Journal of Applied Computing Research, 5(1):32-43
January-June 2016
Unisinos - doi: 10.4013/jacr.2016.51.03

A food allergy risk detection model based on situation awareness[1]

Nelson Manoel de Moura Quevedo, Cristiano André da Costa, Rodrigo da Rosa Righi, Sandro José Rigo

Universidade do Vale do Rio dos Sinos

Av. Unisinos, 950, Cristo Rei, 93022-750, São Leopoldo, RS, Brasil

nquevedo@hotmail.com, cac@unisinos.br, rrrighi@unisinos.br, rigo@unisinos.br

Abstract. Advances in ubiquitous computing are enabling the emergence of opportunities in many areas; among them we highlight the health-related applications. In this area, there is a trend toward developing many applications that enable care wherever you are and whenever you need, called ubiquitous healthcare. A detailed survey of existing and proposed models has shown that none of these applications meets the needs of people who suffer from food allergy. Thus, this article proposes an allergy detection ubiquitous model based on situation awareness. This model, called Allergy Detector, focuses on food allergy, in particular the eight major allergens (peanut, milk, egg, wheat, soy, fish, crustacean and tree nuts) and their derivatives, which cause about 90% of all food allergies. The main scientific contribution of the Allergy Detector is the use of situation awareness for detecting allergies anytime and anywhere in a transparent manner. In order to assess the model, we designed some case studies. We also carried out a profiling, accessing the Allergy Detector in terms of performance and identifying the main bottlenecks. The results showed the feasibility of using the model for detecting allergy ubiquitously.

Keywords: ubiquitous computing, context awareness, ubiquituos health.

Introduction

Ubiquitous computing is enabling the emergence of opportunities in several areas such as health, education, games, shopping, entertainment and transportation (Feltes and Barbosa, 2014). Specifically, in the healthcare area called Ubiquitous Healthcare or "U-Healthcare", these opportunities are very promising today, allowing the provision of medical services anywhere and anytime. The goal of U-Healthcare is to provide convenient healthcare service to both caregivers and patients, and to make it easy to diagnose patient's health conditions (Gelogo and Kim, 2013).

There are models that provide ubiquitous healthcare, such as dietary planning (Antoniou and Nanou, 2003), recommendation of lower calorie foods (Johnson *et al.*, 2014), suggestion of restaurants (Daraghmi and Yuan, 2013), daily food intake (Henricksen and Viller, 2012) and selection of menu restrictions for a safe diet (Iizuka and Okawada, 2012). However, none of the considered models provides support ubiquitously for users who suffer from food allergies.

Food reactions from allergic causes affect 6-8% of children less than 3 years old and 2-3% of adults (according to the Brazilian Association of Allergy and Immunology (Associação Brasileira de Alergia e Imunologia [ASBAI, http://www.sbai.org.br/]).

Moreover, the only therapy proven to be effective against food allergies is the dietary exclusion of the allergen to the user (Silva *et al.*, 2008).

In this perspective, the main objective of this work is to create a model to detect risks for users who suffer from food allergies to

[1] This article is an extended version of the work presented by the authors at the 7th SBCUP -*Simpósio Brasileiro de Computação Ubíqua e Pervasiva,* an event of the 35th *Congresso da Sociedade Brasileira de Computação*, Recife (PE, Brazil), July 20-23, 2015.

the eight major allergens (peanut, milk, egg, wheat, soy, fish, crustacean and tree nuts) and their derivatives, which accounted for over 90% of the cases of food allergy (Pereira *et al.*, 2008). The proposed model, called Allergy Detector, considers what are the proteins contained in the eight major allergens according to the International Union of Immunological Societies (IUIS). The IUIS Allergen Database lists all allergens and isoforms that are recognized (King *et al.*, 1994). The model's major scientific contribution consists of employing situation awareness, based on inferences in an ontology, with the specific purpose of detecting risks to users with food allergies. Situation awareness is defined as the perception of the elements in the environment, considering its space, the comprehension of their meaning and the projection of their status in the near future (Endsley, 1999). The elements in the Allergy Detector are the restaurant's name, the restaurant's dishes and the dishes' ingredients, the restaurant location, information about the user's allergies (profile context) and information about the eight major allergens and their proteins. To detect allergies, the model uses inferences in a developed ontology, which has food allergy information and context information, including profile and location-aware data. In the proposed model, a future projection is the inference about the risk of ingesting the eight major allergens or their derivatives. The model's ubiquitous features give the user mobility and require a minimal interaction for risk detection. To evaluate the model, we proposed a mobile application that transparently identifies risks to the user and signals them whenever the user is entering a restaurant.

This article is organized in six sections. *Food allergy and situation awareness* discusses some fundamental concepts for the work. The proposed model is described in *Allergy Detector's Architecture*. *Evaluation and discussion* presents the evaluation and discussion of the results. The related works are presented in the fifth section (*Related works*). Finally, *Conclusion* presents conclusion and some directions for future work.

Food allergy and situation awareness

Adverse reactions to foods are extremely common and they are usually attributed to allergy, which can be classified as immune mediated or non-immune mediated, and is called food intolerance. Immune mediated reactions can be further broken down into immunoglobulin E (IgE) mediated and non-IgE mediated reactions (Guandalini and Newland, 2011).

The IgE-mediated reactions result from sensitization, which in turn results in the production of allergen-specific IgE-antibodies. The most common examples in this category are skin reactions, gastrointestinal, respiratory and systemic reactions (anaphylactic shock and hypotension) (Silva *et al.*, 2008).

The most comprehensive global allergen database is provided by the International Union of Immunological Societies (IUIS), which is available on the Union's site (http://www.iuisonline.org) and is officially recognized by the World Health Organization (WHO). The IUIS Allergen Nomenclature Sub-committee is responsible for maintaining and developing a single, unambiguous and systematic nomenclature for allergenic proteins. Furthermore, IUIS analyzes the submission of new allergenic proteins and, if they are approved, adds these proteins to the database. The rule for naming allergens is the concatenation of the first three letters of the genus name. This consists of a single letter for the species and a decimal number that identifies the chronologic order of protein allergenic source identification. For example, the cow's milk protein allergen, related to beta-lactoglobulin, was named by IUIS "Bos d 5", which took into consideration the cow's genus name "Bos domesticus", the species, in English "domestic cattle", and the number 5 (Arruda *et al.*, 2013).

