Diminish the Extent of e-Government project collapse with fault tolerance

Jinga B. Prajapati, Nilesh K. Modi

[1] **JJTU Research Scholar (**Acharya Motibhai Patel Institute of Computer Studies, Ganpat Vidyanagar, Kherva, India)
[2] S.V.Institute of Computer Studies Kadi, Gujarat, India.

Corresponding author's information:
Name: Jigna B. Prajapati
Designation: Assistant Professor
Acharya Motibhai Patel Institute of Computer Studies,
Ganpat Vidyanagar, Kherva, India
E-mail: jigna.prajajapati@ganpatuniversity.ac.in
Phone: (M) 91-9925065025

Abstract

Information and communication technologies have a significant budding to help to meet good governance goals in especially in developing countries. eGovernment is a global project of technology transfer, taking designs from one context into a different context and resulting in empowerment of the citizens and increased transparency in public dealings. eGovernment can make a valuable contribution to development, yet it could not possible up to expectation. The study focuses many aspects of eGovernment Project Failure and its issues. The gaps between various entities involved in eGovernment Project have been analyzed. The main root of any project failure is depends upon the faults resided at the system. Here software faults tolerance techniques applied to achieve high reliability and availability to reduce the eGovernment Project Failure.

Introduction

"e-government" or electronic government refers to manage the department with Information and Communication Technologies by government agencies for exchanging information with citizens, businesses or other departments.[1] IT is a digital interaction between a government and citizens (G2C), government and businesses/Commerce (G2B), government and employees, and also between government and governments/agencies (G2G)[2]. The main features of E-Government to increase Speed and more efficient delivery of public services, Improve internal efficiency, reduce costs, increasing revenue, and re-strut of administrative processes. At present, information and communication technology particularly internet is used as a main tool in all areas of government organization for better control [3]. This work shows the concept of transforming internal and external relationships through technology for continuous optimization of service, constituency participation and governance [4, 5].

Different e-Government projects

- Tax control: Notification and calculating
- Social security

- Identify information
- Public libraries
- Inquiries
- Pubic heath issues
- Legislating information
- Official news paper
- Custom offices
- Banks
- Finance

eGovernment Project Failure and it's issues

eGovernment project failure can be classified as total failure, partial failure or succeeded. Total failure means the initiative was never implemented or was implemented but immediately discarded. Partial failure means major goals for the initiative were not attained and/or there were significant undesirable outcomes. And Succeed projects means most of the tasks covered in that projects are achieving the desired output [6]. On the basis of figures provided in surveys, one-fifth to one-quarter of industrialized country government ICT projects fall into the total failure category; something like one-third to three-fifths fall into the partial failure category; and that only a minority fall into the success category[7,8,9]. We may conclude on basis of [7], that there are no of projects is successful but more frequent to fail to fulfill their initial promise. Health information systems in South Africa's public sector: widespread partial failure of high cost systems with little use of data [8]. Failure cases seem to be the norm in Thailand at all governmental levels [9]. Donor-funded public sector ICT projects in China: all were found to be partial failures [10].

One can major the success of e-government projects by the tasks performed in the projects and how these tasks will make our projects satisfied [11]. The gap between the initiated projects and finished projects must cover the maximum output. eGovernment success and failure therefore depends on the size of gap that exists between 'current realities' and 'design of the e-government
project'. How lager the design gap in projects, greater the peril of e-government failure. In reverse How smaller the design gap, greater the chance of success [12]. Analysis of failure of e-government projects indicates has flowing Issues

- ➢ *Information:* lacking of information gathering, dealing, data preparing,
- ➢ *Technology:* frequently change technologies
- ➢ *Processes:* time delay process between Government departments
- ➢ *Staffing and skills:* rotating of staff and skilled persons are not positioned at proper placed,
- ➢ *Management systems and structures:* application flow procedure form higher to bottom approach
- ➢ *Other resources:* time, money, politics, social affairs, external bodies and many more.

The gaps in various bodied may increase the failure of e-government projects. Such gaps can be Hard-soft gaps, Private- public gaps, Country context gaps [13]. Most government organizations are affected by 'soft' factors as people, politics, emotions and culture and, rationality and objectivity which treated as Hard-soft gaps. Many IT systems have been designed in the private sector and used into a public sector reality which operates very differently is Private- public gaps. When we use off the shelf approach which is commonly adopted is set to fail. Infrastructure and mindsets are very different across the world. It is likely that a system designed for New Delhi will not suit Johannesburg is Country context gaps [13].

