Observability and Controllability of Autonomic Computing Systems for Composed Web Services

L. Checiu*, B. Solomon*, D. Ionescu*, M. Litoiu** and G. Iszlai***
* University of Ottawa, Ottawa, Canada
** York University, Toronto, Canada
*** IBM Center for Advanced Studies, Toronto, Canada
lcheciu@ncct.uottawa.ca, bsolomon@ncct.uottawa.ca, ionescu@ncct.uottawa.ca, mlitoiu@yorku.ca, giszlai@ca.ibm.com

Abstract— **Autonomic Computing is a research area whose aim is to embed "intelligent algorithms" in the IT infrastructure management software such that it can adapt to changes in regards to the configuration, provisioning, external attacks, and resource utilization variations at run time. It is therefore, almost natural to consider this IT infrastructure control software framework as being designed upon methods and technologies used for the design of control systems. In this paper the control system design methodology is extended to the analysis of the intrinsic properties of the autonomic system itself. Thus the controllability and observability properties of the computing process itself are defined and examined in more details. These properties are also investigated for the case of cloud services where the serial and parallel composition of these services is considered. These cloud based services are connected through cooperation protocols that define a global process dynamic. Web services are modeled as scheduled computational processes waiting in a queue to cooperate in delivering the service. This paper proposes an input-state-output mathematical model for the autonomic computing model of cloud based services and the observability and controllability are further analyzed on the above models. As an example a Kalman based control is applied to such processes and the general architecture and some simulation results are given.**

I. INTRODUCTION

A service able to self-manage should monitor its state and its execution environment, analyze them, and make decisions to control its own behavior. Usually the behavior is characterized by QoS parameters such as the response time of the application a client request, the corresponding CPU utilization, etc. If the service is in or moving towards an undesired state, then a corrective plan should be devised and executed such that the service is brought back to a desirable state. In many cases, such as self-optimization, this desirable state should also be an optimal state for the performance of the service.

Research approaches into the development of self-managed systems have been using various techniques ranging from artificial intelligence to model based, cost function and policy planning techniques, as well as goal model methodologies in order to devise self-management solutions. Artificial intelligence and machine learning methodologies [14] rely on obtaining data from a running system and then on using it to build a classifier or a clustering system. While using a machine learning approach the system is attempting to induce the full autonomic system behavior based on the way the system behaved in the past. In cases with big perturbations in the behaviour of the system, a machine learning methodology will not respond correctly to that disturbance, unless it has already encountered that behaviour in the past. As such, a machine learning mechanism is only as good and complete as the training data it gathered.

According to the model based autonomic computing [16], decisions in autonomic computing systems rely on the explicit model of the underlying software. Several reference control loops have been designed to control single or collections of services by either tuning parameters, reallocating the load or provisioning new instances of the service. The main control approaches based on the automatic control theory are: (a) linearized feedback control and (b) performance model based adaptive control. The performance control of a Web server using linearized feedback control is presented in [18]. The authors of [19] used the linearized feedback control approach on a linearized dynamic model of the controlled computing system.

The performance model based adaptive control is described in [17] and its schematic representation is depicted in Figure 1:

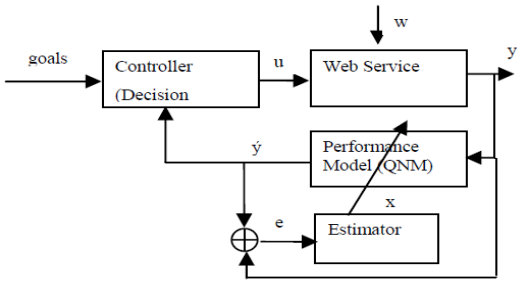

Figure 1. Performance Model Based Adaptive Control

The model is continuously updated by an Estimator which computes the parameters of the system by minimizing the error between the output of the service (Web Service) y and the output of the model. The Estimator can be represented by a Kalman filter [15] and usually works in a feedback loop with the model. The controller generates the command u which will bring the controlled system (Web Service) output closer to the goals in the presence of perturbations w. The command *u* can be expressed as a change in the memory allocation, threading level or other software resources. A perturbation can be a change in the usage patterns.

