SUCCEEDING IN SOFTWARE DEVELOPMENT PROJECTS

Dan Benţa [1]
Ştefan Ioan Niţchi [2]

Abstract

In the past few decades, a large number of different approaches to software development have been introduced in order to adapt and satisfy customer needs and requirements that change during the development project. A wide variety of software development frameworks have evolved, each with its own strengths and weaknesses. A certain approach is not necessarily suitable for use by all software development projects. Choosing the best methodology for software development depends on several factors as project size and complexity, level of innovation, internal and external factors, available technology, organizational culture or project and team considerations. Agile software development methods are accepted worldwide as cost-effective and time-sensitive methodologies for software development in organizations in order to face the new challenges in this competitive environment. The aim of this paper is to identify, analyze and characterize traditional and agile software development approaches with practice use cases from our software development project and lifecycle.

Keywords: software development life cycle, traditional methods, agile software development, Scrum, project and risk management

Introduction

Software development is a complex process to create software starting from a specific set of requirements and involves teamwork and a lot of communication with a major positive effect in terms of innovation. Recent studies for software development focus on methodologies and analyze developers and users satisfaction in a Software Development Project [13], studies that have important implications for managers trying to balance the challenges of managing new software development and making enhancements to existing software, while engaging the user actively in the software development processes.

In our research, we were involved in a company internal project to develop a software solution for risk management department. In a software development project we need to clearly identify and define company preferences [5] in order to make the best choice and to propose the best solution. Any proposed solution, whether is hardware or software, need to integrate into company structure and to respect its policies. We also consider that the human factor is essential for a project to be successfully completed.

---

[1] Dan Benţa is PhD Candidate at Babeş-Bolyai University of Cluj-Napoca, Faculty of Economics and Business Administration, Theodor Mihaly Str., 58-60, 400591, Cluj-Napoca, Romania. E-mail: dan.benta@econ.ubbcluj.ro
[2] Ştefan Ioan Niţchi, PhD, is Professor at Babeş-Bolyai University of Cluj-Napoca, Faculty of Economics and Business Administration, Theodor Mihaly Str., 58-60, 400591, Cluj-Napoca, Romania. E-mail: stefan.nitchi@econ.ubbcluj.ro

This paper describes the software development life cycle and identifies traditional and agile development methodologies in this field. All these methodologies follow the development life cycle and propose a set of methods and steps with different frameworks to develop a valid software product to meet customer requirements. In this paper are listed traditional software development methods and their evolution to agile development methods. From our perspective, best approach for software development is an agile method as uncertainty is inevitable in any project.

Software development life cycle

The software development life cycle (SDLC) process is defined as an organized way involving multiple stages (Figure 1) that starts with determining customer needs and user requirements and ends with maintenance, documentation and personnel training if necessary, and including a way to use feedback for continuous improvement of the processes.

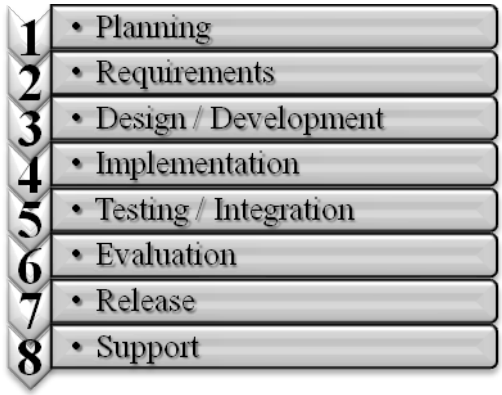


Figure 1 - Software Development Life Cycle Phases

Adapted after [18], we find the following steps:
1. Planning phase - In this first step, an initial planning action is necessary to establish a first view of the intended project; this phase is useful to determine and to identify the customer, user, software goals. This is the phase where the project manager decides to develop an initial Project Management Plan and to define some basic steps according to clients' deadline requirements. Team building, meetings and preliminary discussions are mandatory for a successful project, too. The key output of this phase is to know exactly the goal or goals of the project before funding and resources allocation, Project Management Plan with milestone dates and quality gates definition, and an authorization form for project approve based on the project definition.

