A New Service Oriented Framework for Self-Adapting Smart Applications in Mobile Environment

Ahmed Ghoneim*
King Saud University, Department of Software Engineering, College of Computer and Information Sciences, Riyadh 11543, Saudi Arabia.

* Corresponding author. Tel.: +966(011)-4670685; email: ghoneim@ksu.edu.sa
Manuscript submitted December 01, 2017; accepted February 27, 2018.
doi: 10.17706/jsw.13.3.180-191

Abstract: Successful coupling of service-oriented and agent-oriented architectures can produce applications that are self-adaptive in both behavior and structure over a long life span. In practice, this requires detecting and handling of changes to runtime requirements, consistency in reconfiguration, and separation of the controller from the application itself. To cope with these challenges, we propose a new framework for self-adaptive and dynamically reconfigurable smart applications. This framework is composed of loosely coupled base and controller layers. The base layer consists of a base application and base controller, while the controller layer consists of detector-classifier, inference, and evaluation sub-layers. The detector-classifier layer monitors and detects runtime events within the base application. In the inference layer, an adaptive driver agent creates new adaptation plans. The evaluation layer refines the plan currently in place and compares it with new adaptive plans. We simulated and evaluated the proposed framework behavior in a smart home environment.

Key words: SOA, agents, smart home, reconfiguration, surveillance systems, run-time event.

1. Introduction

It has been widely observed that many modern, user-centric applications (such as those supporting healthcare [1], smart home control [2], [3], surveillance systems, C4I military activities, urban traffic, and the "internet of things" [4] need to continuously adapt their structure and behavior to any changing conditions. For this, a well-defined methodology is desperately needed. We know that to model and implement intelligent application environments requires tracking of the application's behaviors, based on methodologies like SOA (service oriented architecture), and on techniques involving multi-agents or ontological representations of configurable elements. While these approaches have produced positive results, they do not address several key issues [5]-[23], including:

- How to detect and handle changing of application requirements within the smart environment?
- How to reconfigure the software application in a consistent way?
- How to separate the reconfiguration controller from the application itself?

To address these issues, we propose a framework to support self-adaptive and consistent reconfiguration of smart applications. Our framework couples SOA and multi-agent system features to drive adaptation, by evolving a base application through a separate controller. The base application is imported into a base layer

so that its runtime events can be passed through a design style view (base interface). Controller layer agents then handle these runtime events by managing the configuration of representative components in the base application. They do this by proposing a suitable plan and checking whether the representative components are consistent and compatible with the plan. If they are, then the representative components are reconfigured to reflect the plan; otherwise, certain agents at the control level become responsible for creating another plan, or deciding that there is no plan available for the given situation events.

To evaluate the proposed framework, we use a "smart home" case study involving several different runtime scenarios.

The remainder of this paper is organized as follows: in section 2, the proposed framework, its structural components, and the algorithm that describes its role are presented; in section 3, a case study is discussed and some experimental results provided; in section 4, related work is covered; finally, in section 5, we make some concluding remarks and discuss future work.

2. Related Works

To fulfill the complex, dynamic structural requirements of smart application environments, appropriate tools and techniques [5]-[23] are needed. Existing research into modeling and implementing smart application environments can be divided into four categories:

- Daily life monitoring and tracking
- Model-driven engineering for smart environments
- Development of ontological standards for run-time adaptation
- SOA and multi-agent systems for driving adaptation

In this section, we explore these four categories of research in details.

Daily life monitoring and tracking. In [5], the authors proposed a knowledge-driven approach to manage a smart home, using ontology for monitoring daily living activity. In [6], researchers introduced a model that uses a genetic algorithm for clarifying human understanding. The algorithm generates rules representing a human-understandable profile to ensure an acceptable level of performance. The authors of [7] developed an automated assistance and health monitoring technique, and simulated an agent-based smart home to test their technique.

Model-driven engineering for smart environments. In [8], the authors introduced an approach that merges the model-driven and ontology-driven approaches to modeling smart spaces at both design-time and run-time. Authors of [9] proposed an approach that combines both ontology alignment techniques with model-driven engineering to achieve interoperability among smart home devices.

Development of ontological standards for run-time adaptation. The work described in [10] introduces a formal framework for explicit representation of ontological views. In [11], the authors introduced a technique for managing and interoperating among the ontological representations of urban embedded services.

SOA and multi-agent systems for driving adaptation. In [12], a Software Distribution Management System (SDMS) was deployed as a homogenous means of monitoring smart living system services and reconfiguring them to fulfill their domain requirements. In [13], researchers presented a method for performing just-in-time integration of independent living applications and software using SOA.

In [14], an android smartphone application was developed to assist and track elderly people. The application uses distributed components that communicate via cloud-based web services, and so can be fully controlled both inside and outside the application environment. In [15], the impact of dynamic and complex of multi-agent research was studied in the context of smart environments. In [16], the authors built a system that merges multi-agents and ontologies to support behavior changes based on effective user

feedback. In [17] a multi-agent system allowed specific smart applications and their user interfaces to be dynamically updated. In [18], an agent-based framework was used to process user feedback (accept, decline, or change) and to refine the application. In [19], the authors proposed a smart monitoring system, where different home control services were selected using ant-based service selection algorithm. In [20], the authors proposed an SOA architecture for smart-home environments, focusing on service integration and the home environment mobility. Their proposed architecture was based on multiple OSGi (Open Services Gateway Initiative) platforms and P2P (peer-to-peer) communications. For communications and interactions between the different system components, service-oriented methodology and mechanisms were used. For augmenting these interaction methodology and mechanisms, the agent technology and web services were applied. Further, the Mobile Agents are applied for dealing with the dynamic situations to distribute loads from the clients to the multiple service providers. In [21], [22], introduce, how the smart devices integrated with legacy devices using the advantage of SOA at the level of communication for industrial devices networks. The Devices Profile for Web Services (DPWS) implemented for embedded systems and industrial automation. In [23], the authors addressed the interoperability requirements for smart home environments, and used bi-directional management of smart home sub-systems to evaluate the performance of a proposed system.

