IJPIS

SOME COMPONENT GENERATION APPROACHES FOR E-GOVERNANCE SYSTEMS

RATNESHWER
Dept. of Computer Science
MMV, Banaras Hindu University
Varanasi, India
ratnesh@bhu.ac.in

A K TRIPATHI
Dept. of Computer Engineering
IT, Banaras Hindu University
Varanasi, India
anilkt@bhu.ac.in

Abstract

E-governance is a plan for the national interest and in order to meet its objectives within the given timescale, more and more applications, or at least some of their modules, must be reusable by other software. It is possible to reuse the solutions developed for one government, by another government. What is required at this stage is a move from more traditional software development to a reuse based development. Component-Based Development (CBD), 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 some component development approaches for developing software components that can be reused in component based E-governance software systems. The proposed methodologies can be used as a reference model for the development of software components for E-governance solutions.

Keywords: Components, Template Component, Legacy E-Governance system

1. Introduction

A government is responsible for providing the basic services and information to its public. For example, in a country like India, there are over 1500 organizations [Prasad, 2003], functioning directly under the auspices of the Government of India. In addition, similar setups are available in 28 states within the union. Overall, over five million personnel are engaged in day to day activities of the union government and are involved either directly or indirectly and are spanning a large spectrum of activities. It is a challenging task to provide services and information to the public within an appropriate timescale. E-governance, the application of advanced information and communication technologies to improve governance, has proved to be a better option for meeting the objectives. Governments [Backus, 2001] all around the world are attempting to utilize ICT for various purposes. The initial motivation usually comes from the need to improve the efficiency of processes within the

government. An additional reason may come from the need to provide various social services to the citizens and a third reason might be to strengthen the democratic foundations of governance (opinion polls, voting etc). For example, India is a country of diversity, and different types of services are provided by the government, such as central services (Income Tax, Central Excise, Passport/Visa etc), state services (Agriculture, land records, transport etc), and integrated services (Governance gateways, courts etc) etc. The use of ICT in governance, facilitates efficient, speedy and transparent processes for disseminating information to the public and other agencies and for performing the government administration activities. A large number of software, hardware, finance and human resources will be required in this case in order to accomplish the task. It would not be possible to fulfill all the software and hardware requirements in one step and thus a phased approach based on a priority basis is more appropriate. These applications should be developed in parallel in order to reduce both the development costs and the time involved. The rationale behind [Mittal et al., 2004] such a solution is that, as is the case in business that around 85% of the processes are actually the same across those firms within the same industry so it is also the expectation that 85% of the processes should be similar across different governments. Thus it should be possible to reuse the solutions developed for one government, for another government. What is required at this stage is to make a move from a traditional software development to one involving a reuse based development.

Component-Based Development (CBD), an approach to develop a software system with the assistance of reusable software components and this may help to reduce both the development times and costs and will increase reliability and maintainability of such systems. Component development is considered as a separate activity in CBD. In CBD, components developed once, can be reused many times in various applications. Software components for various E-governance services can be developed and stored in a repository and can be used at a later stage in an E-governance software development (using component based approach). The availability of these components would result in both more rapid development times and a reduction in the development costs. The idea is to design E-governance related software components with attributes and functionalities that will always be essential in its any deployment with some scope for the addition of, or modification in, functionalities as per the requirements of specific state/central government applications. It is also possible to modify the straightforward reverse engineering approach to integrate component creation activity and extract components from legacy E-governance software. Such component generation methodologies would be helpful in creating components for E-governance systems. It would be possible to trust these components because they would have been tested many times before being deposited into a Component Repository. E-Governance Software, developed by such components, would be highly reliable. This paper suggests some component development methodologies for developing software components that can be reused in component based E-governance software systems. The proposed methodologies can be used as a reference model for the development of software components for E-governance solutions. For the sake of familiarity, the Indian government services have been used as an example ,but these approaches are applicable for government services of any country. This work initiates a discussion and also requires more extensive research oriented studies to be conducted by professionals and academicians in order to perfect these approaches.

The paper is organized as follows. Related works are briefly described in section 2. In sections 3 and 4, brief introductions to E-Governance and CBSE have been

given respectively. In section 5, CBSE prospective of E-governance has been discussed and some component development approaches for an E-governance software system have been proposed in its various subsections. The discussion and overheads associated with the approaches are given in section 6. The paper is concluded in section 7.

