E-governance Challenges and Strategies for Better-managed Projects

Rajat K Baisya[1] and Siddhartha Paul Tiwari[1*]

ABSTRACT

A study on e-governance challenges and strategies for managing projects has been attempted. Software crisis even today is a widespread affliction. Manifested as over-spent budgets, missed deadlines, and unmet requirements, the software crisis costs is a major threat for the currently developing e-governance sphere. Mismanaging a software development project in case of e-governance can mean ineffectively delivering services to its citizens. Although throngs of texts, papers, and other narratives have been written during the past 40 years to address the subject (and challenges) of software construction and its related activities, there is still no comprehensive solution to the problem. This is because software engineering is a young, complex, and ever-changing discipline. For an individual IT organization with limited resources, solving the long-standing problem of software engineering is improbable. The authors have tried to present a number of strategies and best practices that will help the e-governance and IT industries look towards realistic goals and mitigate the risks inherent to software development projects.

***Keywords**:* E-Governance, software crisis, semantics.

1. Introduction

Governments now have taken up the challenge of creating electronic infrastructure for virtual interaction with citizens, businessmen and others for effective delivery of services with the belief that these would bring visible advantages, such as transparency, accountability, faster and economical delivery of services, convenience, user-friendly access, elimination or minimization of physical interface and a host of other advantages.

E-governance spend, hence, is increasing in every country multifold.

India is not behind in these initiatives. Government has declared important policy initiatives that included setting up of a separate Ministry of Information Technology in 1999, parliament approval of Information Technology Act 2000, allocation 2-3 percent of budget for IT in every government department and most importantly the announcement of National E-governance Plan (NEGP) in 2006. NEGP has stipulated originally with a budget 33000 crores of rupees (above 6 billion dollar) that contained 100000 Community Services Centers (CSC), covering one CSC over 6 villages, State Data Centers, State Wide Area Networks (SWAN) and several mission mode projects in the critical areas such as Police, Customs, Income Tax,

[1] Department of Management Studies, Indian Institute of Technology, New Delhi 110016, India
* *Corresponding Author* (E-mail:Siddhartha@google.com, Telephone: +91-9999258149)

Justice, Treasury etc.

Despite the above emphasis, there are more failure than success. For example, Standish Groups' survey showed that 52.7% of software projects miss their schedule and financial targets, 31.1% of all projects are canceled, and only 16.2% of the projects are completed on time and within the budget (Hayes, 1997). E-Governance projects are unique undertakings that involve degree of uncertainty, since these projects are complex and have a broad scope. Heeks (2002) has compiled results on failure of projects in e-governance. The following results on e-governance projects are an eye-opener. 15% are successes- Projects, which attained their major goals, 50% are partial successes/failures- Projects in which major goals were not attained or significant undesirable outcomes were noticed. 35% are total failures- Projects which were never implemented or implemented but immediately abandoned.

In India, there are several projects which are reported to struggling or branded as failed despite been created with much hype. 'Gyandoot' is an example which was started with fanfare in the central state of India 'Madhya Pradesh' and subsequently won International award (Stockholm Challenge). Also struggling to survive is the big project 'Community Information Center (CIC)' started in 2002 with a budget of 200 Crore and much fanfare by the Ministry of Information Technology in Assam and the North Eastern states of the country. There are 487 such centres in the North East and 219 in Assam. Problem faced by this project is enumerated by Nomita Das[2], an independent researcher, undertook a field study of these project in January 2007 and concluded 'There exists a sense of helplessness and isolation among advocates of the project in the face of the response or lack thereof from the state government. In the last five years since the inception of the project, the government had not shown an iota of interest in establishing a formal system of studying, monitoring and improving the project. There is lack of users' awareness, ease of interface, motivation of the CIC staff (who are low paid, adhoc people). Regularisation of CIC staff jobs is the foremost issue needing attention. CIC personnel made desperate attempts to be heard by writing to local MLAs, MPs, District Commissioners, all the way up to the President of India. They also posted their grievances on various online chat groups. At best they found a sympathetic ear, and a whole lot of opinions'.

Garcia, and Pardo (2005) have tried to build an understanding of the challenges of e-government initiatives and sought to derive key success strategy (KSS). Admitting that there is no single list of challenges to e-government initiatives, there are notable consistencies across the disciplines. These consistent challenges are proposed as primary challenges to e-government and grouped under five categories according to their core aspect: (i) Information and data, (ii) information technology, (iii) organizational and managerial, (iv) legal and regulatory, and (v) institutional and environmental. These challenges are important from the point of view of the factors of effectiveness. Some of these factors are resistance to change and managers' attitudes and behaviour. They have in fact have also suggested strategies to tackle each of these issues and those factors have been named as key success strategies, from which factors of effectiveness and change management can be identified easily. For example, 'ease of use' and 'usefulness' are factors which will boost the effectiveness of the e-governance project. Similarly, 'End user involvement' will reduce the resistance to change.