Note that all of the above works used monitoring and tracking of application functionality within a specific environment, and introduced some forms of SOA, multi-agents, and/or ontological representation to drive adaptations. None, however, addressed the key issues on which our work is focused, namely, detection and monitoring of requirements changes, consistent reconfiguration, and separation of reconfiguration control from the application itself.

3. Proposed Framework

In this section, we present the framework outlined in Fig. 1, detailing its primary layers and corresponding core classes, and discussing the basic connections and databases resources sketched in Fig. 2. The proposed framework is composed of three loosely coupled layers: the base layer, the composite controller layer and physical layer. The base layer structure accords with the Model View Controller (MVC) paradigm, surrounded by an interface for web service interaction with external agents. The model view corresponds to the internal structure of the main component of the smart application (e.g., the class diagram). This model view is obtaining from required information stored in a database associated with smart environment components. The model controller uses its own knowledge base for controlling and managing both the web service base layer interface and the model representing the smart application structure.

The composite controller layer is composed of three main sub-layers: the detector-classifier layer, the inference layer, and the evaluation layer. These sub-layers are also designed in an MVC style and can be executed in parallel.

The detector-classifier sub-layer consists of a layer interface web service containing functions for interacting with higher layers, and three component agents. The first of these agents is the detector, which interacts with the base layer to monitor any changes, such as run-time events or the transition states of the smart application components. The second agent runs the classifier knowledge base. The third agent is the classifier, and uses the classifier knowledge base to determine the component to which a given change corresponds.

The inference sub-layer mediates between the detector-classifier sub-layer and the evaluation sub-layer through a web service. The primary component of the inference sub-layer, the adaptive driver agent, uses an adaptation knowledge base to create a suitable plan for each input from the detector-classifier sub-layer,

and then passes this plan to the base layer interface and to the evaluation sub-layer. The base layer manages the plan by converting it to an in-action plan and passing this to the necessary components. The evaluation sub-layer takes the plan as an input from the inference sub-layer, along with the status of the corresponding in-action plan from the base layer, and creates a general report. If there is inconsistency between the created plan and the handled in-action plan, the evaluation-sub layer sends a negative acknowledgment to the inference sub-layer, triggering creation of a new plan. The process repeats until a consistent plan is acquired, or found to be impossible.

The third layer is the named physical layer, which shows the smart home devices and technologies in mobile environment