2. Related Work

Researchers and practitioners have been considering various aspects of component based E-Governance software solutions and their related issues. Considerable efforts have been spent, both by academia and industry, to advance the state-of-the-art component based E-Governance solutions. Some of these efforts are summarized below.

The visions associated with E-Governance are to provide fast, improved and more efficient services, shared resources and services, productivity increase etc. S. Payrelal has emphasized the need of interoperability between system sources from different vendors as software components from one vendor can be applicable to different e-Governance applications [Pyarelal, 2008]. Vaisya and Tiwari [Vaisya and Tiwari, 2008] have suggested the use of large software components that are reusable across applications. A white paper published by IAC EA-SIG [Butter et al., 2003] has supported the use of Component Based Architecture and given some guidelines so that sound business decisions can be made with respect to this key technological approach and also addressed the requirement for a component repository for E-Governance solutions. The paper [Pyarelal, 2005] promotes reusability through software components in E-Governance software solutions. It also suggested the use of middleware components such as web application server, inter-application communication and messaging and the collaboration of software, data interchange standard etc. [Beer et al., 2006] have proposed a component based software architecture for E-Governance applications. In their approach they modeled software components as work flows and executed these by means of an underlying workflow management. BITS Pilani (India) is working on a project named “eThens”, which is a Component based framework for E-Governance (see slideshow by Sunder and Harsh)**.** With the infrastructure in place, government agencies are able to swiftly develop and deploy E-Government services through reusable software components. For example, with the assistance of the E-Service generator and its reusable components, a scheme in which two million Singaporeans were eligible to apply for shares online was made available within a fortnight [NCS, 2005]. Debnath et al. have presented the feature model for E-Governance systems. They also described an approach to integrate formal specifications with feature models in order to produce guidelines to be used in the context of specification reuse through software components [Debnath et al., 2008].

It is obvious from the above paragraph that Component based development of E-Governance solutions do facilitate faster, improved and efficient services, productivity increase, shared resources and services etc. This paper attempts to extend the above contributions further by proposing some means for component development that would be highly reusable and composable in nature. This initial proposition of such methods may be purposefully employed by the professionals and the corresponding useful feedback may be analyzed. It calls for further extensive research oriented studies, by all concerned, in order to perfect the details of the model.

3. E-Governance

In this section some definitions of E-Governance are given.
According to the World bank website (2005) e-Government can be defined as: "information technologies… that have the ability to transform relations with citizens, businesses, and other arms of government … [and] can serve a variety of different ends; better delivery of government services to citizens, improve interactions with business and industry, citizen empowerment through access to information, or more efficient government management…benefits can be less corruption, increased transparency, greater convenience, revenue growth, and/or cost reduction." E-Government is the term that reflects the use of information and communication technology (ICT) in public administration to change structures and processes of government organizations **[**Lenk and Traunmüller, 2000**].**

E-Governance **[**Bhattacharya, 2002**]** 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 external objective of E-Government is to satisfactorily fulfill the public's needs and expectations on the front office side, by simplifying their interaction with various online services. The use of ICTs in government operations facilitates speedy, transparent, accountable, efficient and effective interaction with the public, citizens, business and other agencies. 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 [Lambrinoudakis et al., 2003]. 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. "E-Government runs wide across all aspects of government, deep within the core of every governmental entity, and will inevitably be a transforming agent for government and governance." [Curtinet et al., 2004]. Given the history of poor governance in developing nations for instance, E-Government applications may provide a feasible and affordable platform to enhance governance in these countries **[**Grönlund and Horan, 2005**].**

The following key issues in E-governance were identified and addressed: E-governance trends, E-governance evolution, E-governance usage, E-governance websites, E-government services, connectivity, E-governance readiness, citizen participation, E-governance technology, change management and funding. The following major challenges in E-governance were identified: trust building in E-governance, ICT management, and privacy and security **[**Palanisamy, 2004**].**

4. 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 Development 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 [Pressman, 2001, p. 847]**:** Domain Engineering and Component Based Development. Domain Engineering explores an application domain with the specific intent of finding functional, behavioural 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 [Panfilis and Berre, 2004] of assembling components and writing code to make these components work together has a name, and of course an acronym, namely *Component-Based Development (CBD)*, while the whole discipline including components' identification, development, adoption and integration in larger software systems is called *Component-Based Software Engineering (CBSE)* **.** Professionals and researchers, both agree that by decomposing a 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 readymade components. Component development is considered as an important activity. The two processes (one for the creation of components and the other for the creation of Component Based Software development) can be performed independently of each other [Tripathi et al., 2008]. 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." [Szyperski, 1999].

