A Distributed Architecture for Supporting *e*-Government Cooperative Processes

Mariangela Contenti[1], Massimo Mecella[2], Alessandro Termini[2], and Roberto Baldoni[2]

[1] Università di Roma LUISS "Guido Carli", Centro di Ricerca sui Sistemi Informativi
mcontenti@luiss.it
[2] Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica
{mecella, termini, baldoni}@dis.uniroma1.it

Abstract. In the last few years the *e*-Government discipline has attracted a growing attention both from practitioners and academics. Although the high number of action plan, projects and conferences spread all over Europe seems sanctioning the achievement of its maturity stage, several organizational and technological issues related with the modernization of service delivery are still far from a comprehensive solution and still require significant efforts. In this paper, the approach followed and the results so far achieved within the Eu-Publi.com research project, are presented. The discussion on the conceptual and design architectures of the Eu-Publi.com distributed, peer-to-peer system is enriched with results about the experimentation conducted on one of its core components.

1 Introduction

In the European Council held in Lisbon in 2000, the Government online was identified as one of the key actions to diffuse and widen the citizens' participation in the Information Society stimulating the use of Internet all over Europe. In its early stage, the enactment strategy for *e*-Government was then mostly focused on a rapid achievement and implementation of electronic service delivery. Many guidelines and benchmark frameworks were published (e.g., [1, 2]), and many projects started following new directions such as one-stop *e*-Government and life-events. Nowadays, researches and governmental activities are moving from *e*-Government to the emergent *e*-Governance: the achievement of a good governance in the Information Society require to exploit ICT resources, no longer only in the administrative activities, but also in the political and legislative ones.

Although this new impulse proves renewed and widened interests, some organizational and technological issues implied by the modernization of service delivery still require significant efforts, being far from a comprehensive solution. Above all, the back office integration for transactional cross-agencies services.

M. Böhlen et al. (Eds.): TCGOV 2005, LNAI 3416, pp. 181–192, 2005.

In the next sections the attempt of the Eu-Publi.com project to design and implement a cooperative system that, interconnecting at application level the several information systems, (semi-)automate inter-country complex *e*-Government services wil be discussed. More in details, in Section 2 the specific requirements driving the project are discussed, whereas Section 3 presents the proposed architecture for supporting cooperative *e*-Government processes, whose novelty lies in the distributed orchestration of different Web Services offered by the cooperating agencies. Section 4 presents the implementation and some preliminary performance tests; Section 5 compares our work with relevant research work, and Section 6 concludes the paper.

2 *e*-Government Requirements

The rapidly increasing number of conferences and projects as well as the first revolutions introduced in the European countries legal frameworks (e.g., the local implementations derived from the 1999/93/EC directive ruling the legal validity of well formed electronic documents) are all signs of the maturity reached by the discipline. At this stage, the core aspects and the more relevant hindrances appear as clearly identified [3]: prerequisites, for an *e*-Government enactment strategy, are the achievement of a technological interoperability of platforms and a deeper cooperation at organizational level; the constraints are related with the environment in which the public agencies operate, strictly constrained by norms and regulations and competitive and result-oriented at the same time.

On this purpose, the focus of the Eu-Publi.com project has been toward back-office integration and related interoperability issues: one of the first steps was to explicit the concepts behind complex *e*-Government services, through the definition of *macroprocess* as aggregation of processes to be jointly executed to satisfy the request of a service from a customer; the following the identification of the key functional requirements on the basis of the academic and end user partners consolidated experiences, enriched with desktop research.

On the hypothesis that each of the processes should be eventually (semi-)automated, the key functional requirement is the need for specific mechanisms, such as a process management systems, through which coordinate, control, monitor and audit the logical and temporal sequence of interdependency existing among the different process instances. Actually, among the several operational activities a complex business service is decomposable on, there is a plenty of control statements; their presence is motivated by the fact that an administrative procedure should exhibit transparency and equity of treatment under predefined conditions, enforcing in this way the citizens' confidence on the PA's operating.