From existing information in allergen databases, we can employ situation awareness to identify potential food intolerances. To detect the risk of allergies, the model uses situation awareness, based on Endsley's model, which employs three levels (Endsley, 1999). In this approach, level 1 is in charge of perception, collecting elements in the current situation. In level 2 this data is processed, generating a comprehension of the current situation. The third level consists in projecting the future status, which, in the proposed model, consists of determining the risk, in the restaurant in which the user is currently, for allergic ingredients and allergenic proteins contained in food. To deal with the level 3 of comprehension, as proposed by Endsley, we chose in this work to use inferences, so we need to develop an ontology (Feltes and Barbosa, 2014).

Allergy Detector's Architecture

The Allergy Detector was developed to detect risks to users suffering from food allergy

caused by proteins of the eight major allergens or their derivatives, using Endsley's model as a reference.

Figure 1 shows the service-based architecture of the proposed solution. On the client side we can see the modules that will run on mobile devices. They are responsible for interacting with the user and acquiring context information, such as the user's current location. The client will interact with a service that processes information, makes inferences and manages situation awareness.

The services that run on the client side are: (i) *Restaurant Registration*: Uniform Resource Locator (URL), in which the restaurant provides information and the dishes' ingredients; (ii) *Profile Registration*: allows the authentication and information about user's allergies; (iii) *Notify*: notifies the user through a traffic light whether the place is secure, uncertain or insecure referring to the user's risk of ingesting foods that cause food allergy; this module also presents the dishes with allergens and their proteins; and (iv) *Check-in*: collects location-aware information, such as the user current whereabouts and the restaurant's name, sending them to the *application server*.

The Allergy Detector Service has two main components: The *Service Ontology* and the *Application Server*. The *reasoner* module in the *Ontology Service* infers information regarding dishes and ingredients used in the restaurant, as obtained in the *check-in* module. This module infers whether the ingredients are part of the eight major allergens or of their derivatives, in addition to identifying the associated proteins. Once this is done, the *reasoner* defines whether the ingredients of dishes can cause allergy to users according to their profile and builds a table with dishes, ingredients allergenic to the user or derived from these and the protein's name (IUIS) related to each allergenic ingredient.

The *Application Server* comprises three main modules: (i) *User Control*: queries the user's allergies in the *profile database* on the basis of the user's credentials informed by the *Profile Registration* module; (ii) *Traffic Light*: infers whether the site is secure (green light), uncertain (yellow light) or insecure (red light) and sends information to the *Notify* module; (iii) *Parser*: gets data about food/restaurant ingredients referring to the URL informed by the *Restaurant Registration* module.

Figure 2 shows a diagram with the three levels that characterize the use of situation awareness, based on Endsley's model. The figure illustrates nine steps that represent the sequential execution of people's allergy risk detection process based on the situation awareness, namely:

(1) Location Context: The *Check-in* module collects the location data over the internet (IPs lo-

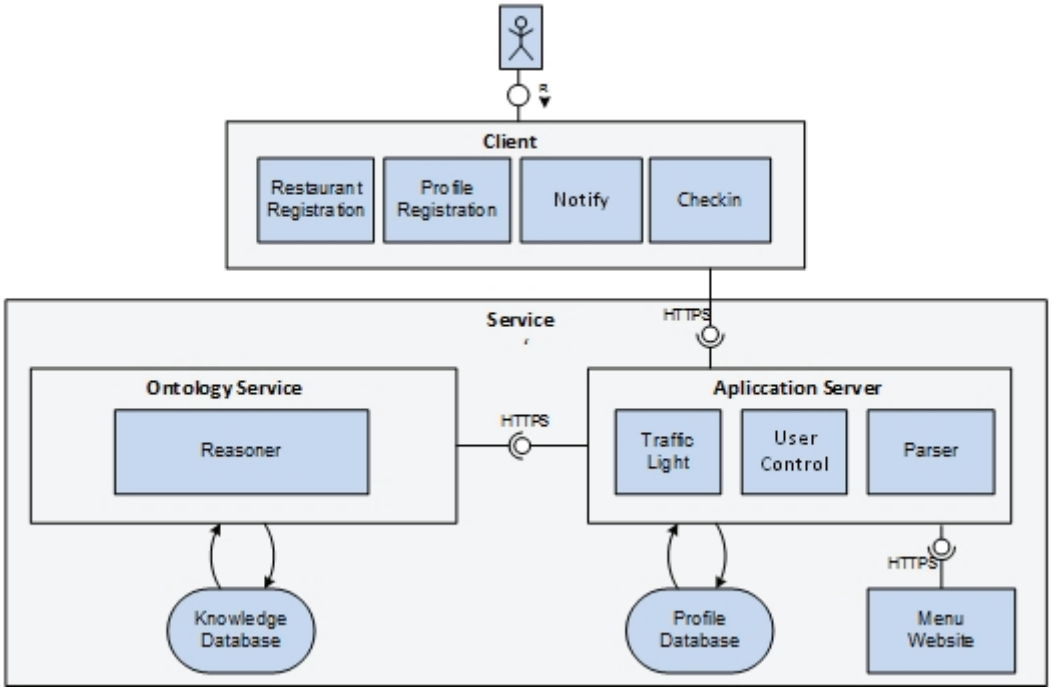

Figure 1. Allergy Detector's Service-based Architecture.

