International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 5, Issue 2 [November 2015] PP: 20-26

Optimization of Resource Allocation using FCFS Scheduling in Cloud Computing

Mr. E. Rajesh[1], Mr. J. Mahalakshmi[2].
1((M.Tech)CSE , Krishna Chaitanya Institute of Technology & Science,Markapur Andhra Pradesh, India)
2(Assoc.Professor, Dept of C.S.E, Krishna Chaitanya Institute of Technology & Science,Markapur Andhra Pradesh, India)

Abstract: As the size of software techniques improves, the methods and data elements of the computations no longer represent the major style issues. When techniques are designed from many elements, the organizations of the overall system—the software architecture—presents a new set of style issues. This level of style has been resolved in various ways such as informal diagrams and illustrative diagrams, UML diagrams, layouts and frameworks for techniques that serve the needs of specific domains, and official designs of element incorporation systems. In this paper we offer presenting the growing area of software architecture. We begin by considering a cloud computing environment, to evaluate the system behavior. Simulators are now widely used in this area and are becoming more complicated. Most of them offer frameworks for replicating application arranging in various cloud environments, others are developed for modeling systems, but only a few of them imitate arranging guidelines. Lastly, we study some of the excellent issues in the area, and consider a few of the appealing research guidelines.

Keywords: *software architecture, design, cloud computing, schedulers.*

I. INTRODUCTION

As the size and complexness of software systems improves, the style issue goes beyond the methods and information components of the computation: developing and specifying the overall system framework comes out as a new kind of issue. Architectural problems include total organizational and global control structure; methods for interaction, synchronization, and information access; task of efficiency to style elements; physical distribution; structure of style elements; scaling and performance; and choice among style solutions.
This is the software structure level of design. There is a considerable whole body of perform on this subject, including module interconnection networks, layouts and frameworks for techniques that serve the needs of specific domains, and official models of component incorporation systems. In addition, an implied whole body of perform prevails in the form of illustrative terms used informally to explain techniques [1]. And while there is not currently a well-defined language or note to define structural components, good application technicians [5] make common use of structural concepts when developing complex application. Many of the concepts signify guidelines [4] or idiomatic styles that have appeared informally over time. Others are more carefully recorded as industry and scientific standards.

It is progressively clear that effective software engineering innovation needs service in architectural design. First, it is important to be able to identify common paradigms [3] so that high-level connections among techniques can be recognized and so that new techniques can be designed as modifications on old techniques. Second, getting the right structure is often important to the achievements of anarchitectural design [2]; the incorrect one can cause to terrible results. Third, specific knowing of application architectures allows the professional to make principled options among style solutions. Fourth, an architectural reflection is often important to the research and information of the advanced stage qualities of a complicated program.
In this paper we are going to discuss about the software architectural design issues in cloud computing environment. The cloud computing environment is simulated by using an efficient simulating tool called as Cloud sim.

Cloud Sim offers the following new features: (i) assistance for modeling and simulator of extensive cloud computing environments, such as data centers, on a single physical computing node; (ii) a self-contained system for modeling clouds, provisioning, service brokers, and allocation policies; (iii) assistance for simulator of system relationships among the simulated system elements; and (iv) service for simulator of federated cloud environment that inter-networks sources from both public and private domains, a function crucial for experiments related to Cloud-Bursts and automated application scaling. Some of the improvements of CloudSim are: (i) accessibility to a virtualization engine that helps in the development and management of several, separate, and co-hosted virtualized solutions on a data center node and (ii) versatility to change between space-

shared and time-shared allowance of handling cores to virtualized solutions. These powerful functions of CloudSim would speed up the development of new program provisioning methods for cloud computing.

II. BACKGROUND AND RELATED WORK

Cloud computing has appeared as a new processing model which is designed to provide efficient, personalized and QoS assured powerful processing surroundings for end-users. With cloud processing, it is possible to allow more versatile support distribution and improve primary IT procedures, such as both customer and program provisioning and systems control. These services are generally separated into three categories: Infrastructure as- a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).