As the introduction of technical and technological resources within organizations it not a so recent phenomena, the codification and enactment of procedures and modus operandi followed through years has been addressed with the more disparate organizational and technological solutions. The adoption of design methodologies and standards, being able to cope with legacy systems for

providing reusable applications and platforms, represents the way through which establish a valuable cooperation and coordination among agencies. Issues to be addressed not only refer to the technological interoperability of ICT resources, but also to the semantic interoperability of business services, the latter reachable through the definition and adoption of linguistic and semantic support (e.g., ontologies). Actually, a suitable platform for *e*-Government cooperation and electronic service delivery should be able to provide support in the management of inconsistencies in the legal and administrative semantics that can occur in the cross border exchange of data, terms and concepts.

Another important requirement is the need, for each of the PA involved, to maintain, not only the autonomy, but also a well defined authority and responsibility on those steps and sub-processes each public agency is entitled for. The enforcement of this separation of concerns reflects the PAs need to maintain a well defined role, on the basis of which proceed in strategy definition and performance analysis. The outcome could be recursively adopted as a reference base for internal and global reallocation of resources and business process re-engineering.

These requirements, completed with valid solution enforcing the security of the system, and the privacy protection of personal and sensitive data exchanged and stored within the cooperative environment, apply to the design and development of the single composing processes as well as the overall process management system.

3 The Eu-Publi.com Architecture

In the Eu-Publi.com application scenario, each organization interfaces the others by offering specific application services independently from their realization, and the inter-organizational cooperation is obtained by sharing and integrating such application services. In this way the several interleaved organizations are loosely coupled and local internal processes re-engineering, being hidden by the service interfaces, does not impact on the overall cooperative process and related applications.

The foundation for the proposed architecture is the Service Oriented Architecture (SOA) model consisting of some basic operations and roles: in the case of Eu-Publi.com, each cooperating administration can act both as provider of its own services and as requestor for services available on other organizations. In addition, the Eu-Publi.com architecture extends the SOA model with further roles and operations. More in details *(i)* a layer supporting macro-process enactment has been introduced in order to cope with the previously highlighted process management needs; and *(ii)* behavioral notions have been developed in order to cope with general QoS for the system, e.g., reliability, and to develop dynamic capabilities of the system[1].

[1] In such a way, the Eu-Publi.com architecture not only resolves specific *e*-Government issues, but contributes also to the current state-of-art of the research by proposing an Extended Service Oriented Architecture.

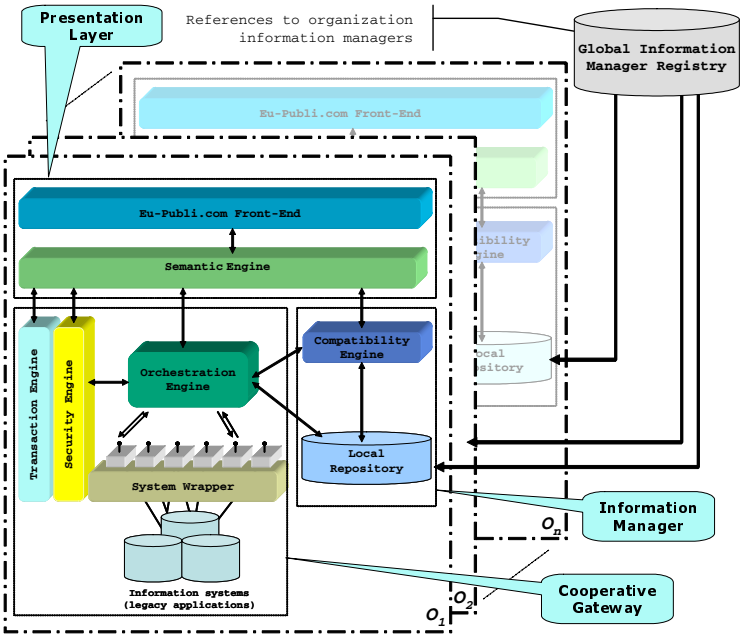


Fig. 1. Eu-Publi.com Architecture

The Eu-Publi.com architecture consists on the specification of how to realize a *cooperative layer*[2] that comprises the set of technologies, application protocols and services enabling the effective cooperation among administrations; it is designed as a peer-to-peer system, in which each organization deploys an instance of the three core macro-components (see Figure 1):

- the **Cooperative Gateway**, representing the back-end layer of the architecture: it *(i)* exports the set of data and application services offered by a single administration and *(ii)* includes the Orchestration Engine component, also deployed in a peer-2-peer fashion, that coordinates the services.
- the **Information Manager**, being the repository of the Eu-Publi.com peer, through which *(i)* service schemas, *(ii)* orchestration schemas and *(iii)* instance data are stored and made accessible.
- the **Presentation Layer**, responsible for the presentation to end users and representing the employees' front-end of the overall architecture.