From software project management angle, some care would be help as enlisted by the researchers from this discipline. This paper outlines some of the important steps needed to ensure project related issues of handling e-government projects with the understanding that a project management approach would help

[2] http://i-quench.org/edevelopment/index.php/profiling/a-primer-on-cic-project-in-assam/ (ISBN: 355 , AUTHOR: Nomita Das, 2007, CATEGORY: General, TRACKBACK: Trackback
UTC: 2007-03-21T20:39:52-25200

handling all sorts of risks associated with IT project.

2. The Software Crisis – An Introduction and History

Less than three decades after the birth of programming; around the 1960's, the software development process reached a pathetic state. Excessive money was spent, discipline was lacking, needs were not met, projects were cancelled and documentation either didn't exist or was not properly managed.

The NATO Science Committee in 1968 assembled dozens of IT experts and leaders to address these issues, these issues were eventually termed as the 'software crisis'. They defined software engineering as the application of a systematic, well organized and quantifiable approach towards the development, operation, and maintenance of software, the committee was not able to decide on the ways how the so called crisis could be tackled. As of today, basic characteristics of software development might have changed but the software crisis still prevails and is costing the IT industry huge amount of resources and money. On an average a software development project misses its completion date by 50%, and 75% of the projects are either not used at all or do not perform as intended. Now, when we apply such a situation to an e-governance project managing pivotal information and performing crucial tasks, we can clearly see that the impact can be of a very high magnitude.

3. The Complex Nature of Software Engineering

Software engineering is a highly complex thing to begin with. Some of the key reasons that make software engineering a challenge will be:

Ambiguous Semantics: One of the most debated questions in the IT field is whether or not the term 'software engineering' justifies the usage of the word engineering in the first place. The National Society of Professional Engineers (NSPE), founded in 1934, is a group encouraging the education and safeguarding the rights of engineers throughout the United States. In the 1990s, the NSPE filed a lawsuit to prevent the usage of the term 'software engineer' as a job title; they eventually won the case a decade later. There argument was that software engineering as a discipline does not meet the minimum criteria that other, more common, engineering fields met.

Engineering can be defined as the application of science and its various manifestations to solve problems in an effective manner. However, some maintain that there is very little science involved in software engineering. They argue that software engineering only involves mathematical algorithms that operate software, something that will be rendered useless if there are no computers; hence there is no manifestation of the science found in nature. According to this definition, anything that is not realizable in a practical sense is not actually engineering.

Nascent Discipline: Software engineering is comparatively a new discipline as compared to the other more traditional branches of engineering. The process of software development comprises of numerous clearly understood moving parts. What is difficult is to assemble these parts into a single unified working part. This drawback of software development can prove to be a bottleneck when we talk of e-governance.

How to measure effectiveness: Imperceptible nature of software makes the process of evaluation of the success and failure of a software development project unclear. What is the baseline for "failed" projects: cost, time, customer satisfaction, throughput, or response time? The Six Sigma quality management technique addresses many of these issues, but in most cases this heavyweight approach is inappropriate for small- and medium-sized software development projects. However, in case of e-governance it is pivotal to measure the effectiveness of software.

Many determine quality solely on the number of defects identified, but there is more to the definition. Traditional engineering branches measure quality in standard ways, such as the temperature a piece of steel can withstand or the speed at which a vehicle can travel. Although some software characteristics can be measured in a comparatively straightforward fashion, other characteristics cannot. For example, how does one determine how modifiable a component really is? How does one define a 'small' error? Research comes to the conclusion that probability-based methods for measuring reliability are ineffective, but statistical methods tend to be more robust.

Uniqueness: The chances of two different software codes being exactly alike is very less, even if they are solving the same problem. In his book, *Facts and Fallacies of Software Engineering*, Robert Glass discusses how a diverse set of problems requires a diverse set of solutions. It is basically an exception to the rule to find a large component that is reusable across applications, let alone domain. Sure, it might be easy to build a reusable component that parses strings and share it across many applications. It is quite a different matter to build a generic component that computes local tax rates that can be reused across the country or the world.

Complexity: As the enormity of the problem increases, so does the complexity of the solution associated with it. This issue reverberates through design, construction, test, and maintenance activities, causing the curve to rise even faster.

Error Replication: In quite a few cases errors in software occur for a very specific case that might be difficult to replicate. What may be considered as a set of unrelated behaviors may eventually be coupled, making debugging almost impossible.

