A Blackboard-Oriented Architecture for e-Government Service Composition

Costas Vassilakis, Anya Sotiropoulou, Dimitrios Theotokis and Dimitris Gouscos
Dept of Computer Science & Technology, University of Peloponnese, Terma Karaiskaki, 22100 Tripolis, Greece,
{costas.anya, dtheo, gouscos@uop.gr}

ABSTRACT

A requirement for electronic government initiatives to succeed is the ability to offer a citizen-centric view of the government model. The most widely adopted paradigm supporting this task is the *life event model*, which combines basic services offered from multiple public authorities into a single, high-level service that corresponds to an event in a citizen's life. This composition is not always straightforward though, because the constituent services are generally developed in an independent fashion, using incompatible input and output formats; moreover the task of synchronising the documents required and produced by the services is tedious to implement and costly to maintain, since changes to requirements and legislation necessitate continuous updates to this scheme. In this paper, we present a blackboard architecture that can be used to deliver life-event oriented services to the citizens. The blackboard proposed for this architecture is an *active* one, undertaking the tasks of conversions, where appropriate. The blackboard couples a data flow approach with event-condition-action rules to enable dynamic formulation of life-event services, decentralising their development and maintenance.

INTRODUCTION

The problem of accommodating change in software systems is a well-addressed issue [Dyer, 2003; Kostkova 2003]. Change occurs because any Information System should be considered as a living organism that should exactly model the environment for which it was created and as such should be able to adapt to changes in this environments. In particular information systems supporting electronic government i.e. the use of ICT tools by public administrations in order to transform their relationships with citizens, businesses and other administrations [World Bank, 2001] are by definition more prone to change than their counterparts from other areas. This is the case not only because of changes that concern the provision of new services or the enhancement of existing ones but also due to changes in legislation as well as in the quality and other non-functional requirements of service beneficiaries [European Commission, 2003].Consequently, systems supporting electronic government should expose a high degree of adaptability to change in terms of transparency, ease of accommodation and tailorability [Stamoulis et al, 2003; Eardley et al., 2003]. Traditional approaches to service composition and service discovery as well as service execution exhibit a number of drawbacks related to the following areas: a) a service's availability or lack of it, b) the concurrent execution of services that contribute to a given task but are in no way related to each other apart from the fact that one service's output is another's input, c) the fact that a particular service may be invoked as many times as it is required to produce a particular document, e.g. a birth certificate, despite the fact that a particular document may be valid for a long period of time unless some change occurs that revokes its validity, d) the *a priori* knowledge of one or in some cases some discrete number of possible compositions that can take place in order to provide a composite service. In other words the absence of mechanisms that can deduce a more cost effective composition in light of the execution flow required for the evaluation of such a composite service.

The *active blackboard* [Nii, 1986; Corkill, 1991] presented in this work addresses these issues by providing a different viewpoint to service oriented computing for electronic government applications. Instead of viewing service execution as a flow of functional evaluations where data serve the purpose of functions, we adopt a data-centric and data-driven perspective managed by data aware entity –the blackboard– where services abide to the principle "perform when all inputs are present in the data aware entity, irrespective of when and by whom they were produced".

The rest of the paper is organised as follows: The next section presents some of the problems related to service composition and service based execution environments based on the experiences gained from a real world application developed to support electronic government functions. Following this, the proposed architecture is presented focusing on its structural and functional aspects while at the same time providing a detailed description of a real world case, where the benefits of this approach as compared to the traditional "functional" understanding of service interaction emerge. Finally, the paper concludes by highlighting the contributions of *active blackboard* architecture to service oriented computing in the context of electronic government while at the same time pointing out further research directions that should be investigated.

