# TRANSPARENCY IN CITIZEN-CENTRIC SERVICES
*A Traceability-based Approach on the Semantic Web*

Ivo J. Garcia dos Santos and Edmundo R. Mauro Madeira
*Institute of Computing, University of Campinas (UNICAMP), P.O. Box 6176, Campinas SP, Brazil*

Keywords: e-Government, Semantic Web Services, Traceability, Middleware.

Abstract: The search for effective strategies to increase the transparency in public administration processes is becoming a key issue for governments and for the success of the new citizen-centric services and applications. Also, the application of service-oriented architectures, semantics and ontologies is gaining momentum as an alternative to fulfill the inherent e-Government interoperability and dynamism demands. Considering the challenges introduced by this new scenario, this paper contributes proposing an approach to monitor and audit composite e-Government services. The solution is based on a set of Traceability Policies modeled and implemented over a semantically-enriched and service-oriented middleware (CoGPlat).

## 1 INTRODUCTION

The demands for the creation of mechanisms to increase the transparency of the public administration processes have dramatically increased over the recent years. It represents a requirement for every democracy identified since the origins of the modern republic: Thomas Jefferson once wrote that ”*whenever the people are well-informed, they can be trusted with their own government*” (Jefferson, 1789). In parallel, new citizen-centric services and applications are becoming more and more investigated, mainly due to the proliferation of the new ICTs (*Information and Communication Technologies*). The use of Service-oriented architectures and, more recently, semantics and ontologies, is also gaining momentum to enable interoperability and to increase the dynamism of e-Government applications. Considering this scenario, this paper contributes by presenting a strategy to achieve transparency in composite e-Government Services based on a set of Traceability Policies modeled and implemented over a semantically-enriched and service-oriented middleware (CoGPlat). The remainder of this paper is organized as follows: Section 2 introduces the concepts and technologies related to the contributions of the paper and presents relevant related work; Section 3 introduces the proposed approach to increase transparency in composite e-Government services; and finally, Section 4 presents the concluding remarks.

## 2 LITERATURE REVIEW

The **interoperability**, defined as ”*the ability of two or more systems or components to exchange information and to use the information that has been exchanged*” (IEEE, 1990), is a fundamental requirement in the context of distributed and dynamic applications. In order to achieve higher levels of interoperability, many systems are being implemented following the Service-Oriented Architecture (SOA) approach (Papazoglou and Georgakopoulos, 2003) and its most famous manifestation, the Web Services technologies. One of the current Web biggest limitations is the lack of support to describe the semantics of data. To overcome this weakness, the so-called **Semantic Web** (Berners-Lee et al., 2001) proposes a scenario where data becomes meaningful to machines and can be automatically processed and understood. In it, ontologies play a fundamental role in the definition of the concepts that are used to annotate data (Medjahed, 2004). Specifications such as RDF (*Resource Description Framework*) and RDF-Schema were among the first W3C (World Wide Web Consortium) initiatives to model meta-data related to Web resources. Later, the OWL (Ontology Web Language) extended RDF, augmenting its vocabulary with the inclusion, for instance, of new class relationships. It is the current W3C standard for specifying ontologies on the Web. The possibility of applying a similar strategy to semantically describe services, opening the road towards their automatic discovery, invocation

J. Garcia dos Santos I. and R. Mauro Madeira E. (2008).
TRANSPARENCY IN CITIZEN-CENTRIC SERVICES - A Traceability-based Approach on the Semantic Web.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - SAIC*, pages 184-189
DOI: 10.5220/0001720601840189
Copyright © SciTePress

and composition, motivated the proposal of different approaches for the definition of what was baptized as the **Semantic Web Services**. OWL-S (*Semantic Markup for Web Services*), for instance, combines a set of inter-related OWL ontologies that define terms used in service-oriented applications. Besides OWL-S (adopted in the work described in this paper) two other proposals already play an important role in the Semantic Web Services scenario: the WSMO (Web Services Modeling Ontology) (W3C, 2005)) and the SA-WSDL (Semantic Annotations for WSDL and XML Schema) (W3C, 2007), recently adopted as a W3C recommendation.