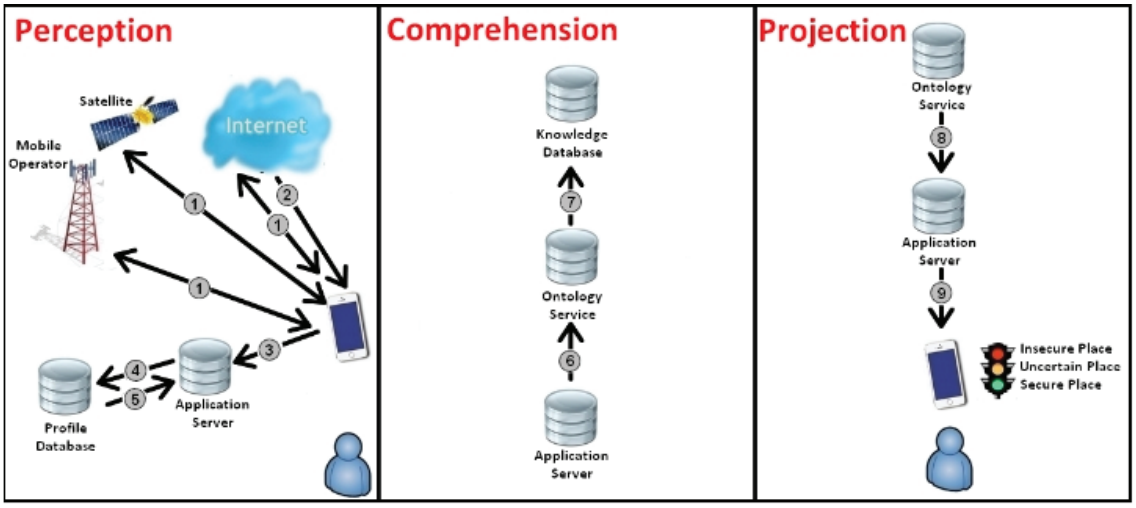

Figure 2. Situation Awareness in the Allergy Detector, based on Endsley's Model.

cation services), triangulation of cell phone towers (General Packet Radio Services [GPRS] from Group Special Mobile [GSM]) or using Global Position System (GPS) (global network with 24 GPS satellites at an altitude of about 20,200 km in the Ultra High Frequency – UHF band);

(2) Location Context – Restaurant identification: The *Check-in* module identifies the restaurant's name by looking at the Google Maps API, using the detected location;

(3) Location Context/Profile - Sending information to the Application Server: the *Check-in* module sends the user authentication data and the restaurant's name to the *application server*;

(4) Profile Context – Allergy Query: based on the validation of the user's credentials, the module makes a query to retrieve the user's allergies from the *profile database*;

(5) Profile Context – Allergies Retrieved: the database returns information about the user's allergies to the *application server*;

(6) Profile/Location Contexts and Knowledge Base – Sending information to the Ontology Service: the module informs the *ontology service* about the restaurant's name and the user's list of allergies;

(7) Context Correlation – Inference of dishes with allergens to user: the *ontology service* queries the *Knowledge Base* looking for food served by the restaurant which contains ingredients that are allergenic to the user, inferring their derivatives and proteins associated with the ingredients and the restaurant's dishes;

(8) Inference – Sending information to the Application Server: the *ontology service* sends the inferred information to the *application server*;

(9) Inference – Place Risk Classification: when the *application server* detects the information about dishes, ingredients and URL, it determines the situation and alerts the user if that restaurant is an "Insecure Place" and shows the user which dishes contain allergenic ingredients and their proteins. If Step 8 returns no value, it determines the situation and alerts the user that the restaurant is an "Uncertain Place", but if it returns an URL only, it alerts the user that restaurant is a "Secure Place".

To allow inferences, we created an ontology on the food allergy domain. Figure 3 illustrates the main classes of the proposed ontology, which contains the classes Restaurants, Restaurant_Dishes, Ingredients, Proteins and subclasses for the Eight_Main_Allergens and Eight_Main_Allergens_Derivates. The inference process is used to decompose basic ingredients of a dish into its derivatives' components. In this way, it allows the detection of allergies beyond those of obvious ingredients already listed in the basic description. Furthermore, sometimes there is also a direct relation between one ingredient and another that the user is allergic to. Finally, there is also a correlation with proteins that could also be harmful to some people.

Evaluation and discussion

A prototype of the model was designed using open and free technologies. On the client side we developed a hybrid application built with Ionic framework (http://ionicframework.com/) and using JavaScript as programming lan-

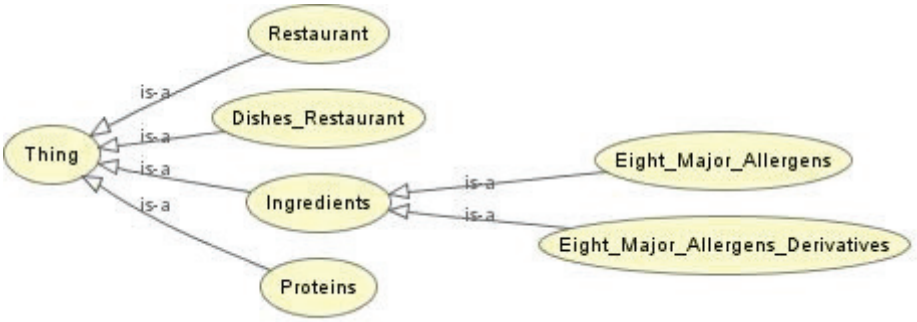

Figure 3. Allergy Detector's Ontology.

guage, HTML as markup language and CSS as style language for smartphones compatible with Android and iOS. On the service side we developed the Application Server, using JavaScript as programming language, and employing NodeJS (https://nodejs.org/) as the execution platform. The Profile Database was stored in the NoSQL database mongoDB. The Ontology Service used the Jena framework (https://jena.apache.org), which uses Java as programming language, allowing the use of SPARQL query language. The Knowledge Base was generated in OWL (http://w3org/TR/owl-guide). The Application Server was stored in OpenShift Platform (https://www.openshift.com) using an instance of the type "small machine", with 512 MB of RAM and 1GB of storage. The Ontology Service and the Knowledge Base were stored in the Amazon Web Service (AWS) Platform (https://aws.amazon.com/pt/) on a t2.micro instance with 1GB of RAM and 30 GB of EBS storage. Finally, mongoDB (https://www.mongodb.org) was hosted in the MongoLab (https://mongolab.com/).