- Infrastructure-as-a-Service (IaaS): it provides clusters, grids, networks, or virtualized servers, systems software designed andstorage to augment or replace the features of an entire data center.
- Platform-as-a-Service (PaaS): it provides servers virtualization on which it can run on existing applications or develop new applications without having to concern about maintaining the operating systems, hardware, load distribution or computing capacity.
- Software-as-a-Service (SaaS): This is the most commonly known and commonly used form of cloud computing. SaaS provides all the features of an innovative traditional program, but through a Web browser, not a locally-installed program. It removes problems about application servers, storage space, database integration and related, common concerns of IT.

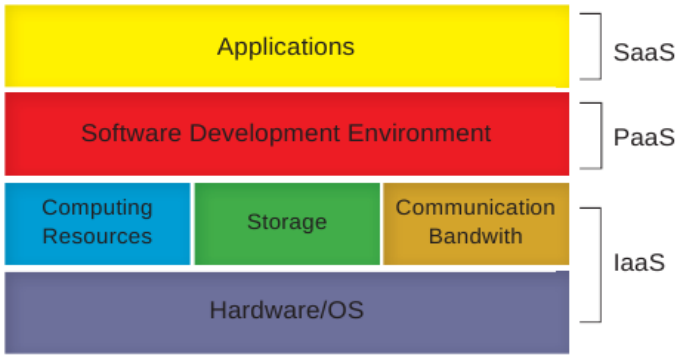

Fig. 1 Layered architecture of Cloud

Cloud computing served in three forms:private clouds,public clouds, and hybridsclouds which are explained in figure 2.

- Private Clouds: A private cloud is one in which the solutions and facilities are managed on a private network. These cloud over the biggest stage of protection and control, but they require the company to still purchase and sustain all the software and facilities, which decreases the price benefits.
- Public Clouds: A pubic cloud is one in which the solutions and facilities are offered off site over the Internet. These cloud over the biggest stage of performance in distributed resources; however, they are also more insecure than private cloud. Many IT division professionals are involved about public cloud protection and stability. Take additional time to make sure that you have protection and governence issues well organized, or the short-term price benefits could turn into a long-term headache.
- Hybrid Clouds: A hybrid cloud has a wide range of public and private options with several suppliers. By growing things out over a hybrid cloud, you keep each part at your company in the most effective cloud possible. The disadvantage is that you have to keep track of several different protection systems and make sure that all factors of your company can connect with each other.

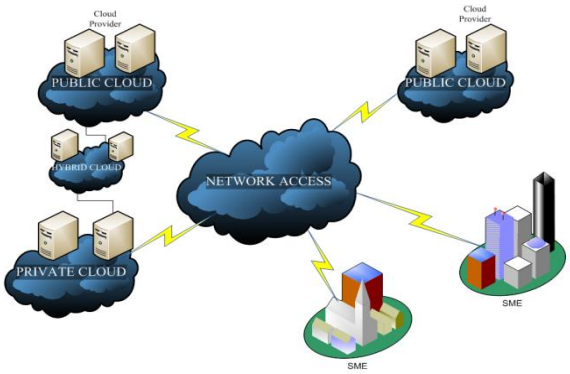

Figure 2: Network of Public, Private and Hybrid cloud

III. ARCHITECTURE OF CLOUDSIM

The basic block diagram of Cloudsim architecture [7] [8] is shown in figure 3. Primary elements are a) User level specification b) Cloudsim elements c) event driven simulation engine. User given specifications describesabout thedetailsof physical, virtual machines and application features. Cloudsim elements provide the abstraction different elements of Cloud computing environment like, physical machine, virtual machine, static placement policies. Event driven simulationengine processes the events and update the system states.

1.1 Modeling of Cloud computing environment

As mentioned previously, we can determine cloud computing as selection of resources which are interconnected and can scaled down or up dynamically. So in simulator, we can subjective cloud computing as selection of resources which are showed by datacenters made up of large number of physical machine. These physicalmachines can use any virtualization technique to support multiple virtual machines. Variety of running virtual machines and their positioning can be changed dynamically over time. Applications are presented to these virtual machines, which generates different amount of work. In cloud computing, virtualization is not an important factor,but to back up powerful provisioning virtualization is required. To absolutely design the cloudcomputing environment, different elements like physical machine, virtualizationtechnology, data center, different policies need to be modeled.