Case Study (1): Our software development project was a great opportunity to work in real project environment with direct impact in economical field. The aim of the project was to develop a software application for Reference Class Forecasting (RCF), to analyze and forecast risks in future or existing projects in terms of delays and/or non-conformance costs. Each of the project members had a distinct specialization, as Graphical User Interface (GUI) Design, project management, XML, XQuery, JavaScript, Java, server configuration, Matlab etc. Project manager allowed a couple of weeks for

terms definition, company policy, social activities designed for improving team performance. This step was expressly designed for positive communication and to develop abilities to work closely together as a team in order to complete the project and to solve all the problems during its implementation.

2. Requirements phase – This step is to extract requirements and to perform requirements analysis. Project goals refinement is recommended in order to define functions and operation of the intended application. According to requirements, we suggest to create a detailed use case diagram and to display the proposed architecture for the intended software on a blackboard so that all team members to see them. The key output for this phase is to clearly state and to summarize in a requirements document what the intended software should do.

Case Study (2): A major requirement in our Software Development Project was to develop a modular application defined in several steps. Our proposed architecture was based on MVC (Model-View-Controller) model. The essential purpose of MVC model is to bridge the gap between the users' mental model (human model) and the digital model (computer model). Our proposal was for a modular application in order to ensure a high level of customization and extendibility. We consider this approach best for cases where, in future, a design team will operate changes for a software application designed by a previous design team. It is a proper approach for application modularity, making components as much as possible independent one from another. In this case a future team will change only a specific module to fit new requirements is not necessary to change the whole application.

3. Design and Development phase – This step describes desired characteristics in detail, including use cases, color and style layout, step by step GUI views, process flow diagrams, software inputs and outputs, pseudo code and other necessary documentation. The key output for this phase is to define a step by step GUI and to clearly define what to do or what software to buy in order to meet requirements, according to proposed architecture.

Case Study (3): For our project, a major challenge was to design a GUI with company *look and feel* specific. In order to successful complete this step in development process, we set weekly meetings with project team and department personnel to discuss and fix a step by step GUI. Meetings and discussions on this subject proved to be very beneficial and we managed to fit our application design in the company pattern.

4. *Implementation phase* – This is actually the step where the code for the software is written. Refinements or adjustments based on user feedback are allowed and prior to a larger scale implementation. The key output for this phase is a validated model that binds all components. Also, additional and specific documents should be created.

Case Study (4): In our software development project for risk management, project manager made sure that the project management plan was followed and each team member respected the deadlines. Each team member had to state what did since last project meeting, what will do until next project meeting and to define if there are any

obstacles or new risks in their activities. Accomplished activities were weekly checked in project management plan and the team ensures that we are ready for our next quality gate meeting and we can advance in our project. In some cases, for complex activities, members with earlier ended activities were assigned to help. In our meetings, we had to bind server-side and client-side components in order to have a working application for the next department presentation. Documentation phase was also performed and we wrote *Software Technical Specifications, Software Requirements Documentation, Software Graphical User Interface, Software Algorithms and Software Help* documents and files. The documentation phase can occur throughout the software development cycle.

5. Testing and Integration phase – In this step all the pieces are collected and connected for testing in a special environment to verify the functionality and to check for errors and bugs ensuring that defects are recognized as early as possible. The key output for this phase is the designed software that respects all requirements for development and fit company policy.

Case Study (5): Our application was designed to analyze and interpret data in a given field. In future, this application should be able to adapt and to analyze data from a different field, for a different client of the department. For this reason a modular application is the best choice. We used platform and device independent languages, for data manipulation we used XML standards, we used XForms to define form data, to store and transport data in XML documents and for data storage and management we used eXist-db. We tested on several datasets with or without merged headers, with or without blank spaces, with or without table format etc.

6. *Evaluation phase* – This step involves client side evaluation for the designed software to ensure that all requirements were met and the software satisfy the client needs. The key output for this phase is gaining the acceptance from the client and the software is put into the real business environment.