Techniques which enable markup and automated reasoning technology to describe, simulate, test, and verify compositions of Web services are discussed in (Narayanan and McIlraith, 2002). The authors define the semantics for a relevant subset of OWL-S in terms of a first-order logical language (*Situation Calculus*). With the semantics in hand, service descriptions are encoded in a Petri Net formalism. The implemented system is able to read in OWL-S service descriptions and perform simulation, enactment and analysis. The use of advanced workflow and activity concepts in the composition of Web services is the proposal of (Fileto et al., 2003). The approach is called POESIA (*Processes for Open-Ended Systems for Information Analysis*), an open environment for developing Web applications using metadata and ontologies to describe data processing patterns developed by domain experts. It supports Web service composition using domain ontologies with multiple dimensions (e.g., space, time, and object description).

The **electronic Government** (e-Government) domain includes the "*set of all processes which serve decision-making and services in politics, government and administration and which use information and communication technologies*" (KBSt, 2006). Recently a new approach to e-Government is gaining momentum: the citizen-centric government, where citizens and businesses are considered customers of the public administration, so that their needs come first, rather than bureaucracy or other imperatives inside the government machine (GOV3, 2006). In this context, usually a government-wide service-oriented architecture is applied to develop a single place that offers access to all government informational and transactional services. Several research efforts in the e-Government domain can be found in the literature. For instance, a system which automatically generates Web services customized to citizens' needs and also to government laws and regulations is presented in (Medjahed and Bouguettaya, 2005). It proposes three levels of service customization: the *Citizen* level, the *Service* level and the *User interface* level. A *metadata ontology*, used to describe e-Government services and operations, is also introduced. An approach for the semi-automated design of data flows between Web Services that are semantically described using different ontologies and data representations is introduced in (Barnickel et al., 2006), including a rule-based mechanism for user-transparent mediation between ontologies spanning multiple application domains.

## 3 ENABLING TRANSPARENCY

As already mentioned, transparency is becoming a fundamental requirement in citizen-oriented applications. One of the possible strategies to improve transparency is to guarantee **traceability** at the service level. Traceability is defined by the International Organization for Standardization (ISO, 1994) as the "*ability to trace the history, application or location of an entity by means of recorded identification*". We introduce a strategy based on traceability policies to regulate the execution monitoring of composite e-Government services and detail it next.

### 3.1 The Middleware

The strategy presented in this paper is modeled and implemented over an e-Government service middleware called *CoGPlat* (***C**itizen-**o**riented e-**G**overnment **Plat**form*) (Santos et al., 2005). Its main goal is to support applications that enable the interaction and collaboration among governmental entities, organizations and citizens and its infrastructure includes (see Figure 1): a **Service Bus**, an interface between the middleware services and the applications; four **core** facilities (described next); a **Service Discovery and Execution Layer**, responsible for selecting the Semantic Web Services to participate in the compositions and also for interacting with those services at process run-time; and a set of **Support Services** which provides security, persistence, reliable messaging and transaction support to the processes running over the platform.

The four middleware core facilities are (see also Figure 1): the **Transparent Services Center**, responsible for dynamically building the service compositions according to the application requests; the **Metamodel Management Center**, which offers services and tools to manage the models, metamodels and ontologies used in the description of services, compositions, processes and entities; the **E-Governance and E-Democracy Center**, which delivers generic ser-

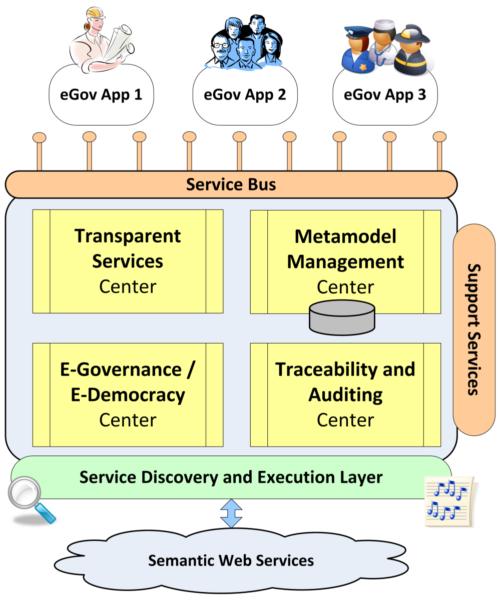
eGov App 1
eGov App 2
eGov App 3
Service Bus
Transparent Services Center
Metamodel Management Center
E-Governance / E-Democracy Center
Traceability and Auditing Center
Support Services
Service Discovery and Execution Layer
Semantic Web Services

