A Model for Component Based E-governance Software Systems

A.SHRABAN KUMAR¹, G.JAYARAO², B.SHANKAR NAYAK³, KBKS. DURGA⁴, A.ESWARA RAO⁵

¹,²,³,⁴ Associate Professor CSE, St.MARTIN’S ENGINEERING COLLEGE, Hyderabad, India,
³Assistant Professor CSE, St.MARTIN’S ENGINEERING COLLEGE, Hyderabad, India,
⁵Assistant Professor CSE, St.MARTIN’S ENGINEERING COLLEGE, Hyderabad, India,

Abstract

E-governance is a plan for the national interest and in order to meet its objectives at least some of their modules, must be reusable by other software Component-Based Development is an approach to develop a software system with the assistance of reusable software components, may help to reduce the development time and costs and will increase reliability and maintainability of such systems. This paper suggests that some component development approaches for developing the software components that can be reused in component based E-governance software systems. The proposed method can be used as a reference model for the development of software components for E-governance solutions.

Keywords: Components, Legacy E-Governance system, electronic government applications.

1. Introduction

E-Governance is a network of organizations to include government, private-sector and nonprofit entities. The importance of E-Governance Standards is now widely recognized globally. The e-governance Standards and specifications with well defined formats will help to achieve communication systems and interoperability of information. To drive both the integration between specific Government functions shared Government functions and data sharing between applications an overall E-Governance application architecture requires to be defined.

E-Governance is defined as the application of electronic means in (1) the interaction between government and citizens and government and businesses, as well as, (2) in internal government operations to simplify and improve democratic, government and business aspects of governance. The objective of E-Government is to fulfill the public’s needs and expectations on the front office side, by simplifying their interaction with various online services. Internal strategic objectives of E-Government in government operations are to facilitate a speedy, transparent, accountable, efficient and effective process for performing government administration activities, and which should result in significant cost savings in government operations. The aim is also to improve the quality of services and to provide greater opportunities for participation in democratic institutions and processes. However, sometimes E-Government is defined as the electronic service delivery to citizens, but those working in the field maintain that E-Government is concerned with far more than simply Making some public information and citizen services available on the Internet.

2. E-Governance Architecture

Present mode of systems development is largely based on proprietary standards that provide limited scope for information exchange and integration with other systems. Legacy systems need to be integrated with the present technologies without compromising the information and data.

2.1. Reusability

A large number of E-Governance applications developed in isolation, resulted in self-contained islands of information. Every application followed its own standards with the main objective of delivering process-centric results irrespective of other applications. Hence, at the outset there is a need for an architecture that is aimed at reuse and customization. Software reuse must become a key part in the software architecture. The continuing need for speeding up the software development life cycle and reducing the cost of development to manageable proportions exists in all large software projects. Component ware Architecture enables efficient reuse of existing application assets, faster deployment of new applications, and improved responsiveness to changing business needs. Reusable software components are the building blocks that make a system able to respond quickly to change.

2.2. Middleware and technology standards

There is a need to introduce the concept of middleware and technology standards as a tool to develop integral, scalable and robust E-Governance solutions, while employing multiple solution providers. The middleware should support processes involving multi-department and multi-agency workflows. For this purpose, it is necessary that the different department offices and also external agencies are interconnected and share the same underlying back-end databases and applications. The middleware also should be able to facilitate integration with legacy systems. Middleware needs to provide services such as identification, authentication, authorization, directories, and security to all applications. By
promoting standardization and interoperability, middleware will make advanced network applications much easier to use. The key middleware components are (a) Web Application Server, (b) Inter-application communication and messaging and collaboration software (c) Language and data interchange standards.

2.3. Service-Oriented Architecture (SOA)

Service-Oriented Solutions: Applications must be developed as independent sets of interacting services offering well-defined interfaces to their potential users. Similarly, supporting technology must be available to allow application developers to browse collections of services, select those of interest, and assemble them to create the desired functionality. A service is generally implemented as a coarse-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled message-based communication model.

3. Requirements of E-Governance applications

A typical architecture for e-governance application is depicted in the Fig. 4. It is well known that, most of the E-Governance solutions are Internet/Intranet based applications except for a few stand alone applications. The web based services are portals as single-point access interface networked with enterprise backend applications, databases, and legacy systems and also to external systems, which basically involves seamless interaction between repository of data and the content management and access systems. The issues that primarily affect the e-governance applications are interoperability between the heterogeneous systems and data repositories, absence of standard taxonomy, absence of compliance to best practices for seamless web-accessibility and lack of internationalization of the multimodal application interface.

![Figure 1: A typical E-Governance application](image)

4. E-Governance and CBSE (Component Based Software Development)

In this section the basic theoretical concepts of Component Based Software Development CBSD) are provided. Firstly, some standard definitions of software components and component based development are given and then a brief description of the activities of Component Based envelopment follows CBSD is a process that emphasizes the design and construction of a software system using reusable software components. CBSE encompasses two parallel engineering activities Domain Engineering and Component Based Development. Domain Engineering explores an application domain with the specific intent of finding functional, behavioral and data components that are candidates for reuse. These components are placed in reuse libraries. Component-Based Development elicits requirements from the customer, selects an appropriate architectural style to meet the objective of the system to be built and then selects, qualifies, adapts and integrates the components to form a subsystem and the application as a whole. The new paradigm of assembling components and writing code to make these components work together has a name, and of course an acronym, namely Component-Based Development, while the whole discipline including components’ identification, development, adoption and integration in larger software systems is called Component-Based Software Engineering. Professionals and
researchers, both agree that by decomposing system into a set of software components, reusability and maintainability can be achieved. This will result in reduced time-to-market and high quality and scalable software applications. CBD includes parallel development activities i.e. development of software components and development of software with ready made components. Component development is considered as an important activity. The two processes can be performed independently of each other. In general terms, a component is considered as a part of a system that performs certain functionality for that system. In a similar analogy, a software component can be defined as an independent executable unit that performs certain functionality when it is plugged into an application software system. Some standard definitions of software components are given below.

