representadas pelas seções (3.1, 3.2.2, 4.1). Na etapa de validação utilizou-se o método *stratified-10-fold-crossvalidation* repetido dez vezes diante diferentes porções do *dataset*, com o intuito de saber a partir de quantas instâncias o modelo iria generalizar. Ficou observado que o melhor resultado obtido foi quando o modelo construído utilizou-se de 45 instâncias do total de 59 instâncias do *dataset*. Então para construir o modelo final, o qual foi implantado no FIM, executou-se o *stratified-10-fold-crossvalidation* (repetido 10 vezes) por diversas vezes sobre 45 instâncias do *dataset* (escolhidas aleatoriamente com seeds diferentes) até que a expressão 4.1 fosse satisfeita, a qual foi composta baseada no melhor resultado obtido anteriormente (o com 45 instâncias). O resultado da expressão satisfeita gerou um modelo classificatório final com $Accuracy = 98.05\%$, $Precision = 98.5\%$ e $Recall = 97.67\%$, como pode ser visualizado na Tabela 4.2.

Desta forma o FIM ficou apto a realizar detecção inteligente de efeitos colaterais indesejáveis para o cenário proposto, mostrando que é possível detectar efeitos colaterais indesejáveis entre dispositivos na visão da IoT.

Como trabalhos futuros, pretende-se realizar um estudo sobre algoritmos de seleção de atributos afim de não mais selecionar um vetor de características (que descreve o problema) de forma manual. Observe que os atributos selecionados de forma manual e visual (seção 3.2.2) para o cenário proposto neste trabalho estavam em um espaço de

2 dimensões (Figura 3.7), algumas vezes um espaço de duas dimensões não é suficiente para descrever um problema, o mesmo acontece para um espaço 3D (o máximo que a visão humana consegue visualizar). Realizado este estudo, então pretende-se executar os experimentos realizados neste trabalho (seção 4.1) em cima dos vetores de características selecionados pelos algoritmos estudados e realizar comparações dos resultados obtidos na seção 4.2.1 com cada um dos algoritmos de seleção de atributos utilizados. Além disso pretende-se criar outro cenário, com mais dispositivos, que seja o mais real possível e com pouca intervenção humana ou nenhuma e, para este realizar todos experimentos do cenário anterior. Feito isto, realizar comparações dos resultados obtidos com o do cenário anterior.

REFERÊNCIAS BIBLIOGRÁFICAS

ALFAKEEH, A. S.; AL-BAYATTI, A. H. Feature interactions detection and resolution in smart homes systems. *International Journal of Electronics and Electrical Engineering*, v. 4, n. 1, 2016.

ASHTON, K. That 'internet of things' thing. *RFID Journal*, 2009.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, p. 2787– 2805, October 2010.

CHANDRAKANTH, S. et al. Internet of things. *International Journal of Innovations and Advancement in Computer Science IJIACS*, v. 3, October 2014.

FRANCA, T. et al. Web das coisas: Conectando dispositivos físicos ao mundo digital. In: . Porto Alegre, RS: SBC: [s.n.], 2011. v. 1, p. 103–146.

GUINARD, D.; TRIFA, V. Towards the web of things: Web mashups for embedded devices. In: *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, International World Wide Web Conferences*. Madrid, Spain: [s.n.], 2009.

HALL, M. et al. The weka data mining software: An update. *SIGKDD Explorations*, v. 11, 2009.

HEFFELFINGER, D. *Java EE 7 with GlassFish 4 Application Server*. Third. [S.l.]: Packt Publishing Ltd, 2014.

MATERNAGHAN, C.; TURNER, K. J. Policy conflicts in home automation. *Computer Networks*, v. 57, 2013.

MELVILLE, P.; MOONEY, R. J. Creating diversity in ensembles using artificial data. *Information Fusion: Special Issue on Diversity in Multiclassifier Systems*, 2004. Submitted.

NAKAMURA, M. et al. Considering online feature interaction detection and resolution for integrated services in home network system. *Feature Interactions in Software and Communication Systems X*, p. 191–206, 2009.

NAKAMURA, M.; IKEGAMI, K.; MATSUMOTO, S. Considering impacts and requirements for better understanding of environment interactions in home network services. *Computer Networks*, Elsevier, v. 57, p. 2442–2453, 2013.

PAPAZOGLOU, M. *Web Services: Principles and Technology*. [S.l.]: Pearson, 2008. ISBN 978-0-321-15555-9.

PAUTASSO, C. Restful web services: Principles, patterns, emerging technologies. In: *Springer Science+Business Media*. New York: [s.n.], 2014. Disponível em: ⟨http://vis.uky.edu/~cheung/courses/ee586/papers/Pautasso2014.pdf⟩.

SUNDMAEKER, H. et al. (Ed.). *Vision and Challenges for Realising the Internet of Things*. [S.l.]: European Union, 2010. ISBN 978-92-79-15088-3.

WEBER, R. H.; WEBER, R. *Internet of Things - Legal Perspectives*. [S.l.]: Springer, 2010.

WILSON, M.; MAGILL, E.; KOLBERG, M. Considering side effects in service interactions in home automation - an online approach. In: _____. *Feature Interactions in Software and Communication Systems IX*. [S.l.]: IOS Press, 2008. p. 172–187. ISBN 978-1-58603-845-8.