Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

E- Governance Monitoring

Laxmi Narayan Padhy
Department of Computer Science & Engg., KIST
Bhubaneswar, India

Abstract— In the era of increasing acceptance of E-Governance systems in different government sectors over the globe, monitoring Citizen Charter is a vital step to increase faith between government and citizens. Citizen charter is a government document setting out standards of service for public and private sector bodies. It contains rules for specifying quality of services, functional and business specification of an organization. Only defining citizen charter without monitoring the satisfiability of the defined rules is of less impact. As government systems and its working processes are most of the time dynamic and volatile in nature, writing monitoring programs for the E-government systems at the design time is of no use. Therefore dynamic auto generation of citizen charter monitoring programs is of vital importance. Automating the monitor program creation process for dynamic systems and dynamically changing monitoring requirements is a real challenge. This paper proposes a formal specification of the citizen charter as well as an auto monitor generation mechanism from the formally specified citizen charter rules .

Keywords— E-Governance, Service Based Systems Monitoring, Formal Language, Compiler

I. INTRODUCTION

Recently many government sectors across the world started their own initiatives to use computing system to offer new electronic channel by means of which citizens and government ministries can interact with one another through automating public services which can allow citizens to more easily access all kind of government services in a automated way that are un- constrained by the locations and schedules; thereby improving government efficiency and effectiveness. This initiative to use Information and communication Technology systems (ICT) to automate government services is called E-Governance.

Research on E-Government spans over many interesting issues covering all the phases of government services [1], [2],[8]. These thesis works propose yet a very interesting research topic: Citizen Charter, its one type of agreement between citizens and government mentioning all the terms and conditions of the service that provided by the organization. Citizen Charter is a government document setting out standards of service for public and private sector bodies. It represents the commitment of the government organizations towards standard, quality and time frame of service delivery.

In general in the countries like India most of the time citizens are dissatisfied by the quality of services they are getting by different government organizations. But, Citizen Charter is one type of service quality assurance and conditions document provided by different government sectors to the citizens. The basic objective of the Citizen Charter is to empower the citizen in relation to public service delivery.

Even if some of the government sectors have their citizen charter, still the citizens are dissatisfied about the quality of services of different government services. The main reason of this is the lack of proper accountability set for the violation of the citizen charter. This is happening due to the absence of service provisioning time proper monitoring of the satisfiability of the citizen charter. Due to the vast nature of the service consumers manual monitoring of citizen charter is a real challenge.

In this work we try to design an automatic monitoring framework for the citizen charter for e-government systems. The ability to set up and monitor citizen charters has been increasingly recognized as one of the essential preconditions for the deployment of service based systems [10]. Citizen charter is a set of terms under which a service is to be offered and the quality aspects that it should satisfy under these terms. The ability to monitor the compliance of a set of agreed upon services at runtime with respect to a charter is crucial both from the point of view of the service consumer and the service producer.

For citizens, monitoring charter is necessary in order to check if the terms of an agreement are satisfied, identify the consequences in the event of violation of certain terms in the charter, and claim penalty for any violations. For government organizations, monitoring of the provisions of a service against the terms specified in a charter is necessary in order to gather evidence regarding the agreed upon service provisions in case of any dispute with a citizen while availing a service. Besides this monitoring is necessary in order to identify problems with the delivery of certain services and take appropriate action before an agreement is violated.

Administrators can make use of the monitoring technology to advertise and offer their service capabilities while citizens can formalize their service level objectives through charter specification. It is in the interest of both the

parties to create and operate citizen charter monitoring with minimum human interaction on one hand and to agree upon legally binding electronic contracts and monitor the contracts on the other hand [7]. Several SLA monitoring frameworks have been pro- posed to cope with the SLA guarantee term specification and monitoring (see e.g. [5], [11], [4], [9], [13], [14], [12]). A novel solution to the problem of citizen charter specification and automatic monitoring of guarantee terms has been proposed in this paper.

