SELF-MANAGEMENT SaaS APPLICATION BY CBR ALGORITHM

Nadir K.Salih*[1], Tianyi Zang*[2]
Electrical and computer engineering department, Karary University, Sudan[1]
School of Computer Science and Engineering, Harbin Institute of Technology, China[2]

*Correspondences should be addressed
nadircom2006@gmail.com[1] tianyi.zang@hit.edu.cn [2]

ABSTRACT: *Take management strain a way from consumers is the main objective in SaaS application. It is the heavy duty centralized in provider level. And we have observed the complexity of the SaaS management from one level. To solve these problems we have design new algorithm base on case base reasoning concept. Which it realize self-management capabilities i.e. self-optimization, self-configuration, self-healing, self-protecting. Depending on our three management levels we have applied this algorithm. The result is increase the SaaS application performance by self-solution problem. However to give more accurate solution we have used accuracy and RMSE adaptation methods. Finally we present a conclusion and future work.*

Keywords: SaaS application, self-management, case base reasoning,

I. INTRODUCTION

In this paper we have described how to evolution SaaS application depends on our new model. That is by applying Case base reasoning (CBR) as Meta-Heuristics (MH) form. We used this algorithm to solve the problem without intervention of human this let our new model of SaaS application manage autonomously. For example the system can self-optimizing for resources, self-configuration for components, self-healing for error by detecting and solve any problem, and self-protecting from any threats. CBR adapts previous solutions for similar problem in solving new problem in hand. For similarity we have used deferent functions that give optimal adaptation. We generate algorithm to self-adaptation our three level models in SaaS application. To realize our opinions we demonstrated three levels management in online booking application. We have summarized our contributions as follows:

- We have been allowed autonomous SaaS systems to continuously and proactively seek for opportunities to improve its execution.
- Using CBR, systems can remember the past effectively solve cases and autonomously decide which MH and parameters to use for the resolution of a new similar problem.
- The system is able to find optimal or near optimal solutions through the use of MH, deal with dynamism.
- By our algorithm of A CBR based on three levels modeling approach increased the performance of SaaS application by quickly selected suitable solution.
- Self-selected of suitable cases for three levels of our model it help system in self-management.
- Adaptation of our algorithm by accuracy and RMSE lead to use the best similar function to optimal solution.

We organized this paper by beginning with the case base reasoning in section II. Depending on our CBR algorithm in three levels modeling approach we have realized autonomic management for SaaS application in section III. To manage SaaS application in runtime we explain the role of adaptation algorithm in section IV. In section V we have described the related work for SaaS application management. We present the conclusion and point to future work in section VI.

II. CASE BASE REASINIG

A case-based reasoned solves new problems by using or adapting [1] solutions that were used to solve old problems. Reasoning that adapts previous solutions for similar problem in solving new problem in hand [2]. Many problems decision makers encountered are similar to old cases. Often it is more efficient to start with the previous solution to a similar problem than to generate the entire solution again from scratch [3].

a. AUTONOMIC BEHAVIOR

We have been allowed autonomous systems to continuous and proactive seek for opportunities that improve its execution. Also through identification and use of opportunities it become more efficient in its performance and cost [4]. Meta-Heuristics (MH) form a class of the powerful and the practical solution techniques for tackling complex, large-scale combinatorial problems producing efficiently high-quality solutions [5]. Using CBR, systems can remember the past for effectively solve cases and autonomously decide which MH and parameters to use for the resolution of a new similar problem. The system is able to find optimal or near optimal solutions through the use of MH, deal with dynamism [6]. Experts solve problem based on previous cases. It contains four steps as depict in figure 1 bellow [7] [8].

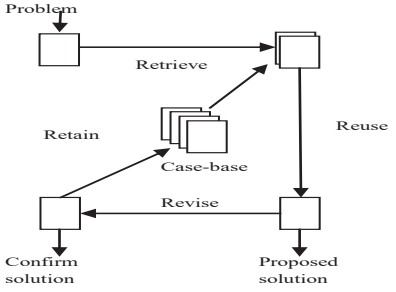

Fig 1 Cycle of Case Base Reasoning

Retrieve: Current case is compared with the existing cases in case base and set of most similar cases are retrieved.
Reuse: Solution of the nearest neighbors is used to devise

solution of the current case.

Revise: need to revise the solution of the current problem. However to make CBR more sensible, adaptation phase should be kept minimal.