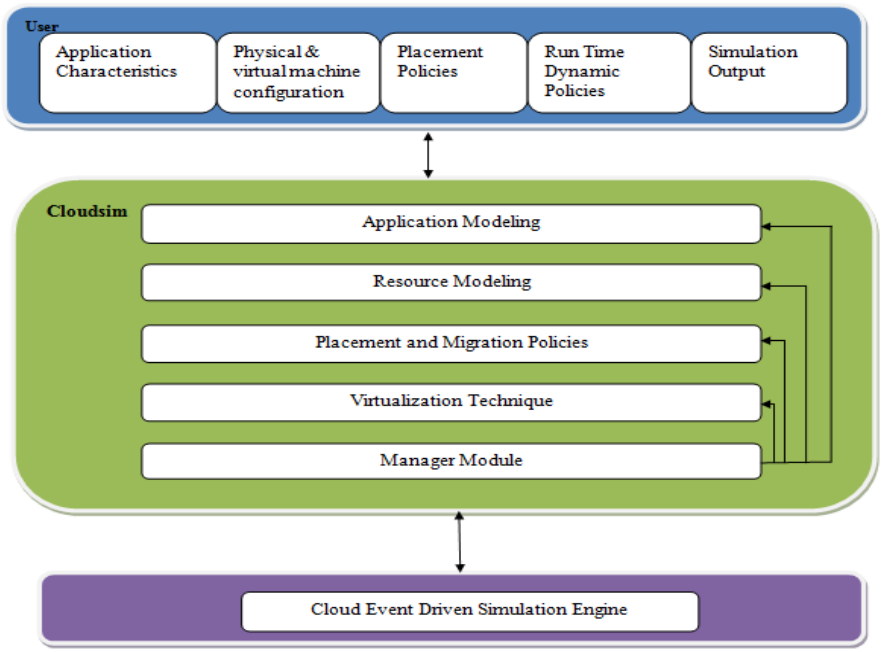

Figure 3: Architecture of Cloudsim

1.2 Modeling the Cloud Application and Workload

Application is modeled as, number of policies to be implemented, number of bytes to read/write from the hard drive, ram size required. So generally application symbolizes different amount of work of different resources over time. We can split this amount of work creation patterns into different types.

- Maximum Utilization Workload:This design symbolizes applications which are batch processin characteristics. Tries to complete as early as possible using highest possible available resource.For example automated listing of some data. This kind of application's execution timedirectly relies on the running system configuration. As the ability of the underlyingsystem improves, execution time-length decreases.
- Distribution based Workload: Here load produced by the application follows a probabilitydistribution. For example number of customers going to crickinfo web page is uniformly distributed between 200 and 300 hourly in the night. Whereas regular number of customer atdaytime is consistently allocated between 2000 and 3000.
- Random based Workload: This design uses the work load at random functions. The random function normally ranges from 0 to 1, denoting 0% and 100%.

1.3 Modeling the Datacenter

Datacenter is collection of computingresources and associated elements like interaction, cooling systems and storage space. Computing resources not only inter-connected with each one another but also with

other datacenters. Basically datacenters are the important of cloud computing environment. So to model datacenter, different elements physical machine, interaction segments, storage space and power consumption segments need to be made.

1.4 Modeling the Physical and Virtual machines

Physical Machine is collection of resources like main memory, processing units, virtualization technology running on it. Virtualization allows creating several virtual machines. Abstraction of virtual machines is same as physical devices, with several virtual machines sharingresources of physical devices. So number of virtual machines reinforced by physical device is limited the requirements. Different policies can be applied to create or eliminate virtual machines. These policies are part of virtualization technological innovation abstraction.

1.5 Manager Module

This module uses the placement and migration policies dynamically Cloudsim implements manager component as broker, which stores the record of physical and virtual machines. Broker submits the record of virtual machines to the physical machines of datacenter, to start execution.

IV. DESIGN ARTIFACTS FOR CLOUDSIM

In this section we are going to deal the fundamental classes of Cloudsim which we call it as building blocks of simulator [6]. The class diagram for Cloudsim is shown in figure 4.