RELATED WORK

The eGov project [Tambouris, 2001] has specified and developed a platform which employs a service repository and a service creation environment that allow developers to create composite, life-event oriented electronic services based on simpler ones. The e-Gov platform is complemented with tools to monitor the flow of composite services and handle asynchronous stages in the composite service workflow. In order to specify a composite service in the framework of the eGov project, the service developer needs to *discover* and *import* the constituent simple services and combine them using service sequences, and "if-then-else" programming constructs. Parameter passing to constituent services is also done programmatically, and code must be expressly written to handle any parameter conversions that may be needed. Within the eGov project framework, changes to inputs, outputs or invocation method of any constituent service necessitate human intervention to rectify the specifications of any composite service that includes the updated simple service. Such dependencies may be hard to be discovered, since no single repository exists in which all dependencies are recorded. Finally, the eGov service specification and execution model lacks any provisions for parallel execution, leaving thus any parallelism inherent to the services unexploited.

THE BLACKBOARD ARCHITECTURE

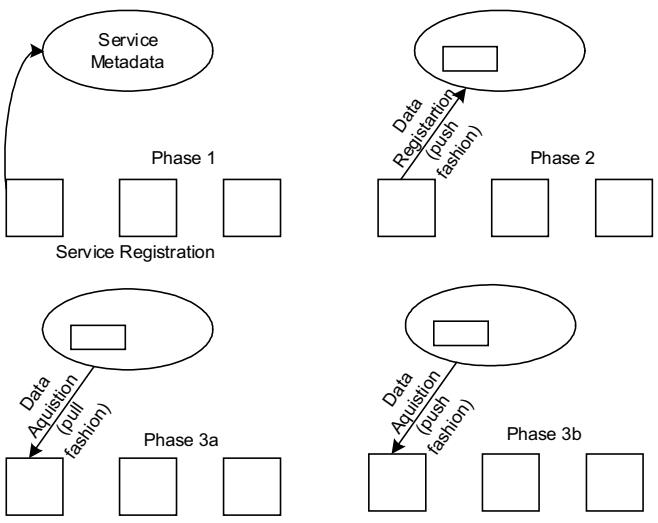
In this section the proposed solution is presented. First an overview of the architecture is given establishing the key characteristics and innovations proposed in terms of the supported functionality. Next an in depth analysis of the components comprising the architecture is given. Finally, using real world examples, a showcase for the proposed approach is given.

Architectural Overview

The active blackboard architecture provides a repository as well as a centralised container for both data, produced or consumed by services, and functionality specifications in terms of workflow sequences. Thus the burden from discovering services to co-operate with is alleviated from the services themselves and becomes the responsibility of the blackboard. In contemporary approaches, services register themselves in a registry and it is the services' responsibility for discovering other services with which they need to co-operate based on some formal or semi-formal specification that each interested service must interpret. Moreover, it is the responsibility of the services to "know" what data other services require, as to guarantee the execution of a service workflow. Instead in our approach this responsibility is solely that of the backboard. Services may either request the data required for their operation or they may be provided the data by the blackboard itself. In as such, our approach exploits the benefits of both "pull" and "push" paradigms while existing techniques for service oriented computing rely on the pull paradigm for service discovery and on the "push" paradigm for inter-service data exchange. This in itself provides a considerable advantage when concurrent independent service execution is required as part of a complex service. Figure 1 depicts this in four phases. Phase 1 concerns the service registration. Upon registration the service provides the blackboard with metadata that identify the service uniquely within the blackboard. In phase 2 a particular service places a datum on the blackboard. The blackboard generates metadata for this datum that will assist other services as well as the blackboard in the datum's manipulation. The datum residence in the blackboard is determined by a time to live property unique for each datum. Finally, in phases 3a and 3b the two different ways of data propagation to services is illustrated. The first one (3a) addresses the "pull" operation whereby a service requests a particular data item. The second one concerns the scenario where the blackboard takes control of the execution sequence and "pushes" the data item to the service responsible for the datum's processing.