This paper proposes an extension of MSL for citizen charter specification. It also describes a framework that we have developed to support the monitoring of guarantee terms (functional, quality of service requirements and business assumptions) which are specified as part of service level agreements. This framework can monitor the provision of services to service based systems i.e., a system that uses one or more external web-services which are coordinated by a composition process. The proposed framework is event based and non- intrusive in the sense that events are collected during the operation of a service based system without interfering in the composition process. The monitoring mechanism is also independent of service execution platform.

The proposed framework uses an extension of MSL [3] which has been have defined to monitor citizen charter. This extension supports description of the functional and quality requirements as well as business assumptions for the services. A language based on XML schema (XMSL) has been developed to specify service guarantee terms as events. In turn XMSL is based on a temporal logic based language called MSL [14]. Specification of service guarantee terms in XMSL can be developed independent of WS-Agreement. The events which are used in the specification of the service guarantee terms in an agreement can only be observed from the service based systems environment (i.e, business layer, service layer and infrastructure layer of the SBS) during the execution of the composition process. The choice of MSL as the language for specifying service guarantee terms is due to its expressiveness as a formal language which allows specification of temporal constraints and the ability to monitor an agreement using well defined reasoning process in the form of inference rules written in first-order logic.

Further, our monitoring framework has been designed with the objective to support non-intrusive, service composition platform independent monitoring of service level agreements as well as service based systems functional and non-functional properties. The framework that we discuss in this paper was originally developed to support the monitoring of functional and non-functional property requirements of web-service based systems and the main formal characteristics of the original form of the framework can be found in [14]. In this paper our focus is to show the usability of this framework in monitoring citizen charter and to introduce extensions to handle additional requirements of service based systems.

The rest of the paper is structured as follows. Section ??, describes the state of the art in service based systems monitoring technologies and existing research approaches. An example scenario is presented in Section ??. Section ?? introduces a formal language for citizen charter specification. Section ?? gives a complete description of the citizen charter monitoring framework. Finally we conclude our work with a mention of our future research direction in Section ??.

II. EXAMPLE SCENARIO: ESI SYSTEM

Here we present a e-government system called Employee's State Insurance Corporation (ESI) which is referred in the rest of our report to explain the concept of monitoring required in the ESI System.

ESI is a self-financing, social security and health insurance scheme for Indian workers. For all the employees earning Rs15000 or less per month as wages, the employer contributes 4.75% and employee contributes 1.75% making a total share of 6.5%. The employee who wants to get the facilities of ESI, needs to fulfil the declaration form and provide the family photograph to the employer. The forms are signed and stamped by the company and sent to the ESI office. The temporary card is given in a week. The permanent card is issued and has to be renewed within a certain interval. After the employee gets the permanent card, he/she can avail the ESI service. The ESI can be viewed as a service based system which involves a service composition process that interacts with the following web services.

- Doctor services (DocS): is provided by ESI system in which the doctors were unable to provide treatment for a particular disease can refer to the different hospitals where the corresponding disease is available.
- ESI Control Service (ESIControlS): is the interface between Doc S and ESIInfoS which is meant for controlling the data.
- ESI information Service (ESIInfoS): contains all the information related to the patient, hospital and ESI.
- Hospital Service (HospS): provides the user friendly frontend for patients to interact.

In this example scenario, the message flows between ESI and the hospital to be referred. The explanation of the given scenario is as follows:

The disease of the patient cannot be cured in ESI so he has to be referred to another hospital where he can be treated properly. For this, the concerned doctor of the patient activates the service for the patient with a ReferRequest- which mention the Patientname and the disease, i.e., ReferRequest(PatientName, disease). Then the ESIControlS checks if there is any hospital available where that disease can be cured. This is done by invoking ESIInfoS by sending IsHospitalAvail (Disease). The ESIInfoS responds to the ESIControlS with the availability status: provided the list of hospitals avail- able and doctors name, i.e., ListAvail(Hospname, Docname) or saying No(Notavailable) if the ESIInfoS cannot find any hospital. The ESIControlS responds the same to the DoctS. Now the doctor chooses one of the hospitals and responds to the patient with a hospital name and doctorname, i.e., HospitalChosen(HospName, DocName) or saying not available if the hospital is not available. The patient activates the

service by sending a registration request, i.e., Register(PatientName, Docname) to the RefHospS and reject if the hospital is not available.