In this section we present the Allergy Detector model's evaluation. The proposed model was evaluated in the following ways: the first evaluation was conducted through a case study and the second one measured the application's performance, using a profiling. In the following, we describe both evaluations.

Case study evaluation

The scientific community has used scenarios to evaluate mobile computing, ubiquitous computing and context awareness projects (Satyanarayanan, 2001; Zaupa *et al*., 2012). To evaluate the model, we developed a prototype that performs situation awareness as proposed by the Allergy Detector model. The ontology was modeled in Protègè (http://webprotege.stanford.edu) and exported to the Ontology Database in OWL format. We used Java language and the Jena library for interaction with the ontology. The following scenario was used for model evaluation:

David Dunn is allergic to peanuts, so he should avoid contact with that allergen source. He has the Allergy Detector application installed on his smartphone, which provides support to check the risk of the place where he wishes to have meals. On a certain day, David Dunn is in Porto Alegre's downtown area when he decides to have a snack at Subway – at the Andradas Street near the Alfandega Square. Arriving there, David starts the Check-in option of the Allergy Detector application to check whether there are snacks containing peanuts as an ingredient. The Allergy Detector shows a red bar on the display of his smartphone, informing him that this is an "Insecure Place". Furthermore, the application also shows the snack name 'Chicken breast with Teriyaki sauce', which has peanut as an ingredient, and presents the peanut proteins contained in it, Ara_h_1, Ara_h_10, Ara_h_11, Ara_h_12 and Ara_h_13 according to Figure 4a. Then David Dunn decides to go to the nearby McDonald's. When arriving there, he activates again the Check-in option, but this time the Allergy Detector presents a green bar in the display. Moreover, it shows the message "Secure Place" on the smartphone's display, according to Figure 4b.

The Allergy Detector was designed to provide support to users like David Dunn who suffer from food allergy to one of the eight most common major allergens. The application requires minimal user intervention; it is enough that users keep their allergy information updated. Whenever users execute check-in at a restaurant where they are located, the system starts the detection of possible allergy sources. First, if this is the first time that this

restaurant is visited by the user, the Allergy Detector searches for the website where the dishes and ingredients of this restaurant are informed. The application then alerts users about the site's risk rating and presents the dishes containing allergenic ingredients or derivatives of these ingredients.

After the user logs in, the Allergy Detector waits for the user to execute the Check-in function. When making the check-in at the restaurant where David Dunn is, the system starts the Geolocation API from Google Maps, obtaining location information. With this information the system detects the restaurant's name through Google Maps API. At this point the Application Server receives the authentication information and the restaurant's name. Then the Application Server executes a query in the Knowledge Base using the restaurant's name to find out the dishes and ingredients that the restaurant serves, and makes another query in the Profile Database using the user's credentials to discover the ingredients that cause allergy to the user. This stage consists in collecting relevant context data.

Figure 4. Screenshots of the Allergy Detector prototype for mobile devices: (a) screen showing an insecure place (b) screen showing a secure place.

At the next stage, the retrieved data are processed to create new information. From the information about user's allergies, the dishes and ingredients served in the restaurant, the system infers whether the eight major allergenic ingredients or of their derivatives are present, i.e. those that cause allergy to the user, and which proteins are associated with these ingredients. To infer this, the Ontology Service executes a SPARQL (http://www.w3.org/TR/rdf-sparql-query/) query using the name of the dishes, as well the related ingredients and proteins contained in the dishes. This query is presented in Figure 5.

The SPARQL query uses a "SELECT" clause informing about all important variables, including "?nameDish", "?nameIngredient", "?nameAllergenicSource" and "?nameProtain", a "WHERE" clause with object properties "ContainedInDish", "ServesDish", "DerivedFrom" and "ContainedProtein" and a filter only to retrieve the sources of the allergenic information. As a result of this query, the server returns all dishes served in the restaurant, selects those that have allergens to the user and proteins matching the ingredients in the IUIS/WHO database.

At the last stage, the *Application Server* makes a query in the *Ontology Service* to retrieve the information inferred in the previous step and through the application of rules, using a code executed in the *Application Server*, classifies the place as "Secure Place", "Uncertain Place" or "Insecure Place". Finally, the application server presents a bar showing the

```
SELECT ?nameDish ?nameIngredient ?nameAllergenicSource ?nameProtein
WHERE { ?ingredient ad:ContainedInDish ?dish.
        ?restaurant ad:ServesDish ?dish.
        ?restaurant ad:Name '"+nameRestaurant+"'.
        ?ingredient ad:DerivedFrom ?allergenicSource.
        ?allergenicSource ad:Name '"+listsIngredients+"'.
        ?allergenicSource ad:Name ?nameAllergenicSource.
        ?protein ad:ContainedProtain ?allergenicSource.
        ?dish ad:Name ?nameDish.
        ?ingredient ad:Name ?nameIngredient.
        ?eightMajor ad:Name ?nameAllergenicSource.
        ?protein ad:Name ?nameProtein.
FILTER ( ?nameAllergenicSource IN ("+listsIngredients+")).}
```

Figure 5. Query of potential allergens contained in dishes at a specific restaurant.

risk of the current restaurant. In the case of insecure places, the application also shows to the user the dishes that contain allergenic ingredients with the proteins associated with these ingredients, as previously shown in Figure 4.

The prototype confirmed the expectation that the application of situation awareness, based on Endsley's model, enables the ubiquitous risk detection, indicating to the user the presence of allergens in food served in restaurants.

Performance evaluation

According to Zahra *et al.* (2013), performance in mobile applications is more crucial than any other desktop or embedded application, and users expect that the processing on mobile devices will be faster and more efficient. The authors also mention that a single early failure in the first contact with the application accounts for non-returning users. Kim *et al.* (2009) also add that performance refers to the degree to which a system or component completes its designated functions within given constraints, such as speed, accuracy or memory usage. They also mention that the performance can be evaluated based on response time and throughput. The authors add that throughput is the measure of the number of events/requests processed in a given amount of time and response time is the time taken by the mobile application to respond back to the user. They add too that it is more meaningful to evaluate the performance of mobile applications using response time.

