ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

Self-configuring Booking SaaS Application

Eman.M-Fageer[1],Nadir K.Salih[2],
University of Computer Science and Technology, Khartoum, Sudan[1]
Department of Electrical and Computer Engineering, Engineering College, Karary University, Sudan[2]
eman_mo_ha@yahoo.com[1]nadircom2006@gmail.com[2]

Abstract *This study aims to demonstrate the impact of self-configuration in system and how systems interact or adapt itself according to environment .The study includes a theoretical introduction to identify autonomic computing, self-configuration and how to make the system easy to be managed .This research tried to implement self-configuration system that help provider to provide multiple configurations for tenants to build their websites according to their requirements without interference between the different tenants requirements. To achieve this we follow three steps: first saas architecture is been designed which divided saas system to three levels and each level divided into four layers to make the management of the system easy and provide services to each level. Second feature model is been designed to illustrate the relationship between the different levels. Third the saas application algorithm(SCAS) to simulate SCAS in our application online hotel booking as case study.The study has concluded the following findings: First the use of architecture so important to manage system and provide services according to level. The provider can manage the tenant and tenant can manage the user. Second, the models determine relationship between the layers and represent input and output clearly .Third, (SCAS) is able to reduce costs without compromising quality and make the system global.*

The study recommends the following: First use reflex technique to make components of application self-configuration .Second apply the (SCAS) algorithm in saas applications.

Keyword: SaaS application, architecture, Self-configuration , Feature model.

1. Introduction

The ideas behind autonomic computing are not new. In fact it is possible to find some aspects of autonomic computing already in today's software products. For instance, Windows optimizes its user interface (UI) by creating a list of most often used programs in the start menu. Thus, it is self-configuring in that it adapts the UI to the behavior of the user, although in a fairly basic way, by monitoring what programs are called most often. It can also download and install new critical updates without user intervention, sometimes without restarting the system. Therefore, it also exhibits basic self-healing properties. DHCP and DNS services allow devices to self-configure to access a TCP/IP network (albeit in a limited Autonomic computing is an initiative derived by IBM 2001, the concept inspired from the function of the human body (e.g nervous system and how the body reacts unconsciously)[1].

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

Autonomic computing is combination of different theories and practices came from several areas that already exist like (control theory, adaptive algorithms, software agents, robotics, fault-tolerant computing, distributed and real-time systems, machine learning, human-computer interaction (HCI), artificial intelligence, and many more).

The need of autonomic computing appears to overcome the rapidly growing and complexity of the computer systems, which make very difficult to manage the large and heterogeneous systems, one error in these complex systems is very difficult to find and to be fixed it may take time and need to employ experts which may cost the company.

The solution was to develop self management computing systems which contains four Characteristics[2] .

2. Autonomic computing Characteristics

Self-optimization: system can optimize it process while it's running without affecting on the performance.

Self-configuration**:** to adapt the system by itself for different environments.

Self-protecting**:** to protect the system from malicious attacks without user intervention.

Self-healing**:** the systems prevent disruptions and provide recovery from malfunctions.

3. Self-configuration

The advances in computing and communication technologies and software tools have resulted in an explosive growth in networked applications and information services that cover all aspects of our life. These services and applications are inherently complex.

Autonomic Computing (AC), which addresses the problems associated with the increasing complexity of computing systems as well as the evolving nature of software[3], is the ability of computing systems to manage themselves and adapt to changes in accordance with business policies and objectives[4].

The essence of autonomic computing systems is self-management and it is derived from a combination of four broad capabilities: self-configuring, self-healing ,Self-optimizing, and self-protecting. Self-Configuration refers to the ability of a system to obtain its configuration parameters and initialize itself in order to provide the expected services. Self-Configuration techniques can be viewed as either Initial Configuration,

methods for specifying initial configuration requirements or Dynamic Configuration,

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

methods for specifying reconfiguration based on given states[5]. For autonomic systems ,self-configuration encompasses the initial configuration of a system as well as dynamic, reactive changes throughout its operational life. Policies are often used to guide these configuration transitions.

4. implement configuration

After having an over view on autonomic computing area and self-configuration characteristic particularly we suggested to use architecture, feature mode and Self-Configuration Algorithm to achieve self-configuration on systems which provide:

Self-configuration (Configure it selves) to system according to environment without interference from user by :

4.1 SAAS Architecture

Saas architecture is been designed which divided Saas application to three levels to make the management of the system easy and provide services to each level [6].