Figure 1: CoGPlat Architecture Overview.

vices which aim to facilitate the decision-making processes and to increase citizen participation in the public administration; and the **Traceability and Auditing Center**, which offers services and tools to monitor running processes and also to audit processes that have already been concluded. The contributions of this paper are concentrated in this last facility.

## 3.2 Policies

To effectively compose e-Government services, issues like the autonomy of the entities, data privacy, process traceability and an efficient identity management usually need to be considered. In order to enable compositions which handle these demands, CoGPlat implements a set of *Interaction Policies* which were first proposed and successfully applied in an e-Business scenario (Santos and Madeira, 2006) and then extended to match the new requirements demanded by e-Government applications. These policies are applied to regulate the *collaborations* that take place over the middleware infrastructure - we consider that two entities *collaborate* when both participate in the same composite service (as providers and/or consumers) and there is at least one information or message exchange between them through the middleware facilities. The *CoGPlat Interaction Policies* are classified into the following categories (all policy terms and relations are specified in an OWL ontology):

1. **Entity Autonomy Policies.** determine the level of control the platform (and therefore the applications running over it) may have over the internal operations of a composite service;
2. **Data Privacy Policies.** determine the collaboration levels on the interactions between two entities that participate on the same composite service, defining how the privacy of the data exchanged between them is handled;
3. **Identity Management Policies.** establish where to provide mechanisms like anonymousness, identity theft protection and non-repudiation for citizens and entities;
4. **Service Traceability Policies.** determine to what extent the operations of a composite service should be tracked by the middleware. This is the category closely related to the contributions presented in this paper and will be further detailed next.

An abstract composition and individual abstract activities can be semantically annotated with the following **Service Traceability** policies:

- **Strong-traceability.** all possible events both at composition and at activity/service level are monitored. Customized traceability requirements can also be specified (e.g.: monitor the behavior of an specific property/value throughout the instance execution);
- **Weak-traceability.** only composition level events are monitored. No customizations are allowed;
- **Zero-traceability.** There is no traceability at all. Indeed, if this policy is specified, there must be no trace (neither during nor after) of a given composition instance.

## 3.3 The Composition Process

One of the main goals of CoGPlat is to facilitate the development and operation of new e-Government applications that present higher levels of dynamism and citizen-orientation. Given the quantity and diverseness of public administration agencies, to integrate and interoperate the services delivered by those agencies become a challenging task. The service composition process described next tries to shift to the middleware level the solution for those integration problems. It is important to understand how CoGPlat builds composite services before a deeper discussion on the Traceability mechanisms is held. The composition process has the following stages (see Figure 2):

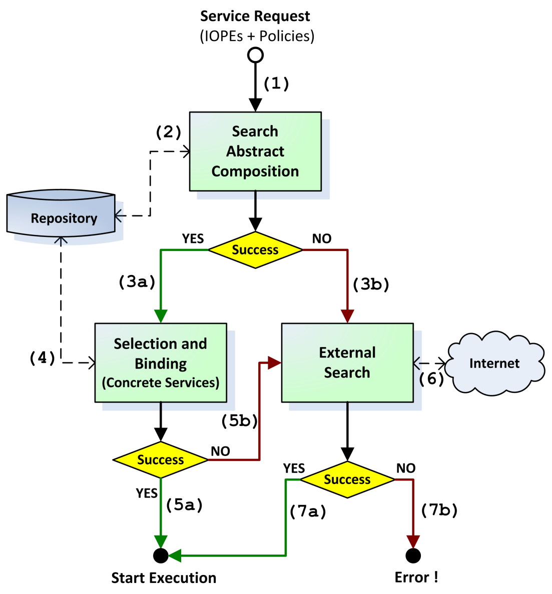

Figure 2: Composition Process in CoGPlat.