The RefHospS responds to the PatSerby sending the date, time, room no., i.e., Surgeryfixed(Date, Time, RoomNo). Once the surgery is fixed, payment is made by the patient to the RefHospS. The RefHospS responds by sending PaymentSuc- cess message when the transaction is successfully completed else it sends a PaymentFail message.

Despite the simplicity of the domain, the ESI service provider may want to include some of the following properties so as to provide better service to the patients. Additionally, the service would like to monitor some of the following properties in order to provide better service. For example:

Property **T**he patient can be referred by a doctor if and only if the doctor cant treat that disease (violation of this Boolean property indicates that there is some malfunctioning in DocS).

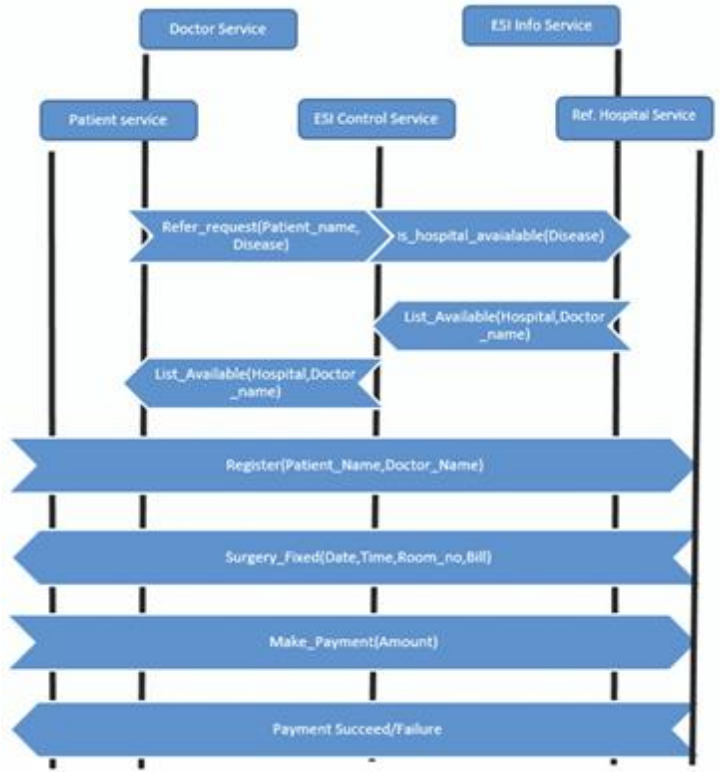

Fig. 1. Employee's State Insurance Corporation (ESI): Scenario

Property Names of those hospitals will be sent where the disease can be cured. (violation of this property indicates that there is some violation of ESIInfoS).

Property Hospital name will be sent only if the doctor of required disease is available (violation of this property indicates that there is violation in ESIIn- foS).

Property Payment is only done after surgery is completed (violation of this property means that there is some functional problem in the system).

The ESI service provider may be interested in counting number of times a given event occurs in the execution of ESI service.

Property Number of times the doctor refers to another hospital. (The information can be helpful for the ESI to see if the doctors are sincere or not).

Property Number of hospitals available for a particular disease in the nearest location : this information will be helpful for the patient to reach the nearest location

Property Number of times the surgery is successful by a particular hospital (It will be helpful for the ESI Control Service to search the efficient hospital). The ESI service provider may be interested in collecting the statistical information, related to all instances of ESI service.

Property Measure the average RHS transaction completion time.