This paper presents the autonomic management of web services which call each other or receive requests in parallel to deliver their part to a more complex service.

978-1-4244-9109-4/11/$26.00 ©2011 IEEE

The contributions of this paper rely on the analysis of web services from a control engineering perspective taking into account their autonomic computing models described in [15]. Observability and controllability concepts are used in this analysis in order to draw the right conclusions with respect to the automatic control approach. Furthermore, these important concepts in the control theory are also used during the analysis of the composed web services systems.

The remainder of the paper is organized as follows: section II describes the performance model for a web service along with its observability and controllability evaluation. Section III describes the composition laws for adaptive services. Section IV presents the autonomic control of web services while section V shows simulation results using the above approach. Finally section VI presents the conclusions.

II. Models For Model Based Control Of Web Services

Figure 2 describes a self-optimization solution for web services. The autonomic computing loop controls and also optimizes the web service operation [10]. For example a web service which is used for airplane bookings would be required to provide a low response time, as large response times could lead to lost sales as users move to other services.

The core of the autonomic computing loop is built taking into account an extended linearized feedback architecture which uses an explicit model of the system whose state vector is estimated using a Kalman filter technique [13]. The web service is modeled as a system responding to users' requests with different response times. The model is non-linear, while the time of the model was considered discrete. Optimal control algorithms are not known to be tolerant to changes in the control system model or in the environment and as such the system has to cope with ever changing model parameters and even with unknown and un-modeled states and disturbances. The goal of the controller is to optimize the hardware usage while at the same time maintain the service quality as required by the Service Level Agreement (SLA) which expresses it as the maximum allowable response time.

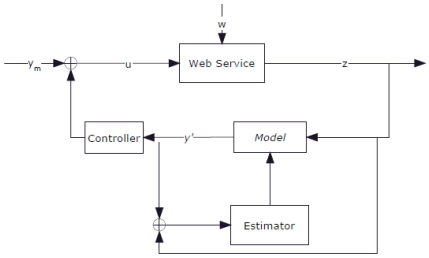

Figure 2. Model Based Service Control

Based on the assumed model a control law is built while the state of the process is estimated using the data produced by the process and of the model. In what follows, the dynamic processes taking place in a server will be analyzed and a model for these dynamic processes will then be built. It is considered that a server receives a request from a client for a service, performs some processing, creates one or more data store requests and finally builds a response which it sends back to the client, or forwards to the next processing web service which is required to cooperate on building the final value of the server response to the request.

A server uses a thread pool which contains a number of threads. Each of these threads can process requests. Once all the threads in the pool are being used, new requests are added in a queue which uses a First Come First Served (FCFS) policy while they await a request under process to complete and a thread in the pool to become available. After the request has been processed, the corresponding reply is added to a queue in order to be sent back to the client. Figure 3 shows the structure of the considered model.

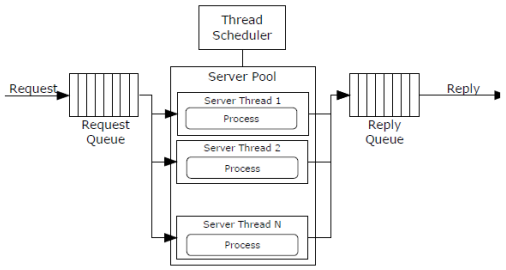

Figure 3. Server Request Model

Considering the following notations:

- A is the system's matrix with elements $a_{i,j} \in R$
- B is the command matrix with elements $b_{i,j} \in R$
- C is the output transformation matrix with elements $c_{i,j} \in R$
- *cpuLoad(t)* is a function representing the CPU load of the system, at the time t, with *cpuLoad(t)*$\in R$ and 0 < *cpuLoad(t)* < 100
- *rt(t)* is the response time expressed in milliseconds, with *rt(t)*$\in R$
- *q(t)* is the size of the buffer queue at the time t
- *u(t)* represents the arrival rate at the time t.
- a, b, c, d are constant values for a system, with a,b,c,d $\in R$