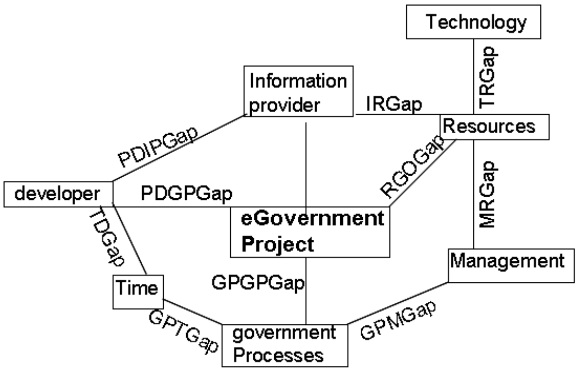

Fig: 1 ***Gap between various entities***

The reason of failure of government projects are many more, but the "GAP" between various object associated in projects is crucial. The "GAP" can between Projects developer and Information provider (PDIPGap), The "GAP" can between Projects developer and government projects Processes (PDGPGap), The "GAP" can between government projects Processes and government organization (GPGOGap), The "GAP" can between government Processes and Management (GPMGap), The "GAP" can between Resources and government organization (RGOGap), The "GAP" can between Technology and Resources (TRGap),The "GAP" can between Information and Resources (IRGap), The "GAP" can between Staff and Information (SIGap),The "GAP" can between Management systems and Information (MIGap), The "GAP" can between Time and Developing phase (TDGap),The "GAP" can between Management systems and Resources (MRGap) and many more.

Proposed ways to deal with Issues

To reduce the gaps(PDIPGap, PDGPGap, IRGap, TRGap, SIGap) among various entities we have to keep in mind about the changes between the design proposal and current reality, degree of change between the design proposal and current reality, complete and radical change between the design proposal and current reality(PDIPGap, PDGPGap) [14]. One must take care information used in the e-government application was somewhat different from the information currently really being used (SIGap), the technology used in the agency comparing the requirements contained within the design of the e-government projects (TRGap), the work processes undertaken in the agency comparing the processes needed for successful implementation of the e-government projects (PDGPGap). It was found that, more time and money required for successfully implement and operate the new projects compared with the time and money really available at specified time (TDGap). These all about the failure of e-government projects, as we discussed it's probable reasons to deal with such failure. It is not only the e-government projects failure but the root is software failure. If we deal properly with software failure and apply same things to e-government projects, then definitely we can decrease the of e-government projects failure.

What is the main cause of software failure? It is the faults reside at any working and performing application. We must enable our projects to work continue even under the faulty environments. Not only working properly but we have to keep in mind about the results producing from that application. The result of the project must satisfy the user and we have to deal the fault reside at the project in such satisfactory way. To do so, here we wanted t use the software fault tolerance techniques.

Software fault tolerance

In order to ensure that the systems operate as indicated, even in extreme conditions, it is important to have fault-tolerant computer systems, hardware and software [15]. Faults may be classified based on locality, cause, duration, phase, system state etc. Locality wise faults as atomic component, composite component, system, operator, and environment where faults reside in some specific location, the combination of more than one component, faults arise from any environmental causes or any user-operators [16]

Techniques which are supported to software fault tolerance may detect the faults, prevent the faults, or remove the faults. Fault prevention: preventing the occurrence or introduction of faults for quality assurance and design methodologies. Fault removal: remove faults after the development stage is completed. This is done by exhaustive and rigorous testing of the final product [17]. Fault avoidance/prevention includes design methodologies which avoid the faults which may not have fault solution [18]. Software fault tolerance techniques are divided into two groups as Single version and multi-version software techniques [19]. Single version techniques focus on improving the fault tolerance of a single piece of software by adding mechanisms into the design targeting the detection, containment, and handling of errors caused by the activation of design faults. Single version techniques are Error detection, Exception handling, Data diversity, Process pair, etc. Multi-version fault tolerance techniques use multiple versions (or variants) of a piece of software in a structured way to ensure that design faults in one version do not cause system failures[19,20,21].

We include Design diversy and Environment diversy. Design diversy is identical service through individual design and implementations. As the exact copy of the software component redundancy cannot improve the reliability in terms of software Design malfunction, we must ensure it [19, 22, 23]. Environment diversity is the new approach to fault tolerance in software. Although this technique has been used for a long time in an ad-hoc manner, only recently has gained recognition and importance [22]. To do so, we are approaching the design diversy with hardware and software fault tolerance.