The ESI service provider may be interested in measuring the time spent to perform certain activities of ESI service

Property Time taken to process the request by the ESI Control Service (It will be helpful for the head of the ESI organization to monitor the system).

Property Time taken by the hospital to respond the request sent by ESI.

Property Time between registration and appointment of surgery given by the hospital (It will be helpful for the doctor during emergency cases)

Property Measure the Referred Hospital Service transaction completion time (sudden increase/decrease of RHS transaction time may lead to some functional problem)

III. MONITOR SPECIFICATION

Citizen charter terma and conditions i.e, functional and non-functional requirements and service process assump- tions are specified using an XML schema that is based on a temporal logic based language called Monitor specifica- tion language(MSL)[14], called XMSL. In the following, an overview of MSL is given and then a complete definition of XMSL.

A. Overview of MSL

As we discussed in Section ??, the SLA guarantee terms that need to be monitored are needed to formally specified. A temporal logic based, executable language called MSL [14] designed by the authors is used to formally specify the SLA guarantee terms. which is defined as follows:

In MSL, SBS properties are specified in terms of *events*. Here, an *event* is something that occurs at a specific instant in time in SBS domain. Events are categorized in to three categories: Service Layer Events, Business Layer Events and Infrastructure Layer Events. For example,

- Sent/received messages by the composed service to/from the atomic services used in the composi- tion. These service to service message communication events are classified as service layer events.
- Something interesting occur in the SBS business do- main which may significantly affect the business of the SBS are categorized as business layer event. For example, if a carnival is takes place in a city then the business of ”Travel Agent Service” of that city may increase. So, the happening of carnival is a interesting event for ”Travel Agent Service”. And since these event occur in the business layer, these events are categorized as business layer events for ”Travel Agent Service”.
- If a service provider uses 4 virtual machines to run the services. If the of one of the virtual machine is exceeding the normal load limit then in near future there is a chance of that virtual machine failure. So this may be a interesting event for a service provider. Since this event is related to service infrastructure, we categorize the events from infrastructure layer as infrastructure layer event.

The grammar for specifying events in MSL is as follows: $event$ ::= $event\ Name$ | $event\ Name$.($condition$)

$Event\ Name$ ::= $[a-z][a-zA-Z0-9]*$

$condition$ ::= $type\ \ var\ \ cond\ \ value$ |

$condition\ V\ condition$ | $condition\ \wedge\ condition$ $type$ ::= int | $double$ | $string$

var ::= $[a-zA-Z0-9]+$

$cond$ ::= $\neq$ | $=$ | $>$ | $<$

$value$ ::= $[-+][0-9]+$ | $[-+][0-9]+.[0-9]*$ | $[a-zA-Z]*$

1) Semantics: From previous discussion we can assume that, finally for the framework the event is a message with a message name with zero or more internal variable with variable name, variable type and variable value. This part of the gram- mar facilitates the specification of events with the condition on the internal variables of the event, *event Name* specifies the message name, *type*, *var* and *val* specifies internal variable type (which may be int/double/string), internal variable name (which is a string) and internal variable value (which may be an integer number/a real number/a string) respectively. Condition is defined as *type var cond value*: where *type* is the data type of the variable(*int/double/string*), *var* is the name of the variable, *cond* is a logical condition ($= / \neq / > / <$) on variable and *value* is a value(number/string) to compare with the variable value.

The following grammar defines the boolean, temporal and statistical formulas. We distinguish boolean formulae *b*, which monitor properties that can be either true or false, and numeric formulas *n*, which monitor properties that define a numerical value (which include temporal and statistical formulae).

b ::= $event$ | $b \vee b$ | $b \wedge b$ | $b \Rightarrow b$ | $\neg b$ | $n = n$ | $n > n$ | $Y\ b$ | $O\ b$ | $H\ b$ | $b\ S\ b$

n ::= $C(b)$ | $T(b)$ | $b?n : n$ | $n + n$ | $n - n$ | $n * n$ | n / n | NUM

NUM ::= $[0-9]*$ | $[0-9]+.[0-9]*$

A boolean formula can be an event, or an event with some comparison between internal variables of the event, or a past LTL [6] formula (operators *Y, O, H* and *S*), or a comparison between numeric formulas, or a logic combination of other boolean formulas. A numeric formula can be either a counting formula (operator *C*), or a time measurement formula (operator *T*), or an arithmetic operation on numeric formulas.

