



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Citizen's Charter Validation

Ajaya Kumar Tripathy^{1,3}, Manas Ranjan Patra², and Sipali Pradhan¹

¹Dept. of Computer Science and Applications, Utkal University, India

²Dept. of Computer Science, Berhampur University, India

³Dept. of CSE, Silicon Institute of Technology, India

Abstract: Citizen Charter is a document of commitments made by a government organization to the citizens in respect of the fundamental procedures being provided to them. The main objective of Citizen Charter is to build trust between the citizen and administration, and to streamline administration in tune with the needs of the citizen. The bureaucratic process flow for different administration services are dynamic and volatile in nature. To make government organizations accountable towards citizen charter plays a vital role. However, making citizen charter without proper validation of fulfilling them is of no use. For electronic validation it is essential to specify them formally. This paper proposes a novel approach for citizen charter Specification and validation.

Keywords: e-Governance, Citizen's Charter, Service Based Systems, Service Based Systems Monitoring

I. Introduction

In general most of the services provided by government to the citizen, involves one or more government organizations. Whereas the involved organizations works independently irrespective of their involvement in inter organizational services. In case of any degradation of quality of service arises in a particular service none of the organizations are taking responsibility. To make government accountable towards quality and on time service delivery Citizen Charter plays an important role.

Research on e-governance [1, 2] spans over many interesting issues covering all the phases of government service lifecycle. In this article, we focus on yet another very interesting research topic: Citizen Charter specification and validation.

A Citizens' Charter represents the commitment of the government organization towards standard, quality and time frame of service delivery, grievance redress mechanism, transparency and accountability. Citizens' Charter one type of agreement between citizens and government organizations mentioning the terms and conditions of the services provided by the organization. For government organizations validation of Charter is necessary in order to gather evidence regarding the proper service provisioning in case of any dispute with the citizen.

Beside these reasons monitoring of government organizations are necessary for the administrators to make the system effective by taking actions before violation of charter. It is in the interest of administrators and organizations to create citizens' charters with minimum human interaction on one hand, generating electronic charters and monitor the charter on the other hand [3, 4, 5]. Several monitoring frameworks have been proposed to cope with the Web Service Based System monitoring (see e.g., [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]). We believe that in the literature no effort has been given to formally specify and monitoring Citizens' Charter. A novel solution to the problem of formal specification and provisioning time validation of Citizens' Charter has been proposed in this paper.

This paper proposes a Citizens' Charter specification and validation framework based a Web Service Based System monitoring framework [4,5]. The Citizens' Charter specification is based on a formal specification language called Monitor Specification Language (MSL) [5]. MSL Specified formulas are monitored at runtime using the monitoring framework [4,5]. The monitoring framework is an event based and non-intrusive in the sense that events are collected during the operation any organization. MSL is a temporal logic based language. The choice of MSL as the language for specifying Citizens' Charter is due to its expressiveness as a formal language, which allows specification of temporal constraints and the ability to monitor Citizens' Charter using well defined reasoning processes in the form of inference rules written in first order logic.

The rest of the paper is structured as follows. Section II, introduces an example scenario. The Citizens' Charter formal specification has been described in Section III. Section IV, gives an overview of the monitoring framework. Finally we conclude our work with a mention of our future research directions in Section V.