To deal with a possible unavailability of a blackboard, the architecture in question provides replicated blackboards as snapshots of the master blackboard. A primary blackboard is the one that all services know to be the centralised repository. Secondary blackboards are provided as optional information to each service to be utilised in the event that the primary one is not available. To ensure that all backboards are synchronized, a blackboard content propagation scheme is employed. Whenever a datum of any kind –service registration or service datum– is placed on a blackboard the receiving blackboard forwards this data item to the other blackboards. This ensures that all blackboards contain the same information at a given time.

Figure 1. Blackboard Architecture and Its Primitive Functions



Deploying Services

Using the blackboard architecture, simple services can be deployed by simply *registering* the developed service to the blackboard (Figure 1 phase a). Such a registration activity will include a name for the service, the input parameters (names and types) that the service needs for its execution to commence, the outputs produced (types) by the service and a calling method. This is the indispensable information for the blackboard, in order to be able to (a) determine when all data for invoking the service are available (b) deduce which results may be returned to the citizen and/or which other services may be executed upon the completion of the execution of the specific service and (c) how the input parameters will be passed to the service and the service execution be initiated, and how results will be collected. Provisionally, a service registration may provide *metadata* for the service such as a textual service description, the actions performed by the service (e.g. change/update, register etc.), the entity that actions are performed on (e.g. citizen address, enterprise, car and so on), offering organisation, expected response time. Metadata may be exploited by the blackboard for service composition (detailed later), optimisation (e.g. choose minimal expected response times), enhance service presentations to the users and so on. Metadata may also be assigned to service inputs and outputs, providing semantic information on the parameters, multilingual labels that may be used for describing the parameters to human users of the service, security requirements (e.g. require digitally signed versions of parameters) etc. For output parameters, especially, the parameter metadata may include a *time to live* designation, which is used by the blackboard to determine the time frame within which the produced datum can be returned to the requesting citizen or be used as input to other services (Figure 1 phase 2).

Composite services, i.e. services that involve the execution of more than one simple service are deployed by *registering the individual components* that make up the composite service. Note that there is no requirement for the development of each individual service to take into account the cooperation with the other web services, or the communication with the blackboard: the blackboard is responsible for handling communication with the web service, as specified via the *calling method* provided in the registration, whereas communication with other services is again coordinated by the blackboard, through matching of input and output parameters (Figure 1 phase 3a). For example, in order to deploy a *passport request service*, which requires a *birth certificate* and a *proof of residence certificate*, it suffices to deploy the three individual services, i.e. the *passport request service*, the *birth certificate service* and the *proof of residence certificate service*. No need exists to include explicit logic for interconnecting the individual services, since the blackboard derives from the inputs required by the *passport request service* that the *birth certificate service* and the *proof of residence certificate service* must be completed before the *passport request service* execution can commence. The blackboard also arranges for collecting the results from the first two services and providing them as input to the latter. In this example, there is no dependence between the *birth certificate* service and the *proof of residence certificate* service, thus the blackboard may schedule their concurrent execution, exploiting thus the inherent parallelism in the composite service workflow. Parallelism can also be exploited in the cases where *multiple input providers*, i.e. in cases that an input parameter required by a composite workflow stage can be produced by more than one service. In these cases the blackboard *may* schedule the parallel execution of these tasks, collect the first reply that is returned and use it for the next workflow stage, while the rest of the replies will simply be dropped (or the relevant service executions will be pre-emptively aborted). Such a parallel execution can increase the overall response time for a composite service, however two issues must be taken into account:

- This policy introduces redundant computations, thus the load of the service deliverers may increase to such a level that the overall system performance is actually degraded. Thus, the blackboard may

consult the system load and throughput capacity of service deliverers before scheduling redundant service executions.
- The various input providers may themselves need different input parameters, which should be requested from the user (for instance one birth certificate provider may need the social security number and place of birth, while the other may need the name, surname, father name and date of birth). Requiring all candidate parameter sets to be provided can prove a nuisance for users and discourage them from using the services provided.