The operators $\vee, \wedge, \neg, =, >, <$ and $\Rightarrow$ have the same meaning as $logical\ \vee$, $logical\ \wedge$, $logical\ \neg$, $logical\ =$, $logical\ >$, $logical\ <$ and $logical\ \Rightarrow$. Past LTL formulas have the following meaning: *Y b* means ”*b* was true in the previous step”, *O b* means ”*b* was true (at least) once in the past”, *H b* means ”*b* was true always in the past” and *b1 S b2* means ”*b1* has been true since *b2*. Numeric formula *C(b)* counts the number of times that boolean formula *b* has been true since the creation of the process instance. Formula *T(b)* counts the sum of the time-spans the formula *b* remains true.

B. XMSL

A citizen charter specification protocol has been proposed in this section. This protocol uses standard XML and monitor specification language and is called XMSL. Figure 2 depicts guarantee term specification protocol.

charter.xml

```
<DCC>
<!DCC: digital citizen's charter>
    <Organization>
        Employee's State Insurance
        Corporation: ESI
    </Organization>
    <Rule Number ="1">
        <EnglishDescription>
            Rule definition in simple language
        </EnglishDescription>

        <MSL_Definition>
            Rule definition in MSL
        </MSL_Definition>

        <Penalty>
            <PenaltyProvider>
                Rule violation responsible authority.
            <PenaltyProvider>
            <PenaltyDescription>
                Description of penalty imposed upon
                violation of the rule.
            </PenaltyDescription>
        </Penalty>
    </Rule>
        .
        .
        .
    <Rule Number ="n">
        ...
    </Rule>
</DCC>
```

Fig. 2. XMSL Protocol

C. MSL SPECIFICATION FOR ESI SERVICE SCENARIO

This section introduces the MSL formulae of the definition of the properties mentioned in Chapter II. The properties along with the MSL formulae are as follows:

Property The patient can be referred by a doctor if and only if the doctor cant treat that disease (violation of this Boolean property indicates that there is some malfunctioning in DocS).

ReferReqValid ⇒ O(ESIDocTreatUnavail)

Property Names of those hospitals will be sent where the disease can be cured. (violation of this property indicates that there is some violation of ESIInfoS). C(HospitalsDiseaseTreatment)

Property Hospital name will be sent only if the doctor of required disease is available (violation of this property indicates that there is violation in ESIIn- foS).

HospitalName ⇒ H(RequiredDoctor)

Property Payment is only done after surgery is completed (violation of this property means that there is some functional problem in the system). Make Payment ⇒ O (Surgery Successful) The ESI service provider may be interested in counting number of times a given event occurs in the execution of ESI service.

Property Number of times the doctor refers to another hospital. (The information can be helpful for the ESI to see if the doctors are sincere or not). C(Refermade)

Property Number of hospitals available for a particular disease in the nearest location : this information will be helpful for the patient to reach the nearest location
C(HospDiseaseTreatNearLoc)

Property Number of times the surgery is successful by a particular hospital (It will be helpful for the ESI Control Service to search the efficient hospital). The ESI service provider may be interested in collecting the statistical information, related to all instances of ESI service.
C(SurgerySuccessfulParticularHosp)

Property Measure the average RHS transaction completion time.
The ESI service provider may be interested in measuring the time spent to perform certain activities of ESI service.
AddAll(T((SurgerySuccessful) S (RHSRegister))
/ AddAll(C(RHSRegister))