In order to get the information distributed on all the organizations joining the Eu-Publi.com architecture, a further component, the ***Global Information Manager Registry***, is designed: it points to specific Information Managers containing specific information (as shown in Figure 1).

[2] As a prerequisite, the different organizations joining the Eu-Publi.com architecture must deploy a transport layer interconnecting each other, mainly based on standard Internet and Web protocols.

The cooperation of different agencies is achieved by making them responsible for exporting some views of its own information system as services; the *Cooperative Gateway* represents "where" and "how" services are deployed; it includes the definition on how different cooperating organizations are structured and connected and how pre-existing legacy applications (Local IS in the figure) can be integrated in a common cooperative process (the *System Wrapper* is the component wrapping the PA's legacy system).

The *Orchestration Engine* sub-system is responsible for coordinating all the services involved in a cooperative process; this component, working in a peer-to-peer fashion, acts at two different layers. Through "cooperative process definitions" (technically referred to as *orchestration schemas*) it dynamically finds and links suitable services; it also interacts with the different service instances deployed on the different cooperating organizations. Moreover, during the orchestration of a specific business process and whether needed, an Orchestration Engine can communicate with another Orchestration Engine deployed onto a different organization to move the responsibility of the process control (e.g., this could occur due to some legal constraints that oblige a particular administration to control the process).

The *Compatibility Engine* is in charge of managing service substitution during orchestration: this feature can be used in different scenarios, such as for handling different versions of the same service, in case of using a new service which offers improved functionalities with respect to the one currently used, during runtime substitution of an unavailable service with another one, and, in general, to enforce QoS to the system, e.g., load balancing [4].

The *Front-End System* is the component representing the GUI of the employees. This component works in a dynamic fashion: according to the specific process, it dynamically build the corresponding GUI.

The architecture also defines additional components that manage the set of non-functional requirements for the system:

- as Eu-Publi.com offers B2B integration to organizations located in different countries, the orchestration of the processes involves also cultural and linguist aspects. Therefore the existence of a component that will cope with the semantics of the exchanged information both in the presentation and the communication level is needed. With this aim, the *Semantic Engine* consists of mechanisms that enable the translation and the correspondence of terms.
- the *Transaction Engine* provides transaction and reliability features, which are mainly independent from the business domain. It adopts classical long running transaction models [5].
- the *Security Engine* is in charge of managing security-related issues, such as security and privacy of the data exchanged or communication security among the architectural components. This component also manages the access-policy for identification, authentication and authorization mechanisms envisioned by the architecture [4].

4 The Eu-Publi.com Prototype

The current implementation of the Eu-Publi.com architecture prototype relies on *Web Services technologies* [6], that can be viewed as the natural "software artifacts" of the service abstraction. Each PA joining the Eu-Publi.com system must export its functionalities as Web Services (in this case the Cooperative Gateway is the application server in which the PA's Web Services run); once an organization exports its services, an Orchestration Engine is needed in order to coordinate them into macro-processes. The implementation of such a component is based on BPEL4WS [7], i.e., the orchestration schema is a BPEL file. Moreover, in order to implement the distributed features of the architecture (not included in the BPEL specification), a specific protocol to "send" and "accept" the orchestration responsibility has been designed (see Section 4.2). The current implementation includes a simple Information Manager, based on UDDI, a Semantic Engine implementation based on OWL-S and UDDI, and a Front-End System implementation based on XForms. Finally, for what concern the Transaction Engine and the Security Engine, the W3C standards for security and transactions (i.e., WS-Security, WS-Coordination and WS-Transactions) have been considered.