In certain cases, the input required by a stage within a composite service may not be directly compatible with the output produced by any other service, it may however be *transformed* to the needed form. Consider for instance a service for registering to the VIES database, in order to conduct trading between EU member states. This registration requires a *European VAT number*, which is composed of a country code (e.g. IT for Italy) and a country-specific VAT number. If a specific country has a service for obtaining a valid VAT number for that country, this can be transformed to a European VAT number by prepending the appropriate country prefix. The blackboard handles automatically such transformations, by exploiting appropriate *transformation rules*, which can coerce between different formats of the same data (represented by separate parameter/result types). In the former example, the transformation rule between the Italian VAT number and the European VAT number could be stated as:

```
EU_VAT_NUMBER ← ITALIAN_VAT_NUMBER:
    concatenate("IT}", ITALIAN_VAT_NUMBER)
```

Note that such a facility could alternatively be provided by a *registered service*, which accepts a single input parameter of type ITALIAN_VAT_NUMBER and yields a single result of type EU_VAT_NUMBER. The special provision for type transformations has been provided for optimisation purposes, since the blackboard can coerce between the two types without any overhead for parameter passing and communication with the registered service. The transformation rules can also be exploited in the context of *metadata maintenance*, in which transformations enable the grouping of metadata pertaining to alternative forms of the same information into a single pack. Complex transformations, or transformations requiring repository lookups (e.g. determine the VAT number given the social security number) can still be modelled using registered services.

The execution model described above follows a *pull approach* (Figure 1 phase 3a), i.e. when the result of some service *S* is requested, the services it depends upon are executed to provide the necessary inputs. However a *push approach* can prove useful in cases where arrival of specific data should trigger the execution of registered services. Consider for example the case that an *announcement of moving* service has been deployed by some municipality. Through this, a citizen that changes her residence may provide her new address for the municipality registry to be updated. It would be though of service to the citizen if the registries of other authorities (e.g. social security, tax office, etc) were also updated using the same data, through similar *announcement of moving* services that these authorities will have deployed. Notice that such a "broadcast" cannot be directly modelled using the previously described scheme, because the different *announcement of moving* services do not depend on one another. To cater for such cases, the blackboard allows for *event-condition-action* (ECA) specifications to be registered with it. An *event* in such a specification may be a *service invocation*, a *service invocation completion*, a *service invocation failure* or a *temporal event* (some predefined time point has been reached or a periodic timer has expired). The *condition* is a boolean expression evaluating to *true* or *false* and may reference any valid context related to the service of the event part (if any). *Valid contexts* include the service's input parameters and metadata (in all cases), the service's results (if the event is *service invocation completion*), the failure reason (if the event is *service invocation failure*). The action part may designate the invocation of one or more services, either as *additional actions* or as *alternative actions*. The former category may serve the purpose of "broadcasting" information to multiple services stated in the above example (upon completion of the municipal *announcement of moving* service, invoke the respective services offered by the social security and the tax office, using the same parameters); parameters can also be subject to *transformations*, as discussed above (Figure 1, phase 3b). The latter category is useful for specifying failover services (e.g. if a birth certificate cannot be obtained from a municipality, the respective service of the central government may be tried).

The proposed scheme also simplifies maintenance of composite services, since constituent services can be updated independently. The blackboard needs to be notified only in the case that the service interface (input and output parameters or calling method) or the pertinent metadata are updated. Linkage to other constituent services of the composite service will be automatically arranged by the blackboard, based on the inputs and outputs. ECA specifications referencing the updated service also need to be located and (possibly) adjusted accordingly. The task of locating the affected ECA rules is facilitated due to the fact that all ECA rules are stored in a blackboard repository, thus a simple query suffices for this information to be retrieved.