DESIGN DIVERSITY

As the exact copy of the software component redundancy cannot get better reliability in terms of software malfunction, we must ensure diversity in the development and implementation of software. Here is the goal to provide diverse through variants design to minimize the indistinguishable causes of errors.

Recovery Blocks (RcB)

The basic RcB scheme is one of the two original diverse software fault tolerance techniques. The RcB is categorized as a dynamic technique [24]. It chooses one alternate accepted execution path. If selected execution path and selected execution test passed then it would end execution, otherwise it will divert to another alternative. Such process will repeated up to the no. of alternative to passed successful. A less effective option (s) is consistently in the main block and is called (secondary) alternate or alternative units [25].

N-Version Programming:

NVP was suggested by Elmendorf in 1972 and developed by Avizienis and Chen in 1977–1978. NVP gives us the different version of the same system [26]. Compared with RCB, NVP is static method. It use different version of the same piece of code. Hardware fault tolerance architectures, associated with NVP is N-modular [26]. The Processes can run simultaneously on different computers or sequentially on one computer [25].

Implementation

No basis on the observation that the failure of most eGovernment project transient in nature, the environment and the diversity of approaches require re-implementation or re-executing of programs in a different environment. It offers diversity of the environment effectively with Heisenberg by exploiting its definition and nature [19]. Adams has proposed restarting the system as the best approach to masking software faults [22]. Environment diversity is a generally give a way of restarting. This was proposed as cheap but effective techniques for fault tolerance in software. There are three components that determine the behavior of the process or the execution of.

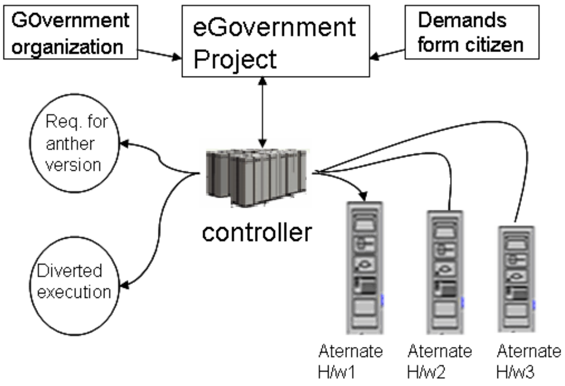

Fig: 2 ***eGovernment with H/W and Design diversy***

Transient deficiencies occurred at eGovernment project because of design faults in software which result in unacceptable and erroneous states in the OS environment. When software fails, it will be restarted in another, unmistakably OS environment that is achieved by some cleaning operations. Examples of methods of protection of environment diversity include the operation again, restart the system and rebooting the host, and though if the problem was not handling it will use a different h / w for specific hardware failures [15].

If any outside treatment is absent and does not work, we can use alternative equipment. As we have discussed an approach to handling software errors, mistakes and problems, or diversion of flow execution. In Robotic surgery the working of system is very most important in any situation l when it faces S / W difficult to answer in any way, distract performance S/W will not cause a system failure as we already discussed approach for handling software errors, faults and problems by either diverting the flow of execution. In eGovernment project When such part facing s/w problem to answer, the divert execution of the s/w will not cause system fail. We can place some piece of information which provide the same type of dealing with inputs and assure with required output. The RcB uses different acceptance test. When execution started first it check the primary execution , if it is according to the requirements or meet the satisfactory level then ok otherwise go to another alternative. The RcB gives us many alternatives until few extend. eGovernment project is used continues for crucial data so it need Sound availability and reliability in order to deal with faults by providing many alternatives of the any part of the system. Using different versions in NVP, the execution began with first version of the system, if any fault occurred it will use another version of the same piece of problem. It intimates about completing executing by successful handling the occurred faults.

CONCLUSION

eGovernment project is an important element for the growing country. eGovernment projects implemented with fault-tolerant system, it can extensively improve the reliability of governments organization working. The important is evaluation and deployment planning properly. We discussed eGovernment project failure and issue. Then, we addressed software and hardware fault tolerance technique which either improved the traditional techniques or took a new approach to solve the issues of eGovernment project and make it more reliable and available. In conclusion, the techniques and approach used in software fault tolerance will make any system more adaptable. These are the most adaptable solutions for dealing with different types of faults. Thus applying software fault tolerance techniques to eGovernment project will decrease the failure Project. The RcB and N-version work much effectively at any consign eGovernment Project approached by mentioned techniques will increasing sound reliability and availability.