- **(1)** An application sends a request (an OWL-S profile) to the *Transparent Services Center* for a composite service. The request includes the IOPEs (Input, Output, Precondition, Effects) and the demanded interaction policies;
- **(2)** The *Metamodel Management Center* performs a search in the repository for an abstract composition that satisfies the application requiremens. An OWL-S matchmaker is used in this stage;
- **(3a, 4)** If an abstract composition is found, a search for concrete services that will execute the abstract composition activities is started. The interaction policies are, together with the IOPEs, used as selection criteria. **(3b, 6, 7a, 7b)**: Else, a search for external services (not preregistered in the platform) is tried;
- **(5a)** If a concrete composition has been successfully built, its specification is prepared for execution. **(5b, 6, 7a, 7b)**: Else, an external search is tried (as in **(3b)**).

The chances of success on the initial stages of the composition process (**1** and **2** in Figure 2) are directly related to the amount and diversity of abstract compositions registered in the platform. These abstract compositions describe the generic public processes that may run over the platform combining services from different agencies. No concrete binding is necessary in design time - actually only service classes, semantically annotated with IOPEs and Policies are included in the abstract composition flow.

## 3.4 Traceability and Auditing Center

The **Traceability and Auditing Center** implements the necessary mechanisms to guarantee that the active traceability policies are respected during the execution of the composite services. Its internal infrastructure is composed of the following elements (see also Figure 3): the **Standard Tracking Service** which collects all data from the generated tracking events, according to the active policy, and saves it into a repository; the **Real-time Tracking Service** which collects only representative data from the tracking events, also according to the active policy, but provides it in real-time to the platform client applications instead of saving it to a repository; the **Auditing Service** which offers mechanisms to access tracking data of already concluded processes; and a **Tracking Data Repository** that stores the information generated by the tracking service events.

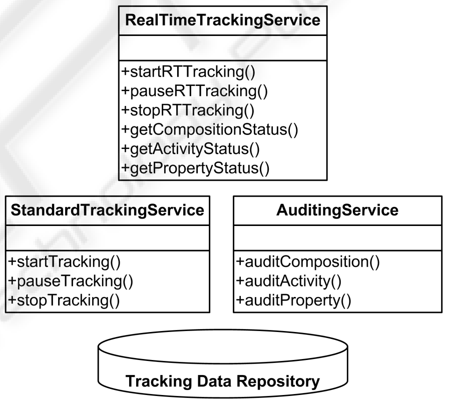

Figure 3: Traceability and Auditing Center Infrastructure.

## 3.5 Composition Example

In order to illustrate how the Traceability Policies are evaluated and executed in CoGPlat, let's follow the example of a generic process of requesting a document. The process is composed of the following steps: **(1)** Validation of the requester personal information; **(2)** Payment of administrative fees; **(3)** Forward of the request to the responsible government office (determined by the document type).

The composition process follows the steps previously presented in Figure 2. First, the application sends a request containing IOPEs and demanded policies. A search is then performed in the abstract compositions repository. In Figure 4 the selected abstract composition is shown. Note that the composition and the individual activities are annotated with IOPEs and with the supported traceability policies.

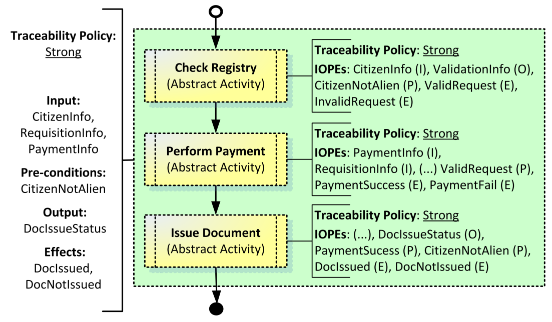
Traceability Policy: Strong
Input: CitizenInfo, RequisitionInfo, PaymentInfo
Pre-conditions: CitizenNotAlien
Output: DocIssueStatus
Effects: DocIssued, DocNotIssued
Check Registry (Abstract Activity)
Traceability Policy: Strong
IOPEs: CitizenInfo (I), ValidationInfo (O), CitizenNotAlien (P), ValidRequest (E), InvalidRequest (E)
Perform Payment (Abstract Activity)
Traceability Policy: Strong
IOPEs: PaymentInfo (I), RequisitionInfo (I), (...) ValidRequest (P), PaymentSuccess (E), PaymentFail (E)
Issue Document (Abstract Activity)
Traceability Policy: Strong
IOPEs: (...), DocIssueStatus (O), PaymentSucess (P), CitizenNotAlien (P), DocIssued (E), DocNotIssued (E)

Figure 4: Abstract Composition with Policies and IOPEs.