The web service model, with the assumptions made in this paper, comprises the following process variables: the CPU load, the response time, the buffer size - the buffer being the one in which all queued tasks are stored, the client requests represented as the arrival rate which is the stimulus/input of the computing processes and the response of the application server which is the output of the computing process. According to [15], the web service model can be expressed by the following equations:

$$\dot{cpuLoad}(t) = a_{1,1} * cpuLoad(t) + b_{1,1} * u(t) \quad (1)$$

$$\dot{rt}(t) = a_{2,1} * cpuLoad(t) + d * q(t) * rt(t) \quad (2)$$

$$\dot{q}(t) = -a_{3,1} * cpuLoad(t) + q(t) + b_{3,1} * u(t) \quad (3)$$

The trend of the CPU utilization depends on the current CPU utilization and the arrival rate. The response time trend depends on the current CPU utilization, the response time and the buffer queue size. The trend of the buffer queue length depends on the CPU utilization and the current buffer queue along with the arrival rate. At high CPU utilization values the buffer is emptied faster.

Equation (2) is not linear; therefore it needs to be linearized around a $t_0 \in R$ using the Jacobian of $f(cpuLoad(t), rt(t), q(t)) = a_{2,1}*cpuLoad(t)+d*q(t)*rt(t)$ as follows:

$$\dot{rt}(t) = a_{2,1}*cpuLoad(t0)+d*q(t0)*rt(t0) + \left[a_{2,1} \quad d \cdot q(t_0) \quad d \cdot rt(t_0)\right] \cdot \begin{bmatrix} cpuLoad(t) - cpuLoad(t_0) \\ rt(t) - rt(t_0) \\ q(t) - q(t_0) \end{bmatrix} \quad (4)$$

The following notations will be used in order to simplify and homogenize the notations:

$$a_{2,2} = d*q(t_0)$$

$$a_{2,3} = d*rt(t_0)$$

$$b_{2,2} = -d*rt(t_0)*q(t_0)$$

The equations (1),(2) and (3) can be written as follows:

$$\dot{cpuLoad}(t) = a_{1,1} * cpuLoad(t) + b_{1,1} * u(t) \quad (5)$$

$$\dot{rt}(t) = a_{2,1} * cpuLoad(t) + a_{2,2} * rt(t) + a_{2,3} * q(t) + b_{2,2} * u_{-1}(t) \quad (6)$$

$$\dot{q}(t) = -a_{3,1} * cpuLoad(t) + q(t) + b_{3,1} * u(t) \quad (7)$$

where $u_{-1}(t)$ is the step function.

Obviously, the measured data of our system relates to the *cpuLoad* and *rt* therefore the system can be modeled from a control engineering perspective as follows:

$$\begin{bmatrix} \dot{cpuLoad}(t) \\ \dot{rt}(t) \\ \dot{q}(t) \end{bmatrix} = \begin{bmatrix} a_{1,1} & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} \\ -a_{3,1} & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cpuLoad(t) \\ rt(t) \\ q(t) \end{bmatrix} + \begin{bmatrix} b_{1,1} & 0 \\ 0 & b_{2,2} \\ b_{3,1} & 0 \end{bmatrix} \cdot \begin{bmatrix} u(t) \\ u_{-1}(t) \end{bmatrix} \quad (8)$$

$$\begin{bmatrix} cpuLoad(t) \\ rt(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} cpuLoad(t) \\ rt(t) \\ q(t) \end{bmatrix} \quad (9)$$

The system is described by the following model:

$$\dot{x}(t) = A \cdot x(t) + B \cdot u_c(t) \quad (10)$$
$$y(t) = C \cdot x(t)$$

A. Observability and Controllability

The concept of observability shows the possibility of finding, or at least estimating the internal state variables of a system from input and output variables. The concept of controllability (or state controllability) describes the possibility of driving the system to a desired state, i.e. to bring its internal variables to certain values (*cpuLoad* at 10%, response time at *rt* =5s, for example, and buffer queue at *q*=1000) in finite time using only the input variable which is the arrival rate *u* of clients requests.

