A SERVICE ORIENTED DESIGN APPROACH FOR E-GOVERNANCE SYSTEMS

Rama Krushna Das [1] and Manas Ranjan Patra[2]

[1]National Informatics Centre, Berhampur, India
ramdash@yahoo.com
[2] Department of Computer Science, Berhampur University, India
mrpatra12@gmail.com

ABSTRACT

Today electronic Governance (E-governance) is no more a buzzword but a reality as countries all over the worldwide have shown interest in harnessing governance with state-of-the-art information and communication technology(ICT), in order to foster better governance. However, the inherent complexities of E-governance systems remain as a challenge for the architects to develop large scale, distributed, and interoperable E-governance applications. Besides this the dynamic nature of such applications further complicates the system design. In this paper, we present a design approach based on the service oriented paradigm for building E-governance systems. We also formalize concepts like service environment, service composition, and service collaboration which are some of the important ingredients of our design approach. In the sequel we highlight the suitability of our approach through some E-governance service provisioning scenarios.

KEYWORDS

E-governance, Service oriented design, service window, service provisioning

1. INTRODUCTION

Electronic Governance is an application of Information and Communication Technology (ICT) for delivering government services, facilitating information exchange among various stakeholders, and integrating different stand-alone systems with a view to foster better governance. It is an approach in which the Government and its citizens, businesses, and other arms of government can transact certain activities using different ICT tools and techniques. The objective of e-Governance is to make government services available anywhere, anytime to the citizens, employees, businesses, and other nongovernmental agencies in a convenient, efficient and transparent manner. It attempts to reach out to the stakeholders, even in remote areas, to provide information in real time. It tries to enhance operational efficiency and productivity by operating seamlessly among government departments and concerned agencies. The majority of e-government initiatives are aiming at improving government processes by cutting process costs, managing process performance, making strategic connections in government and creating empowerment within the government architecture. Accordingly, connection between governments and citizens (and other stakeholders) can be improved [6]. In addition, the interactions associated with the use of such rich pool of organizational and technological platforms in e-governance, creates a "task-oriented" forum of engagement between governments and other stakeholders [3][7][8]. Successful implementation of e-government initiatives depends not only on the availability of "resources" but most importantly on the adoption of appropriate implementation-oriented paradigms that describe the growth and evolution of e-governments

[11]. The following sections of the paper explains about independent servicers and interdependent services in Government departments and the approach to achieve interoperability, the ability to handle heterogeneity and scalability.

1.1. Design challenges in E-governance systems

E-governance can be viewed as a digital means of public administration which essentially facilitates the process of delivery of information and services to the public. Such systems keep evolving as governments at different levels strive to adopt new governance style for achieving higher degree of effectiveness. In practice E-governance systems are inherently incremental in nature and it is hard to foresee all future requirements at a much early stage of development. Therefore, one really finds it difficult to integrate applications developed by different parties using varieties of technologies. Typical characteristics of E-governance systems includes, they are highly interoperable, large-scale, distributed, and heterogeneous systems cutting across geographical boundaries and administrative domains. Therefore, achieving interoperability among E-governance applications towards seamless integration and information exchange is of paramount importance.

The above characteristics of E-governance systems necessitate the choice of a suitable design approach to build different applications to realize a pragmatic system. Some of the requirements of the design approach are:

I. the ability to handle heterogeneity so that independently developed systems can be integrated
II. the ability to achieve interoperability so that applications exchange information seamlessly
III. the ability to handle scalability so that new applications can be added and the system can support growing number of users
IV. the ability to provide transparency so that complex processing details can be hidden from the users thereby achieving a higher level of abstraction

2. SERVICE ORIENTED DESIGN PARADIGM

In recent years, Service Oriented Architecture (SOA) has emerged as an architectural style that advocates reuse and integration of software components irrespective of their location and implementation. It has been advocated as a suitable paradigm for crafting next-generation large scale applications. While the SOA approach strongly reinforces well-established, general software architecture principles such as information hiding, modularization, and separation of concerns, it also adds additional themes such as service orchestration, service choreography, service repositories, and concepts like service bus.