Next, according to the process in Figure 2, the semantic annotations are used to select concrete services that will implement each of the activities. When the composition starts to execute, the **Standard Tracking Service** of the **Traceability and Auditing Center** is activated. In the example, the composition and all activities require a strong traceability policy, so all composition and service level events are tracked and stored in the **Tracking Data Repository** (a database). While the composition is running, it is possible monitor its status in real-time. This is achieved in the platform through the following steps (see Figure 5):

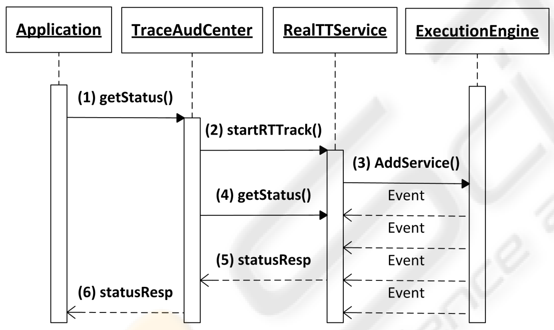

Figure 5: Finding the current status of a running process.

**(1)** The *Application* calls the *getStatus()* operation from the **Traceability and Auditing Center** (*TraceAudCenter*) service;

**(2)** The *TraceAudCenter* informs the **Real Time Tracking Service** (*RealTTService*) that a given composition instance should be tracked following a given policy;

**(3)** The *RealTTService* subscribes to the tracking events generated by the WF Runtime Engine (*ExecutionEngine*) for the given instance. It uses a custom profile that implements the active traceability policy (see also Section 3.6). From this moment, the *events* specified in the profile start to be delivered to the *RealTTService*;

**(4)** The *TraceAudCenter* asks the *RealTTService* for the status of a given composition (or also of an activity or property);

**(5)** The status response (*statusResp*) is sent back to the *TraceAudCenter* and then to the *Application* **(6)**.

The next time the application requests real-time status information of the same composition instance, the steps (2) and (3) do not need to be repeated (the flow starts then from step (4)) as the instance is already being monitored. It is also possible to audit the process execution anytime after its conclusion using the **Auditing Service**, which offers to the applications a convenient way to access the tracking information stored in the repository. Note, though, that only information in accordance with the selected traceability policy will be present there.

## 3.6 Implementation Issues

The *CoGPlat* middleware infrastructure is implemented with the support of the following technologies: the **Service Bus** exports traditional Web Services (WSDL + SOAP/HTTP); the middleware **core** runs over the Microsoft .NET framework and is written in C#; the **service discovery** layer uses the OWLS-MX matchmaker (Klusch et al., 2006); and the **composition executions** are performed by the *Windows Workflow Foundation* (WF) runtime engine (Bukovics, 2007). The **Traceability and Auditing Center** implementation relays on the WF *Tracking Service* functionalities. It provides three event categories: **Workflow Events** which correspond to changes in the composition instance status; **Activity Events** which correspond to changes in the execution status of individual activities (services); and **User Events** that can be generated at any point in the life cycle of the composition instance (those events are customized and defined during the construction of the abstract compositions). The default WF Tracking Service behavior generates events for all workflow and activity types. Therefore, to implement a behavior that corresponds to the chosen traceability policy, *CoGPlat* defines three custom WF tracking profiles, one for each of the possible traceability policies (**strong-**, **weak-** and **zero-** traceability). Note that as the first policy (strong) allows customization, additional tracking behavior might be specified during the construction of the abstract compositions. Besides what is (and what is not) being monitored, another important question is what to do with the information produced by the tracking service. The Traceability and Auditing Center uses the default WF *SQLTrackingService*, which provides ready-to-use functionalities to record data to a SQL Server database. This recorded data can be later used for auditing purposes. In addition, CoGPlat implements a custom tracking service which allows the on-line monitoring of an executing composition.