The observability property of a linear system is evaluated by calculating the rank of the observability matrix *O*, defined as follows for the autonomic computing model used in this paper:

$$O = \begin{bmatrix} C \\ C \cdot A \\ C \cdot A^2 \end{bmatrix} \quad (11)$$

The system is completely observable if and only if the rank of the observability matrix is equal to the number of state variables, in our concrete case *cpuLoad*, *rt* and *q*. Using Symbolic Math Toolbox from Matlab®: *rank(O) = 3*, therefore the system is observable. The past evolution of the system can always be known.

The controllability property of a system can be as well established using the controllability matrix *S* defined as follows:

$$S = \begin{bmatrix} B & A \cdot B & A^2 \cdot B \end{bmatrix} \quad (12)$$

The system is completely state controllable if and only if the rank of the controllability matrix is equal to the number of the state variables: *cpuLoad*, *rt*, and *q*. The rank of controllability matrix S is computed using Symbolic Math Toolbox from Matlab®: *rank(S) = 3* therefore our system is completely state controllable.

III. COMPOSITION OF WEB SERVICES

This section describes several possible composition laws for autonomic services. As general assumption, it is considered that web services can be composed in two basic ways, namely a serial composition when a web service master calls at its turn another web service, called slave, to solve the clients' requests, and a parallel composition when a request is distributed to many web services in parallel in order to solve a client request.

A. Serial Composition of Web Services

In the serial services approach, each service computes a value added result based on the output of the previous service. The same example as in [15] is now considered, a service for a travel agent, whose goal is to make travel and hotel reservations for the customers. The service can be split into two sequential services: one which does the hotel reservations, and upon successful hotel reservation a second which performs the travel reservations based on the hotel reservation dates. Figure 4 presents the generic example of serial web services. A client makes a request to a service, and the output of that service is forwarded to a second service which computes its output based on the previous service's output and sends the final result to the client.

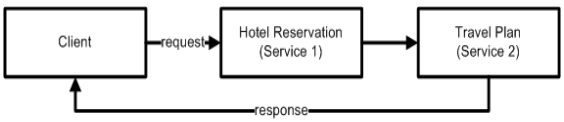

Figure 4. Serial Web Services

Each web service system ("Hotel Reservation" and "Travel Plan") can be described by the following equations:

$$\dot{x}_1(t) = A_1 \cdot x_1(t) + B_1 \cdot u_c(t) \quad (13)$$
$$y_1(t) = C_1 \cdot x_1(t)$$

$$\dot{x}_2(t) = A_2 \cdot x_2(t) + B_2 \cdot u_{c2}(t) \quad (14)$$
$$y(t) = C_2 \cdot x_2(t)$$

In a serial configuration $u_{c2}(t)$–the input to the "Travel Plan" web service – is actually the output of the "Hotel Reservation" web service, $y_1(t)$. Therefore the equations (16) and (17) can be considered in one system:

$$\begin{aligned} \dot{x}_1(t) &= A_1 \cdot x_1(t) + B_1 \cdot u_c(t) \\ \dot{x}_2(t) &= A_2 \cdot x_2(t) + B_2 \cdot (C_1 \cdot x_1) \\ y(t) &= C_2 \cdot x_2(t) \end{aligned} \quad (15)$$

The resulting system of the two web services connected in a serial configuration can be described by the following equations:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ B_2 \cdot C_1 & A_2 \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} \cdot [u_c(t) \quad 0] \quad (16)$$

$$y(t) = [0 \quad C_2] \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

The output of the composed system (*y(t)*) depends only on the state vector of the "Travel Plan" web service ($x_2(t)$). The response time of the "Hotel Reservation" web service ($rt_1(t)$) is practically hidden to an observer of the composed system. The state of the "Hotel Reservation" web service ($x_1(t)$) cannot be estimated from the output of the composed system and furthermore the global state $[x_1(t), x_2(t)]^T$ of the composed system cannot be entirely estimated from the output of the composed system (*y(t)*). Therefore – intuitively for now – the composed system is not observable.