Service-orientation is a new way of thinking about software systems that require interoperability. It helps one to identify certain high-level functionalities that can be deployed as logical units. One who is interested in availing a specific functionality can possibly do so by requesting the logical unit through the provided interfaces(s) without bothering about how the functionality is actually worked out. This feature enables one to provide common interfaces to applications, thereby enhancing the interoperability and reusability of already available functionalities of applications running on different platforms. Eventually, this would help in rapid integration of applications and automate most of the business processes of organizations.

2.1. The Notion of Service

The notion of service has been defined in different way in the literature. Service has been described as an encapsulated unit of functionalities [10]. It has also been considered as a logical manifestation of some physical resources (like programs, databases, devices etc.) grouped as a process that an organization exposes to the network [9]. A pragmatic definition of service can be found in [4] wherein Service is defined as an externally observable behaviour of a software/hardware component which possibly hides the internal processing details that is required for the realization of a requested service, and is accessible only through a set of well-defined interfaces. In the context of web services, a service is viewed as an application or business logic that can run on a web server by exposing its functional capabilities to clients [2]. One thing that is apparent in all these definitions is: service is a conceptual entity that facilitates certain actions in response to a set of stimuli from its environment. The stimuli can be in the form of messages or explicit invocation programs that trigger some internal processing at the service provider, which may not be visible to the service requestor.

Based on the notion of service, software applications can be designed in an implementation independent manner using the abstract concept of service as the fundamental design entity. Each service encapsulates certain clearly specified functions while hiding implementation details of the service under reference. Such service entities can be reused and even combined to build systems/subsystems to implement higher level services. In this approach software development begins by analysing an application domain and identifying a set of services to be provisioned. These services form the fundamental design objects upon which a complete system can be built.

2.2. Service-Oriented Software Engineering

The notion of service is undoubtedly going to play a very important role for organizations that try to model their applications as a set of service entities providing certain services that can be consumed by those who request for it. In order to take benefit of the service-oriented paradigm it is necessary to develop a precise understanding of the related concepts and use them consistently right from the requirements elicitation phase through the design phase up to implementation. In other words, there is a need for developing a software engineering approach to specify what service is and systematically follow a methodology to implement applications based on the service-oriented concepts. A typical service oriented architecture is presented in the Figure-1.

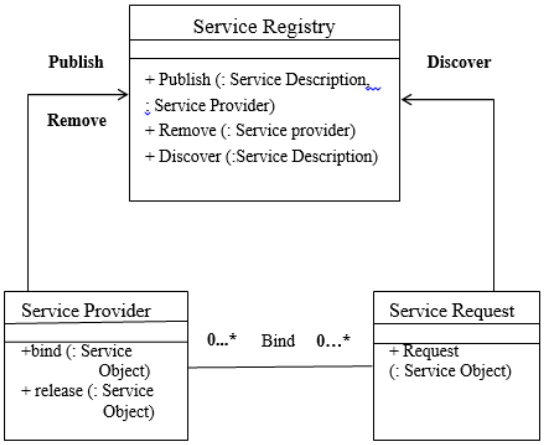

Figure 1. Service Oriented Architecture: Conceptual Model

This concept is based on an architectural style that defines an interaction model between three primary parties: the service provider, who publishes a service description and provides the implementation for the service, a service consumer, who can either use the uniform resource identifier (URI) for the service description directly or can find the service description in a service registry and bind and invoke the service [1].