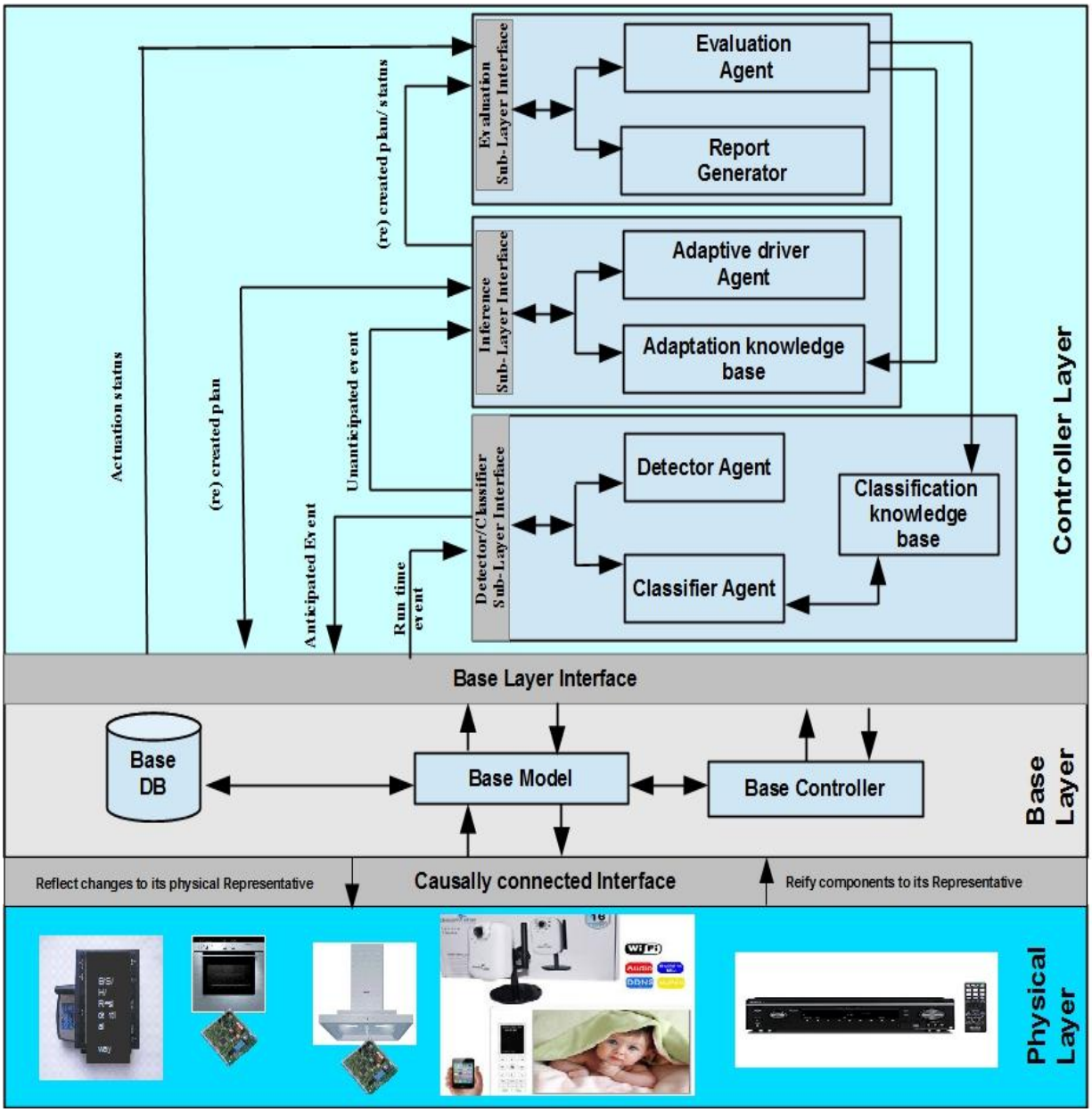

Fig. 1. The proposed framework.

The main execution flow for the framework is monitoring and detection of run time events for a given smart environment application. The control layer receives these events from the detector agent, and a suitable plan is created by applying adaptive-driver agent in which it calls adaptation knowledge based to

extract the required adaptation knowledge and rules. The following step is to reflect the created plan to the web-service base layer interface. At the final stage, the evaluation agent receives the in-action and the created plans from the base environment and from the adaptive driver agent to check their consistency value. If the consistency result is valid, the evaluation agent sends the base controller in the base environment a positive acknowledgment requesting it to execute the in-action plan. On the other hand, if the consistency result is invalid, the evaluation agent sends a negative acknowledgement to the adaptive driver agent to recreate another plan. This process is repeated until either the consistent result is a valid value or the number of requesting other plans exceed a fixed known number by the system.

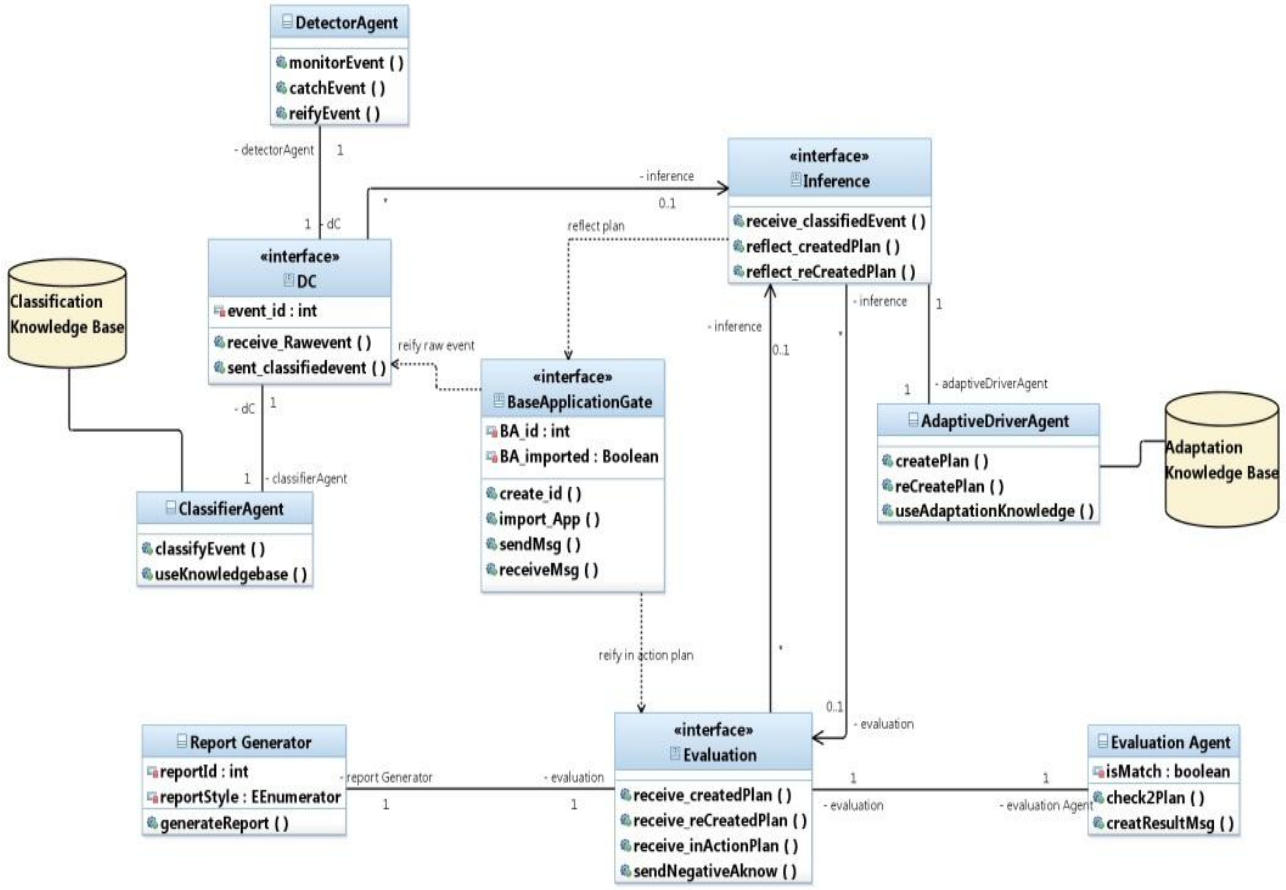

Fig. 2. Main classes and connections in the proposed framework

4. Case Study

To simulate the proposed framework and evaluate its run-time functionality, we employed a case study of a smart home environment, as illustrated in Fig. 3. Note that, the main packages are devoted to the smart home gate and home automation. The smart home gate includes a packet control interface that can receive external signals from a various channels including LANs, Bluetooth (PICONET), telephone networks, the Web, base stations, and mobility towers. A packet-controller handler receives these signals and, based on a predefined signal table, locates related home-device components to deal with them. The smart home gate also includes a configuration management sub-package for discovery and re/configuration of home components devices. The home automation package is responsible for managing security and controlling multimedia systems [24]. A class diagram for the smart home case study is provided in Fig. 4.