The first step made for the performance assessment process was to do an analysis of the application's features, in order to identify which one has more influence on the processing load and the application's response time. We concluded that the check-in function would be the one most frequently used, especially at lunch and dinner times, which are considered peak times.

The check-in function directly involves client communication with the application server (in NodeJS) and with the Ontology Service (implemented in Java). Thus, it was decided to create for-loop ties, running "n" times, where "n" is the amount of requests made simultaneously. We chose to run the application tests in Google Chrome, because Ionic makes this possible through the "ionic serve" command. After running this command, the application is opened in the browser. To enable recording the response times, we used the "Inspect element" from Google Chrome and the "Network" option, as shown in Figure 6.

The used approach allowed us to save on an HTTP file (HAR format) the response times for each set of requests generated in the

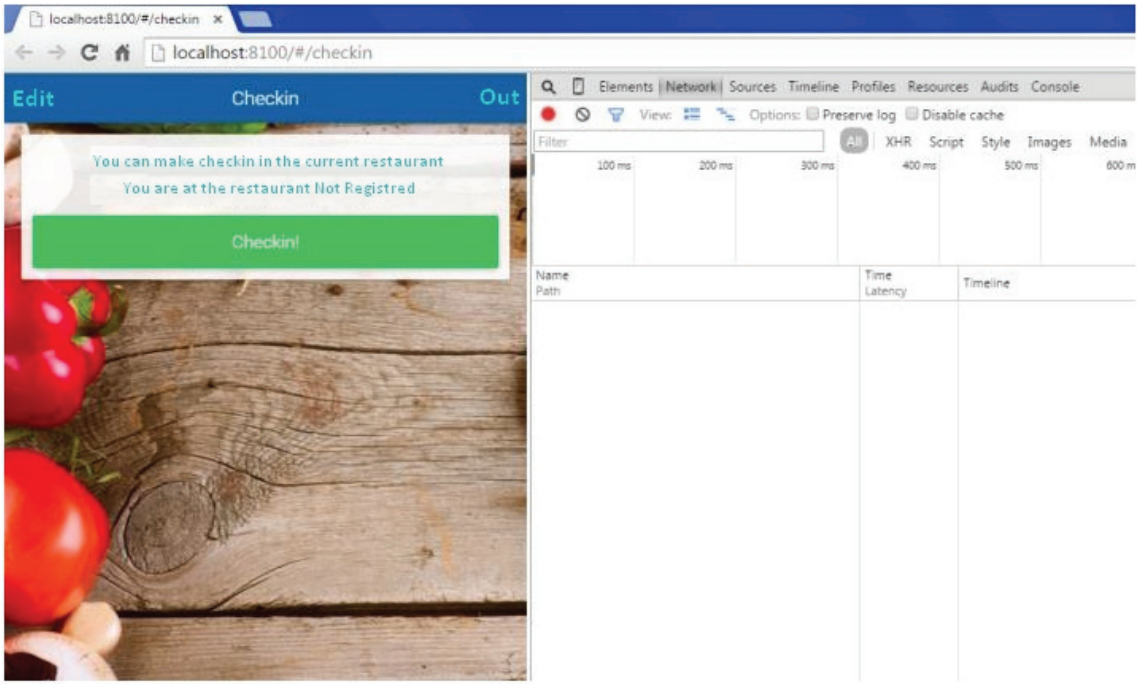

Figure 6. Allergy Detector's profiling using Google Chrome's inspect element option.

for-loop ties. The HAR files are based on the HTTP Archive specification, which allows the capture of load information from a web page in a JSON format.

First we executed the check-in function once, recording the response time of one request. Subsequently, we executed 20 ties, generating 20 sets of requests. These sets started with 5 simultaneous requests and were increased by 5, up to a total of 100 simultaneous requests. With these collected values, we calculated the average response time and the chart was plotted in a histogram format, as shown in Figure 7. Analyzing the graph, we can see that the response time increases at a linear rate until 100 requests; from this mark onwards, we find that the time grows at an exponential rate.

We attribute this exponential rate to the fact that the simulation of simultaneous requests is running in only one computer. Furthermore, the data links used in communication between PaaS platforms and the implemented prototype has not considered the performance requirement. A possibility of identifying whether the results are related to the computer's performance that generates the requests would be to generate a single request from multiple computers simultaneously, using the same data link and measuring the response time. The next test would be to repeat the previous step, but using a dedicated data link for each computer. A possibility of identifying whether the performance problem is related to PaaS platform scalability would be to do an upgrade in the settings of PaaS platforms. And finally we could optimize the code by decreasing the number of iterations.

As reported before, the check-in function involves HTTP requests directly from the Application Server (NodeJS) and the Ontology Service (Java). The NodeJS was hosted in the OpenShift, in a Virtual Machine (VM) with 512 MB of RAM memory, with the Red Hat Enterprise Linux Server operating system. And the Ontology Service was hosted in the Amazon AWS, in a t2.micro VM instance, with Intel Xeon turbo processor high frequency until 3,3GHz and with a RAM memory of 1 GB, with the Ubuntu 14.04.2 LTS operating system. As both servers have the Linux operating system, we used the TOP command to measure the percentage load of both components. This command is used for process management in Linux, displaying a list of all processes in execution with CPU usage, process ID (PID), process name, memory usage, etc. Figure 8 shows the return of the TOP command executed in the AWS instance, listing the data for the Ontology Service (PID 776) process, showing the load of 0.3% of the total CPU.