On the other hand, the arrival rate at the "Hotel Reservation" web service has no impact on either the CPU utilization, response time or buffer queue size of the "Travel Plan" web service. This fact gives the indication that the composed system is not controllable.

1) Observability and Controllability of Serial Composition of Web Services

The observability and controllability matrices of the composed system must have the rank equal to the number of all the state variables ($cpuLoad_1$, rt_1, q_1, $cpuLoad_2$, rt_2, q_2) for the resulting system to be observable and controllable, respectively. The observability matrix of the serial composition (*O_Serial*) is constructed as described in Section II.A. It cannot be presented explicitly using the elements of *A1* and *A2* matrices because of the space limitations. With the help of Symbolic Math Toolbox from Matlab® the rank of this matrix can be computed and its value is *rank(O_Serial)*=5. The resulting system of the serial composition is not observable. In the same manner, the rank of the controllability matrix (*S_Serial*) is *rank(S_Serial)=5*, therefore the resulting system of the serial composition is not completely state controllable.

The controller should not regard the resulting system as a black box; it should also have access to the internal connection of the two web services in order to preserve the observability and controllability properties for the resulting system.

A proper controller should consider the composed system with both two inputs and the two outputs in order to retrieve the proper information about the system. In a decentralized control approach of serial web services [15] every web service has its own controller with the proper access to the right input and output variables and the observability and controllability of the whole system is preserved. Even in the case of the centralized control approach of serial web services [15] the global controller measures the required data from each of the individual services, computes the needed changes to be made to the services and takes appropriate actions. In this manner the observability and the controllability of the resulting system is also preserved.

B. Parallel Composition of Web Services

In the parallel service approach, requests are sent to multiple services and each processes the request independent of all other services. The final result is obtained by combining the responses from the services.

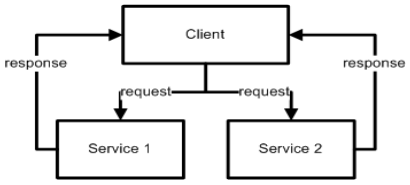

Figure 5. Parallel Web Services Composition

Considering the same travel agent as an example, this travel agent sends out the request for reservation to several hotels. Any two web services connected in a parallel configuration are described by the following equations:

$$\begin{aligned} \dot{x}_1(t) &= A_1 \cdot x_1(t) + B_1 \cdot u_c(t) \\ y_1(t) &= C_1 \cdot x_1(t) \end{aligned} \quad (17)$$

$$\begin{aligned} \dot{x}_2(t) &= A_2 \cdot x_2(t) + B_2 \cdot u_c(t) \\ y_2(t) &= C_2 \cdot x_2(t) \end{aligned} \quad (18)$$

The composition of the two web services connected in a parallel configuration is represented in this system of equations:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \cdot [u_c(t)] \quad (19)$$

$$y(t) = [C_1 \quad C_2] \cdot \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

The parallel configuration implies that the global output is a combination of all the outputs hence $y(t)=y_1(t)+y_2(t)$.

The CPU utilization for each of the web services cannot be estimated from (19) having only the global CPU utilization value. This is an indication the composed system in a parallel configuration is not observable. Furthermore, the two web services are independent of each other therefore their state variables (CPU utilization, response time and buffer queue length) are independent of their counterparts in each web service. This means that it is almost impossible to drive the CPU utilization of the first web service to 50% and the second CPU utilization to 30% for instance using the same arrival rate. Intuitively this is an indication that the composed system in a parallel configuration is not controllable.