Case Study (6): For our software, this step was the final meeting with the head (the big boss) of our department and the presentation for our client with real analyze on real data and with results and solutions. In the final presentations we had to prove that our analysis is correct and to clearly define features (from hundred of features and thousand combinations) that provide delays in their projects. In this case, the value for a specific project to forecast was between 250mil and 800mil Euros and one day delay was a penalty of several hundred thousand of Euros.

7. *Release phase* – This is the project signing off point and final reports writing as lessons learned reports or others.

Case Study (7): This step was performed after the final presentation for department. We completed the final reports and lessons learned report where we made a brief description of the project, we listed good and bad experiences, what must be improved and we presented a short resume of the project.

8. *Support phase* – In this step a set of future requirements can be proposed; includes the maintenance phase that goes on seemingly forever.

Case Study (8): Finally, in our software development project for risk management, we made a list and a presentation where we clearly defined what we planned, what we achieved and we proposed future improvements.

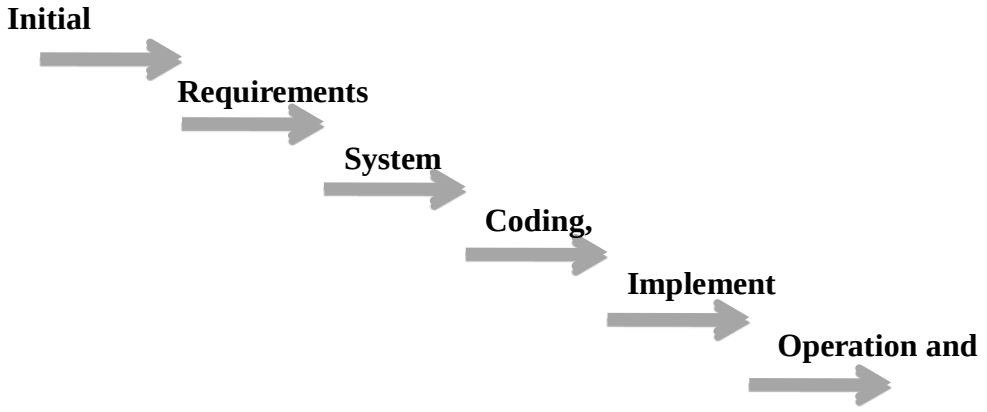
Traditional software development methods

A software development method refers to the used framework for planning and controlling the process of developing a software application. In traditional software development methods are included: Waterfall Approach [15], Prototyping Approach [12], Spiral Approach [4], Incremental Approach [6], and Rapid Application Development (RAD) Approach [8]. Choosing the best approach for a Software Development Project depends on the project type and context, internal and external environment, technical specifications, organizational features and policies. Basic frameworks and basic principles for Waterfall, Prototyping and Spiral approaches are presented in Figure 2 and *Table 1*.
Incremental and RAD methodology are derived or mix and match Waterfall, Spiral or Prototyping methods. *Incremental* methodology was developed as a response to the weaknesses of the waterfall model. The primary objective of this method is to reduce inherent project risk by breaking a project into smaller parts. The initial software concept, requirements analysis, and design of architecture and system core are defined using the Waterfall approach, followed by iterative Prototyping, which culminates in installation of the final prototype [17]. *RAD* methodology involves iterative development and the construction of prototypes for fast development and delivery of a high quality product at a relatively low investment cost. This compromise can lead to lower overall system quality [17]. This approach was a response to customer needs, Waterfall approach wasn't enough to meet client needs because applications took so long to build that requirements had changed before the system was complete.