2.3. Suitability of Service Oriented Design Approach for Developing E-governance Systems

There is a whole range of E-governance Systems which have been implemented by various governments all over the world. The most prominent ones can be categorized as G2C, G2E, G2G and G2B. While G2C refers to a set of services exchanged between government and the citizen, G2E is set of services exchanged between government and government employees, G2G is a set of services exchanged between government agencies, and G2B is set of services exchanged between government and the business community. The common aspect in all these applications is the delivery of service. Thus, the notion of service is already inherent to the E-governance Systems. This characteristic of E-governance Systems makes it suitable to be modelled using service oriented design approach. Moreover, one can extend or change the design objects on demand. SOA based solutions are composed of reusable services, with well-defined, published interfaces. It also provides a mechanism to integrate existing legacy applications which is an important requirement of E-governance Systems.

3. PROPOSED SERVICE ORIENTED DESIGN APPROACH

Considering the nature of E-governance applications, here we have proposed a design approach that is pragmatic and can address most of the requirements of the current E-governance system development. Our focus is to develop an approach that is realistic and very closely corresponds to the manner government services are offered to a wide variety of user groups and accessed by the citizens. It should also take into account services at different levels of abstraction, in the sense that certain services can be provided straightway whereas certain other services may require invocation of other related services in order to provision the requested service. Some of the key concepts that we employ in our design approach are: the notion of service window, service composition, service collaboration, and service enactment. Each of these concepts is formalized in the following sections using the RAISE specification language. The benefit of having such a formal description in the form of abstract specifications is to provide a precise understanding of the design concepts which can be systematically concretised to an implementation model.

3.1. Service Types

Services requested by users may require different levels of processing. Depending on the complexity of processing requirement we categorize services into three different types, namely, readily available services, composable services, and collaborative services.

Let us consider the following services of different Government departments for the province of Odisha in India, for explaining service types.

S_0: Below Poverty Line(BPL) service, provided by Panchayat Raj Department.
S_1: Antyodaya Anna Yojana (AAY) service, provided by Panchayat Raj Department
S_2: National Rural Employment Guarantee Act (NREGA) service, provided by Panchayat Raj Department

S_3: Rashtriya Sawasthya Bima Yojna(RSBY) service, provided by Finance Department
S_4: Indira Awaas Yojana(IAY) service, provided by Panchayat Raj Department
S_5: Kutir Jyoti Yojna(KJY) service, provided by Energy Department
S_6: Mamta Service, provided by Health Department

The use case diagram in Figure-2 below describes the interaction between the user and the various services described above. The RSBY Service is interdependent on NREGS Service and BPL Service, whereas KutirJ Service is interdependent on IAY Service and BPL Service. When the user invokes AAY and Mamata Service they are dependent only on BPL Service.

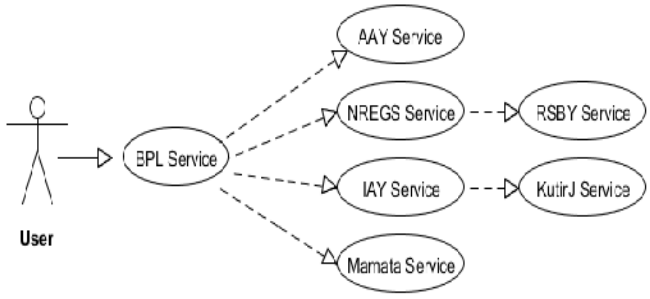

Figure 2. Use case diagram of the referred services

3.1.1. Readily available services

These are services that do not require complex processing. For instance, a citizen applies for a residence certificate. Such a request can be serviced after verifying some standard documents and can be made available to the citizen. The other similar services form Revenue Department are Nativity Certificate, Income Certificate, Social Status(caste) certificate etc.; from Health Department Birth Certificate , Death Certificate, Disability Certificate etc.; from Agriculture Department identification of small farmer and marginal farmer; from Social welfare Department Old Age Pension, Widow Pension etc.; from Welfare Department Student Scholarship, Free study books etc.; from Panchayat Raj Department NREGA Job card, BPL card etc.; The following class diagram in Figure-3 explains the attributes and operations of BPL service.