1) Observability and Controllability of Parallel Composition of Web Services

The observability and controllability matrices of the composed system must have the rank equal to the number of all the state variables ($cpuLoad_1$, rt_1, q_1, $cpuLoad_2$, rt_2, q_2) for the resulting system to be observable and controllable, respectively.

The observability matrix of the parallel composition (*O_Parallel*) is constructed as presented in section II.A. It cannot be presented explicitly using the elements of *A1* and *A2* matrices because of the space limitations. Its rank is lower than 6 (*rank(O_Parallel)=5*) therefore the resulting system is not completely observable. The rank of the controllability matrix has the same value as the rank of

observability matrix (*rank(C_Parallel)=5*) therefore the resulting system of the parallel composition is not completely state controllable.

The controller should also have access to the individual outputs of the two web services in order to preserve the observability and controllability properties for the resulting system. In the same manner as in the serial configuration, the decentralized approach for parallel service control approach [15] has a controller for each service.

IV. WEB SERVICE CONTROL USING AUTONOMIC COMPUTING CONTROL LOOPS

The controllability and observability results obtained above dictate the architecture to be used in the case of composed web services when these services are offered on the cloud (SaaS). This architecture should be structured such that the measured values are always available for the estimator in order to perform a proper control.

Extended Kalman filters are used in this paper to predict the state estimated value such that assuming a stochastic dispersion of the estimated values the error is minimal. In this section we consider the discrete case of the system defined by (7), (8) and (9).

Considering the discrete Kalman filter and its notations presented in [13]:

$$x(k) = A \cdot x(k-1) + B \cdot u(k-1) + w(k-1)$$
$$z(k) = H \cdot x(k) + v(k) \quad (20)$$

The measured data is represented in this case by the CPU utilization (*cpuLoad*) and the response time (*rt*), therefore:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The Kalman gain K_k at the step k is computed as follows:

$$K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R)^{-1} \quad (21)$$

where P_k^- is the *a priori* estimate error covariance and R is the measurement noise covariance.

A. Kalman Filter for Composed Services

This section presents the Kalman filter for both serial and parallel composition of web services in centralized control architecture. The composition rules are now applied in the design of the controller for each case of serial and parallel composition.

1) Kalman Filter for the Serial Composition of Web Services

Figure 6 describes the centralized control of the serial composition of "Hotel Reservation" and "Travel plan" web services:

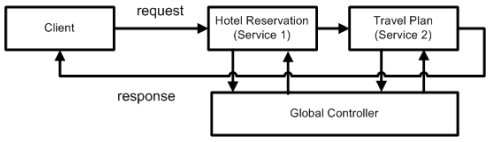

Figure 6. Serial Web Services Centralized Control

Equation (21) gives the model of this serial composition. Based on this, Kalman gain K_k can be computed according to equation (28) and considering H=[0 C_2] where C_2 is the output transformation matrix of the "Travel Plan" web service.

2) Kalman Filter for the Parallel Composition of Web Services

Figure 7 describes the centralized control of parallel composition of web services.

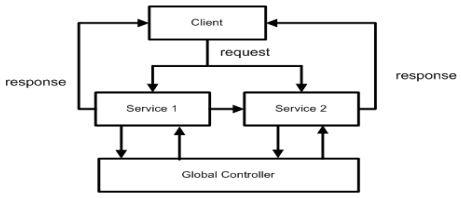

Figure 7. Parallel Web Services Centralized Control

V. SIMULATIONS RESULTS

All the simulations have been performed on a computer based on the following configuration: Intel Core Duo 2.8 GHz, 4GB, Windows 7 32bit operating system. The software tool used was Matlab® version 7.7.0.471.