- Host:This class symbolizes the characteristics of a physical machine. It encapsulates details of main memory, processing unit, and disk and network bandwidth, virtualization monitor requirements.Also details about provisioning policies for main memory, disk, cpu, network to virtual machines are specified in this class.
- VM: This class symbolizes you to virtual machine. Information such as main memory, processing power, network and disk bandwidth, hosting physical machine are exemplified. Research about the resource consumption by running applications are also showed by this class.
- Cloudlet Scheduler: This abstract class describes the plan of cloudlet (applications) performance.Depending upon the policies of cloudlets is implemented simultaneously or sequentially.CloudletSchedulerTimeShared and CloudletSchedulerSpaceShared are prolonged sessions whichallow applications to perform simultaneously and sequentiallyrespectively. CloudletScheduler-DynamicWorkload class based on CloudletSchedulerTimeShared allows powerful resourceload generation.
- Cloudlet: this class denotes the applications running on VMs. It contains the information of number of instructions which is going to be executed and the amount of data transfer to complete the task. Cloudlet class provides identification of guest VM on which it is operating and load generation model.

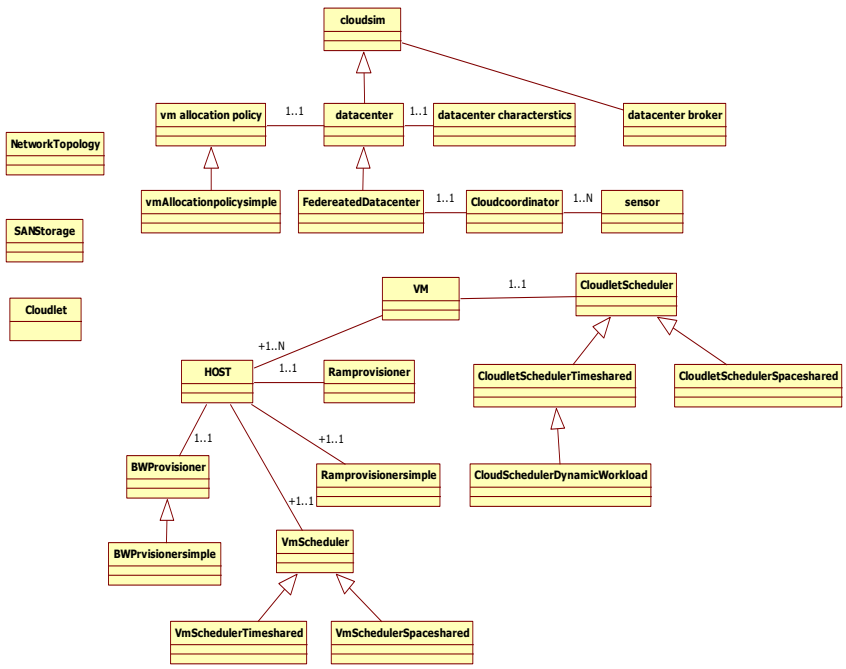

Figure 4: Simplified Class diagram for Cloudsim

1.6 Scheduling policies of Cloudsim

Scheduling is a process of allocating tasks with resources to achieve specified goal.The objective of cloud computing scheduling is to get the maximum scheduling submitted by the user, It should try to progress the overall throughput of cloud computing with Specific goals include the Quality of Service(QoS), maximum makespan, economic principles, load balance and so on.

1.6.1 Usecase Metamodel for Scheduling

The usecase diagram [11] consists of actors, usecases and their relationships. In the scheduling process the user which we consider as actor. The broker or resource scheduler we call it as usecase and different usecases will be connected to the broker. The public, private and hybrid clouds are mentioned as set of usecases [9] which is connected to the broker.

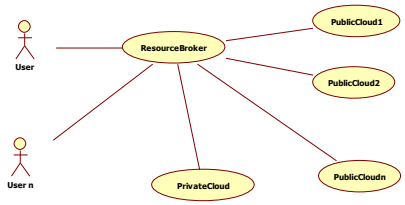

Figure 5: Use case Metamodel for scheduling policy

1.7 Analysis of First Come First Serve (FCFS) Scheduling

FCFS is a simple efficient and error free scheduling policy that saves valuable cloud resources.It uses nonpreemptive scheduling in which the tasks is automatically queued and giving out occurs according to an incoming request.FCFS develops its concept from real-life customer service. In the cloud computing environment the cloudlet contains the tasks which has to be scheduled to the resources. The resource broker will be there to dispatch the tasks into concerned resources in order. FCFS broker will be there to execute the policies of FCFS scheduling algorithm. This process is explained in detail in the class diagram which is shown in figure 6 and the sequence diagram for FCFS is shown in figure 7 [14][15].