Retain: This is a beauty of the CBR approach that recently solved problem is immediately available in the case base for future use.

Components of CBR: components will be included in case base reasoning representation by retrieval cases measure similarity. And it uses heuristic rule for adaptation cases if the solution is not good. After adaptation the case can retain it and organize it case base with the indexing.

b. SIMILARITY FUNCTIONS

Similarity functions can be used to get a group of the nearest neighbor solutions for current case problem [9]. That will be lead to catch accuracy by different functions in various using. Here in table 1 bellow can see example of this function are Hamming distance, Manhattan distance, Euclidean distance, Sim1 , Sim2 , Canberra distance, Bray–Curtis distance, Squared Chord distance, Squared Chi-Squared distance and Jaccard similarity function [10].

Function	Function Formula
Hamming similarity	$sim_{ij}=\frac{matches_{k-1}^{m}(P_{ik},P_{jk})}{m}$
Manhattan distance	$d_{ij}=\sum_{k=1}^{m}W_k\lvert P_{ik}-P_{jk}\rvert$
Euclidean distance	$d_{ij}=\sqrt{\sum_{k=1}^{m}(W_k\,(P_{ik}-P_{jk}))^2}$
Sim1	$sim_{ij}=1-\max(IP_{ij},OP_{ij}),\ IP_{ij}=\max(\min(P_{ik},P_{jk})),OP_{ij}=\min(\max(P_{ik},P_{jk}))$
Sim2	$sim_{ij}=1-t_1*t_{2,}\ t_1=\min(IP_{ij},1-OP_{ij}),\ t_2=\frac{IP_{ij}+(1-OP_{IJ})}{2}$
Canberra distance	$d_{ij}=\sum_{k=1}^{m}\frac{\lvert P_{ik}-P_{jk}\rvert}{P_{ik}+P_{jk}}$
Bray–Curtis distance	$d_{ij}=\frac{\sum_{k=1}^{m}\lvert P_{ik}-P_{jk}\rvert}{\sum_{k=1}^{m}P_{ik}+P_{jk}}$
Squared Chord distance	$d_{ij}=\sum_{k=1}^{m}(\sqrt{P_{ik}}-\sqrt{P_{jk}})^2$
Squared Chi-Squared distance	$d_{ij}=\sum_{k=1}^{m}\frac{(\lvert P_{ik}-P_{jk}\rvert)^2}{P_{ik}+P_{jk}}$
Jaccard similarity coefficient	$sim_{ij}=\frac{C_i\cap C_j}{C_i\cup C_j}$

Table 1 Similarity Functions

A distance measure is needed to determine the closeness of instances. Classify an instance by finding its nearest neighbors and picking the most popular class among the neighbors. K-nearest neighbour methods are mainly suitable for problems with numerical data (typically real, floating point numbers).

c. ALGORITHM FOR AUTONOMIC

Initial retrieval starting with some attribute for case, compute similarity that weight sum of attribute retrieval case is the same. Adaptation rule is the rule obtains from engineering session with expert appraises. By additional feature cause the different between subject and retrieved cases. That will adjust to better reflect property value for final selection. Aggregate selection cases and combine to produce estimate of the value of subject the result will justification reliable value

Listing 1 Case base reasoning algorithm:

Input: CB containing n cases, C_{np} case new problem, SM similarity measure, SA solution algorithm, N_n nearest neighbor.
Output: select solution for case C_{np}.
For each $C_j \in CB$
simi = find similarity(C_{np}, SM)
w = all simi cases.
retrieve w.
End for
N_n= find nearest solution
Revision(N_n)
Retention
CB= CB +N_n
End.

The steps of our algorithm retrieve, reuse, revise and retain as defined in listing 1 it realize autonomic characteristic. Because it do some processes without user intervention.