In the following sections, some details about the current prototype are described; the focus is concentrated on the distributed orchestration mechanism, that is one of the most innovative result gained within the project.

4.1 Eu-Publi.com Architecture Deployment

As previously described, the Eu-Publi.com architecture is designed as a peer-2-peer system, in which each organization deploys an instance of the core components, namely the Cooperative Gateway, the Information Manager and the Front-End. As in a peer-2-peer system, there is no a clear separation between clients and servers: each organization can be both client and server.

In Figure 2 a simple process regarding the request of a license is shown; in the process three different administrations (organization A, B and C) are involved, each of them exporting some Web Services (WS_{A1} exported by A, WS_{B1} and WS_{B2} exported by B, and WS_{C1} export by C). It is supposed that the process is provided by the organization A, so the process started under organization A's responsibility from a client request (step 1). The organization A invokes the WS_{A1}, and then, for continuing the process, it passes the responsibility to the organization B (for example due to some legal or system restrictions for the access to the Web Services WS_{B1} and WS_{B2}). The Web Service WS_{C1} is then invoked by the Orchestration Engine running on the organization B (step 4-5, 8-9, 12-15) that sends back the responsibility to the organization A that concludes the process (step 18).

4.2 Orchestrator Engine Design

As previously pointed out, the implementation of the Orchestration Engine is based on BPEL4WS; unfortunately the BPEL specification does not cover all the aspects needed by this component: in particular no distribution of process is

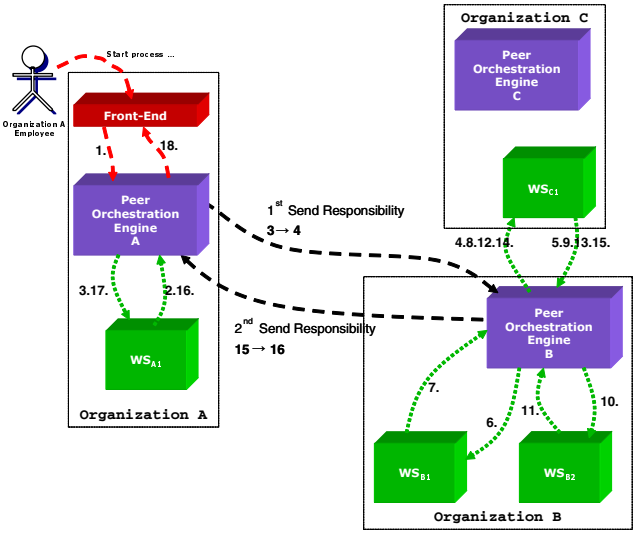


Fig. 2. An example process execution

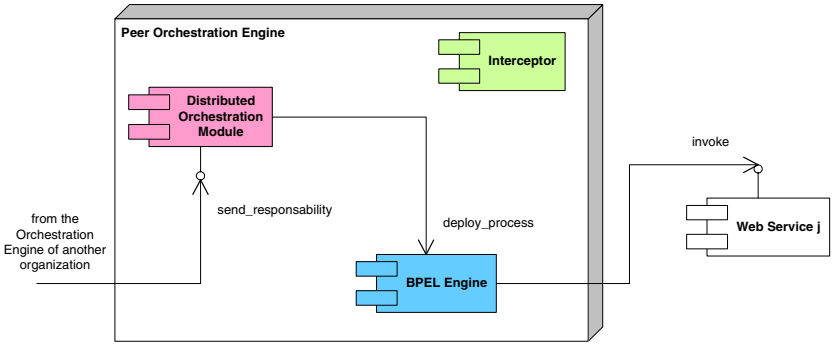


Fig. 3. Orchestration Engine Design

taken into consideration in the specification. This implies the need for a specific module managing the distributed orchestration. Figure 3 shows the design of the Orchestration Engine in the Eu-Publi.com architecture.

The **BPEL Engine** is the module containing the orchestration logic, i.e., it executes the scripts of the processes and it coordinates the involved Web Services, routing the messages in the correct order and maintaining the state of the process instances.