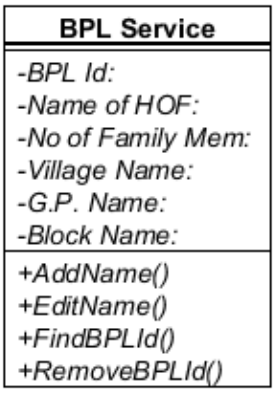

Figure 3. BPL Service Class Diagram

3.1.2. Composable Services

These services are not readily available but require further processing, possibly invoking a set of related services in the same department. For instance, permission for a building construction cannot be given immediately. This would require verifying the ownership of land, check construction regulation policies, approval of the design, etc. All such related activities/services can be invoked to facilitate the requested service. Other similar examples are: for enrolling in Swasthya Bikas Yojona, it is necessary to verify that the citizen has got NREGA job card and BPL card. In order to provide 35Kg of one Rupee per Kilo rice every month under AAY scheme it is necessary to verify that he/she has BPL card. The following diagram in Figure 4 explains the operation FindBPL() invoked for using AddAAY() operation from another service.

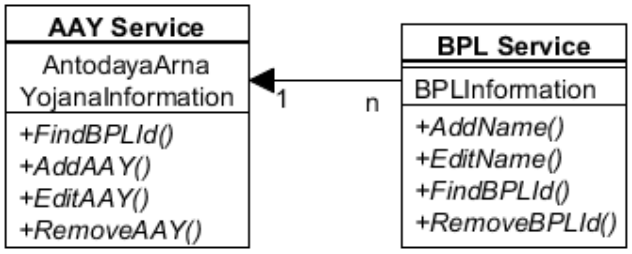

Figure 4. Class diagram of BPL and AAY Services

The sequence diagram in Figure 5 illustrates different interactions to add a Citizen name in AAY.

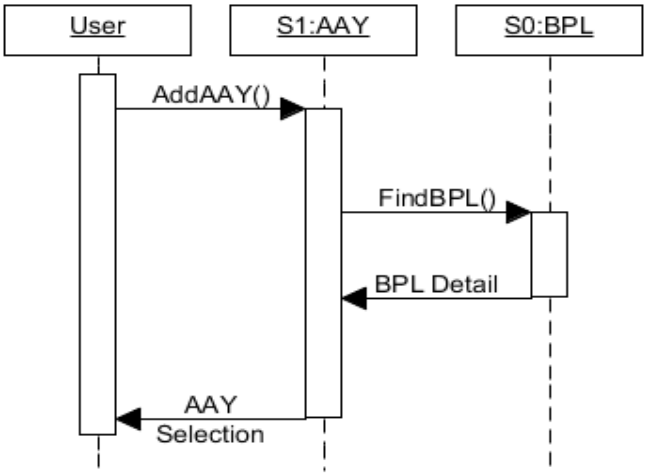

Figure 5. Sequence diagram to add a member under AAY

3.1.3. Collaborative Services

These are services which are neither available readily nor can be composed using the services available at a service window. In such a case, a service window needs to explore whether the requested service can be provided using the services of other service windows. For instance, to avail free electricity of one point household connection under Kutir Jyoti Yojona(KJY), one must

have a house constructed under Indira Awas Yojona(IAY). To have a house under IAY one must be a BPL card holder. The BPL and IAY services are provided at the service window of Panchayat Raj Department whereas the Kutir Jyoti service is provided at service window of Energy Department. The request needs to come from Energy Department service window to Panchyat Raj Dept. service window to add a new member in Kutir Jyoti Yojona. The following sequence diagram illustrates different interactions to add a new member in Kutir Jyoti Yojona.