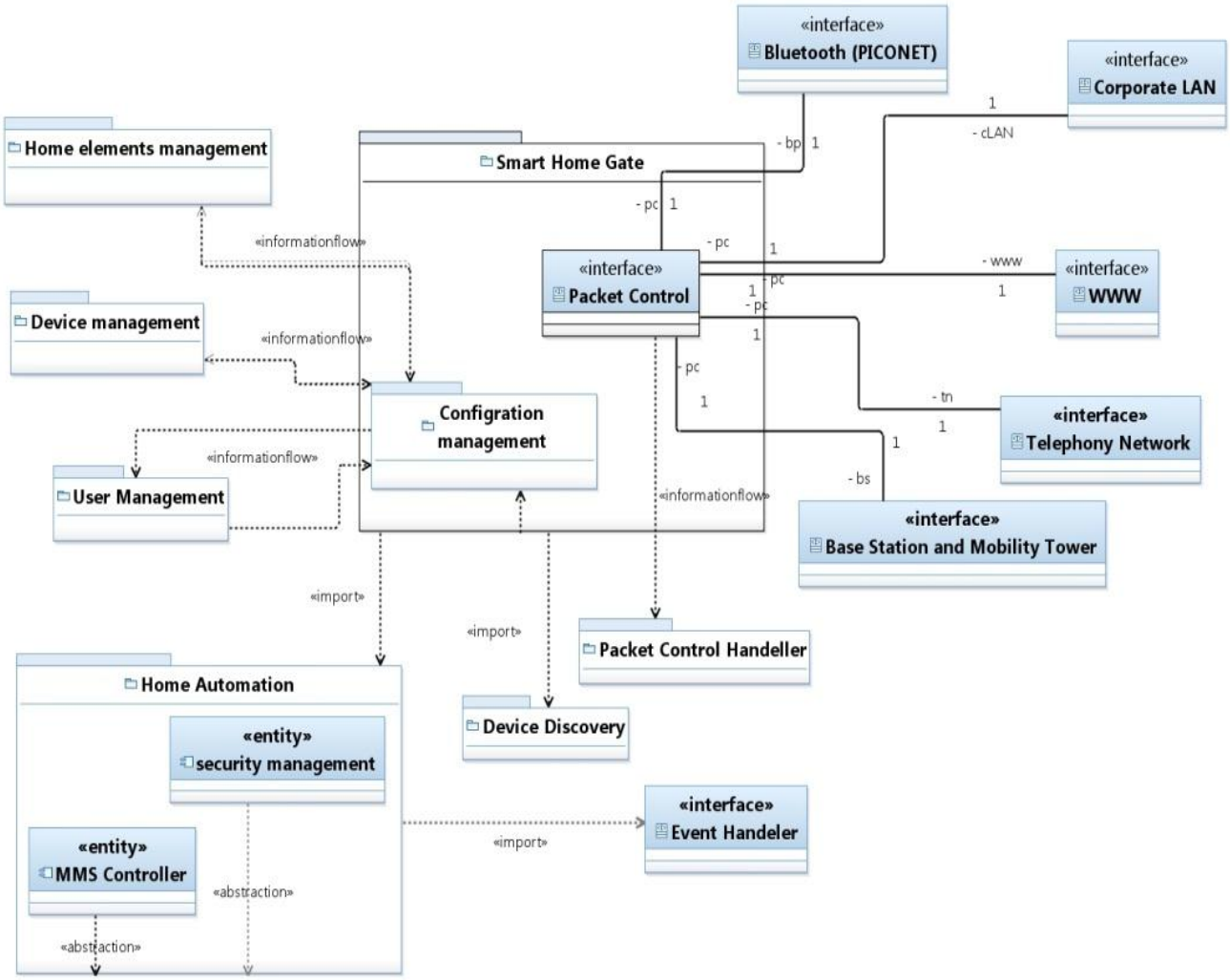

Fig. 3. Schema of simulated smart home and connected external systems.

To illustrate how the proposed framework applies to this case study, consider the scenario described in Fig. 5. When the user requests the system for viewing and playing a video, the current smart application uses self-operation to locate appropriate videos. If self-operation finds this event, it provides a list of videos to the user, who can then select one of the videos for playback (e.g., mms_id=12).

To process this kind of operation, our detector and classifier component would receive a raw-event, its parameters, and its required components, and check the latter for availability. This availability depends on (1) identifying the related category of the event, (2) extracting the specification of this category, and (3) checking whether the required components and the related category specification are matched or not. If they do match, the detector and the classifier change the status of the event from raw-event to classified-event. Otherwise, it is sent back to the base application to re-verify event requirements, a process that is repeated until verified and valid requirements are acquired.

Next, the inference interface receives the classified-event, creates a suitable action plan through the adaptive driver agent, and reflects this plan in the related home devices. In addition, this figure shows the evaluator agent rules as shown in Fig. 6.