II. Online Passport Application Scenario

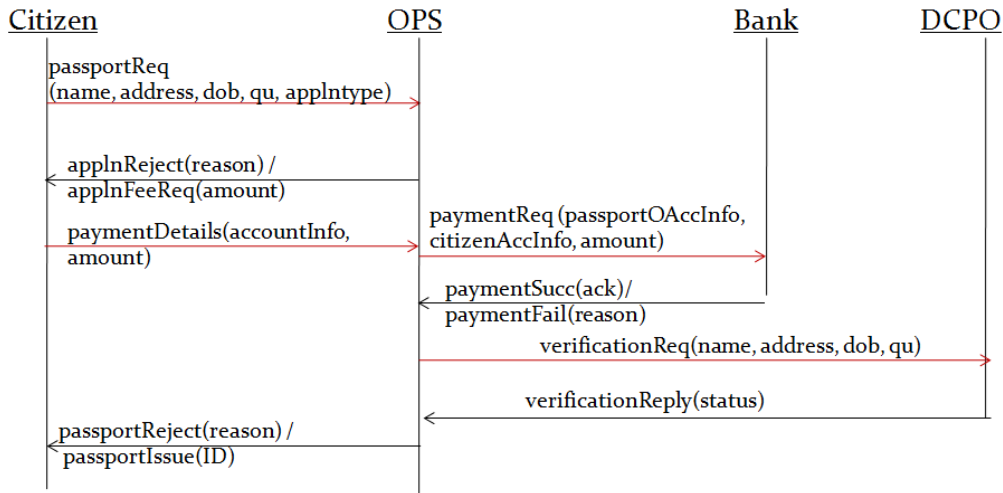


Figure 1: Online Passport Application Service Message Flow

This section depicts an online government passport issue service provided to the citizen as a visionary e-governances system. To apply for a passport citizens make an online application to the passport office with personal details. The passport office make preliminary verification of the eligibility of the application and either reject the application directly or ask the citizen to make a application fee payment, though an online payment system. Upon receiving the payment the passport office send the citizen details to DCP office for residential and character verification .The DCP office make a verification at their level and replies with a positive or negative result upon receiving a positive verification result the payment office make further processing and issue a passport to the citizen otherwise replies the citizen with a reject information .Even if this is a very simple government service scenario the citizens, administrators and the passport office my required to monitor following properties as a citizen charter rules :

- Rule 1. Passport should be issued only if the verification certification status is positive.
- Rule 2. Verification duration should be less than 10 days.
- Rule 3. Passport issue process should be less than 30 days.
- Rule 4. Count the no of times the payment process fails.
- Rule 5. Average response time of passport application process should be less than 20 days.
- Rule 6. Verification process starts after payment process success
- Rule 7. Count the percentage of application rejected.

In this example scenario, we assume the message flows of online passport service (OPS) are as follows. Citizens starts the passport application process by sending **passportRequest(Name,add,age,que.apptype)** to the OPS. OPS makes a preliminary verification and replies the citizen by sending a **reject(Reason)** or accept the application by sending a application fee payment request by **paymentRequest(amount)** by this message upon receiving this message the client send the payment details by sending **paymentDetails(accInfo,amt)**. After receiving this OPS use a online banking service (**Bank**) for payment by sending **paymentRequest(officeAccInfo,citizenAccInfo,amount)** message. The **Bank** processes the request and replies with a payment success message or payment fails message. Based on the bank message the OPS acknowledge the citizen about the application fails of application success status. Then the OPS send a verification request to Deputy Commissioner of Police office (DCPO) for citizen residential and charter verification by sending **verificationRequest(Name,add,que,age)**. After verification DCPO replies with a positive or negative status. In case of negative verification status OPS send the citizen a rejection message (**passportReject(reason)**). In case of positive verification status OPS makes further processing and issue a passport to the citizen by sending **passportIssue(ID)** message to the citizen. The complete message flow has been shown in Figure 1.

III. Monitor Specification Language

The citizen charter need to be monitor as expressed in a temporal logic based, executable language, which was proposed in [4, 5], which is described as follows. In this language the properties are specified in terms of *events*. Grammar:

$event ::= eventName \mid eventName.(condition),$
 $eventName ::= [a - z] [a - z A - Z 0 - 9]^*$

$condition ::= type\ var\ cond\ value \mid condition \vee condition \mid condition \wedge condition$
 $type ::= int \mid double \mid string$
 $var ::= [a - z\ A - Z\ 0 - 9]^+$
 $cond ::= \neq \mid = \mid > \mid <$
 $value ::= [-\ +][0 - 9]^+ \mid [-\ +][0 - 9]^+ [0 - 9]^* \mid [a - z\ A - Z]^*$

This part of grammar facilitates the specification of the events with the condition on the internal variables of the event, *eventName* specifies the message name, *type*, *var* and *value* specifies the internal variable name (which may be int, double or string), internal variable name (which is a string) and internal variable value (which can be real number, integer or a string) respectively. Condition is defined as *type var cond value* : where *type* is the data type of the variable (int or double or string), *var* is the name of the variable, *cond* is the logical condition ($= \mid < \mid >$) 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 formula 'b' which monitors the properties that can be either true or false, a numeric formula 'n' which monitor properties that define a numerical value (which includes temporal and statistical formulas).