III. A CBR BASE ON THREE LEVELS MODELLING APPROACH

CBR it contains retrieving, reusing, revising, and retaining cycle for obtains a good matching case. Similar functions add in order to selecting the best alternatives. We observe the important thing in CBR to determine by looking at similarities of data. For that in primary define the main attributes of cases and ordering of them. From the weight of attribute the system calculate distance value between cases by similar function mention in table1. The minimum distance indicates the best similar case to the new case. For that we have preferred to use CBR because it reducing the knowledge acquisition, avoiding mistake, learning in runtime, and it is very suitable for incomplete or imprecise data. We management our SaaS model by provider level will manage tenant level or travel agencies and travel agencies will administrate for user level as depict in figure 2. We define our algorithm to solve any problem for any level just by enters the new case. By case criteria's algorithm will retrieve similar cases and reuse this case to propose solution. After that it revises to

confirm solution. In the last retain this solution in case base, because it can be used in future to solve new problem. We take example for every level to explain our algorithm as in following:

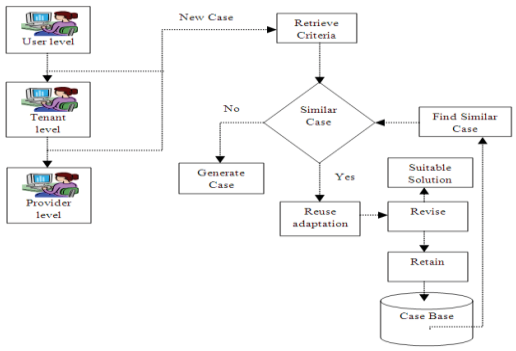

Fig 2 CBR Architecture for Three Levels

1. USER LEVEL MANAGEMENT

In user level the SaaS application can help user to now and expect booking price by show some cases that define by some requirements.i.e. customer location (CL), number of room (NR), booking level (BL) hotel (H) ,period (P) that the customer will be stay in hotel, transportation(T) percent will be give as service to customer, total costing (C)of booking as in table 2

Cases	CL	NR	BL	H	P	T	C
C_1	Jamoci	1	High	meridian	2 week	50%	30,000
C_2	Harbin	2	Middle	grand village	3 week	100%	20,000
C_3	Harbin	1	High	meridian	3 week	50%	30,000
C_4	Jamoci	3	Low	Manhattan	2week	25%	15,000
C_5	Harbin	3	Middle	grand village	2week	100%	25,000

Table 2 User Level Cases

Assuming that there are n criteria that determine each case, the distance measure can be composed as Manhattan distance see formula 1:

$$d_{ij} = \sum_{k=1}^{m} W_k |P_{ik} - P_{jk}| \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where W_k it is the related weight of criterion A.
P_{ik} is the requested case and P_{jk} is one of the cases in the knowledge base. All the criteria of the P_{ik} have to be compared with all the criteria of each case in the knowledge base. The calculation of the distance requires that each instance of criteria is represented with a numeric value.

2. TENANAT LEVEL MANAGEMENT

This level will be managed by tenant level or by travel agencies in our running example. We take some information from customers' requirements to make variation between them. By other meaning this requirements will be look as attributes in cases, i.e. type of holidays (TH), number of travelers (NT), region to visit (RV) transportation mode (TM), duration (D), Season (S), accommodation (A). The table 3 bellow shows some sample of cases in our example.

Case	TH	NT	RV	TM	D	S	A
C_2	mountain	3	Tianjin	plane	7	Feb	low
C_3	historical	2	Suzhou	plane	4	June	mid
C_4	island	2	Wuhan	plane	4	Aug	high

Table 3 Tenant Level Cases

The group of cases that has the most similar criteria values with the new case is retrieved to give the expected booking rate.

3. PROVIDER LEVEL MANAGEENT

This level in our SaaS model it administrator by provider level. We can look in our example to show the requirements of travel agencies that will demands some resources to achieve what they want. We can take some attributes, i.e. service name (SN), web server (WS), application server (AS), and database server (DS). The table 4 bellow defines some cases of the services event from travel agencies level.

Cases	SN	WS	AS	DS
C_1	Cancelling booking	1	1	0
C_2	Display user form	0	1	1
C_3	Modify tariff	1	1	1
C_4	Reporting	0	1	1

Table 4 provider level cases

The table take some cases includes samples of the services for travel agencies that it used some resources. The number (1) it means the service used this resource and (0) not used. Case base reasoning here can measure or select best costing for services or it can show the configuration of components. In addition if some services make error in case it can be avoid in new cases.