## 4 CONCLUDING REMARKS

The proposal of effective solutions to increase the public administration transparency is becoming a key issue for the success of the new citizen-centric services and applications. Considering the challenges introduced by this new scenario, this paper contributes proposing a strategy to monitor and audit composite e-Government services. It includes the proposal of a set of Traceability Policies, used to semantically annotate the services, specifying the desired traceability behavior. The paper also contributes by presenting an application scenario and by discussing its implementation, based on the *Windows Workflow Foundation Tracking Service*. The work presented in the paper is in the context of the *Traceability and Auditing Center*, one of the core facilities of the *Citizen-oriented e-Government Platform* (CoGPlat). Future works may include the extension of the traceability policies, the implementation of new applications and a study on the effects of the tracking mechanisms on the performance of time-critical compositions, not so frequent in the e-Government domain. There are still many challenges to be faced before a fully citizen-centric e-Government becomes a reality. However, the visioned changes create the expectations of a future where public administration processes may become really transparent and efficient, stimulating the continuous research efforts in this field.

## ACKNOWLEDGEMENTS

The authors would like to thank the Brazilian agencies CAPES and CNPq for the financial support. Additional thanks to the Fraunhofer-FOKUS Institute (Berlin, Germany).

## REFERENCES

Barnickel, N., Fluegge, M., and Schmidt, K.-U. (2006). Interoperability in egovernment through cross-ontology semantic web service composition. In *Workshop Semantic Web for eGovernment, Montenegro*.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.

Bukovics, B. (2007). *Pro WF: Windows Workflow in .NET 3.0*. Apress.

Fileto, R., Liu, L., Pu, C., Assad, E. D., and Medeiros, C. B. (2003). Poesia: An ontological workflow approach for composing web services in agriculture. *The VLDB Journal*, 12(4):352–367.

GOV3 (2006). *Citizen Centric Government (White Paper)*. The GOV3 Foundation - Intel, www.intel.com/go/government.

IEEE (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY.

ISO (1994). *ISO 8402:1994 - Quality management and quality assurance - Vocabulary*.

Jefferson, T. (1789). *Personal communication to R. Price*.

KBSt (2006). *SAGA - Standards and Architectures for eGovernment Applications - Version 3.0*. German Federal Ministry of Interior, www.kbst.bund.de/saga.

Klusch, M., Fries, B., and Sycara, K. (2006). Automated semantic web service discovery with owls-mx. In *Proc. of the 5th Intl. joint conference on Autonomous agents and multiagent systems (AAMAS)*, pages 915–922, USA. ACM.

Medjahed, B. (2004). *Semantic Web Enabled Composition of Web Services*. PhD thesis, Virginia Polytechnic Institute and State University.

Medjahed, B. and Bouguettaya, A. (2005). Customized delivery of e-government web services. *IEEE Intelligent Systems*, 20(6):77–84.

Narayanan, S. and McIlraith, S. A. (2002). Simulation, verification and automated composition of web services. In *Proc. of the 11th Intl. WWW Conf.*, pages 77–88. ACM Press.

Papazoglou, M. and Georgakopoulos, D. (2003). Service-oriented computing. *Communications of the ACM*, 46(10):25–28.

Santos, I. J. G. and Madeira, E. R. M. (2006). Applying orchestration and choreography of web services on dynamic virtual marketplaces. *International Journal of Cooperative Information Systems*, 15(1):57–85.

Santos, I. J. G., Madeira, E. R. M., and Tschammer, V. (2005). Towards dynamic composition of e-government services - a policy-based approach. In *5th IFIP Intl. Conf. on e-Commerce, e-Business and e-Government (I3E)*, pages 173–185. Springer.

W3C (2005). *Web Service Modeling Ontology (WSMO) Primer*. http://www.w3.org/Submission/WSMO-primer/. Member Submission.

W3C (2007). *Semantic Annotations for WSDL and XML Schema (SA-WSDL)*. http://www.w3.org/TR/sawsdl/.