The **Distributed Orchestration Module** is the module implementing the distributed features not included in the BPEL specification The `send_responsibility` operation is implemented as a simple Web Service operation, i.e., this module is a Web Service exporting operations able to pass the responsibility or to accept the responsibility from another peer.

From a technological point of view, when a centralized process is created, it is deployed onto the engine of the initial organization; if there is any `send_responsibility` operation, i.e., the process is enacted in a peer-2-peer way, the process is divided into subprocess, and only the first of them will be deployed onto the initial organization. During the execution of a distributed process, the BPEL file describing the process is modified and transferred to a different Orchestration Engine, that *(i)* deploys the sub-process to execute, *(ii)* restores the state of the computation (explicitly sent by the old peer), and *(iii)* continues the computation from the point at which the old peer has interrupted its activities. From this point, the old peer is no longer involved into the process (excluding if another peer newly passes to it the responsibility for the orchestration).

Finally, the **Interceptor** is a module that analyzes the messages flow to/from the Orchestration Engine; the main feature of this module is to forward asynchronous reply to the correct destination in those cases in which the corresponding process instance is running on another Orchestration Engine, due to a previous `send_responsibility`.

4.3 Orchestration Engine Performance Analysis

Preliminary performance analysis carried out on the prototype are shown below.

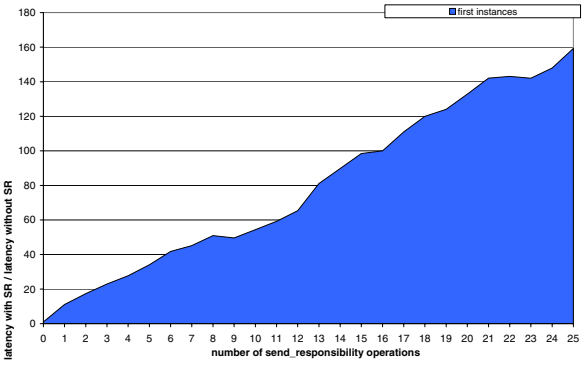
Testbed Platform. The testbed environment consists of 4 Intel Pentium IV 2,5GHz and 1GB RAM workstations running Microsoft Windows XP Professional as operating system and equipped with Java Development Kit v.1.4.2_03, Apache Jakarta Tomcat Web Server v.5.0 as application server, and Collaxa BPEL Server v.2.0RC7 as BPEL Engine. The PCs are interconnected by a 100Mbit Ethernet LAN setted-up as a single collision domain.

Experiment Aim and Description. The aim of the experiment was to evaluate the overhead introduced by the `send_responsibility` operation in a process; the overhead is evaluated comparing the latency of the distributed orchestration with respect to the centralized one. In particular, the ratio between the average latency of the process execution with `send_responsibility` operation and the average latency of the same process with no `send_responsibility` operation is shown. In the experiment the number of `send_responsibility` operation in the distributed process, has been varied from 1 `send_responsibility` to 25 `send_responsibility`.

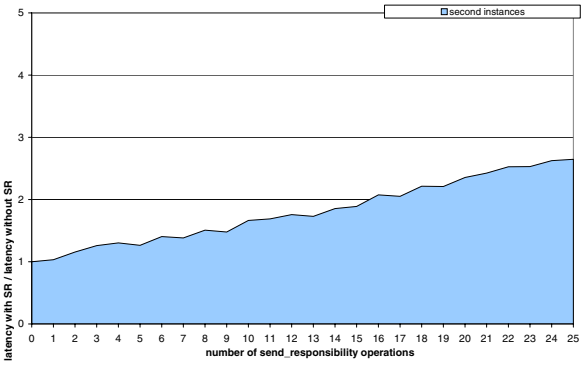
Moreover, two distinct sets of measure were conducted: the former measures the latency of the first instance of the process; the latter the latency of the other instances[3]; this separation is due to the run-time deployment of the distributed process pointed out in the previous section[4].

[3] For the first instances the average is evaluated by invoking 10 times the process, while for the other instances the average is evaluated by invoking 100 times the process.