> "A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a component standard." [Councill and Heineman, 2001, p. 7].

4.1. Activities of the Component Based Software Development

From a component-based perspective the process of system design involves the selection of components, together with an analysis regarding which components can be acquired from external sources (e.g. COTS) and which must be developed from scratch. The engineering of component based systems can be considered to be primarily an assembly and integration process. The vertical partitions depicted in figure 1 describe the central artifact of component based systems- the components in various states.

4.1.1. Off-The-Shelf Components [Brown and Wallnau, 1996]

Components come from a variety of sources, some are developed in-house (perhaps used in a previous project), others are specially purchased from a commercial vendor. Selection and evaluation of software components are the key activities that take place

at an early stage in the life cycle of a Component Based System. The evaluation approach typically involves a combination of paper based studies of the components, discussion with other users of those components, and hands on benchmarking and prototyping.

4.1.2. Qualified Components [Brown and Wallnau, 1996]

Typically, a component is described in terms of an interface that provides access to component functionality. In many cases this interface is the main source of information about a component. To make use of a component aspects of the components performance, reliability, usability etc. must also be understood. While a great deal of information relating to a component and its operation can be found through hands-on evaluation, a number of gaps will remain. In this process, components are qualified according to their functional and performance requirements.

4.1.3. Adapted Components [Brown and Wallnau, 1996]

The variety of sources for components leads to a number of problems. These arise because stand alone components are being used to construct a system where the components must cooperate. The result is a number of conflicts with respect to concerns such as sharing resources, version control, environmental setup etc. As a consequence of these conflicts, components must be adopted based on a common notion of component citizenship (i.e. based on rules that ensure conflicts among components are minimized). This usually involves some form of component wrapping.

4.1.4. Assembled Components [Brown and Wallnau, 1996]

The assembled components are integrated through some well defined infrastructure. This infrastructure provides the binding that forms a system from the disparate components.

4.1.5. Updated Components [Brown and Wallnau, 1996]

As with any system, a Component Based System must, over time, involve the fixing of errors and the addition of new functionality. In order to repair an error, an updated component is swapped for its defective equivalent thus treating components as plug replaceable units. Similarly when additional functionality is required it is embodied in a new component which is added to the system.

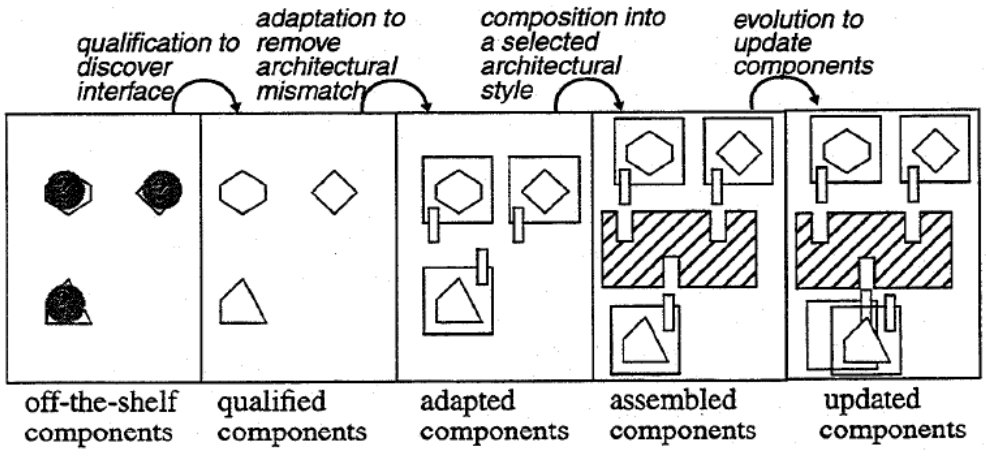

Figure 1: Model for architectural assembly of components.

5. E-Governance and CBSE

In this section a component based approach for E-Governance applications is to be discussed. Some approaches to software component development (apart from conventional component development approaches) are also suggested.