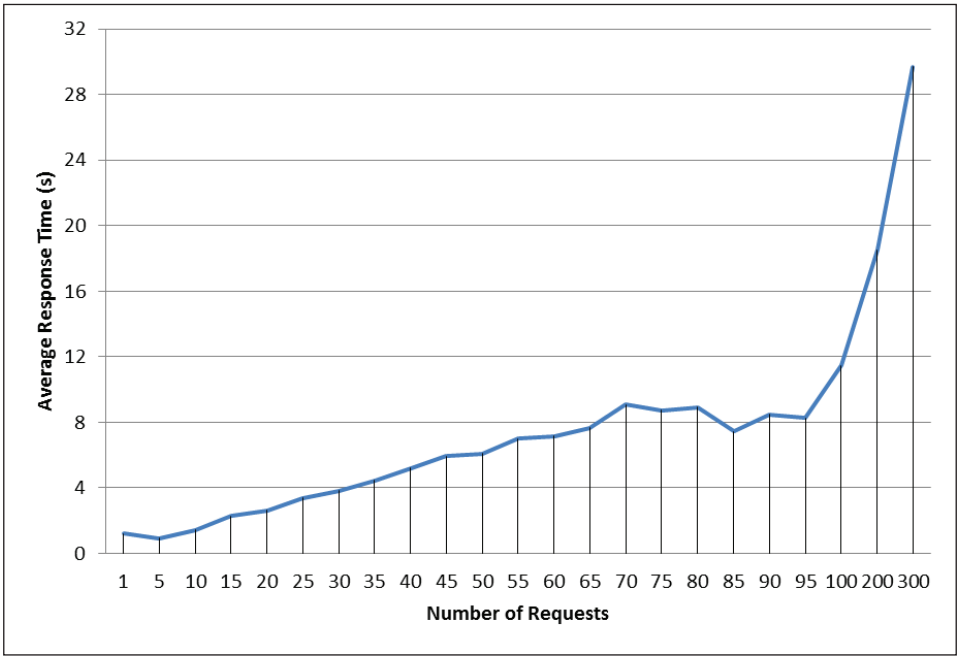

Figure 7. Mean Response Time for the proposed set of requests, varying from 1 to 300 (using an interval of 5 requests).

We used the TOP command to register the CPU load consumption in OpenShift generated for three sets of simultaneous requests to 100 requests, 1,000 requests and 2,000 requests. These data were plotted, generating the three graphs of Figure 9. Figure 9a shows the percentage of CPU consumption during the 15s that the OpenShift Platform takes to process 100 simultaneous requests, whereas Figure 9b shows the percentage of CPU consumption during 180s to process 1,000 simultaneous requests. And lastly, Figure 9c shows the percentage of CPU consumption during the 265s that the OpenShift Platform takes to process 2000 simultaneous requests.

We found that the maximum load of 100 requests was less than 4% of CPU use, for 1000 requests it was close to 5% of CPU use and for 2000 requests it was 6% of CPU use. Thus, we concluded that the resources provided by OpenShift had a wider margin than the Application Server's needs. We also used the TOP command to measure the consumption of CPU resources of the Ontology Service in AWS, recording the values for 100, 1,000 and 2,000 simultaneous re-

```
top - 18:36:22 up 77 days, 20:11,  1 user,  load average: 0.04, 0.03, 0.05
Tasks:  98 total,   1 running,  97 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   1016292 total,   716640 used,   299652 free,   153128 buffers
KiB Swap:        0 total,        0 used,        0 free.   338348 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  776 ubuntu    20   0 2163900 128272  13728 S  0.3 12.6  58:59.37 java
    1 root      20   0   33504   2872   1492 S  0.0  0.3   0:12.42 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.55 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kworker/u30+
    7 root      20   0       0      0      0 S  0.0  0.0   0:10.48 rcu_sched
```

Figure 8. CPU use as reported by TOP app in the AWS instance.

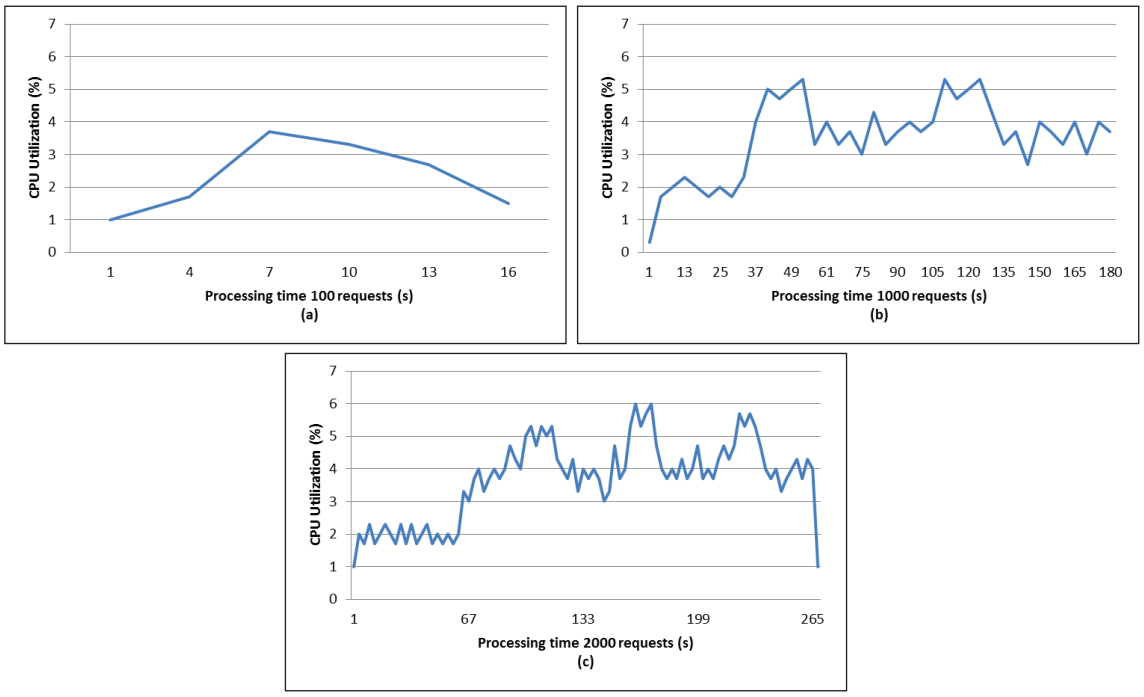

Figure 9. CPU used by the Application Server when varying the number of requests.