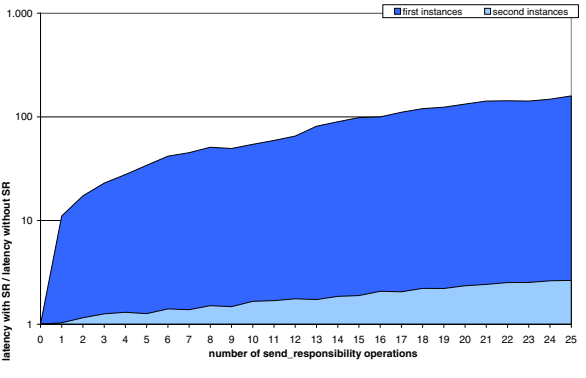
[4] In the latency of the first instances is included the time for the deployment of the sub-process.



(a) Latency of the first instances.



(b) Latency of the second instances.



(c) Comparison between latencies.

Fig. 4. Experiment Results

Experiment Result. The tests results are depicted in Figure 4; in details whereas Figure 4(a) shows the latency for the first instance of the process used in the experiments, Figure 4(b) shows the latency of the second instance.

Both figures show that latency increases in a linear way, i.e., the overhead introduced by the `send_responsibility` operation is proportional to its number. This behaviour can be predicted because, from a technological point of view, the `send_responsibility` operation is an invocation on a Web Service (more the `send_responsibility` operations are, larger the distributed process is, and more the time required for its execution is).

Figure 4(c) (in logarithmic scale) also shows that the latency of the first instance of the distributed process is about 150 times slower than the latency of the second instance: this behavior is due to the run-time deploy of each subprocess (the deploy of a process involves many activities and it results very slow with respect to the invocation on a Web Service).

5 Related Work

The Eu-Publi.com architecture is based on the notion of Cooperative Information System (CIS); various approaches have been proposed in literature for the design and development of CIS's, e.g., schema and data integration [8], agent-based methodologies and systems [9], or business process coordination and service-based systems [10]. This last approach is the one adopted in the Eu-Publi.com project, and a general architecture focusing on those characteristics typical of the *e*-Government context has been designed.

Systems adopting such an approach are typically based on a SOA, in which a clear distinction is made between an internal architecture and an external one. The first is the one allowing back-end systems to be exported as Web Services; respect on this, the Eu-Publi.com architecture provides specific support in the presentation layer and in the system wrappers. But, as argued in [6], major research work is needed in the external one, which is the one allowing the peer-2-peer cooperation among different organizations; on this purpose, the Eu-Publi.com Orchestration Engine, implementing a distributed orchestration, is a novel proposal and a first, yet simple, solution to such an issue.

The issue of supporting cooperative processes based on Web Services is quite similar to the problem of inter-organizational workflows. In the last years, several systems and approaches has been proposed to support inter-organization workflows, by extending traditional workflow management system (WfMS) technology to distributed, Internet-based scenarios: CrossFlow [11], WISE [12], MENTOR-lite [13], e-ADOME [14]. The main difference is in that the Eu-Publi.com approach assumes that services offered by different organizations are "black boxes" to be coordinated. Conversely, all such proposals adopt a "white box" approach, in which the service internal workflow schemas are knwon and used for coordinating the overall process.

With respect to systems such as *e*-Flow [15], WebBIS [16] and AZTEC[17], the novelty of the Eu-Publi.com architecture is in the distributed nature of the

orchestration engine; to the best of our knowledge, SELF-SERV [18] is the only system currently addressing a similar issue.

6 Conclusions

The paper presents the Eu-Publi.com architecture, specifically focused on *e*-Government service provisions based on orchestration of services. The approach is based on the concept of cooperative process, as unifying element among different inter-country agencies providing value-added services to European citizens.

The consideration and methodologies underlying this solution relies on a *technological improvement approach*. This approach, not requiring initial radical changes, allows to overcome the strictly constrains of norms and legal frameworks which hinder the rapid reconfiguration, in terms of interdependencies and involved actors, of business processes in the *e*-Government domain. The proposed architecture also enhances current state-of-the-art research in Service Oriented Computing, by providing a distributed orchestration mechanism which could be adopted also in many other scenarios and application domains. In *e*-Government, for instance, the distributed orchestration feature, relying on general considerations, could be reused and find applications also in other recent IST research projects such as TERREGOV [19] or SmartGov [20].