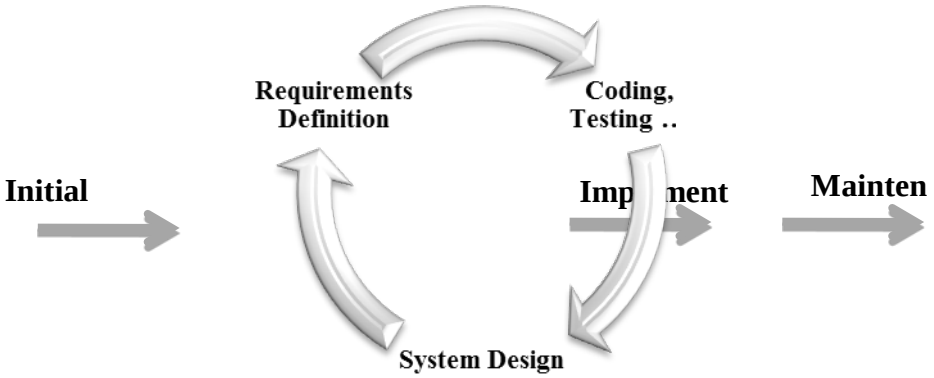
Agile software development methods

Changes in business and technology environments force software developers to adapt and find new solutions for successful projects and software development performance. Recent literature presents studies that analyze agile development approach and combine qualitative and quantitative data analyses for process performance in terms of on-time completion, on-budget completion, and software functionality [7].

WATERFALL



PROTOTYPING



SPIRAL

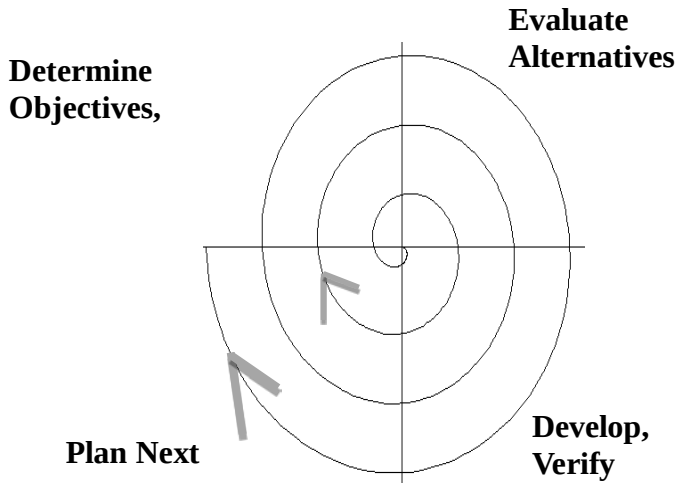


Figure 2 - Waterfall, Prototyping and Spiral Approaches: Frameworks

Table 1 - Waterfall, Prototyping and Spiral: Principles[1]

Waterfall	Prototyping	Spiral
• Project is divided into sequential phases, with some overlap and splash back acceptable between phases; • Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time; • Tight control is maintained over the life of the project through the use of extensive written documentation, as well as through formal reviews and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.	• Not a standalone, complete development methodology, but rather an approach to handling selected portions of a larger, more traditional development methodology (i.e., Incremental, Spiral, or RAD); • Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process; • User is involved throughout the process, which increases the likelihood of user acceptance of the final implementation; • Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements; • While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system; • A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.	• Focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle; • "Each cycle involves a progression through the same sequence of steps, for each portion of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program." [4]; • Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration. [4]; • Begin each cycle with an identification of stakeholders and their win conditions, and end each cycle with review and commitment.

[1] Adapted after [17]

The needs of the customers in terms of functions and quality constantly evolve leading to high requirements volatility which requires the software companies to be highly flexible [10]. Therefore, more and more software companies started to adopt incremental and agile methods.

The term of agile software development (ASD) was introduced in 2001 by Agile Manifesto [16] and describes a methodology where requirements and solutions evolve through collaboration between cross-functional teams. During the Software Development Project, user requirements can change dramatically and require software development in an agile and flexible environment; this is an efficient and effective approach comparing with traditional software development. The ASD shares the following principles, as stated in Agile Manifesto [16]:

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software;
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
- Business people and developers must work together daily throughout the project;
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done;
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation;
- Working software is the primary measure of progress;
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely;
- Continuous attention to technical excellence and good design enhances agility;
- Simplicity--the art of maximizing the amount of work not done--is essential;
- The best architectures, requirements, and designs emerge from self-organizing teams;
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Case Study (9): Rigid specifications are defined in a traditional software development process but in an agile process all changes are accepted and in some cases it can actually cost less than ensuring requirements will never change. In our Software Development Project, changes occurred almost weekly and usually those changes were the most important to implement. To avoid future troubles we suggest tracking all changes in a software requirements document. Weekly meetings are an essential factor for project evolution. On major milestones for the project we defined quality gates and we discussed requirements to be fulfilled to proceed to next step.