It is often the case that [Chandrashekhar, 2005] similar applications are developed by different groups but there is no possibility of interaction between these functions within the organization or with outside agencies. It is possible to develop these functions to be sharable across systems by developing them as reusable components. These components can be developed independently from user interfaces. CBSE can play a very significant role in developing E-Governance software. The idea is to analyze a particular E-governance domain (for example Income Tax, Health Systems, Police etc) and identify the common functionalities within that domain. Such a common functionality can be developed as a component and reused in all applications belonging to that domain. Using the open system approach and reuse technology, it is possible to develop the reusable software components and make them widely available so that vendors and software developers can create their own customized version by using them. Since software components are accessed through their interfaces and these are written in interface definition language, it should prove relatively easy to plus them into software systems. It should also be relatively easy to maintain them. Software components are designed and implemented to be reliable and hence ensure a high reliability of a system. The major objective should be to identify common functions in governance services and to develop software components and make use of them in as many software applications as possible. Software components can be developed for a variety of purposes including database components (citizen records, land records etc., for example a citizen database component can be used in more than one application (within a state) such as a voter list application, health application and other similar functions may require such databases.) , entity components (components that map real world objects such as passports, court, offices etc) and service components (various calculation functionalities, mathematical and relational functionalities, transfer of data/control etc). Components can also be developed for networking and operating system level services. These components may be stored in a commercial repository in order for it to be possible for any user to use the components. At the time of software development, appropriate components can be selected and be integrated into the application.

In the following sub-sections we propose some reuse based component development approaches (apart from conventional component development approaches) that may be useful for E-Government solutions.

5.1. Component Development by Existing Frameworks of E-Governance Systems

Frameworks are just one of many reuse techniques [Johnson et al., 1997]. 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 [Fontoura et al., 2000]. 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 realting 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.

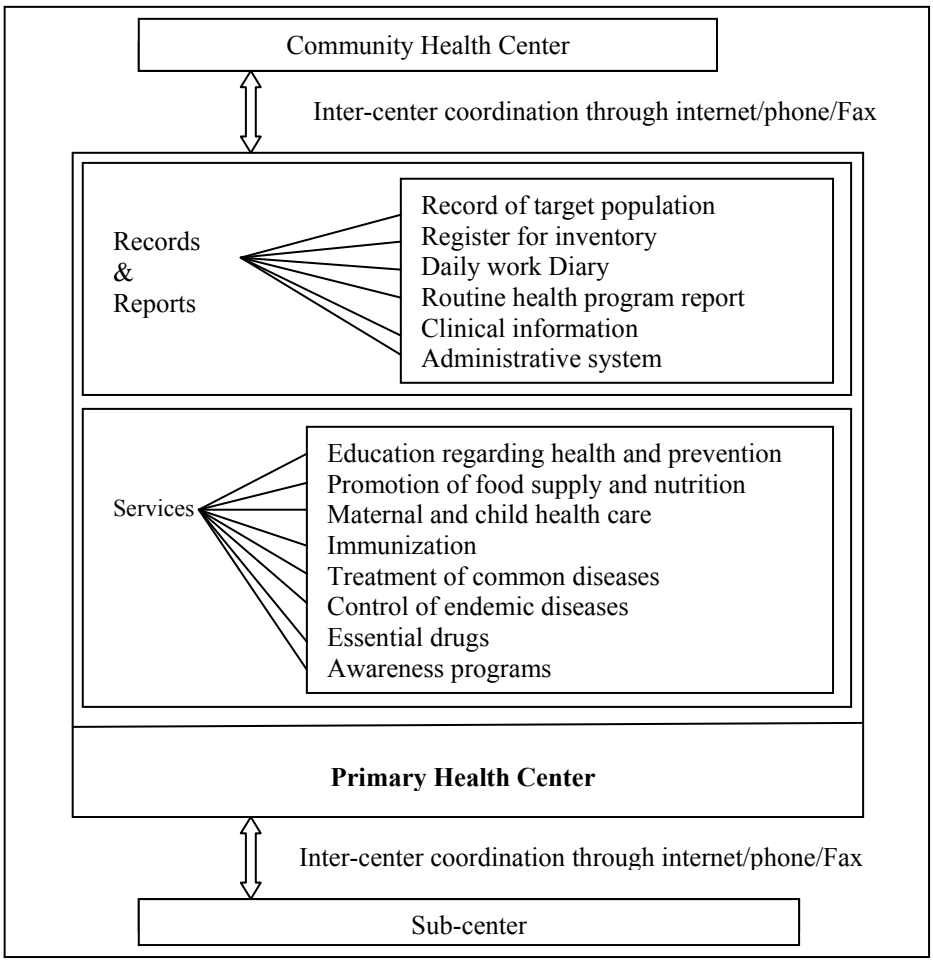

Figure 2: An example framework of a Primary Health Center.