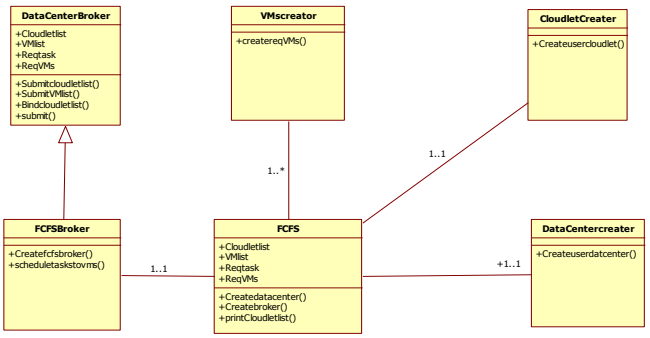

Figure 6 Design model: Class Diagram of FCFS Scheduling policy

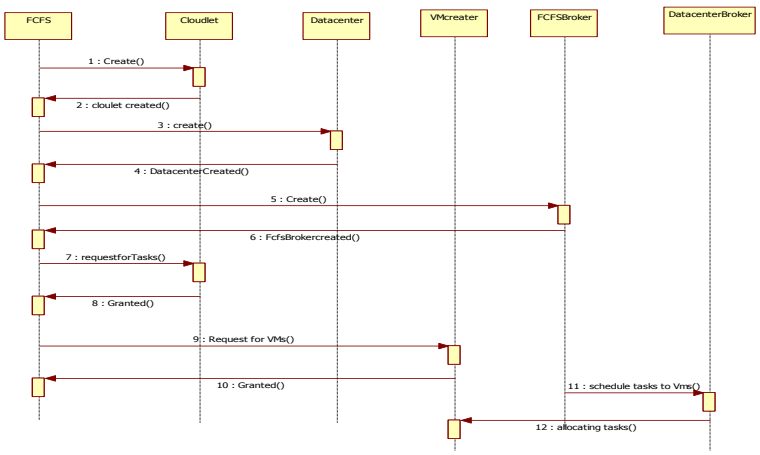

Figure 7: Display FCFS Scheduling sequence diagram

V. EXPERIMENTAL DESIGN

The simulation results setup has carried in a simulator named Cloudsim. Cloudsim is a java based work environment in which it is used for cloud application development.

1.8 Experimental setup

The FCFS is tested in cloudsim in terms of the execution time and cost, MIPS,host type, VMs, Response Time and the results are shown in Table 1 and 2.

Table 1: Simulation design in cloudsim

Host	10
VMs	10-50
RAM	512
Bandwidth	1000
Scheduling	FCFS

Table 2: Simulation results for FCFS Scheduling

Cloudlets	VMs	Execution Time (sec)	Response Time (ms)
10	15	52.4	41.3
	25	46.2	37.2
	45	41.9	32.4
20	15	64.3	52.9
	25	53.7	48.5
	45	47.2	44.6
30	15	73.5	61.4
	25	63.1	57.9
	45	57.3	53.8
40	15	81.8	72.7
	25	73.1	67.3
	45	67.2	62.8

Here we are using the constant VMs count which is taken as 15, 25 and 45. The experimental results for FCFS scheduling are shown in table 2.

1.9 Testing Phase of FCFS Scheduling

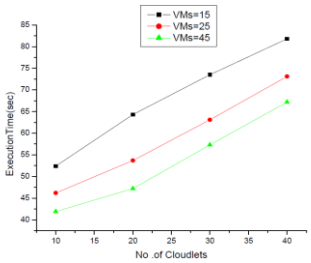

Figure 8:FCFS test results for Execution time

In figure 8 it shows the comparison graph of execution time and no. of cloudlets which is participated in the workflow scheduling. It defines that execution time is increased automatically when the no of cloud lets increases. The average response time is also shown in figure 9. It can variable when we are using random VMs.

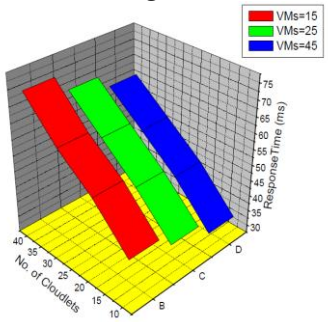

Figure 9: FCFS test results for Response Time

1.10Results and Findings

Our research theories state that using the suggested functionality guidelines for software development for Cloudsim helps to reduce development time, improves execution time, and decreases response time[10] [12]. Regarding the results showed the developers who are interested to design Cloud Computing environment they can follow this architectures to design to develop more quickly and helpful for creating error free environment[13].