$b ::= event \mid b \vee b \mid b \wedge b \mid b \Rightarrow b \mid \sim b \mid n = n \mid n > n \mid Yb \mid Ob \mid Hb \mid b\ S\ b$
 $n ::= C(b) \mid T(b) \mid b\ ?\ n : n \mid n + n \mid n - n \mid n * n \mid n / n \mid NUM$
 $NUM ::= [0 - 9]^* \mid [0 - 9]^+ [0 - 9]^*$

A. MSL Specification

The monitoring properties we have introduced in Section II can be define by the following MSL formulae:

Rule 1. Passport should be issued only if the verification certification status is positive.
passportIssue $\Rightarrow 0(\text{verificationRes} \text{ (status = positive)})$

Rule 2. Verification duration should be less than 10 days.
 $T(\sim \text{verificationRes}\ S\ \text{verificationReq}) < 10 * 24h$

Rule 3. Passport issue process should be less than 30 days.
 $T(\sim(\text{passportReject} \vee \text{passportAccept})\ S\ \text{applicationSucess}) > 30 * 24h$

Rule 4. Count the no of times the payment process fails.
 $C(\text{paymentFails})$

Rule 5. Average response time of passport application process should be less than 20 days.
 $SUM(T(\sim(\text{passportReject} \vee \text{passportAccept})\ S\ \text{applicationSucess})) / C(\text{passportAccept} \vee \text{passportReject}) < 20 * 24H$

Rule 6. Verification process starts after payment process success.
 $\text{verificationReq} \Rightarrow 0(\text{paymentSucess})$

Rule 7. Count the percentage of application rejected.
 $C(\text{passportReject} \vee \text{passportIssue}) * 100 / C(\text{passportRequest})$

IV. Monitoring Framework

For monitoring citizen charter we have used the monitoring framework proposed in [4, 5], which is depicted in Figure 2.

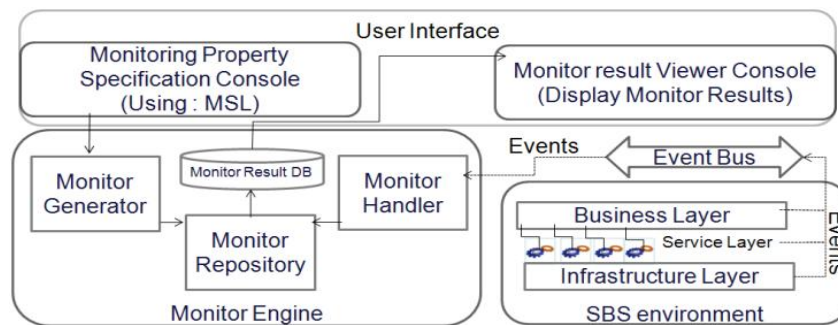


Figure 2: Service Based System Monitoring Framework

This framework has 7 main components namely *Event Bus*, *MSL Specification Interface*, *Monitor Generator*, *Monitor Repository*, *Monitor Handler*, *Monitor Result DB*, *Monitor Result Viewer*.

Event Bus: The “*Event Bus*” collects events from the service based system to be monitored and puts the events in an event queue. The monitors consume the events from the queue. The types of events the *Event Bus* receives are: messages received from or sent to the services by one of the component services of the process, interesting events from business layer/Infrastructure layer.

MSL Specification Interface: The “*MSL Specification Interface*” is used by the framework to input the MSL formulas for creation of monitors.

Monitor Generator: The “*Monitor Generator*” is a MSL compiler, designed using Bison [15] as parser generator and Flex [16] as lexical analyzer generator. This compiler translates the MSL specified formula to a C program named MonitorID.c and stores it in the *Monitor Repository*, where ID is the serial number of the monitor. Also the name of the created monitor (i.e, MonitorID) is registered (i.e, stored) in the *Monitor Repository* (i.e, a registry which stores name of created monitors).