CONCLUSION

In this work we have demonstrated that the use of an active blackboard architecture in service oriented computing for electronic government provides a number of advantages compared to existing approaches for this purpose. These advantages include the utilisation of both the push and pull paradigms for data exchange, the exploitation of the inherently concurrent nature of service execution when viewed as a workflow sequence, the independence of the "service to data" and the "service to service" relationship, the adaptability and tailorability of the final service composition, the ability to express complex service composition in terms of simple "if-then-else" like constructs but most importantly the shift from the "function" based approach found in existing approaches to a data centric one where data "live" and are exploited whenever the need occurs by whoever requires them. Apart from performance gains the proposed approach provides a clear and sound separation between the function and the subject of the function. This implies that the function itself may evolve independently of its subject and visa versa. As such change to both function and data can be accommodated transparently without the disruption of the function of the underlying system.

The blackboard acts as a mediator between services not only with respect to data provision but also as the execution coordinator of services participating in a service composition. A major contribution of the proposed approach is that each service need not to possess the knowledge regarding other services that are required to perform a certain task.

The ability of the blackboard to determine the shortest path for a given task (service evaluation) and balance the load of request across the available paths makes the proposed approach cost effective. Moreover the replicated nature of the blackboard introduces makes the architecture failsafe since the failure of a particular blackboard does not jeopardise the completion of a service execution.

Finally, although technologies such as WSDL, UDDI, and SOAP [Newcomer, 2002] are not core to the architecture they can complement it. For instance UDDI can be utilised for data-centric discovery instead of service description based discovery. WSDL can be employed to provide the metadata required by the blackboard. However, in order to fully support the principles addressed by the active blackboard architecture further research needs to be undertaken as to enhance both UDDI and WSDL to support data-centric discovery. Incorporation of concepts addressed by WSCAF [WSCAF, 2003] although present in the proposed approach are addressed from a higher level perspective and are provided not as a feature *per se* but as a means to support the data-centric approach to service composition and interoperability.

REFERENCES

Bunting D. et al. (2003) Web Services Composite Application Framework (WS-CAF), Ver1.0, http://developers.sun.com/techtopics/webservices/wscaf/primer.pdf

Corkill D. D. (1991) Blackboard Systems *AI Expert* 6(9):40-47

Dyer S. L. (2003) A Conceptual Framework Enabling Assessment of Software Flexibility in Business Applications. In N. Patel (ed) Adaptive Evolutionary Information Systems, Idea Group Publishing.

Eardley A. et al. (2003) Methods for Developing Flexible Strategic Information Systems: Is the Answer Already Out There? In N. Patel (ed) Adaptive Evolutionary Information Systems, Idea Group Publishing.

European Commission (2003) The Role of e-Government for Europe's Future, Brussels, 26.9.2003, COM(2003)567 final, available at http://europa.eu.int/information_society/eeurope/2005/doc/all_about/egov_communication_en.pdf

Stamoulis D et al. (2003) Ateleological Development of "Design-Decisions-Independent" Information Systems. In N. Patel (ed) Adaptive Evolutionary Information Systems, Idea Group Publishing.

Kostkova, P. (2003) Support for Dynamic Trading and Runtime Adaptability in Mobile Environments. In N. Patel (ed) Adaptive Evolutionary Information Systems, Idea Group Publishing.

Newcomer, E. (2002) Understanding Web Services: XML, WSDL, SOAP, and UDDI, Addison Wesley Professional, ISBN: 0201750813

Nii H. P. (1986) Blackboard Systems. The Blackboard Model for Problem Solving and the Evolution of Blackboard Architectures. *AI Magazine 7(2)*. pp. 38-53

Tambouris, E. (2001) An Integrated platform for Realising Online One-Stop Government: The eGov Projet, in: Proceedings of the DEXA International Workshop "On the Way to Electronic Government", IEEE Computer Socity Press, Los Alamitos, CA, p. 359-363

World Bank (2001) World Bank, A Definition of e-Government, September, (http://www1.worldbank.org/publicsector/egov/).