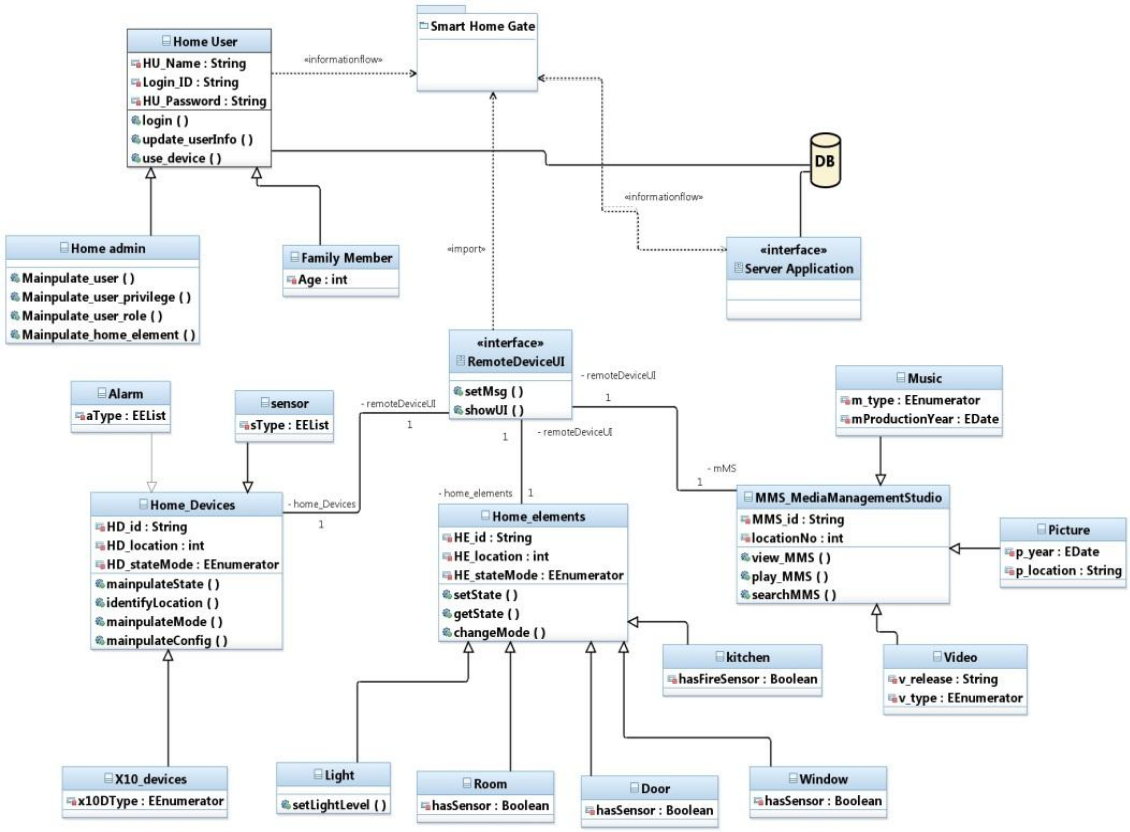

Fig. 4. The smart home runtime class structure

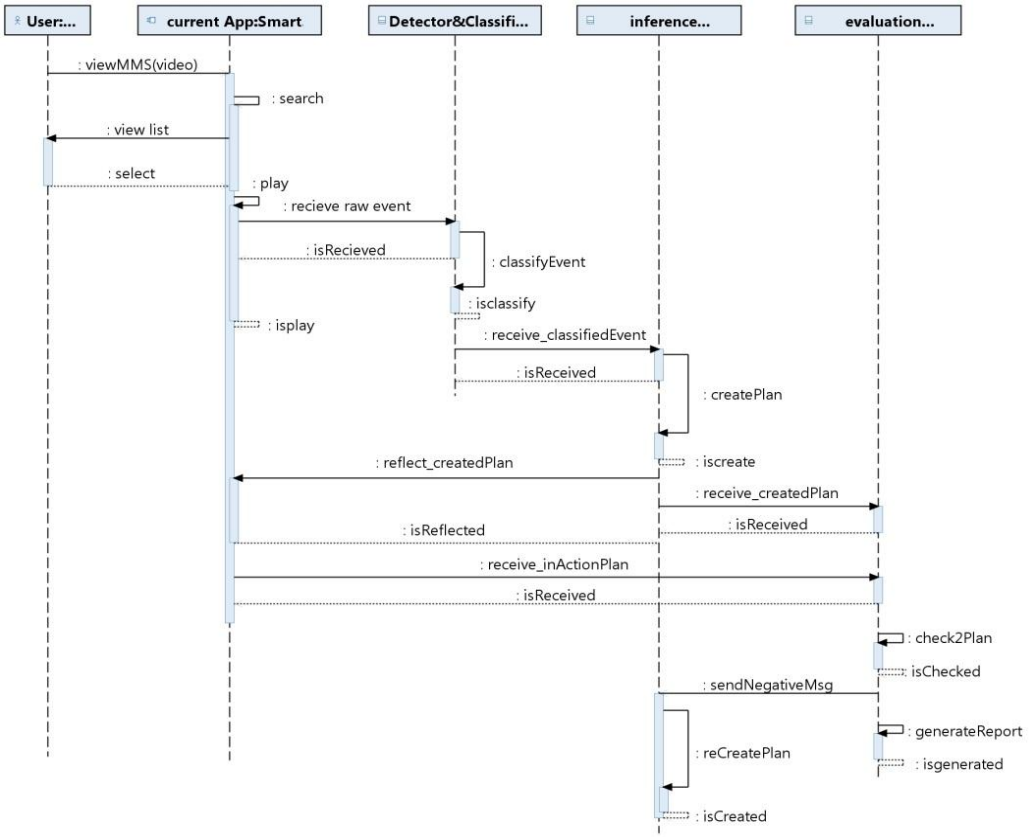

Fig. 5. Video playing scenario

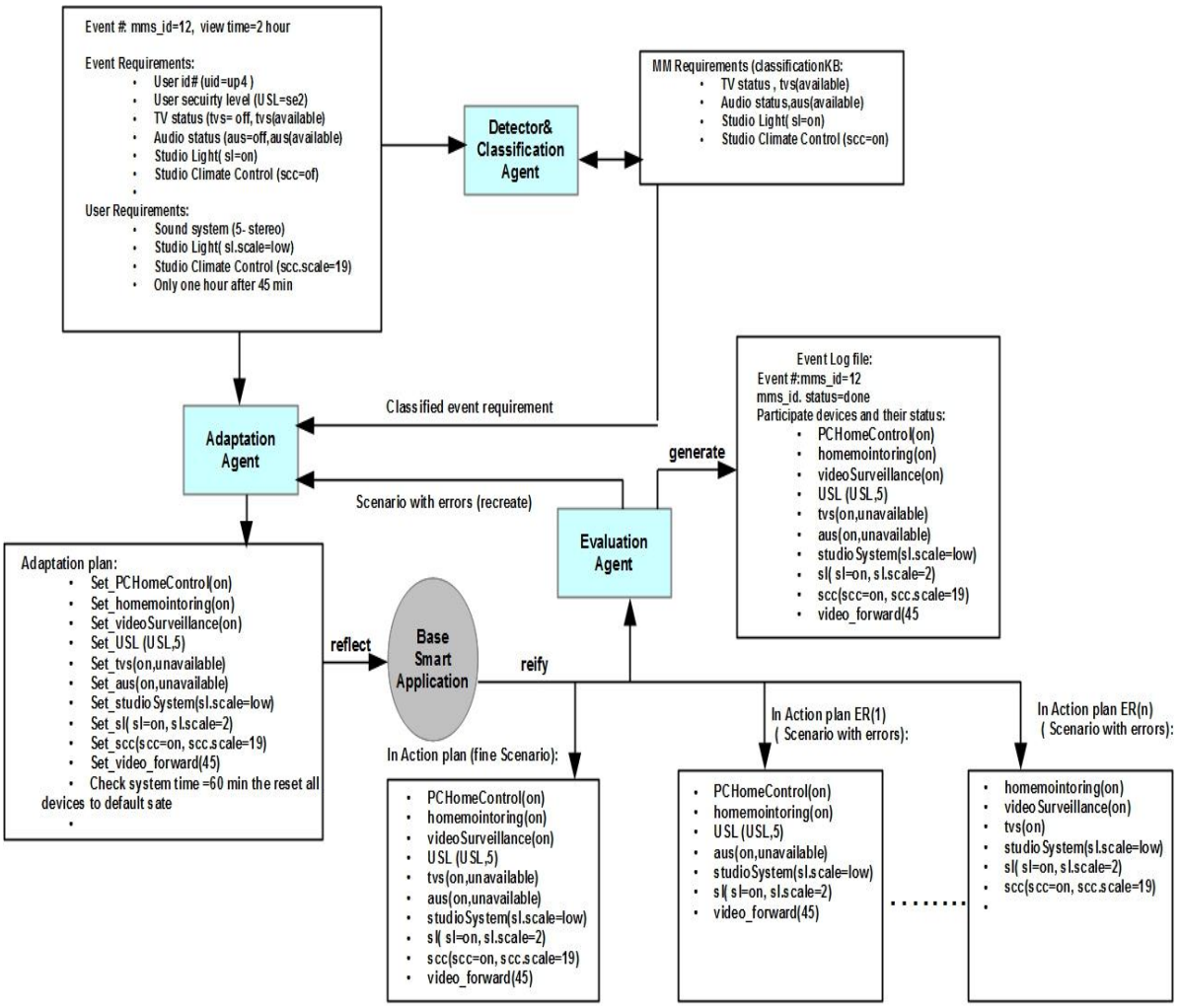

Fig. 6. Runtime output of the video playing scenario

5. Experimental Results

In order to evaluate the proposed smart home system and to validate the suitability of the proposed adaptive smart applications, a java based simulation experiments have been conducted to find the scalability with regards to smart home service requests. Then a usability study has been conducted with both mobile users and non-mobile users.