Best Practices for improving the software development process in the e-governance scenario:
There does not exist one solution to solving all the problems that have hindered software development for decades. According to Frederick Brooks, known for his landmark contributions to computer architecture, operating systems, and software engineering. Brooks asserts that past developments in software engineering address only the byproducts of the software engineering process, such as disparate technologies and programming environments or the slow speed inherent in certain design paradigms. What really needs to happen is for the software developers and managers to address the essence of software engineering. In other words, they need to acknowledge that developing software is complex and software is inherently intricate. Approaches that address the nature of software engineering include the following:

Utilize Incremental (or Iterative) Techniques: Some think of software as being evolved, rather than built. As the variables are very complex, it is rare that the requirements are understood well enough to build the system accurately in one attempt. Clear and stable requirements are an unattainable goal. Furthermore, the stakeholder may not even have control over his or her own environment. Therefore, review requirements early and often. Trying and integrating lessons learned from previous iterations into subsequent iterations is one of the best ways to learn something. There are a number of schools of thought on where to start early iterations — with the largest components, the riskiest components, or the least understood components. In any case, assume the first iteration will be a waste to be thrown out.

Procure Great Designers and Developers: Subscribe to the notion that developing software is all about people, process, and tools. While finding the right talent is only one third of this equation; it definitely is one of the most important components. The IT industry continues to look the other way and assume that people can be swapped in and out of a project without recourse. Instead of finding qualified personnel, the industry often falsely relies on tools and techniques to fill in the gaps made by unskilled developers, all in an effort to save time and money. The idea of giving special attention to software developers is not new, as computer scientist Raymond Rubey addressed in 1978, "When all is said and done, the ultimate factor in

software productivity is the capability of the individual software practitioner." In the context of more contemporary frameworks, author and agile development practitioner James Highsmith explains that when the façade of formal and rigid process is removed, the reason projects are successful is because of the people.

Reuse Components: The IT industry is trying to define creation of software as a design of algorithms and modules from the scratch. Similarly, many leaders and academicians believe that software should be assembled and not written. There are very few atomic pieces of functionality that have not already been authored, and developing a new system is just a matter of assembling components that have been produced by other vendors or that are products of previous implementations. While beyond the scope of this article, component-based development (CBD) and, more recently, a service oriented architecture attempt to address these issues. While these techniques are certainly a right step in the right direction to achieve these goals, component reuse has its own set of risks that need to be understood and managed. Few challenges related with component based development include:

- Who will own the component?
- Who will maintain the component?
- Who will guarantee the component's reliability and security?
- What happens if the component becomes obsolete?
- How do you design a component without knowing how it will be used ahead of time?
- What happens if not all components are available at the time of assembly?
- What happens if the component does not exactly meet user requirements?
- Who incurs the cost of developing the component?

Additionally, in a dynamic E-Governance environment, it is important that stakeholders realize the importance of moving away from the narrow mentality and adopt architectures that are for the overall betterment of the industry. Individual business units may have a myopic view of the organization and can tend to consider only their own needs. In scenarios where the project sponsors include the IT organization, addressing this issue at an organizational level may not be as much of a problem. It is not until these governance questions are answered, that CBD will provide a viable "cookbook" method for constructing software to the extent that many hope it will be.

Talking more practically; developing components for reuse is harder than developing single-use components. Reusable components increase in complexity to accommodate their generic nature. According to a few estimates, reusable components should be tested in a minimum of three environments before they are considered viable.

Use Tools Effectively: Tools are the enablers of reuse and reengineering, but remember this one caveat: a poorly understood, improperly implemented, or forced usage of a tool can sometimes cause as much, if not more damage than not using tools at all. Limitations and dangers of tools can include stuff like cost of acquiring & applying the tool, deficiencies of the tool, false performance expectations, and inappropriate usage plus it leads to over dependency on the tool.

Bringing such tools into the enterprise requires a culture change and buy-in from all effected parts of the organization. Without the proper communication, selection processes, and training, tool implementations will result in failure. When these concepts are understood, effective use of tools can have a significant impact on the success of the e-governance project. To use tools effectively, be sure to:

- Identify tool dependencies
- Use simple tools and simple configurations of complex tools—don't over-engineer use of the tool
- Avoid inadequate tools — don't use a tool for the sake of using a tool

- Develop and share tool-specific knowledge
- Communicate with management about the requirements for tools
- Keep tools upgraded
- Avoid the tool pit — Use combinations of tools as necessary, instead of looking for the "silverbullet" tool. Make sure the tools are there only to support the people and processes, and use only the features that improve efficiency.
- Remember that paper and pencil might be the best tools after all.