The preliminary test of the Eu-Publi.com prototype, strictly focused on the distribution feature of the Web Service orchestration, proves the approach feasibility in term of performances: an *e*-Government process in a real scenario is a long-running process and the overhead due to its distribution became negligible respect to its overall execution time; moreover, rarely, in the real world a macro-process would need tenths `send_responsibility` operations.

Acknowledgments

This work has been supported by the European Commission under Contract No. IST-2001-35217, Project Eu-Publi.com *(Facilitating Cooperation amongst European Public Administration Employees)* (`http://www.eu-publi.com/`). The authors would like to thanks the other project partners: ICE (Italy), IBERMATICA (Spain), ALTEC, NATH and CERTH/ITI (Greece). A special thank goes to Alessia Candido and Adriano Bernardini for their contributions to the development and experimentation of the prototype.

References

1. Australian National Audit Office, "Electronic Service Delivery, Including Internet Use, by Commonwealth Government Agencies", The Auditor-General, Audit Report No.18 1999-2000, 1999.
2. European Commision, "*e*-Government Indicators for Benchmarking *e*-Europe", 2001.

3. R. Traunmuller, Ed., *Electronic Government*, Springer Verlag, LNCS 3183, 2004.
4. The EU-Publi.com consortium, "Deliverable D2.1: The EU-Publi.com Architecture", 15^{th} August, 2004 (please contact authors for a copy).
5. Morgan Kaufmann, Ed., *Database Transaction Models for Advanced Applications*, A. Elmargarmid, 1992.
6. Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, *Web Services: Concepts, Architectures and Applications*, Springer-Verlag, 2004.
7. "Business Process Execution Language for Web Services Version 1.1", Document. `ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf`.
8. M. Lenzerini, "Data Integration Is Harder than You Thought", in *SIGMOD/PODS 2002*, USA, 2002.
9. J. Castro, M. Kolp, and J. Mylopoulos, "Towards Requirements-driven Information Systems Engineering: the Tropos Project", *Information Systems*, vol. 27, no. 6.
10. U. Dayal, M. Hsu, and R. Ladin, "Business Process Coordination: State of the Art, Trends and Open Issues", in *Proc. VLDB 2001*, Roma, Italy, 2001.
11. P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig, "CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises", *International Journal of Computer Systems Science & Engineering*, vol. 15, no. 5, 2000.
12. A. Lazcano, G. Alonso, H. Schuldt, and C. Schuler, "The WISE approach to Electronic Commerce", *International Journal of Computer Systems Science & Engineering*, vol. 15, no. 5, 2000.
13. G. Shegalov, M. Gillmann, and G. Weikum, "XML-enabled Workflow Management for *e*-Services across Heterogeneous Platforms", *VLDB Journal*, vol. 10, no. 1, 2001.
14. D.K.W. Chiu, K. Karlapalem, and Q. Li, "E-ADOME: a Framework for Enacting *e*-Services", in *Proc. VLDB-TES 2000*, Cairo, Egypt, 2000.
15. F. Casati and M.C. Shan, "Dynamic and Adaptive Composition of *e*-Services", *Information Systems*, vol. 6, no. 3, 2001.
16. B. Benatallah, B. Medjahed, A. Bouguettaya, A.K. Elmagarmid, and J. Beard, "Composing and Maintaining Web-based Virtual Enterprises", in *Proc. VLDB-TES 2000*, Cairo, Egypt, 2000.
17. V. Christophides, R. Hull, G. Karvounarakis, A. Kumar, G. Tong, and M. Xiong, "Beyond Discrete *e*-Services: Composing Session-oriented Services in Telecommunications", in *Proc. VLDB-TES 2001*, Rome, Italy, 2001.
18. B. Benatallah, Q.Z. Sheng, and M. Dumas, "The Self-Serv Environment for Web Services Composition", *IEEE Internet Computing*, vol. 7, no. 1, 2003.
19. TERREGOV project website, "`http://www.terregov.eupm.net/`", 2002.
20. SmartGov project website, "`http://www.smartgov-project.org/`", 2002.