VI. CONCLUSION

As outlined in this paper, integrating functionality features with a high-impact on softwarelogic is not an uncomplicated process. Some appropriate research has been designed in this route, but this is still an open question. In this paper, we set out to promote this area by suggesting functionality recommendations for software growth explaining a possible solution for integrating some of the best-known functionality features into Cloudsim. The key policies of artifacts specify the obligations that the system and its areas must meet to comply with these functionality features, making them straight implementable from design.

Initial validation results show a significant decrease in development time; designers who applied the recommendations built their design more quickly. Furthermore, implementing the recommendations also reduced testing time. Using the recommendations also assisted designers produce improved designs, especially designs with better liability allowance. Finally, the suggested recommendations assisted designers understand functionality features as not being extremely complex.

REFERENCES

[1] K. Nebe and V. Paelke, "Usability-Engineering-Requirements as a Basis for the Integration with Software Engineering," Proc. 13th Int'l Conf. Human-Computer Interaction, Part I: New Trends, pp. 652-659, 2009.

[2] L. Bass, B.E. John, and J. Kates, "Achieving Usability through Software Architecture," Technical Report CMU/SEI-2001-TR- 005, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, Penn., Mar. 2001.

[3] J.A. Cruz-Lemus et al., "Assessing the Influence of Stereotypes on the Comprehension of UML Sequence Diagrams: A Family of Experiments," Information and Software Technology, vol. 53, no. 12, pp. 1391-1403, 2011.

[4] A. Davis, O. Dieste, A. Hickey, N. Juristo, and A. Moreno, "Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review," Proc. 14th IEEE Int'l Conf. Requirements Eng., 2006.

[5] L. Carvajal, "Usability-Oriented Software Development Process," PhD thesis, Computer Faculty, Universidad Polite´cnica de Madrid, http://oa.upm.es/10599/, 2012.

[6] E. Folmer, J. van Gurp, and J. Bosch, "Software Architecture Analysis of Usability," Proc. Int'l Conf. Eng. Human Computer Interaction and Interactive Systems, pp. 38-58, 2005.

[7] Buyya R, Ranjan R, Calheiros RN. InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010)*, Busan, South Korea. Springer: Germany, 21–23 May 2010; 328–336.

[8] Avetisyan AI, Campbel R, Gupta I, Heath MT, Ko SY, Ganger GR, Kozuch MA, O'Hallaron D, Kunze M, Kwan TT, Lai K, Lyons M, Milojicic DS, Lee HY, Soh YC, Ming NK, Luke J-Y, Namgoong H. Open cirrus: A global cloud computing testbed. *IEEE Computer* 2010; 43(4):35–43.

[9] A. Vescan and C. Grosan, "A Hybrid Evolutionary Multiobjective Approach for the Component Selection Problem," Proc. Third Int'l Workshop Hybrid Artificial Intelligence Systems, pp. 164-171, 2008.

[10] M. Tang and L. Ai, "A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem in Web Service Composition," Proc. IEEE Congress Evolutionary Computation, pp. 1-8, 2010.

[11] ISO IEC 9126-1:2001, "Software Engineering-Product Quality- Part 1-Quality Model," 2001.

[12] B. John, L. Bass, E. Golden, and P. Stoll, "A Responsibility-Based Pattern Language for Usability-Supporting Architectural Patterns," Proc. First ACM SIGCHI Symp. Eng. Interactive Computing Systems, 2009.

[13] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato, "How Developers' Experience and Ability Influence Web Application Comprehension Tasks Supported by UML Stereotypes: A Series of Four Experiments," IEEE Trans. Software Eng., vol. 36, no. 1, pp. 96-118, Jan./Feb. 2010.

[14] J.A. Cruz-Lemus et al., "Assessing the Influence of Stereotypes on the Comprehension of UML Sequence Diagrams: A Family of Experiments," Information and Software Technology, vol. 53, no. 12, pp. 1391-1403, 2011.

[15] S. Alpert et al., HCI Patterns, http://www.hcipatterns.org, 2012.