REFERENCES

[1] http://www.nisg.org.

[2] Jeong Chun Hai, Ibrahim. Fundamental of Development Administration. Selangor: Scholar Press, 2007.

[3] OECD. The e-government imperative: main findings, Policy Brief, Public Affairs Division, Public Affairs and Communications Directorate, OECD, 2003

[4] Koh, C.E., Prybutok, V.R. The three-ring model and development of an instrument for measuring dimensions of e-government functions, Journal of Computer Information Systems, Vol. 33 No.3, 2003, 34-39.

[5] Gartner Group. Key Issues in E-Government Strategy and Management, Research Notes, Key Issues, 23 May 2000.

[6] Korac-Boisvert N. and Kouzmin A. Transcending soft-core IT disasters in public sector organizations. Information Infrastructure and Policy 4(2): 1995, 131-161.

[7] Avgerou, C., and Walsham G. IT in developing countries. In Information Technology in Context , eds. C. Avgerou and G. Walsham, Aldershot, UK: Ashgate, 1-8.

[8] Braa J. and C. Hedberg. Developing district based health care information systems. In Information Flows, Local Improvisations and Work Practices, Proceedings of the IFIP WG9.4 Conference 2000, Cape Town: IFIP.

[9] Kitiyadisai K., 2000. The implementation of IT in reengineering the Thai Revenue Department. In Information Flows, Local Improvisations and Work Practices , Proceedings of the IFIP WG9.4 Conference 2000. Cape Town: IFIP.

[10] Baark, E. and R. Heeks, 1999. Donor-funded information technology transfer projects. Information Technology for Development 8(4):185-197.

[11] Karl Weigers, 21 project management success tips, project smart 2000-2010, pp 5-6. (www.projectsmart.co.uk/21-project-management-success-tips.htm.)

[12] Causes of E-Government Success and Failure: Design-Reality Gap Model, R Heeks , 2003.

[13] CRITICAL ISSUES IN e-GOVERNANCE Summary of Discussion using Issue Process Methodology Panel discussionIn 5th International Conference on e-Governance (ICEG 2007), Dec 2007, 28-30.

[14] Most e-Government-for-Development Projects Fail How Can Risks be Reduced? By: RICHARD HEEKSIDPM, 2003.3-6.

[15] LAPRIE, J. -C., et al., "Hardware and Software Fault Tolerance: Definition and Analysis of Architectural Solutions," Proceedings of FTCS-17, Pittsburgh, PA, 1987, 116–112

[16] Randell B. "System Structure for Software Fault Tolerance" IEEE Transactions on Software Engineering 1975 SE-1(2) 220–232.

[17] Jean-Claude Laprie. Dependability- its attributes, impairments and means. InB. Randell, J.-C. Laprie, H. Kopetz, and B. Littlewood, editors, Predictably Dependable Computing Systems, ESPRIT Basic Research Series, pages 3–18. Springer Verlag, Berlin, 1995.

[18] CSE 598D: Software Fault Tolerance Instructor: Mahmut Kandemir, 23-24.

[19] P. A. Lee and T. Anderson. Fault Tolerance: Principles and Practice Second Edition, Springer-Verlag. 1990.

[20] Software Fault Tolerance: An Evaluation by Anderson, T., Barrett, P.A., Halliwell, D.N. Moulding, M.R. In Software Engineering, IEEE Transactions on,2006,1502 – 1510.

[21] A SURVEY OF SOFTWARE FAULT TOLERANCE TECHNIQUES by Zaipeng Xie, Hongyu Sun and Kewal at citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.1320.

[22] Gray and D.P. Siewiorek, "High-availability computer systems," IEEE Computer, vol. 24, no. 9, 1991, 39-48.

[23] Avizienis A., Kelly JP. Fault tolerance by design diversity - Concepts and experiments. Computer. Aug. 1984, Vol. 17, 67-80.

[24] Zaipeng Xie, Hongyu Sun and Kewal. A Survey Of Software Fault Tolerance Techniques. (citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.1320)

[25] Avizienis A. On the Implementation of NVersion. Programming for Software Fault- Tolerance during Execution. COMPSAC '77, Chicago, IL, 1977, 149–155.

[26] Chen L., Avizienis A. N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation. Proceedings of the 8th International Symposium on Fault Tolerant Computing System (FTCS-8), Toulouse, France, 1978, 3–9.