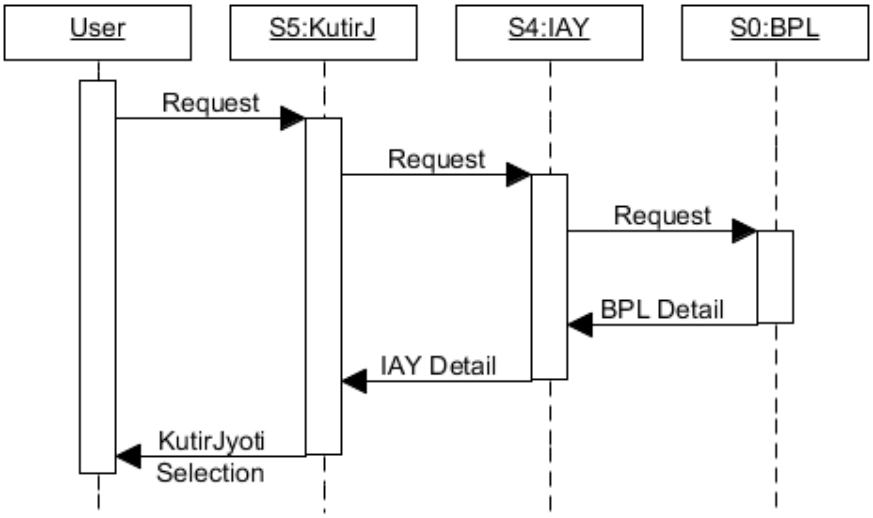

Figure 6. Sequence diagram to add a member under Kutir Jyoti

3.2. Service Window

We introduce the notion of a service window, which can refer to a physical location or a web portal where users can submit their requests for service. Typically, a service window provisions a set of well-defined services which can be accessed by interested users through appropriate service requests. Next, we conceptualize the notion of service environment which is a loosely coupled distributed environment consisting of a group of service windows. Each service window is identified by a unique identification.

The Service Environment is formally specified as a RSL scheme which includes a set of abstract types and a record type called Service window. The record type specifies entries that contribute to the service provisioning at a service window, namely, a set of readily available services, a set of service requests that it can handle, and other internal components that facilitate a requested service. It also contains information about other service windows whose services may have to be invoked in situations when a requested service is not readily available at a service window where the service request has been made. Another important component of the service environment is the service registry which is akin to a yellow page that contains information about all publicly known service windows. The service registry is described as a map type that maps each service window unique Id to a service window designed to facilitate a set of services. The change type associated with each entry of the record type above indicates that new services can be added and some existing services may be removed. Similarly, new service windows can be created and some service windows may be withdrawn.

3.3. Service Composition

```
Scheme ServiceEnvironment =
  class
        type
        Service,
        ServiceWindowId,
        ServiceRequest,
        ServiceWindow ::
                readyServices : Service-set ↔ chg_services,
                serviceRequests : ServiceRequest-set ↔ chg_servReq,
                compserv : ServiceRequest m̄ ServiceComp-set ↔ chg_servcomp,
                colbserv : ServiceRequest m̄ ServiceWindowId-set ↔ chg_servwindow,

        ServiceRegistry = ServiceWindowId m̄ ServiceWindow
        value
        deliver_service: ServiceWindow × ServiceRequest ⇀ Service
        axiom
          sw: ServiceWindow, sr: ServiceRequest •
                deliver_service (sw,sr) ≡ {s| s:Service • s ∈ readyServices (sw)}
        pre
        sr ∈ serviceRequests (sw)
    end
```

On receiving a service request a service window checks whether the service request can be complied with using a set of readily available services without much processing. The deliver_service function takes care of such requests. However, if the request needs further processing then it invokes a service composition procedure which tries to facilitate the requested service making use of relevant services accessible within the service window. This involves selecting the services and determining the order in which those are to be executed for realizing the service under consideration. The following specification describes the service composition process.

```
Scheme
        ServiceComposition = extend ServiceEnvironment with
   class
      value
        compose_service: ServiceWindow × ServiceRequest ⇀
ServiceComp-set
        axiom
          sw: ServiceWindow, sr: ServiceRequest •
                compose_service(sw,sr) ≡rng compserv (sw)
        pre
        sr ∈ serviceRequests (sw)
     end
```