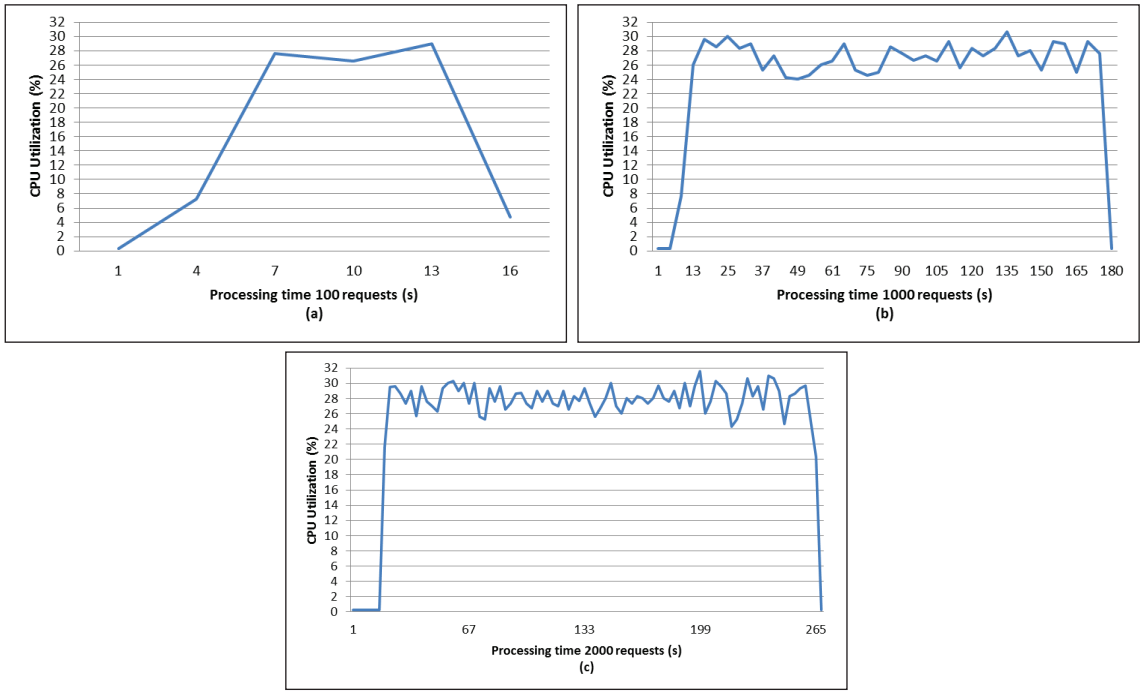

Figure 10. CPU used by the Ontology Service when varying the number of requests.

quests. These values were plotted and are shown in Figure 10. Figure 10a shows the percentage of CPU consumption during the 15s that the AWS Platform takes to process 100 simultaneous requests, whereas Figure 10b shows the percentage of CPU consumption during 180s to process 1,000 simultaneous requests, and Figure 10c shows the percentage of CPU consumption during the 265s that the AWS Platform takes to process 2,000 simultaneous requests.

The graphs show as percentage of maximum CPU loads of 28, 30 and 32% for sets of 100, 1,000 and 2,000 simultaneous requests respectively. In the case of the Application Server component, the resources consumed by the Ontology Service component (in Java) are also not significant.

In our experiments, we obtained an acceptable behavior until 100 simultaneous requests. From the point of view of capacity, the system consumed very few resources on both platforms, which leads us to conclude that, for this specific application, the architecture is not considered a bottleneck. We believe that the limit of 100 simultaneous requests may be related to the response time of SPARQL queries between the Application Server and the Ontology Service. It can also be related to network latency, throughput and network failures. Future studies should further investigate those causes.

Related works

For this work, we considered computer models that are related to health, with a focus on food allergy. Shokuping3 (Iizuka and Okawada, 2012) is a model that proposes a food menu selection support to a safe dietary life considering constraint conditions. The users included are those who are taking drugs, suffer from food allergies and people who are on dialysis. The model uses the user's profile as context, has interactivity with a Database for Food-Drug Interaction and a Food Composition Database, and is accessed via the Web and mobile devices.

PMR (Daraghmi and Yuan, 2013), Personalized Mobile Restaurant System, proposes a model that supports the choice of restaurants by consumers with different profiles (Religions, Cultures, Allergies, Health Condition, Diet, Preferences and Dislikes). The model presents location, profile and nutrition contexts, provides recommendations (restaurants/dishes/ingredients), allows Web and mobile device interactions, ordering of a meal and remote configuration.

Food Diary (Henricksen and Viller, 2012) proposes a model designed to support families dealing with children who suffering from food allergy and food intolerance. The model presents interactions with users, serves as a repository of food ingested and exchange of

information between users, such as tips on restaurants and recipes.

Food Tracker (Johnson *et al.*, 2014) is a model that provides access to restaurant menus for users via mobile devices and the Web, informing nutritional data of food items and tips for healthy eating. The model presents location and nutritional contexts and restaurant recommendation.

Considering related works, we can see that all use some form of context, either location, profile or both. However, none of them uses situation awareness as a way of combining different contexts. In the Allergy Detector we employed an ontology for decoupling food ingredients in their basic components and proteins and correlating them with the user's location and allergies.

Conclusion

The Allergy Detector model allows us to detect risks of ingesting foods that trigger allergic reactions, through the use of mobile devices and situation awareness, for users who suffer from allergy to eight allergenic ingredients (peanut, milk, egg, wheat, soy, fish, crustacean and tree nuts) and their derivatives. In the proposal, we employed contexts for setting information about situation, allergy database, profile, location, and restaurant menus (dishes/ingredients). Moreover, we used inferences in an ontology, allowing the detection of situations where there are elements allergenic to users.

The model's main scientific contribution is to present a way of using situation awareness in the scope of ubiquitous computing to detect food allergy. At the first level, using the context location, it was possible to obtain information about the name of the restaurant / food served and the ingredients in the dishes. At the second level, the model contextualized information about the dishes/ingredients to identify the presence of one of eight major allergens. Then it made a correlation between this new contextualized information and the user's profile. At the last level, the model was able to infer, by correlating contexts of the level 2, whether the site brings risks to the user. In addition, the model also has an important contribution to society by detecting the risks to users who suffer from allergy to eight allergenic ingredients or their derivatives.