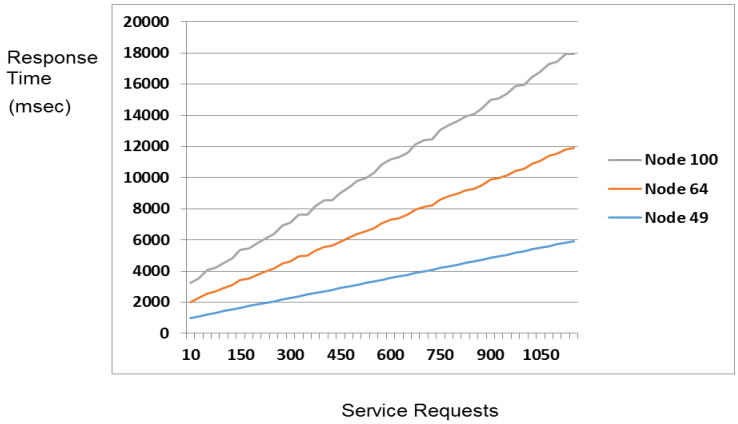

Fig. 7. Scalability for the number of concurrent requests against different smart services (100, 64, 49).

Three different computational power settings have been used in our simulation experiments. For each setting, multithreaded clients have emulated the same number of concurrent requests (1050 requests) and have sent them to smart home servers. Ten simulation experiments have been conducted for each setting, where the time span of each was almost two hours. In such experiments, all smart home servers have been configured to stream data at a maximum capacity.