3.3. Service Collaboration

In some cases the requested service is neither a readily available service nor a composable service, i.e., the service window cannot facilitate the requested service by using its own service capabilities. In the context of E-governance the effort is always to provide all requested services at a service window as far as possible using the concept of one-stop service. Thus, in our design approach we incorporate a mechanism whereby a service window can collaborate with other service windows to facilitate the requested service. This is possible with the help of the service registry which maintains a list of service window IDs along with the service requests they can handle. This concept is geared towards realizing interoperability among service windows to facilitate a whole range of services for the benefit of the users. The following specification describes the service collaboration process.

```
Scheme
        ServiceCollaboration = extend ServiceEnvironment with
  class
    value
        collaborate_service: ServiceWindow × ServiceRequest ⇸ ServiceWindowId -set
        axiom
            sw: ServiceWindow, sr: ServiceRequest •
                collaborate_service (sw,sr) ≡rng colbserv (sw)
        pre
        sr ∉serviceRequests (sw)
    end
```

4. Service Provisioning

In the previous sections we have introduced the basic design components around which an E-governance system can be developed. The proposed service window embodies all these design elements to handle different types of service requests submitted by the users time to time. A logical structure of a service window is depicted in figure 7.

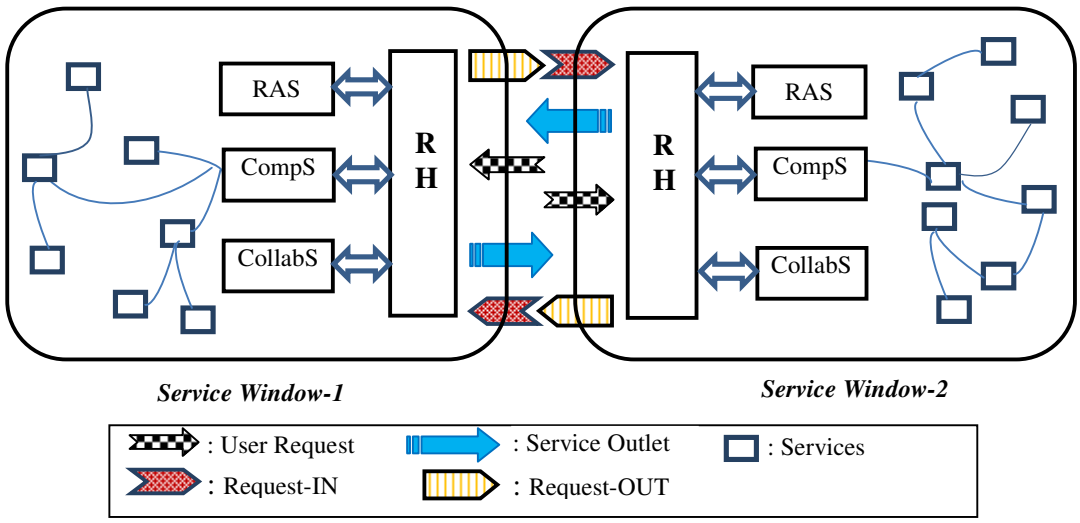

Figure 7. Interaction between Service Windows

Each service window is associated with a Request Handler (RH) module which accepts service requests and analyses them and takes appropriate action to process it. On receiving a service request, the RH module checks if the requested service is one of the readily available service categories; in that case it passes the control to the *RAS module* (Readily Available Service) to take appropriate action to respond to the user's request. However, if the requested service does not fall into this category then it checks whether it requires further internal processing, in which case the control is passed on to the *CompS module* (Composable Service) to perform necessary service composition function. But, if the service request cannot be processed locally and requires collaboration with other service windows, in that case the control is passed on to the *CollabS* module (Collaborative service). The CollabS module in turn refers to the service registry and selects the relevant service window(s) with which collaboration is necessary to realize the requested service. Each service window is augmented with four different types of channels: *User Request*, *Service Outlet*, *Request-IN*, and *Request-OUT*. Users make their service requests through the *User Request* channel. Service is delivered to users through the *Service Outlet* channel. A service window sends all its service collaboration requests through the *Request-OUT* channel and receives such collaboration requests from other service windows through the *Request-IN* channel. Collaborative service requests are processed according to some service level agreements (SLAs) between the requesting and partner service windows. In a way, a service window acts as a one-stop service provisioning point which does necessary processing on the background to facilitate services requested by users.