Property Time taken to process the request by the ESI Control Service (It will be helpful for the head of the ESI organization to monitor the system). T((HosplistUnavail) S (ReferRequest))

Property Time taken by the hospital to respond the request sent by ESI.
T((HospitalNotResponded) S (ESIRequestSent))

Property Time between registration and appointment of surgery given by the hospital (It will be helpful for the doctor during emergency cases)
T((AppointmentNotGiven) S (RHSRegister)

Property Measure the Referred Hospital Service transaction completion time (sudden increase/decrease of RHS transaction time may lead to some functional problem)

T((SurgeryUnsuccessful) S RHSRegister)

An XMSL example for ESI has been given in Appendix

IV. AUTOMATIC MONITOR GENERATOR

Automatic monitor generator framework has been designed with the objective to support three different key monitoring features for service based system. The three key features of this approach are: (i) the monitoring is performed in parallel with the operation of an service based system without affecting its performance, (ii) non-intrusive service based system monitoring (i.e, monitoring without interfering the service based system process execution or with out changing the original service based system), and (iii) the monitoring framework is independent of the service process execution platform.

The framework facilitate the specification of citizen charter created between service provider and service consumer, as well as specification of service based system properties for monitoring. The framework monitor the operations of service based system at run-time to see whether certain specified properties and/or citizen charter guarantee terms are satisfied or not and indicate any deviations once they are detected.

A. Monitor Engine

Monitor Engine is the most important part of framework. It consists of 4main parts:• Monitor Generator

- Monitor repository
- Monitor handler
- Monitor result DB

1) Monitor Generator: Monitor generator generates the monitor. It is a MSL compiler designed using Bison (Parser Generator) and flex (lexical analyzer generator). The compiler transmits the MSL file to a C program which is stored in monitor repository which is the storehouse for all the monitors. The ID is the serial number of the monitor. The C file contains two parse trees, one of MSL formula and other of updating function. Each node along with its child subtree represents formula and each node stores the formula values. The root node contains the total formula. When monitor handler wakes up a monitor by sending an event, the update tree function updates the formula value at each node of the parse tree of the corresponding monitor according to formula value update rules.

2) Monitor Repository: A Monitor Repository is a real time database that consolidates data from a variety of clinical sources to present a unified view of a single patient. It is optimized to allow clinicians to retrieve data for a single patient rather than to identify a population of patients with common characteristics or to facilitate the management of a specific clinical department. Typical data types which are often found within a monitor repository include: clinical laboratory test re- sults, patient demographics, pharmacy information, radiology reports and images, pathology reports, hospital admission, dis- charge and transfer dates, ICD-9 codes, discharge summaries, and progress notes.

3) Monitor Handler: Monitor handler receives events from the event bus and sends the received events to appropriate monitors in the Monitor Repository and Monitor Result DB stores the results of the monitors.

V. MONITORING FRAMEWORK SETUP

In our work we have investigated the use of the proposed ”Web Service Based System” monitoring framework in Citizen charter specification and monitoring which was originally proposed in [15], [14]. We took their framework and test our charter monitoring and found out that it is a realtime monitoring and effective framework and is very suitable for dynamic monitoring requirements.

A. MSL Compiler Generation

We took the initial Lex (mongen.l) and Bison (mongen.y) files from [15], [14] and generated the compiler using follow- ing steps.

We setup the Bison and Flex in Ubuntu 14.10 and followed the following execution steps.

- Generate the parser generator using following command.
  `bison d mongen.y`
  It generates a file called *mongen.tab.c*.
- Generate the parser generator using following command.
  `lex mongen.l`
  It generates a file called *lex.yy.c*.
- Generate the compiler using following command.
  `cc g lex.yy.c mongen.tab.c –11 o mongen`
  It generates a object file called *mongen*.

B. Monitor Generation and Execution

Compiler execution and testing steps.