Brooks' Law: As per the Brooks' Law, while programming work performed increases linearly with the number of programmers, the complexity of a project increases exponentially with the number of programmers. Therefore, numerous numbers of programmers working on a project will become entangled in a web of confusion. Brooks himself states: "Adding manpower to a late software project makes it later." Hence, careful planning and consideration should be made to decrease the severity of a late project.

The Only Thing Not Susceptible to Change Is the Fact That Things Change: Even for the most basic of implementations, gone are the days when we can assume that only when requirements gathering activities are complete, accurate, and immutable can we start design activities. Changes made late in the schedule are considerably more expensive to address than those uncovered earlier. It is probably a safer bet to choose a more flexible process that accommodates for a changing environment and presents stakeholders with artifacts and work products for review earlier, rather than later. Conversely, do not assume that if you expect requirements to change to a minimal or moderate degree, that you will necessarily need to use an agile development framework. While understanding the needs we might change the design such that it is able to accommodate making any future changes an easy to implement task.

4. Concluding Remarks

With reference to E-governance never underestimate the organizational impact and business changes that implementing a piece of software can have. Let's not forget fundamental project management tenets when it comes to E-Governance projects. It is critical that communication takes place in the organization before, during, and after the project is implemented. All too often it is the case where users find out only during an implementation that changes are on the horizon. In many cases this results in reluctant and resentful users. Certainly problems will occur and eliminating them in their entirety is unlikely to happen anytime soon. Instead, set realistic goals. Take the appropriate steps to mitigate risks by not overdoing the process, being flexible.

References

1. Gibbs, W. Wayt, "Software's Chronic Crisis," *Scientific American,* September 1994.
2. Liu, Lay, "Is Software Engineering Actually Engineering?" *The Iron Warrior,* October 2003.
3. Cox, Brad J. (1990), "Planning the Software Industrial Revolution," *IEEE Software Magazine* — Software Technologies of the 1990s, The Institute of Electrical and Electronics Engineers.
4. Glass, Robert (2003), *Facts and Fallacies of Software Engineering,* Addison Wesley
5. Woodfield, Scott N., "An Experiment on Unit Increase in Problem Complexity," *IEEE Transactions on Software Engineering,* March 1979.
6. Brooks Jr., Frederick P., "No Silver Bullet — Essence and Accidents of Software Engineering," Computer, 20, no. 4, The Institute of Electrical and Electronics Engineers, Inc., April 1987, 10–19.
7. Mills, H.D. (1971), *"Top-Down Programming in Large Systems," Debugging Techniques in Large Systems,* ed. R. Ruskin, Prentice-Hall
8. Hohmann, Luke (1997), Journey of the Software Professional — A Sociology of Software Development, Prentice-Hall.
9. "Brooks'Law," http://en.wikipedia.org/wiki/Brooks%27_law
10. Gil-Garcia, J. R. and Pardo, Theresa A. (2005) "E-government success factors: Mapping practical tools to theoretical foundations", *Government Information Quarterly* Vol 22 pp- 187–216

11. Heeks, R. (2003), "Most eGovernment-for-Development Projects Fail: How Can Risks be Reduced?" *Institute for Development Policy and Management*, University of Manchester, UK.

About the authors

R. K. Baisya is serving as Professor of Marketing & Strategic Management and also served as Head of Department of Management Studies at IIT- Delhi for three years till September 2004) and engaged in teaching , research , training and consultancy activities. Working as consultants to many private and public sector undertakings. Honorary Visiting Professor of Strategic Management of a leading European Business School (ENPC International Business School , Paris). Prof Baisya has served for over twenty eight years in industry in very senior capacity with large Indian and multinational corporations. He has long association with large Indian Business houses having served as President & CEO of Emami Group of Companies , Senior Vice President (Business Development) of Reckitt & Colman of India Ltd (now known as Reckitt Benckiser India Ltd.) , General Manager- Projects and CEO of International Business of Escorts Ltd , Controller (Corporate Planning & Coordination) and also as Technical Controller of United Breweries Group , Project Engineering Manager of Best Foods International (formerly known as Corn Products India Ltd) , a Unilever company , Standards & Quality Control Manager of Herbertsons Ltd and as Development Engineer and also as Project Manager of Parle Exports Pvt. Ltd. He made significant contribution in all these industries during his tenure there having responsible also for launch and development of many successful new products.

Siddhartha Paul Tiwari is currently Product Specialist at Google India, Inc, based out of the Gurgaon office. He is also a doctoral student at IIT, Delhi. Prior to joining Google, Siddhartha has had wide experience in information systems. He has also served as a visiting faculty to Indore University. Siddhartha has served as a consultant to Impetus Technologies. He has been actively involved with Government in various high level committees. In year 2005 Siddhartha was awarded the IT Guru Award for his efforts in the field of IT education. His current interests lie in the e-business industry and IT strategy.