Application layers

- Graphical User Interface.

Graphical User Interfaces use pictures and graphics instead of just words to represent the input and output of a program. The program displays certain icons, buttons, dialogue boxes . on the screen and the user controls the program[7].

- Business Process.

A business process is a collection of activities that takes one or more type of input and creates an output that is of value to the customer. Includes any information about the end product or service .Clearly identified inputs and outputs

- Service.

A Service Level Agreement (SLA) is a contract between Service Providers and tenant, what are the services that Service ,Provider will provide and what limit the Service Provider will pay if he cannot meet the committed goals[8].

- Database.

The hardware infrastructure

- Network.
- Storage.

The autonomic management

Provider is the first levels consist of four layers:

- The Graphic User Interface (GUI) has different styles for the GUI as agencies need or select
- Business Process (BP) the provider puts the business. It can be variable from agency to agency
- Services (S), can be different for the service dispatcher and service catalog between agencies. Then cataloged which means the sending, indexing services from provider to tenant are not the same. The variation in the service level agreement for

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

many services can be according to agency requirements.

- Data Base (DB), in this layer defines as any agency by a unique identifier.

Tenant is second level consist of four layer level:

- Graphic User Interface (GUI) travel agency show a suitable style interface for the users .
- Business Process (BP), the business logic can be different from user to user so that travel agency can use a different formula
- Service layer the travel agency sending and indexing the services depends on the type of user, and the classification of the service for different costs.
- Data Base the travel agency defines any user by an identifier because it is unique for every user, and attributes data can be different from user to user.

User is third level consisting of four layers:

- variation in graphic user Interface is defined by the tenant or travel agency.
- Business process put by the travel agency.
- Service layer introductions from travel agency are according to user requirements.
- Data base travel agencies give all users id

Management process

Contains self-configuration.

In our research we try to implement some parts of the architecture to demonstrate configuration among the application according to SCAS algorithm.

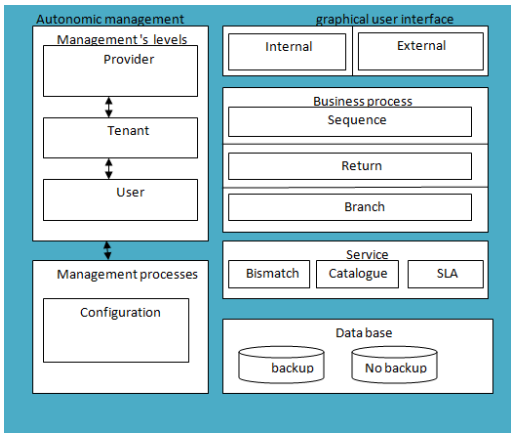

Fig 1 Architecture of SAAS application

4.2 Feature Model for Saas application

Shows relation of each layer and the description of the configuration for the provider level .we have observed that there are many variations in layers . Those will help SaaS application to give multiple choices and provide many tenants as shown in the following figure 2.

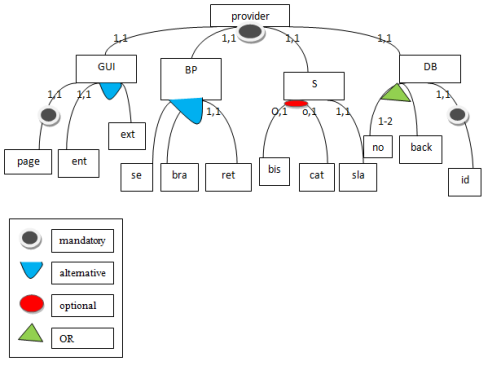

Fig 2 Feature model

4.3 SCAS Algorithm:

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

Give multiple choices and provide many tenants with different Configuration

Demonstrate relation between the four layers, contains four concepts which are:

- Mandatory (1,1)
- Optional(0,1)
- Alternative(1,1)
- OR ( )

After having an over view on architecture of SAAS application that divided the system to three Levels and feature model show the relation between layers then the implementation in our application(online hotel booking) and simulate algorithm of SAAS application to represent self-Configuration in online hotel booking system.

Online booking with the hotel as the provider level and managed at the tenant level or by the travel agencies that managed users as case study.

We implemented the concepts of the architecture, feature model and SCAS algorithm and we have using (ASP.net-SQL server) to simulate the algorithm in application

.