A. Single Web Service Control Simulation Results

The arrival rate (*u(k)*) has been considered as a uniform distributed random variable. The following graphic represents the evolution of the CPU utilization (cpuLoad) over 50 steps of iteration:

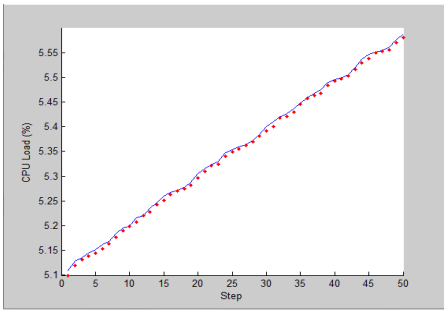

Figure 8. CPU Utilization – Single Web Service Control

In this case, the dots represent the Kalman filter prediction and the continuous line represents the actual value of the CPU utilization. The CPU utilization increases continuously because the web service receives a request for every step of simulation.

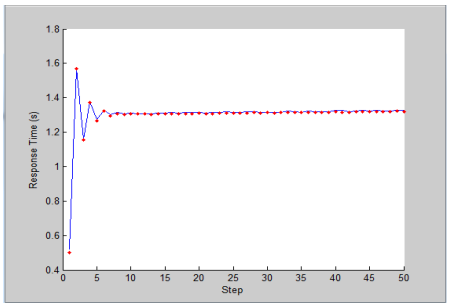

Figure 9. Response time in a single Web Service control configuration

Similarly in Figure 9, the dots represent the Kalman filter prediction and the continuous line represents the actual value of the response time.

B. *Control of Two Web Services in a Series Configuration*

For this case, the arrival rate (u(k)) at the service 1 is still considered as a uniform distributed random variable.

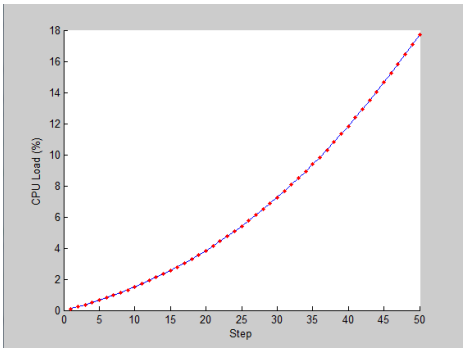

Figure 10. CPU utilization 2 in a centralized control of Series Configuration

The shape of the CPU utilization for the second web service graph in Figure 10 is different than the one for a single Figure 8 having a slower increase at the beginning and then the increase becomes steeper. The dots represent the Kalman filter prediction and the continuous line represents the actual value of CPU utilization 2.

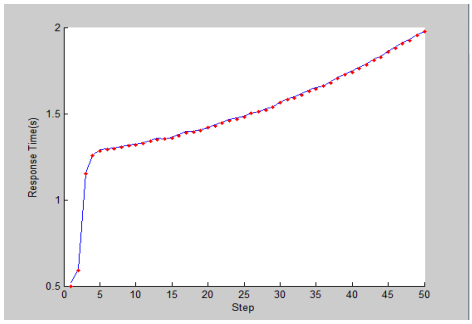

Figure 11. Response time for the second web service in a centralized control of Series Configuration

VI. CONCLUSIONS

The focus of the investigation was on the analysis of the autonomic computing models for web services and on modeling the composition of web services in two possible cases: the serial and the parallel models. These models were studied from a control engineering perspective, i.e. their observability and state controllability. The resulting system from either a serial or a parallel composition is not completely observable and state controllable even though the consisting sub-systems are completely observable and state controllable. Future work will focus on the control and the control properties for the deployment of web services on cloud based distributions.

REFERENCES

[1] W. H. F. Aly and H. Lutfiyya. Feedback control theoretic technique for data centers. In ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems, page 38, Washington, DC, USA, 2007. IEEE Computer Society.