IV. ROLE OF ADAPTATION ALGORITHM

All autonomic methods that are used on recent researches ignore current dataset and dynamic learning in a runtime. For that this techniques could not manage unseen cases, didn't learn from mistakes haven't continuous improvement in learning. The big issue that not applied in most techniques is how to formalize and structure knowledge for testing and solving problem in autonomic management system. We found case base reasoning can be a good option to manage SaaS application in runtime [11]. It can be familiar in formalize knowledge to obtain all autonomic capabilities self-configuration, self-optimization, self-healing, and self-protecting that should give SaaS application self-management feature.

a. CASE BASE REPRESENTATION

To evaluate the performance of solution proposed if a problem has been solved for one service it can be reapplied and adopted for any other service. This advantage leads us to make CBR very powerful in SaaS application. In addition CBR can realize autonomic architecture element by:

Monitoring component, the service get monitored by CBR to collect metrics of CBR-attributes that create the case for CBR. The CBR attributes have to be careful chosen to solve different problems.

Analysis: the collected CBR-attributes to given a time and analyzed.

Plan: the CBR create in analysis phase to get used for finding and good solution.

Execution: in order to execute operation for the service. Execution component translates the more abstract solution in case base operation.

CBR implementation by cases representation as a set of n attributes and case base represented as a set of m cases. As in our SaaS model we have described in section III three levels defined in online booking example. In user level we have represented attributes case as:

C_u = (CL, NR, BL, H, P, T, C).

In Tenant level we have represented attributes case as:

C_t = (TH, NT, RV, TM, D, S, A).

In Provider level we have represented attributes case as:

C_p = (SN, WS, AS, DS).

b. WORKING ALGORITHM

To configure any level we have needed to invoke three files contains i.e. cases, attributes, and mapper for every level. This can be appeared in config class as in listing 2 bellow.

Listing 2 config class

```
package CaseBasedReasoning;
/* Config for runtime application*/
public class Config {//config for display log message
        public static boolean DEBUG      = true;
        //config for cases filename
        public static String caseFilename =
"providerlevel.cases";//config for atribut filename
        public static String atributFilename =
"providerlevel.attributes";//config for mapper filename
        public static String mapperFilename =
"providerlevel.mapper";"
```

We have taken Manhattan distance as similar function and can input weight of attributes and input new case. The system will display the new best fit case and the match's case content by smallest similarity distance as it work in solver listing 3 bellow. At first our algorithm defined solver calss to show new cases with attributes and weights for our model in three levels userlevel, tenantlevel, and providerlevel.

c. EXPERIMENTAL RESULTS

We have taken result for every level from running algorithm at first user level or the customer in our running example: Event will be do in future from any travel agency. Here we can make self-healing to the service by look at similar attributes of nearest case.

d. ADAPTATION ALGORITHM

Limitation measure of the nearest case for problem and retrieve it by function of similarity measure it represent for the developers want to know which the best solution to make decision. And the algorithm demand which the attributes that will help to measure similarity. To improve the efficiency of our system we can use optimal nearest neighbor algorithm that has been proposed to determine the optimal solution. We can depend on accuracy and root mean squared error (RMSE) to show the different between similarity functions. To analysis our adaptation algorithm we supposed ten cases for user level and eight cases for tenant level, provider level as the same. These cases it has defined customer whose want to book online from travel agencies. And these travel agencies has controlled by administrator or provider level to monitor all events. Our algorithm dynamically it select the nearest similar solution for new case according to different similar functions as it represented in table 5 algorithm output. The bold in every

row showed the best solution for the new case.
We have obtained the optimal solution by accuracy formula 2 to give maximum value by similar functions. The optimal solution for every level used accuracy of our model it represented in table 6.

$$\text{Accuracy} = \frac{\text{total correct prediction}}{\text{total cases}} * 100 \ldots\ldots.(2)$$

Model level	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	Select case	Similar function
Userlevel	23.8	1.6	23.5	**0.3**	1.6	23.8	1.6	1.6	23.8	1.6	4	Manhattan
	34.1	2.5	33.2	**0.94**	2.5	34.1	2.5	2.5	34.1	2.5	4	Euclidean
	1.48	0.4	0.88	0.6	0.4	1.48	0.4	0.4	1.48	**0.4**	10	Canberra
	29.5	0.83	28.5	1.0	0.83	29.5	0.83	0.83	29.5	**0.83**	10	Squared chord
	43.4	1.6	41.6	1.8	1.6	43.4	1.6	1.6	43.4	**1.6**	10	Squared chi
Tenantlevel	0.5	0.4	0.3	0.8	0.5	0.2	0.3	0.6			6	Manhattan
	1.3	1.0	0.76	1.78	1.39	**0.63**	0.76	1.34			6	Euclidean
	0.8	0.53	0.34	0.8	0.8	**0.33**	0.34	0.6			6	Canberra
	1.1	0.44	0.17	1.1	1.1	0.34	**0.17**	1.0			7	Squared chord
	2.0	0.86	0.34	2.0	2.0	0.66	**0.34**	1.8			7	Squared chi
Providerlevel	0.2	**0.1**	0.3	0.8	0.5	0.4	0.3	0.5			2	Manhattan
	0.44	**0.31**	0.54	1.75	0.99	0.86	0.54	1.2			2	Euclidean
	0.33	**0.33**	1.0	2.33	1.3	1.3	1.0	1.3			2	Canberra
	0.17	**0.17**	1.0	2.3	1.17	1.17	1.0	1.3			2	Squared chord
	0.33	**0.3**	1.0	2.6	1.3	1.3	1.0	1.6			2	Squared chi