Our application has three levels: first, provider is first level has environment ready to tenants to build their website which is ready to user to select hotel or complete reservation

Results and discussion

As we mention our SaaS model has three levels of management including the provider, tenant and user. Every level has different managements for the application layers. the provider is the administrator for all travel agencies. The travel agencies look like tenants and Customers are users that want to book a travel service.

We took an on-line hotel booking example to demonstrate this configuration as shown in the following figures.

The provider environment ready for tenants to reserve a name for their agency

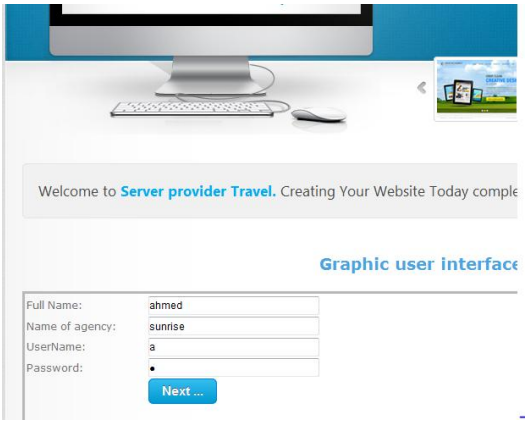

Fig 3 provider level

This second step to create pages by agencies as need

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

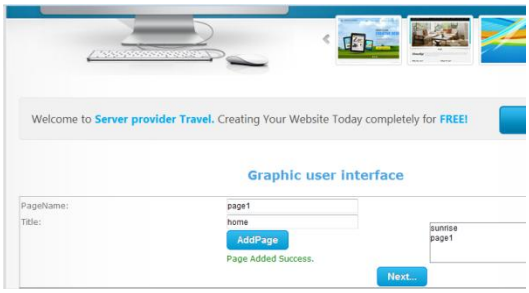

Fig 4 provider level

The third step show the alternative concept of SCAS. There are two options tenent must select one of them (internal-external).

Fig 5 provider level

The fourth step represent business process and show alternative concept of SCAS ,must select one of the options (return-sequence-branch).

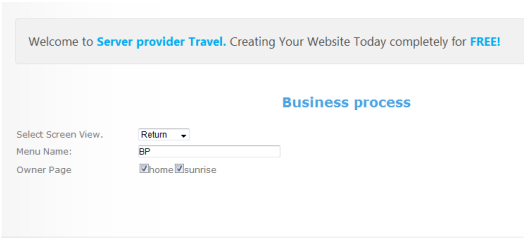

Fig 6 provider level

The fiveth step show optinal services that mean tenant can celect one of services or two or nothing.

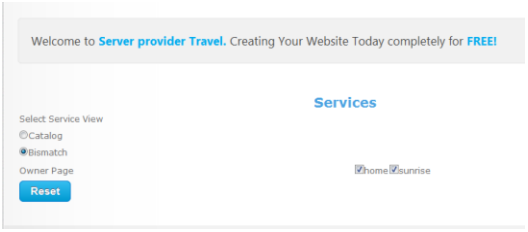

Fig 7 provider level

The seventh step is database,it implement OR consept , tenant can select part of it or select all.

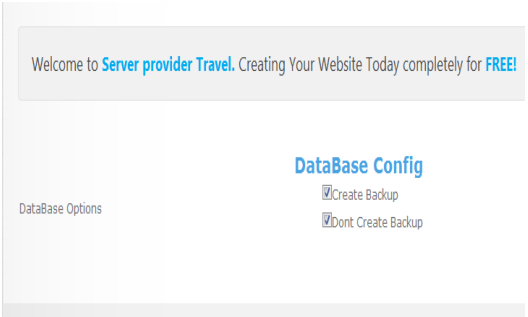

Fig 7 provider level

The eight step show the web site of tenant as requreired and ready to users to select suitable hotel.