Monitor Repository: The “*Monitor Repository*” is used to index and store the created monitors for on time consumption of events and validation of the rules.

Monitor Handler: The “*Monitor Handler*” is responsible for receiving new *events* from *Event Bus*, creating the required new instances of the existing monitors in the *Monitor Repository* and waking up appropriate monitors to consume the incoming *event*.

Monitor Result DB: The “*Monitor Result DB*” is responsible to store the updated results produced by the MSL formula monitors after consuming the *Events* from the *Event Bus*.

Monitor Result Viewer: The “*Monitor Result Viewer*” is an interface to display the results stored in Monitor Result DB at real time.

V. Conclusion and Future Work

In this paper, we have presented an approach for monitoring Citizen Charter for Government systems. An event based approach has been proposed that separates functional logic from the monitoring functionality and Citizen Charter monitoring. Further, a monitoring language has been discussed to formally specify the Citizen Charter .The specification is automatically translated into an executable C program which is used by the framework while monitoring the specified behavior of the system .In future we will try to implement and test the framework in a real like scenario.

References

- [1] M R Patra, A. K. Tripathy and R. K. Das. Monitoring of Service Based e-Government Systems. In *the 5th International Conference on Theory and Practice of Electronic Governance: ICEGOV 2011* Estonia, Pages 353-354 ACM 2011.
- [2] A. K. Tripathy, M. R. Patra. MSL Based Service Based e-Governance Systems Monitoring. In *International Journal of ACM Jordan: The research Bulletin of Jordan ACM Volume II (II)*, Page 90-101 *Jordan* 2012.
- [3] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. In *ICWS*, volume 6, pages 63-71 2006.
- [4] 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.
- [5] A. K. Tripathy and M. R. Patra. Service Based System Monitoring Framework. In *International Journal of Computer Information Systems and Industrial Management Applications: IJCISIM*, Volume 3, Page 924-931, 2011, **Dynamic Publisher USA**.
- [6] L. Baresi, C. Ghezzi, and S. Guinea. Smart monitors for composed services. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 193–202. ACM New York, NY, USA, 2004.
- [7] A. K. Tripathy and M. R. Patra. Modeling and Monitoring SLA for Service Based Systems. In *Proceedings of the International conference on Intelligent Semantic Web - Services and Applications – ISWSA 11*, Jordan, pages 60-65. ACM 2011.
- [8] L. Baresi and S. Guinea. Towards dynamic monitoring of WS-BPEL processes. *Lecture Notes in Computer Science*, 3826:269, 2005.
- [9] A. K. Tripathy and M. R. Patra. Automating WS-Agreement Monitoring. In *International Journal of ACM Jordan: The research Bulletin of Jordan ACM Volume II (I)*, Page 99-113 *Jordan* 2011.
- [10] A. K. Tripathy, M. R. Patra and S. K. Pradhan. Run-Time Monitoring of SLA for WSBS at Application Layer. In *International Journal of Software and Web Sciences*, Volume 9(2), Page 100-104, Aug 2014, IASIR.
- [11] K. Mahbub and G. Spanoudakis. A framework for requirements monitoring of service based systems. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 84-93. ACM, 2004
- [12] A. K. Tripathy, M. R. Patra, M. A. Khan H. Fatima and P. Swain. Dynamic Web Service Composition with QoS Clustering. In *21st IEEE International Conference on Web Services: ICWS*, pages 678–679, Alaska, USA, 2014.
- [13] A. K. Tripathy and M. R. Patra. Cost Effective, Requirement Oriented Web Service Composition and Adaptation. In *the International Journal on Recent Trends in Engineering & Technology: IJRTET*, Volume 05, No. 01, Page 95-98 Mar 2011. **ACEEE USA**.
- [14] S. Satpathy, A. K. Tripathy, S. K. Pradhan and M. R. Patra. A Survey & Analysis of Web Service Composition. In *International Journal of Software and Web Sciences*, Volume 9(1), Page 23-27, Aug 2014, IASIR.
- [15] C. Donnelly, and R. Stallman. The YACC-compatible parser generator. The Free Software Foundation. 1992.
- [16] V. Paxson and others. Flex-fast lexical analyzer generator. Free Software Foundation, 1988