Table 5 CBR Algorithm output

Model level	Accuracy	Function
Userlevel	0.103	Manhattan
	0.148	Euclidean
	0.007	Canberra
	0.122	Squared chord
	0.181	**Squared chi**
Tenantlevel	0.0045	Manhattan
	0.0122	**Euclidean**
	0.0056	Canberra
	0.0067	Squared chord
	0.0057	Squared chi
Providerlevel	0.0038	Manhattan
	0.0082	Euclidean
	0.0111	Canberra
	0.0103	Squared chord
	0.0117	**Squared chi**

Table 6 Accuracy of Similar Function

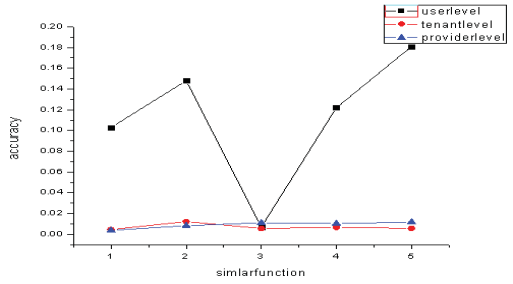

Fig 3 Optimal Solution by Accuracy

The result of accuracy it has appeared in figure 3 the number in x axis denoted to similar function 1 Manhattan, 2 Euclidean, 3 Canberra, 4 squared chord, 5 squared chi .By the meaning of the accuracy maximum value for user level is 0.181 it realized by squared chi function. In tenant level the best function is Euclidean because it gives 0.0122 as a big value. Squared chi function again gives 0.0117 as optimal value in provider level.

The selection of optimal function will increase the SaaS application performance, because the suitable function realizes perfect solution. Another way we have used to select the optimal solution of our algorithm is a root mean squared error (RMSE). It denotes by give minimum value of the error for similar functions see formula 3.
Here the optimization of the result depends on the less value of similar functions as we bolded value in table 7.

$$\text{RMSE}(X_1, X_2) = \sqrt{\frac{\sum_{i=1}^{n}(X_{1,i} - X_{2,i})^2}{n}} \ldots\ldots(3)$$

Model level	RMSE	Function
Userlevel	14.84	Manhattan
	20.86	Euclidean
	0.6139	**Canberra**
	23.86	Squared chord
	26.15	Squared chi
Tenantlevel	**0.275**	**Manhattan**
	0.628	Euclidean
	0.311	Canberra
	0.650	Squared chord
	1.160	Squared chi
Providerlevel	**0.351**	**Manhattan**
	0.684	Euclidean
	0.681	Canberra
	1.071	Squared chord
	1.115	Squared chi

Table 7 Optimal Value by RMSE

We have showed the RMSE in figure 4 to depict the optimal value for every level. Like we observed the less

value is 0.6139 by the Canberra number function for user level. For tenant level the best value is 0.275 by Manhattan function. And for provider level the value is 0.351 by Manhattan function.

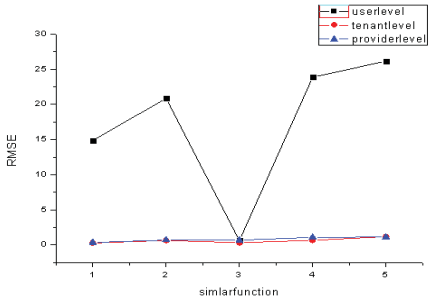

Fig 4 Optimal Solution by RMSE

Here we have adapted our algorithm to give more performance for management SaaS application.