The most popular methods for ASD are: Extreme Programming (XP) [2] and Scrum [14], [11]. The XP methodology is by no means suitable everywhere and is aimed for small and medium sized teams limited between three and a maximum of twenty project members with strong communication and coordination between team members located in the same place considering also the business culture and the technology [1], [3].

Scum methodology is for small teams of less than ten team members and if more people available, multiple teams should be formed [1], [11]. Scrum process is described (Figure 3) in three phases [1], [11]:

- Pre-game - this phase includes two steps: (1) Planning which includes the definition of the system being developed and a Product Backlog list with all requirements and the definition of the project team, tools and other resources, risk assessment and controlling issues, training needs and verification management approval and (2) Architecture/High level design where architecture is planned based on the current items in the Product Backlog;
- Development - this is the agile part of the Scrum approach where unpredictable is expected and the different environmental and technical variables are constantly controlled in order to be able to flexibly adapt to the changes; the process is developed in iterative cycles where the functionality is developed or enhanced to produce new increments, called Sprints, with traditional phases of software development;
- Post-game - this phase contains the closure of the release when no more items and issues can be found nor can any new ones be invented and requirements are completed.

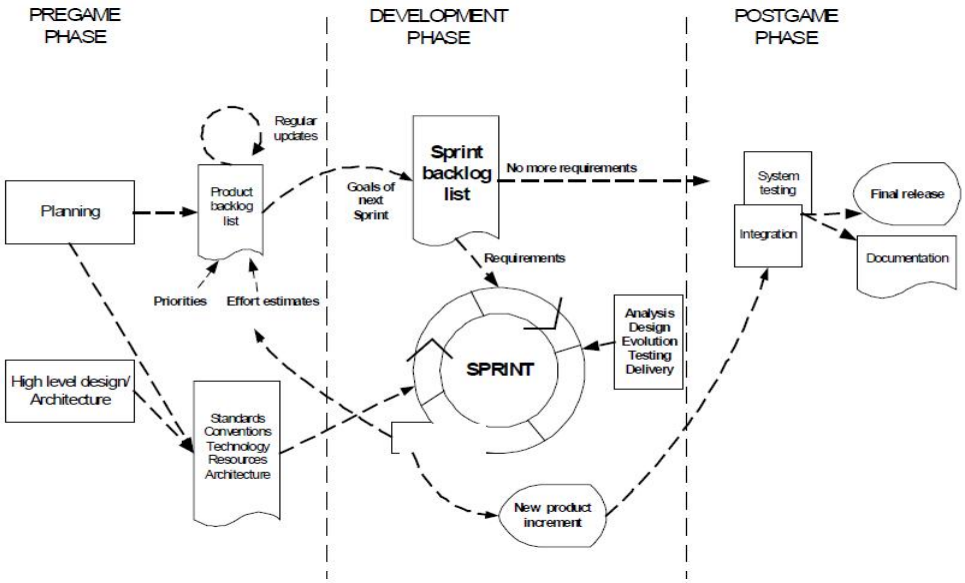


Figure 3 - Scrum phases[1]

As traditional software development methods, those agile software development methods comply with the same software development life cycle and have the same goal to deliver an optimal product or service in order to satisfy customers' needs. There are a variety of methodologies, each with its own specific and adapted to a specific process. Choosing the

[1] Adapted after [1] and [11].

best software development method depends on a lot of internal and external factors; major aim of a Software Development Project is to develop a final product in predefined cost, time, and quality. Failure in software can have potentially catastrophic effects, leading to injury or even death in case of medical device software and risk management area is very important throughout the software lifecycle, as presented in [9].

Conclusions

We consider that agile development methods are preferred when a team decides to implement a software solution and some of those methods share many benefits and common features but some of them are more focused than others. Traditional solutions and methodologies for software development are not used in the real software development environment.