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party software component is a software element that conforms to component model and can be independently deployed and composed. Without modification according to a component standard

4.1. Component Development by Existing Frameworks of E Governance Systems

Frameworks are just one of many reuse techniques. They are an object-oriented reuse technique and are an intermediate form, part code reuse and part design reuse. “Frameworks are widely used to reuse design of all or part of a system that is represented by a set of abstract classes and the way their instances interact”. A framework is a collection of several fully or partially implemented components with largely predefined cooperation patterns between them. If frameworks for the E-Governance services are available, then it would be a good idea to develop reusable software components from the existing frameworks. Since frameworks are designed after making a proper domain analysis, it would thus be very easy to sort out common services from these frameworks and make software components for those services. Components developed from Frameworks, will simplify the task of developing component for E-governance based solutions.

To demonstrate this idea, we have used an example framework (based on literature) for a Primary Health Center (PHC) and the primary health care system of India has been used as an example. India’s rural healthcare system is based on the Primary Health Centre. The main activities of a PHC are the promotion of food supply and nutrition, education relating to both health problems and their prevention, maternal and child health care, immunization, awareness programs, treatment of common diseases, making essential drugs available, early warning of epidemics, deployment of control measures etc. A PHC manages different type of records including those relating to local citizens’ records, disease records and daily work routine. A PHC collects data from sub-centers and also sends/receives data/information to the respective community health centers. An example framework of PHC services is depicted in Figure 2 given below.

![Figure 2: An example framework of a Primary Health Center.](image-url)
By conducting a proper analysis of the framework, the following common services can be identified. Extracting Components by Component Oriented Reverse Engineering Due to the constant change in technology and business rules, the majority of the legacy E-governance systems require extensive patches and modifications. In the majority of cases, the original developers are not available and development methods used in legacy E-governance systems may also become outdated. It will become too difficult to maintain such systems. In spite of this, it is not possible to ignore the importance of legacy E-governance systems.

Legacy E-Governance application systems, particularly some of their functions, are too valuable to be discarded and too expensive to reproduce. Software Reverse Engineering attempts to understand the existing S/W by reverse engineering and redesigning and scaling it up into a new S/W as per newer demands. In this case there is the opportunity to create components that will not only become part of this new E-governance system, to be obtained by forward engineering, but will be reused multiple times in many E-governance systems to be engineered in the future.

A straightforward reverse engineering approach should be suitably modified to integrate component creation activity. The procedural nature of Legacy E governance software may naturally yield components based on procedural thinking. Such components may not have many features of modern component technologies. Thus, these components may not be very suitable for component-oriented programming. A two-way approach, wherein the structure of the business component may be separately identified whereas its methods (functions) and attributes may be extracted from the legacy E-governance software by modifying the traditional reverse engineering process, is required in this case. The following are the activities necessary according to this process model. Analyze the E-governance domain and decompose it in terms of component. Firstly, the problem must be analyzed and decomposed in terms of components. Define the component structure. A component should be specified such that it covers all the necessary information to assist a developer in integrating components into their systems. The interface and semantic specification parts can be designed at an earlier stage and the implementation part of the component can be brought from the legacy systems. Perform component identification from legacy E-governance systems. The component identification exercise firstly requires the software developer to gain an understanding of the legacy E-Governance system. A software system can be understood in the following terms: the different elements of the system and the relationship that exists between those elements. Adding non-functional and other semantic properties in components. After refining the computing unit, the next step is to place them into a component framework. It includes the addition of code for non-functional requirements, providing some specific documentation, customizing it according to specific requirements etc.

5. Limitations and Overheads of Proposed Approaches
Component based E-governance systems would be easily modifiable as it is possible to change components with the newer versions. Such systems would be easily maintainable as a component can also be added, deleted and replaced in a system at run time. Component based E-governance systems would be more modular, so it is very easy to test the software. The description above has highlighted the need for lightweight compostable components and raises a few important questions concerning the solution of the proposed component development models. The idea of these component development models is very useful to generate lightweight components. The realization of this idea will assist a component development organization to develop components according to the requirements of a specific E-Governance application. The question is whether the complete automation of the 'Component Generation Model' is possible or whether some software environment would be necessary for this purpose. The Component Development Life Cycle for such components would be different to that for the current development and delivery model. The domain analysis for the identification of components would also be different. Requirements would be classified in two parts: the fixed part and its variants. A clear separation of common attributes and functionalities of a generic component and its possible variants is necessary. It should be very clear which information should be in the generic components and which should be in their variants. Design, coding and unit testing processes may have some newer issues that would not appear in a conventional E-Governance development process. The question is whether the proposed components would be tested before depositing them into the repository or whether the testing process will be performed after the development of a final component. We propose the idea of component development models. The process of design and development of the proposed component development models are difficult. The nature of the repository for storing such components would also be different. The management of these repositories within the organization would involve some extra overheads. This is also an important issue as it involves how much time, effort and expertise will be required in order to realize the idea of the proposed component development models for developing components for E-Governance systems.

6. Conclusion
The aim of this work is to point out the possibilities of applying the component generation methodologies for the development of components for component based E-governance software systems. The use of such components in software development will result in both quality improvements and cost reductions. We propose some methods for component creation that address the question, albeit in rather limited sense. The approaches suggested in this case are only at the theoretical level and more work is required by both the academicians and professionals in order to perfect
these approaches. Future work may include the development of a process model for such Software Components and a study of other software component related issues.

References