V. RELATED WORK

SaaS contributes to improve relationships between vendors and customers. In the traditional model once the software is sold, it is largely up to the customer to make it work. in [12] they defined Multilayered customization framework supporting continuous testing and recoverability for database layer. Support SaaS customization with two-layer database partitioning at the top level, there is a partition for each tenant, Lower level, a larger component for column partitions. In [13] they designed Conceptual architecture of a SaaS platform to Executing of configurable and multitenant SaaS application. Author in [14] they overviewed the multi-tenancy architecture it increased utilization of hardware resources and improved ease of maintenance. In [15] thy made flexible frame work. To self adaptive business process in SaaS application depends on handled the exceptions according to rules provided by tenant. In [16] they Configurable business process, and isolation database to Leverage SMEs. Researchers in [17] Innovated multi-layered customization framework to Manage the variability of SaaS applications and tenants-specific requirements. Ontology is used to derive customization and deployment information for tenants cross layers. In [18] authors used Context-oriented Programming to achieves a higher customization flexibility by Single-version application code base. In [19] they Identified requirements for such a runtime architecture addressing the individual interests of all involved stakeholders. SaaS reference architecture must support at design time as well as at runtime. They Self-optimize the runtime environment according to a tenant's configuration. Authors in [20] they designed Three architectural patterns that support variability in multi-tenant SaaS environments. In [21] they depicted the design space and represent the common and variant parts of SaaS architectures. Feature model enhances the understanding of SaaS systems, and supports the architect in designing the SaaS application architectures. The research in [22] Defined architecture by SaaS application layers for flow customization business process. Integration requirements and roadmap between SaaS application and on-premise applications are introduced in [23] by frame work architecture in SBM services. In [24] they designed Systematic approach and corresponding tool support for guiding the design of SaaS application architectures. Selected features are related to design decisions and a SaaS application architecture design is derived. Separate Data Servers for Tenants, Separate Application Server, and one Distribution Server. They implemented architecture in [25] to define a SaaS application as an organization-based service composition and helps SaaS vendors, tenants in modeling and managing their applications. In [26] they change application to SaaS architecture with Business logic should be substituted with simple integration to the business services of the platform. The main issue in [27] is SaaS service architecture maturity model should give a concrete way of constructing a strategy to build practical SaaS service. In recent SaaS application reserches we didn't found any research apply self-management for SaaS application.

VI. CONCLUSION AND FUTUREWORK

We have founded Case Base Reasoning can be a good option to manage SaaS application in runtime. It can be familiar in formalize knowledge to obtain all autonomic capabilities self-configuration, self-optimization, self-healing, and self-protecting that give SaaS application self-management feature. From our new algorithm we have done experiments the ouput is increase the performance of SaaS application. in user level it select suitable booking in price as customer's requirements. For travel agencies or tenants level we predict for them the expected booking rate. By retrieved some nearest neighbor case that use resources to serve and mointor tenant services provider level can self-management the problems.

In future work we will look at optimization our new algorithm to be fully cotrol for SaaS application management by apply some fuzzy concepts.

REFERENCES

[1] P. lama, X. zhou. Autonomic Provisioning with Self-Adaptive Neural Fuzzy Control for Percentile-Based Delay Guarantee. ACM Transactions on Autonomous and Adaptive Systems, Vol. 8, No. 2, Article 9, 2013.

[2] M. Wang, H. Mei, W. Jiao, J. Jie, J. Ma. A Self-Adaptive Mechanism for Software Configuration Based on Case-based Reasoning and Policy. International Conference on Artificial Intelligence and Computational Intelligence. IEEE,2010,pp.250-255.

[3] E. Bou, M. L. anchez, J.A. R. Aguilar. Using Case-Based Reasoning in Autonomic Electronic Institutions. Springer-Verlag Berlin Heidelberg, 2008, pp. 125–138.

[4] M. J. Khan, M. M.Awais, S. Shamail. Enabling Self-Configuration in Autonomic Systems using Case-Based Reasoning with Improved Efficiency. International Conference on Autonomic and Autonomous Systems. IEEE, 2008, pp.112-117.

[5] S. Nasir, M. Taimoor, H. Gul, A. Ali. M. J. Khan. Optimization of Decision Making in CBR Based Self-Healing Systems. International Conference on Frontiers of Information Technology. IEEE, 2012, pp. 68-72.