As shown in Figure 7, in response to the increased number of concurrent requests, the response time and the average response time gradually and linearly increased. It is shown that the scalability is a linear until it reaches a certain threshold value. As seen from Figure 7, at 500 concurrent requests, the average response time slightly is increased in the system with 49 smart service requests (around 3001.466 ms), compared to the system with 64 services (around 3100.208 ms).

Table 1. The Comparison table

No. of selected smart services		0	1	2	3	4	5
Probability of getting best smart home service (%)	Experiment setting no 1	0	87.87	96.06	98.87	100	
	Experiment setting no 2	4.5	75.35	94.60	93.78	99.72	93.56
	Experiment setting no 3	48.13	68.45	83	92.45	94.72	87.23

The summary of the three experiments settings data is described in Table 1. It may note that the number of services selected from each smart home service group (appliance selection to social media section) varies from 0 to 5.

The above table shows that the results of the experiment setting no 1 are a bit better than those of experiment setting no 2, while the results from experiment setting no 3 are not as good as those of experiment setting no 2. It could be as a result of the user's requirement, which is clarified in the following points:

- Experiment setting no 1 satisfies the user's selection criteria,
- Experiment setting no 2 does not satisfy the user's requirements, while
- Experiment setting no 3 satisfies the users' service selection criteria.

Before presenting the usability study, we considered four important questions that are related to the adaption of smart applications in mobile environment. A questionnaire has been conducted and distributed on 100 students and researchers at different level. 50% of the participants were in the age group 13-17, 20% of them were in the age group 17-30, and 30% were in the age group 30+. Firstly, the participants have been asked to provide their preference of using smart systems (with mobile devices or without mobile devices). Following that, interaction history data has been recorded. Finally, the participants have been requested to fill up a questionnaire (Table 2), to give their opinion on the ratings of the system ease of use, preferences, response time, and likeness.

Table 2. User satisfactory questionnaires of the smart system services

Question No	Question
Q1	Smart service interaction is easy to use
Q2	Using the smart system through mobile devices

Q3	Response time of the smart system is acceptable while using mobile devices
Q4	The composed self-adaptive smart service is mostly liked by users

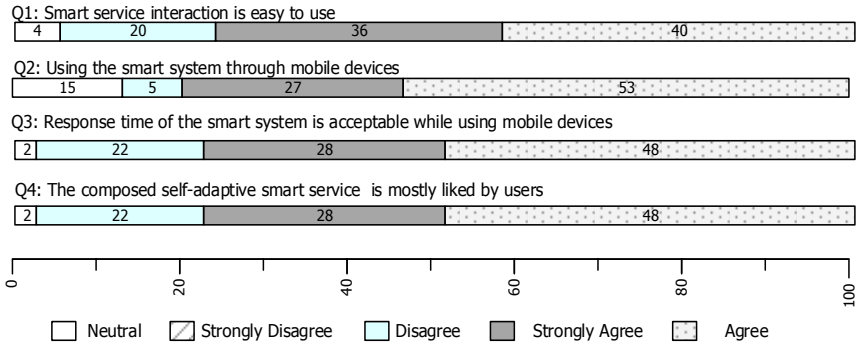

Fig. 8. Overall user response.

As shown in Fig. 8, the response of participants satisfaction based on the Likert five-point scale [25]. Feedbacks from users have been summarized and the average of user responses has been calculated. The questionnaire results have shown that the majority of users like to use self-adapting smart applications in mobile environment.

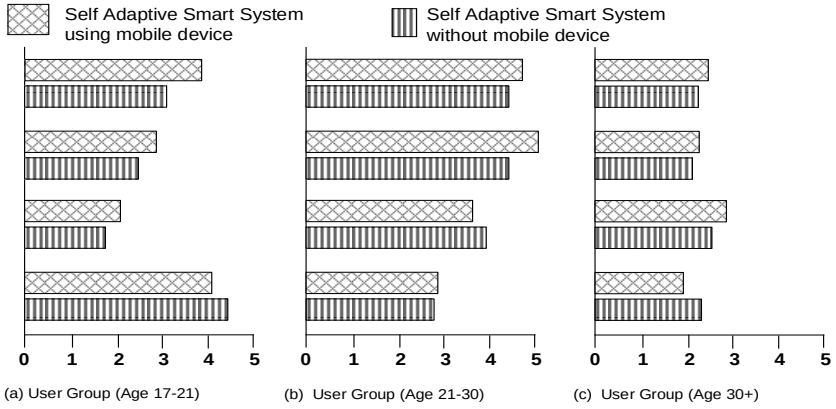

Fig. 9. Different user responses of three age groups three in Table 3

In table 3, the overall performance scores of users' satisfaction are summarized, where the higher mean (μ) and standard deviation (σ) values for ease of use, likeability and preferences, denote acceptable user satisfactory responses, while the medium mean values for response time shows a good user satisfaction. Also, acceptability satisfaction has been conducted to ascertain the suitability of the system for different age groups: ages 17~21, ages 21~30, and ages 30+ responses have been documented, as shown in Fig. 9. By comparing older age group (30+) and younger age group (17-21), users from the middle age group (21~30) seem to be more eager to use smart systems.

Table 3 User Satisfaction Evaluation

	μ	σ	μ Percentage
Ease of use (Q1)	4.21	0.89	82.2%
Preferences (Q2)	4.34	0.78	85.3%
Response time (Q3)	3.78	0.98	71.7%
Likeability (Q4)	4.17	0.83	83.3%

6. Conclusion

Our framework for dynamic reconfiguration of smart applications is based on monitoring the base application environment to detect both runtime events and environmental changes, where suitable plans of dynamic adaptation, for acceptance or rejection by a configuration controller, have been proposed.