In software development life cycle, initial software requirements may suffer major changes and developers must consider those changes in order to achieve customers' needs. From this point of view, an agile software development method is indicated and capable to solve all the problems. An agile software development approach is proper to satisfy customer needs that are changing during the software development process according to their principle for high-quality and low-cost products or services; it is also a proper approach for financial benefits and to achieve company goals.
Another aspect to consider is that new software development for a company has to integrate into its policy and rules in order to encourage its use and to avoid additional costs. That's why it is very important for example that during the development process to have meetings with top managers and final users of the software. In this way they will be familiar with the utility of the application and the training process will be more easily.

Acknowledgments

This work was supported by Romanian National Authority for Scientific Research under the grant no. 12119.2/01.10.2008 **INVITE.**

Bibliography

1. P. Abrahamsson, S. Outi, J. Ronkainen & J. Warsta, *Agile software development methods. Review and analysis*, Espoo, VTT Publications 478, 2001, 107 p.
2. K. Beck, "Embracing Change with Extreme Programming" in *IEEE Computer*, 32(10), pp.70-77, 1999.
3. K. Beck, *Extreme programming explained: Embrace change*, Reading, Mass.:Addison-Wesley, 1999.
4. B. Boehm, "A Spiral Model of Software Development and Enhancement" in *ACM SIGSOFT Software Engineering Notes*, ACM, 11(4), pp.14-24, August 1986.
5. S. Jecan, D. Benţa & L. Muresan, "Clustering companies profile and preferences" in *The Proceedings of the Ninth International Conference on Informatics in Economy*, Bucharest: ASE Publishing House, 2009.
6. C. Larman and V.R. Basili (June 2003). "Iterative and Incremental Development: A Brief History" in *IEEE Computer (IEEE Computer Society) 36 (6)*, pp. 47–56, June 2003.

7. G. Lee and W. Xia, “Toward Agile: An integrated analysis of quantitative and qualitative field data on software development agility” in *MIS Quarterly Vol. 34 No. 1*, pp. 87-114, March 2010.
8. J. Martin, *Rapid Application Development*, New York: MacMillan, 1991.
9. F. McCaffery, J. Burton & I. Richardson, “Risk management capability model for the development of medical device software” in *Software Qual J 18*, Springer Science+Business Media, LLC., pp.81–107, 2010.
10. K. Petersen and C. Wohlin, “The effect of moving from a plan-driven to an incremental software development approach with agile practices. An industrial case study” in *Empir Software Eng 15*, Springer Science+Business Media, LLC., pp. 654–693, 2010.
11. K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, Upper Saddle River, NJ: Prentice-Hall. 2002.
12. M.F. Smith, *Software Prototyping: Adoption, Practice and Management*, London: McGraw-Hill, 1991.
13. R. Subramanyam, F.L. Weisstein & M.S. Krishnan, “User Participation in Software Development Projects”, vol. 53/3 in *Communications of the ACM*, pp.137-141, March 2010.
14. H. Takeuchi and I. Nonaka, *The New Product Development Game*. Harvard Business, pp.137–146, Review Jan/Feb 1986.
15. W. Royce, "Managing the Development of Large Software Systems" in Proceedings of IEEE WESCON, The Institute of Electrical and Electronics Engineers, pp.1-9, August 1970.
16. K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R.C. Martin, S. Mellor, K. Schwaber, J. Sutherland & D. Thomas, *Principles behind the Agile Manifesto*, Available on-line: http://agilemanifesto.org/iso/en/ principles.html, [Accessed 05 Nov., 2010].
17. Selecting a Development Approach, Centers for Medicare & Medicaid Services, Office of Information Services, Department of Health & Human Service, USA, Original Issuance: February 17, 2005, Revalidated: March 27, 2008, Available on-line: http://www.cms.gov/SystemLifecycleFramework/Downloads/Selecting DevelopmentApproach.pdf, [Accessed 10 Sept., 2010].
18. United States House of Representatives, 111th Congress, 2nd Session, U.S. House of Representatives, Systems Development Life-Cycle Policy, Executive Summary, Available on-line: http://www.house.gov/cao-opp/ref-docs/SDLCPOL .pdf, [Accessed 10 Oct., 2010].