5. CONCLUSIONS

New ICT opportunities and achievements are constantly emerging, which needs to be adopted rapidly for effective results in e-Governance. Service oriented architecture (SOA) is a design approach to organize existing IT assets such that the heterogeneous and distributed e-Governance applications can be transformed into an integrated and simplified single window service centre for common citizens. A systematically executed SOA project using software engineering principles as discussed will help organisations to build stronger connection with service consumers and provider and provide accurate and readily available information for better governance. It will help common citizens in sharing the available information in an easy and affordable manner. These software engineering principles can further be improved to develop citizen centric e-Governance SOA models, so that the outputs of the business processes become visible to the end-customer, who will be able to select and combine different services to configure unique solutions that meet his/her personal needs.

REFERENCES

[1] Arsanjani, Ali, (2004), Service Oriented Modeling and Architecture, Web Service Centre of Excellence, IBM
[2] D. Greenwood, M. Calisti, Engineering web services-Agent integration, IEEE proc. 2004.
[3] Davies, T. R. (2002). Throw e-gov a lifeline. Governing, 15(9), 72
[4] G. Lomow , E. Newcomer, Introduction to SOA with web services, Addison Wesley, 2005.
[5 G.P. Kumari, B. Kandan, A.K. Mishra, "Experience sharing on SOA based Heterogeneous Systems Integration." 2008 IEEE Congress on Services 2008 Part-I.
[6] Hussein Al-Omari and Ahmed Al-Omari "E-government readiness assessment model," Journal of Computer Science, (2006). Vol. 2, No. 11, pp. 841-845
[7] John Carlo Bertot, Paul T. Jaeger, Charles R. McClure "Citizen-centered e-government services: benefits, costs, and research needs." DG.O 2008: 137-142.
[8] Klaus Lenk, Roland Traunmüller (2002) Electronic Government: Where Are We Heading? EGOV 2002: 1-9

[9] Webber, J. & Parastatidis, S. (2004), "Demystifying service oriented architecture", web services journal.
[10] V. Talwar, Q. Wu et al, "Approaches for service deployment", (2005) IEEE Internet computing, March-April, pp 70-80.
[11] Tagelsir Mohamed Gasmelseid: "A Multiagent Service-oriented Modeling of E-Government Initiatives." (2007) IJEGR 3(3): pp 87- 106

Authors

Rama Krushna Das is Technical Director (Scientist-E) working with National Informatics Centre (NIC), Department of Electronics and Information Technology, Government of India. His research and professional career spans over twenty five years of coding, research and capacity building in computing, e-governance and related subjects. His expertise is primarily in the domains of Electronic Governance, Implementation Architectures and Strategy, and Software Technology. He is presently involved in development and implementation of different E-Governance projects of NIC. He has published several peer-reviewed papers as journal articles, book chapters, and contributions to conference proceedings. His research interests include e-governance, software engineering, Service Oriented Architecture and Cloud computing. He is a life member of Computer Society of India (CSI) and a professional member of the Association for Computing Machinery (ACM).

Dr. Manas Ranjan Patra holds a Ph.D. Degree in Computer Science from the Central University of Hyderabad, India. Currently he is an Associate Professor in the Post Graduate Department of Computer Science, Berhampur University, India. He has about 25 years of experience in teaching and research in different areas of Computer Science. He had visiting assignment to International Institute for Software Technology, Macao as a United Nations Fellow and for some time worked as assistant professor in the Institute for Development and Research in Banking Technology, Hyderabad. He has about 90 publications to his credit. His research interests include Service Oriented Computing, Software Engineering, Applications of Data mining and e-Governance. He has presented papers, chaired technical sessions and served in technical committees of many International conferences.