We have demonstrated the applicability of the proposed framework by using a case study that involves smart home systems. From the experimental results, ease of use, likeability and preferences have reached acceptable user satisfactory responses, also acceptability satisfaction has been conducted to ascertain the suitability of the system for different age groups, where users from the middle age group seem to be more eager to use smart systems.

In future work, we plan to enhance quality of service by increasing the quantity and quality of monitored measurements. We also plan to identify a new method for handling the synchronization of multiple events.

References

[1] Hossain, M. S. (2015). Cloud-supported cyber–physical localization framework for patients monitoring. *IEEE Systems Journal*, 1-10.

[2] Wu, S., Rendall, J. B., Smith, M. J., Zhu, S., Xu, J., Wang, H., ... & Qin, P. (2017). Survey on prediction algorithms in smart homes. *IEEE Internet of Things Journal*, *4(3)*, 636-644.

[3] Pan, Z. (2017). A context aware anomaly behavior analysis methodology for building automation systems, Doctoral dissertation, The University of Arizona.

[4] Hossain, M. S., & Muhammad, G. (2016). Cloud-assisted industrial internet of things (IIoT) – Enabled framework for health monitoring. *Computer Networks*.

[5] Chen, L., Nugent, C. D., & Wang, H. (2012). A knowledge-driven approach to activity recognition in smart homes. *IEEE Transactions on Knowledge and Data Engineering*, *24(6)*, 961-974.

[6] Fahim, M., Fatima, I., Lee, S., & Lee, Y. K. (2013). EEM: Evolutionary ensembles model for activity recognition in Smart Homes. *Applied intelligence*, *38(1)*, 88-98.

[7] Das, S. K., & Cook, D. J. (2004). Health monitoring in an agent-based smart home by activity predition. *Proceedings of the International Conference on Smart Homes and Health Telematics* (pp. 3-14).

[8] Soylu, A., & De Causmaecker, P. (2009). Merging model driven and ontology driven system development approaches pervasive computing perspective. *Proceedings of the 24th International Symposium on In Computer and Information Sciences*, pp. 730-735.

[9] Kaed, E., Denneulin, C., Ottogalli, Y., G., F., & Mora, L. F. M. (2010). Combining ontology alignment with model driven engineering techniques for home devices interoperability. *Software Technologies for Embedded and Ubiquitous Systems* .

[10] Xue, Y., Ghenniwa, H. H., & Shen, W. (2012). Frame-based ontological view for semantic integration. *Journal of Network and Computer Applications*, *35(1)*, 121-131.

[11] Gregor, D., Toral, S. L., Ariza, T., & Barrero, F. (2012). An ontology-based semantic service for cooperative urban equipments. *Journal of Network and Computer Applications*, *35(6)*, 2037-2050.

[12] Chen, I. Y., & Huang, C. C. (2007, April). A reconfigurable software distribution framework for smart living environments. *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering*.

[13] Shen, W., Xue, Y., Hao, Q., Xue, H., & Yang, F. (2011, October). A service-oriented system integration framework for community-based independent living spaces. *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics* (SMC).

[14] Fahim, M., Fatima, I., Lee, S., & Lee, Y. K. (2012). Daily life activity tracking application for smart homes using android smartphone. *Proceedings of the 2012 14th International Conference on Advanced*

Communication Technology (ICACT) (pp. 241-245).

[15] Cook, D. J. (2009). Multi-agent smart environments. *Journal of Ambient Intelligence and Smart Environments*, *1(1)*, 51-55.

[16] Benta, K. I., Hoszu, A., Văcariu, L., & Creţ, O. (2009, June). Agent based smart house platform with affective control. *Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship* (p. 18).

[17] McNaull, J., Augusto, J. C., Mulvenna, M., & McCullagh, P. (2012, June). Multi-agent system feedback and support for ambient assisted living. *Proceedings of the 2012 8th International Conference on Intelligent Environments* (pp. 319-322).

[18] Cavone, D., De Carolis, B., Ferilli, S., & Novielli, N. (2011). An agent-based approach for adapting the behavior of a smart home environment.

[19] Hossain, M. S., Hossain, S. A., Alamri, A., & Hossain, M. A. (2013). Ant-based service selection framework for a smart home monitoring environment. *Multimedia tools and applications*, *67(2)*, 433-453.

[20] Wu, C. L., Liao, C. F., & Fu, L. C. (2007). Service-oriented smart-home architecture based on OSGi and mobile-agent technology. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *37(2)*, 193-205.

[21] Jammes, F., Mensch, A., & Smit, H. (2005). Service-oriented device communications using the devices profile for web services. *Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing* (pp. 1-8).

[22] Zeeb, E., Bobek, A., Bohn, H., & Golatowski, F. (2007). Lessons learned from implementing the devices profile for web services. *Digital EcoSystems and Technologies Conference*..

[23] Perumal, T., Ramli, A. R., Leong, C. Y., Mansor, S., & Samsudin, K. (2008). Interoperability for smart home environment using web services. *International Journal of Smart Home*, *2(4)*, 1-16

[24] Trilles, S., Calia, A., Belmonte, Ó., Torres-Sospedra, J., Montoliu, R., & Huerta, J. (2017). Deployment of an open sensorized platform in a smart city context. *Future Generation Computer Systems*, *76*, 221-233.

[25] Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*.

Ahmed Ghoneim received his M.Sc. degree in software modeling from University of Menoufia, Egypt, and the Ph.D. degree from the University of Magdeburg (Germany) in the area of software engineering, in 1999 and 2007 respectively. He is currently an assistant professor at the department of software engineering, College of Computer Science and Information Sciences, King Saud University. His research activities address software evolution; service oriented engineering, IoT, software development methodologies, Quality of Services, Net-Centric Computing, and Human Computer Interaction (HCI).