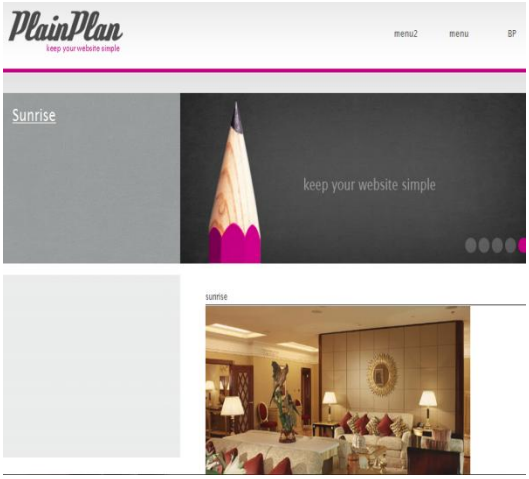

Fig 8 Tenant level

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

The provider is the administrator for all travel agencies

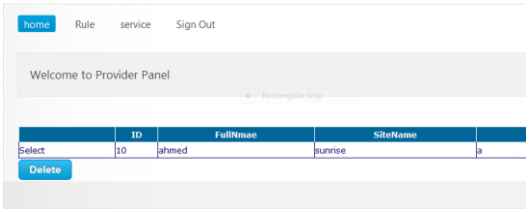

Fig 9 provider

Results of Self-configuration Algorithm for Provider Perspective

Configuration of provider GUI:

Config1= { PGUI, page, internal}. Config2= { PGUI, page, external }.

Configuration of provider BP:

Config1= { PBP, sequence }.

Config2= { PBP, branch }.

Config3= { PBP, return }.

Configuration of provider Service:

Config1= { PS, bismatch, catalog, SLA}.

Config2= { PS, bismatch, SLA}.

Config3= { PS, catalog, SLA }.

Config4= { PS, SLA}.

Configuration of provider database:

Config1= { DB, ID, share, isolate}.

Config2= { DB, ID, share(no backup) }.

Config2= { DB, ID, isolate (backup)}.

5. Related Work

[6]In this paper searchers design new model for SAAS application and discovered a new algorithm for self configuration, classifies the SAAS application management in three layer demonstrated the relationship between them

[9]in this paper authors introducing user-centric policies that guide its autonomic behavior These user-centric policies provide a way for users to state preferences and concerns that influence service provisioning.

[10]In this paper authors formalized a fundamental self-configuration problem. It forms a basis for managing the cross interdependencies of configurationally items, assessing the system-wide impacts of changes, and making dynamic decisions about new configurations.

To support the embedded reasoning and decision services that enable dynamically self-configuring automotive software systems, authors formalized and proposed a preliminary solution to the task self-configuration problem, which seeks to decide which tasks should be active on ECUs subject to resource constraints and task dependency.

[5] In this paper Authors present a new paradigm based on the principles of autonomic computing that can handle efficiently complexity, dynamism and uncertainty in networked systems and their applications. authors developed a general Component Management Interface (CMI) to enable autonomic behaviors and operations

ISSN: 1858-7658 (Online)
ISSN: 1858-7690 (Print)

of any legacy resource or a software component.

Authors successfully implemented the CMI in XML format and validated the effectiveness and performance of their approach to develop automated and self-configuring security patch deployment services

6. Conclusions

This research is depend on architecture and feature model for SaaS application to build online hotel booking defined four layers to composite system and showed the associations and dependencies of the layer elements.

In our application classify services according to three levels. We have increased the quality of the system by showing it has different level services and

can serve by important ordering. In addition we have simulated the self-configuration algorithm to dynamically configure application components

In future work we will see the effect of reflex technique to make components of application self-configuration in the application

7. References

[1] A. Computing, W. Paper, and T. Edition, “An architectural blueprint for autonomic computing .,” no. June, 2005.

[2] O. Jeffrey and M. David, “The Vision of,” no. January, pp. 41–50, 2003.

[3] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, J. O. Kephart, and I. B. M. Thomas, “An Architectural Approach to Autonomic Computing,” 2004.

[4] N. K. Salih, “Autonomic Management for Multi-agent Systems,” vol. 8, no. 5, pp. 338–341, 2011.

[5] H. Chen, S. Hariri, F. Rasul, C. Port, and O. Port, “An Innovative Self-Configuration Approach for Networked Systems and Applications University of Arizona,” pp. 537–544, 2006.

[6] N. K. Salih and T. Zang, “Modeling and Self-Configuring SaaS Application.”

[7] S. Wiesmann, “Graphical User Interface - Layout and Design Responsible persons : Regula Stopper René Sieber,” 2012.

[8] E. Marilly, O. Martinot, S. Betge-Brezetz, and G. Delegue, “Requirements for service level agreement management,” *IEEE Work. IP Oper. Manag.*, vol. 00, no. C, pp. 57–62, 2002.

[9] H. W. Networks, “Self-Configuration and Self-Optimization Process in Heterogeneous Wireless Networks,” pp. 425–454, 2011.

[10] L. Feng, D. Chen, and T. Martin, “Self Configuration of Dependent Tasks for Dynamically Reconfigurable Automotive Embedded Systems.”