[6] S. Montani ,C. Anglano. Achieving self-healing in service delivery software systems by means of case-based reasoning. Springer Science+Business Media, 2007, pp.139-152.

[7] S. Hassan ,D. McSherry , D.Bustard. Autonomic self healing and recovery informed by environment knowledge. Springer Science+Business Media, 2007, pp.89-101.
[8] S. Montani , C. Anglano. Case-Based Reasoning for Autonomous Service Failure Diagnosis and Remediation in Software Systems. Springer-Verlag Berlin Heidelberg 2006, pp. 489–503.
[9] Pereira, A. Madureira. Self-Optimization module for Scheduling using Case-based Reasoning. Applied Soft Computing, 2012, pp. 1419–1432.
[10] M. J. Khan , M. M. Awais , S. Shamail , I. Awan. An empirical study of modeling self-management capabilities in autonomic systems using case-based reasoning. Simulation Modelling Practice and Theory, 2011, pp. 2256–2275.
[11] Ivanka Menken ,Gerard Blokdijk. SaaS - The Complete Cornerstone Guide to Software as a Service Best Practices. The Art of Service.2009.
[12] Wei. Tsai, Q.Shao, Y. Huang, X. Bai. Towards a Scalable and Robust Multi-tenancy SaaS. Second Asia-Pacific Symposium on Internetware. ACM, 2010.
[13] S. Kang, Sungwon Kang, S. Hur. A Design of the Conceptual Architecture for a Multitenant SaaS Application Platform.IEEE, 2011.
[14] C. Bezemer, A. Zaidman. Multi-Tenant SaaS Applications: Maintenance Dream or Nightmare, Report TUD-SERG,2010.
[15] Y. Shi, S. Luan, Q. Li, H. Wang. A Flexible Business Process Customization Framework for SaaS.IEEE, 2009.
[16] Irene S. Harris, Z. Ahmed. An Open Multi-Tenant Architecture to Leverage SMEs. European Journal of Scientific Research. Vol.65 No.4, pp. 601-610, 2011.
[17] W.Tek Tsai, Q. Shao, W. Li. OIC: Ontology-based Intelligent Customization Framework for SaaS. International Conference on Service-Oriented Computing and Applications (SOCA) IEEE,2010.
[18] E. Truyen, N. Cardozo,S. Walraven, J. Vallejos, Bainomugisha, S. Gunther, T. Hondt, W. Joosen. Context-oriented Programming for Customizable SaaS Applications. Proceedings of the 27th Annual Symposium on Applied Computing ACM, 2011, pp. 418-425.
[19] Julia Schroeter, Sebastian Cech, Sebastian Götz, Claas Wilke, Uwe Aßmann. Towards Modeling a Variable Architecture for Multi-Tenant SaaS-Applications. Sixth International Workshop on Variability Modeling of Software-Intensive Systems.ACM, 2012.
[20] Jaap Kabbedijk, Slinger Jansen. Variability in Multi-tenant Environments: Architectural Design Patterns from Industry, Springer, 2011.
[21] K. Öztürk, B. Tekinerdogan, Feature Modeling of Software as a Service Domain to Support Application Architecture Design. International Conference on Software Engineering Advances. IARIA, 2011.
[22] Ralph Mietzner, Frank Leymann. Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors, International Conference on Services Computing.IEEE, 2008.
[23] Y. Zhang, S. Liu, X. Meng. Towards High Level SaaS Maturity Model: Methods and Case Study. IEEE, 2009.
[24] B. Tekinerdogan, K. Ö., A.Doğru, Modeling and Reasoning about Design Alternatives of Software as a Service Architectures. Ninth Working IEEE/IFIP Conference on Software Architecture.IEEE,2011.
[25] M. Kapuruge, A. Colman, . J. Han. Achieving Multi-tenanted Business Processes in SaaS Applications. Springer-Verlag Berlin Heidelberg ,2011, pp. 143–157.
[26] D. Concha, J. Espadas, D. Romero, A. Molina. The e-HUB evolution: From a Custom Software Architecture to a Software-as-a-Service implementation. Elsevier,2010, pp 145–151.
[27]S. Kang, J. Myung, J. Yeon,S.-w. Ha, T Cho, J.-man Chung, S.-goo Lee. A General Maturity Model and Reference Architecture for SaaS Service. Springer-Verlag Berlin Heidelberg, 2010, pp. 337–346.