By conducting a proper analysis of the framework, the following common services can be identified.

Database Components: Local Citizen Database, Local Disease Record, Inventory Record Format, Daily Work Diary format, Routine Health Program Report format, Clinical Information, Employee records, Traditional medical expert systems, Disease record in rural language, Symptoms record for disease, Special diseases for rural areas spread by cattle etc. etc.

Awareness program Components: education for common diseases and their cure, importance of clean atmosphere, precautions at delivery time, prevention from endemic diseases, information about various government schemes etc.

Entity Components: Management of PHC, Clinical Record, Medical Prescription, Touch Screen Information System etc.

*Service Components***:** Information Retrieval System, Data Transfer System, Data Modification System, Printing System, Middleware Components etc.

These common services, extracted by the framework, can be developed as components and can be stored in a central repository. In India, the actual delivery of health care is the duty of the state governments within the framework laid down by the central ministry of health. In India, there are 28 states and primary health services for states are more or less similar. Software components developed for one state can be reused in other state E-Governance software systems. If any organization prepares a central repository of such components then a software developer has the ability to take advantage of these components and is thus able to develop software systems in less time and at a reduced cost.

5.2. 'Component Template' for Component Based E-Governance systems

In the majority of the E-governance services provided by the states, the basic functionality is the same, but some business rules may be different for different states. For example, the basic function of revenue collection system would be same in the majority of the states but some minor variations nay occur in a particular state. Software components should be developed in such a manner that they can be used in all states E-Governance solutions. One possibility is, to add all possible business rules (of state governments) within the component's internal logic. In practical terms, this would be inefficient as the size of the component would become extremely large. The other possibility is to use lightweight components that express the internal logic of a particular E-governance service in a base class and use a common generator function that will generate the component according to the variations or modifications required by the specific state's E-governance application. The idea is to develop a component in semi-code form (the component will have attributes and functionalities, some of them possibly in abstract form, common to all the applications) and store it in a repository. A component may have different variations in functionality for different states' applications. In another part of the repository, all the possible variations in functionality (for different state services) of a component will be stored. For example, consider the case of a software component 'Student Marking System'. The functionality of the component may have some variations for different states such as statistical grading might be used in one state whereas absolute grading is used in another state; there may be a semester system in one state or a year system in another state etc. A generic component of "Student Marking System" (having common

attributes and functionalities)must firstly be developed and this will be stored in a repository. In another part of the repository there would be the possible functions (such as one variant for statistical grading, one variant for absolute marking, one variant for semester system, one variant for yearly system etc) corresponding to this 'generic component'. A numeric binding scheme can be used to show the relation between the generic component and its corresponding functional/non-functional properties. For example a generic component, in one repository, is coded a 'C' then the corresponding properties, in another part of the repository, can be termed as C1.1, C1.2, C1.3 etc.

If a software developer requires a specific component for his/her application, then the component developer will use some sort of generation function that will choose the generic component and the desired functionality (as required by the specific state application) and by combining both, the final component will be delivered. The 'generic component' can be transformed into a lightweight component as per the requirements of the application and it itself carries only those functionalities and attributes required in this particular application.

The basic idea is that:

1. An E-Governance service component should be designed with attributes and functionalities that will always be essential in any deployment, and

2. It should have scope for the addition of, or modification in, other functionalities as per the requirements of the specific application for the different state governments.