- Run the compiler using following command. `./mongen`
- Enter the required MSL formula to generate the required monitor program. Example: $paymentSuccess \Rightarrow O\,(paymentRequest)$.

It will automatically generate a c program (*mon.c*) to monitor the MSL property.

- Compile the monitor i.e., *mon.c* using following com- mand.
  cc mon.c
  It will create a object file called *a.out*.
- Run the monitor i.e., *mon.c* using following command.
  `./a.out`
- Enter the events and observe the monitoring output.

Figure 3 shows the execution console snapshot of all this process.

Fig. 3. Execution console snapshot

VI. CONCLUSION

An event based, non-intrusive monitoring framework has been proposed that separates business logic from the monitoring functionality and supports cross-layer citizen charter monitoring. The proposed framework does not depend on the service execution platform. A protocol using MSL and XML has been developed to formally specify citizen charter guarantee terms. The charter guarantee terms expressed in the form of XSML specifications are automatically translated into an executable C program which is used by the framework while monitoring the specified behavior of the system.

We continue our work to extend the proposed framework for service based system failure-handling, repair and adaptation triggered by information provided by the monitors. Further, we plan to provide an experimental evaluation of the usability and practical effectiveness of the proposed framework in different application domains.

REFERENCE

[1] F. Aina, D. Faniran, and K. Olaniyan. The imperative of smart technology for timely release of official data for national development: a focus on nigeria's national bureau of statistics. In Proceedings of the 8th International Conference on Theory and Practice of Electronic Governance, pages 20–23. ACM, 2014.

[2] H. Almagwashi, A. Tawileh, and A. Gray. Citizens' perception towards preserving privacy in e-government services: a cross-sectional study. In Proceedings of the 8th International Conference on Theory and Practice of Electronic Governance, pages 24–27. ACM, 2014.

[3] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification (WS-Agreement). In Global Grid Forum. The Global Grid Forum (GGF), 2004.

[4] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. 2006.

[5] T. Chau, V. Muthusamy, H. Jacobsen, E. Litani, A. Chan, and P. Coulthard. Automating SLA modeling. In Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds, pages 126–143. ACM, 2008.

[6] E. Emerson. Temporal and modal logic. Handbook of theoretical computer science, 8:995–1072, 1990.

[7] P. Hasselmeyer, H. Mersch, B. Koller, H. Quyen, L. Schubert, and P. Wieder. Implementing an SLA negotiation framework. In Proceed- ings of the eChallenges e-2007 Conference, Hague, Netherlands, 2007.

[8] O. O. Ikiz, M. Z. Sobaci, N. Yavuz, and N. Karkin. Political use of twitter: the case of metropolitan mayor candidates in 2014 local elections in turkey. In Proceedings of the 8th International Conference on Theory and Practice of Electronic Governance, pages 41–50. ACM, 2014.

[9] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management, 11(1):57–81, 2003.

[10] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck. Web service level agreement (WSLA) language specification. IBM Corporation, 2003.

[11] K. Mahbub and G. Spanoudakis. Monitoring ws-agreements: An event calculus-based approach. Test and Analysis of Web Services, pages 265–306, 2007.

[12] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. End-to- end support for qos-aware service selection, binding, and mediation in vresco. Services Computing, IEEE Transactions on, 3(3):193–205, 2010.

[13] A. Sahai, V. Machiraju, M. Sayal, A. Van Moorsel, and F. Casati. Au- tomated SLA monitoring for web services. Management Technologies for E-Commerce and E-Business Applications, pages 28–41, 2002.

[14] A. K. Tripathy and M. R. Patra. An event based, non-intrusive monitoring framework for web service based systems. In Proceedings of the 6th International Conference on Next Generation Web Service Practices: NWeSP 2010, pages 201–206. IEEE, 2010.

[15] A. K. Tripathy and M. R. Patra. Service based system monitoring framework. International Journal of Computer Information Systems and Industrial Management Applications: IJCISIM, 3:924–931, 2010.