[2] D. Ardagna, M. Trubian, and L. Zhang. SLA based resource allocation policies in autonomic environments. Journal of Parallel and Distributed Computing, 67(3):259{270, 2007

[3] L. BenMohamed and S. Meerkov. Feedback control of congestion in store-and-forward datagram networks: the case of a single congested node. In Decision and Control, 1992., Proceedings of the 31st IEEE Conference on, volume 1, pages 991{996, 1992.

[4] G. B. Dantzig. Linear Programming and Extensions. Princeton University Press, Princeton, 1998.

[5] V. Guens and G. Bastin. Optimal adaptive feedback control of a network buffer. In American Control Conference, 2005. Proceedings of the 2005, volume 3, pages 1835{1840, June 2005.

[6] D. Ionescu, B. Solomon, M. Litoiu, and M. Mihaescu. A robust autonomic computing architecture for server virtualization. In International Conference on Intelligent Engineering Systems, 2008. INES 2008, pages 173{180, Feb. 2008.

[7] D. Ionescu, B. Solomon, M. Litoiu, and M. Mihaescu. Web service distributed management framework for autonomic server virtualization. In SVM '08: Proceeding of the 2008 Systems and Virtualization Management. Standards and New Technologies, pages 61-71. Springer Berlin Heidelberg, 2008.

[8] K. Kontogiannis, G. A. Lewis, D. B. Smith, M. Litoiu, H. Muller, S. Schuster, and E. Stroulia. The landscape of service-oriented systems: A research perspective. In SDSOA '07: Proceedings of the International Workshop on Systems Development in SOA Environments, page 1, Washington, DC, USA, 2007. IEEE Computer Society.

[9] M. Litoiu. A performance analysis method for autonomic computing systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2(1):1-27, 2007.

[10] M. Litoiu, M. Mihaescu, D. Ionescu, and B. Solomon. Scalable adaptive web services. In SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments, pages 47-52, New York, NY, USA, 2008. ACM.

[11] J. Rao and X. Su. A survey of automated web service composition methods. Semantic Web Services and Web Process Composition, pages 43-54, 2005.

[12] B. Solomon, D. Ionescu, M. Litoiu, and M. Mihaescu. Towards a real-time reference architecture for autonomic systems. In SEAMS '07: Proceedings of the 2007 International Workshop on Software Engineering for Adaptive and Self-Managing Systems, pages 1-10, 2007.

[13] G. Welch and G. Bishop. An introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, July 2006. [Accessed: May 2009].

[14] R. Zhang and A. J. Bivens. Comparing the use of bayesian networks and neural networks in response time modeling for service-oriented systems. In SOCP '07: Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches, pages 67-74, New York, NY, USA, 2007. ACM.

[15] B. Solomon, D. Ionescu, M. Litoiu, G. Iszlai Autonomic Control of Composed Cloud Based Services. In SEAMS '10 Proceedings of the 2010 International Workshop on Software Engineering for Adaptive and Self-Managing Systems.

[16] Litoiu M., Woodside M., Zheng T. Hierarchical Modelbased Autonomic Control of Software Systems. Proceedings of ACM ICSE Workshop on Design and Evolution of Autonomic Software, St. Louis, May, 2005

[17] Litoiu M., Zheng T., Woodside M. Service System Resource Management Based on a Tracked Layered Performance Model, Proceedings of IEEE International Conference on Autonomic Computing, Dublin, Ireland, June 2006

[18] Abdelzaher, T., Shin, K.J and Bhatti, N., Performance Guarantees for Web Server End-Systems: A Control Theoretical Approach. IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 1, Jan 2002.

[19] Diao, Y., Lui, X., Froehlich, S., Hellerstein, J.L., Parekh, S.and Sha, L. On-Line Response Time Optimization of AnApache Web Server. International Workshop on Quality of Service, 2003.

[20] Zheng T., Yang J., Woodside M., Litoiu M., Islzai G. Tracking Time-Varying Parameters in Software Systems with Extended Kalman Filters, Proceedings of CASCON 05, ACM Digital Library, October 17-20, 2005