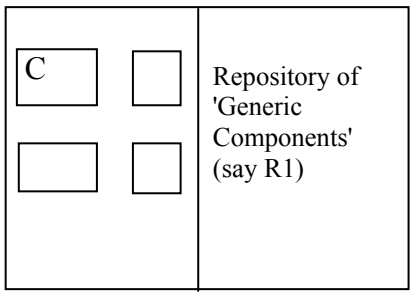

Figure 3(A): Repository of Generic Components

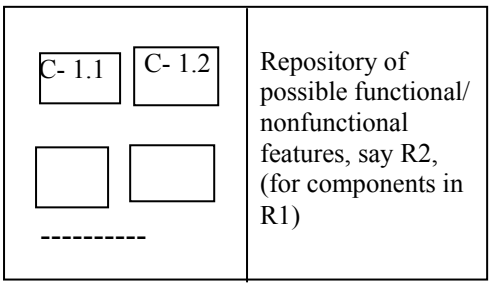

Figure 3(B): Repository of possible functional/nonfunctional properties corresponding to components in R1

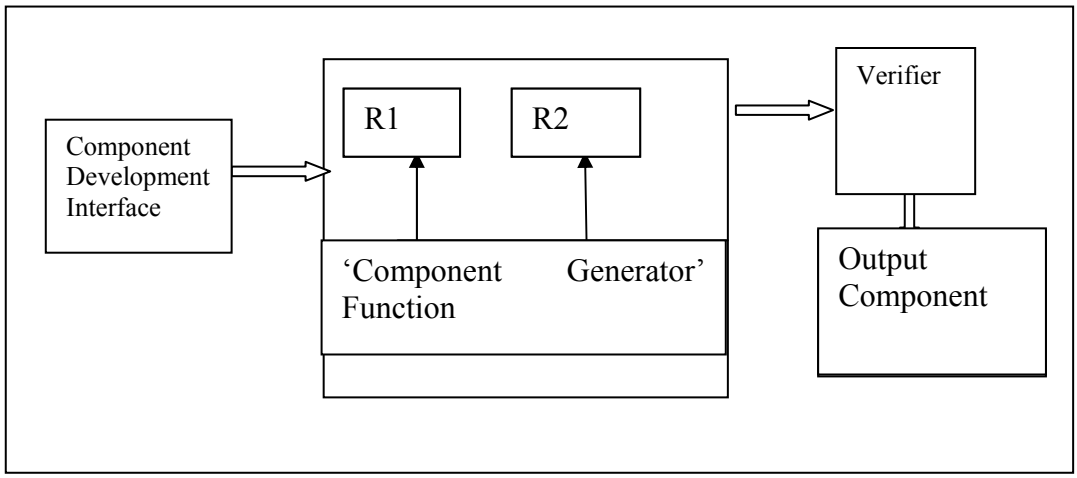

Figure 4: Proposed idea of 'Component Generation Modal'.

The idea of a "*Component Generation Model*" is being proposed as follows.

> In the first step, a "Generic component" would be worked out by identifying and implementing the most common attributes and functionalities required across, and common to all, the application types. There would be a repository, say R1, in which all the identified "generic components" will be stored.
>
> There would be another part of the repository, say R2, which would have the possible functional/non-functional properties, required as variations in the different application types (for different states).
>
> A component developer will specify the requirements, that s/he receives from a component user, to be submitted to the 'Component Development Interface' being proposed as part of this Component Generation framework.
>
> Now the proposed "*Component Generator*" function will choose the "generic component" (for example- Student mark-sheet) from repository R1 and the required properties (as desired by the component user) from the repository R2 (for example say Absolute grading mark etc) and then generate the component (by combining them both) according to the requirements of the specific application.
>
> There would be a 'Verifier' that will verify the non-functional aspect of the developed component.
>
> Finally the Component Generation Model may deliver the component.

The realization of this idea will assist a component development organization to develop components according to the requirements of a Component User. The question is whether the complete automation of the 'Component Generation Model' is possible or whether it is necessary for some software environment (to be handled by professionals with some human interaction) to exist for this purpose.

5.3. 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. Inthe 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 [Bennett, 1995]. 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 (creating documents etc. that may not have existed) and redesigning and scaling it up into a new S/W as per newer demands. In this case there is the opportunity tocreate 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 (e.g. programs, jobs, data files etc), 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.

6. 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 composable 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 (to be handled by professionals with some human interaction) would be necessary for this purpose.

The Component Development Life Cycle (identification, designing, coding and testing of a component) 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.

7. 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 a 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

Backus, M (2001), E-Governance and developing countries: Introduction and examples. Research Report, IICD Research Brief, No. 3, April 2001.

Bennett (1995), Legacy Systems: Copying with Success, IEEE Software, pp June 19-23.

Beer, D., Kunis, R., and Runger, G. (2006). 'A Component Based Architecture for E-Government Applications', In proceedings of the First international Conference on availability, Reliability and Security (April 20-22, 2006), Vienna.