We developed a prototype to conduct two evaluations. The first assessment used scenarios to demonstrate that the application of situation awareness makes it possible to alert users to the place's risk of serving dishes with allergens. The second evaluation conducted a performance measurement, showing that with the proposed architecture there is no large consumption of CPU capacity and that the model is suitable for up to 100 simultaneous requests. However, since the model is dependent on the processing of an ontology, further studies should be made to evaluate its performance with additional restaurants and more complex menus.

This model has some other opportunities for future works. One possibility is to expand the model to include all types of allergies. Secondly, we could further explore other vocabulary components in the ontology to ease integration with external sources. One possibility is exploring RDF-a (http://www.w3.org/TR/rdfa-syntax/), a standard that is becoming popular for tagging information in web sites. Finally, we should expand the evaluation by assessing usability aspects with real users.

References

ANTONIOU, I.; NANOU, T. 2003. An intelligent system for the provision of personalized dietary plans and health monitoring. *In:* International IEEE EMBS Special Topic Conference on Information Technology Applications in Biomedicine, 4, Birmingham, 2003. *Proceedings…* p. 70-73. http://dx.doi.org/10.1109/ITAB.2003.1222440

ARRUDA, L.K.; MORENO, A.S.; FERREIRA, F. 2013. Molecular diagnosis of allergy: ready for clinical practice. *Brazilian Journal of Allergy and Immunology (BJAI)*, **1**(4):187-194.

http://dx.doi.org/10.5935/2318-5015.20130024

DARAGHMI, E.; YUAN, S. 2013. PMR: Personalized Mobile Restaurant System. *In:* International Conference on Computer Science and Information Technology (CSIT), Aman, 2013. *Proceedings…* p. 275-282.

http://dx.doi.org/10.1109/CSIT.2013.6588792

ENDSLEY, M. 1999. Supporting situation awareness in aviation systems. *In:* D.J. GARLAND; J.A. WISE; V.D. HOPKIN, *Handbook of Aviation Human Factors*. New York, IEEE, p. 257-276.

FELTES, L.H.; BARBOSA, J.L.V. 2014. A model for ubiquitous transport systems support. *Latin America Transactions, IEEE* (Revista IEEE América Latina), **12**(6):1106-1112.

http://dx.doi.org/10.1109/TLA.2014.6894007

GELOGO, Y.; KIM, H. 2013. Unified ubiquitous healthcare system architecture with collaborative model. *International Journal of Multimedia and Ubiquitous Engineering*, **8**(3):239-244.

http://dx.doi.org/10.1.1.306.9974

GUANDALINI, S.; NEWLAND, C. 2011. Differentiating food allergies from food intolerances. *Current Gastroenterology Reports*, **13**(5):426-34. http://dx.doi.org/10.1007/s11894-011-0215-7

HENRICKSEN, K.; VILLER, S. 2012. Design of software to support families with food-allergic and food-intolerant children. *Proceedings of the 24th Australian Computer-Human Interaction Conference*, p. 194-203.

http://dx.doi.org/10.1145/2414536.2414571

IIZUKA, K.; OKAWADA, T. 2012. Food menu selection support system: considering constraint conditions for safe dietary life. *In:* ACM Multimedia 2012 Workshop on Multimedia for Cooking and Eating Activities, Nara, 2012. *Proceedings…* p. 43-48.

http://dx.doi.org/10.1145/2390776.2390786

JOHNSON, T.; VERGARA, J.; DOLL, C. 2014. A mobile food recommendation system based on the traffic light diet. Disponível em: http://arxiv.org/abs/1409.0296. Acesso em: 10/10/2014.

KIM, H.; CHOI, B.; WONG, W. 2009. Performance testing of mobile applications at the unit test level. *In:* International Conference on Secure Software Integration and Reliability Improvement, 3rd, Shangai, 2009. *Proceedings…* IEEE Computer Society, p. 171-180.

http://dx.doi.org/10.1109/SSIRI.2009.28

KING, T.; HOFFMAN, D.; LOWENSTEIN, H. 1994. Allergen nomenclature. *International Archives of Immunology*, **105**(3):37-49.

http://dx.doi.org/10.1159/000236761

PEREIRA, A. da S.; MOURA, S.; CONSTANT, P. 2008. Alergia alimentar: sistema imunológico e principais alimentos envolvidos. *Semina: Ciências Biológicas e da Saúde*, **29**(2):189-200.

http://dx.doi.org/10.5433/1679-0367.2008v29n2p189

SATYANARAYANAN, M. 2001. Pervasive computing: Vision and challenges. *Personal Communications*, IEEE, **8**:10-17.

http://dx.doi.org/10.1109/98.943998

SILVA, L.R. *et al*. 2008. Consenso Brasileiro sobre Alergia Alimentar: 2007. Documento conjunto elaborado pela Sociedade Brasileira de Pediatria e Associação Brasileira de Alergia e Imunopatologia. *Revista Brasileira de Alergia e Imunopatologia*, **31**(2):64-89. Disponível em: http://www.funcionali.com/php/admin/uploaddeartigos/Consenso%20Brasileiro%20sobre%20Alergia%20Alimentar.pdf. Acesso em: 10/10/2014.

ZAHRA, S.; KHALID, A.; JAVED, A. 2013. An efficient and effective new generation objective quality model for mobile applications. *IJMECS*, **5**(4):36-42.

http:/dx.doi.org/10.5815/ijmecs.2013.04.05

ZAUPA, D.; COSTA, C.; SILVA, J. 2012. Implementing a spontaneous social network for managing ubiquitous interactions. *In:* Symposium on Computer Systems (WSCAD-SSC), 13th, Petrópolis, 2012. *Proceedings…* p. 163-170.

http://dx.doi.org/10.1007/s13278-014-0158-8

Submitted on November 23, 2015

Accepted on June 6, 2015