Bhattacharya, J. (2002). Middleware and technology standards for E-Governance. India Research Lab, IBM, 2002.

Brown, A. W. and Wallnau, K.C. (1998). Current State of CBSE. [Electronic Version]. IEEE Software, Vol. 15, No. 5, pp.37-46.

Butter, J. C., Mago, D. R. and Weiler, J. (2003). 'Succeeding with CBA in E-Government', Conceptual Level White Paper, Developed for the Federal Enterprise Architecture Program Management Office (FEA-PMO), IAC, EA-SIG, March 2003.

Chandrashekhar, R. (2005), 'National e-Governance Action Plan (2003-07)', National e- Governance Plan: -Workshop with States and UTs, 11-12 March, New Delhi.

Crnkovic I. and Larsson M.(2002), Building Reliable Component Based Systems, ISBN 1- 58053327-2, Artech House Publishers.

Councill, W. T. and Heineman, G. T. (Eds.). (2001). Component Based Software Engineering: Putting the Pieces Togather, Reading, MA: Addison Wesley, 2001.

Curtin, G. G., Sommer, M. H., and Vis-Sommer, V. (Eds.) (2004). The World of E-Government. Haworth Press: New York.

Debnath, N., Felice, L., Montejano, G. and Riesco, D. (2008), 'A Future Model of E-Governance Systems Integrated with Formal Specifications', Fifth International Conference on Information technology: New Generations, Las vegas, Nevada, April 7-9, 2008.

Fontoura M., Braga C., Moura L. and Lucena C., (2000) Using Domain Specific Languages to Instantiate Object-Oriented Frameworks, IEE proc-Softw., Vol 147, No.4, August 2000, Page(s) 109-116.

Grönlund, A. and Horan, T. A. (2005), Introducing e-Gov: History, Definitions, and Issues, Communications of the Association for Information Systems, Vol.15, 2005.

Johnson R.E., Components, Frameworks, Patterns, Proceedings of 1997 Symposium on Software Reusability, 1997, pp 10-17.

Lambrinoudakis, C., Gritzalis, S., Dridi, F., and Pernul, G. (2003). "Security requirements for egovernment services: a methodological approach for developing a common PKI-based security policy", in Computer Communications, 26(16), pp. 1873-1883.

Lenk, K. and Traunmüller, R. (2000). "A Framework for Electronic Government", in Proceedings of DEXA 2000, London/Greenwich, UK, September 4-8, 2000, pp.340-345.

Mittal, P.A., Kumar, M., Mohania, M.K., Nair,M.,Batra, N.,Roy, P., et al, (2004), A Framework for E-governance solutions, IBM Journal of Research and Development, Volume 48, Issue 5/6 (September/November), 717-733.

NCS, (2005). Large Scale Deployment of E-Governance Services. NCS- A Member of SingTel group, NCS Hub, Singapore. URL: http://www.ncs.com.sg.

Palanisamy, R., (2004). Issues and challenges in e-governance planning Electronic Government, an International Journal, 2004 - Vol. 1, No.3, pp. 253 – 272.

Panfilis, S. D., and Berre, A.J.(2004), 'Open issues and concerns on Component Based Software Engineering', Published in proceedings of 'Ninth International Workshop On Component Oriented Programming (WCOP 2004)', 14-18 June, Oslo, Norway.

Pressman R. (2001) S., Software Engineering: A practitioner approach, Sixth Edition, ISBN 007-124083- 7, TMH, 2001, pages: 847-857.

Prasad, T.V., (2003), 'e-Governance and Standardization', published in proceedings of 'Conference on Convergent Technologies for Asia Pacific for Asia Pacific Region (TENCON 2003), 15-17 Oct, vol 1, 198- 202.

Payrelal, S., (2008), 'Standard and Interoperability: for E-Governance", Project Report, Management Development Program for NIC, national Informatics Center, New Delhi (India).

Payrelal, S., (2005). 'Technical Standards and E-Governance architecture', Approach paper, Government of India, National Informatics Center, New Delhi. URL: http://egovstandards.gov.in

Szyperski, C. (1999). Component Software- Beyond Object Oriented Programming, Reading, MA: Addition-Weslay.

Sunder, B. and Harsh, J. eThens- A Component Based Framework for E-Governance', BITS C461, Software Engineering project, BITS Pilani, India. URL: www.slideworld.org/viewslides.aspx/2129735