Evaluation of the progress in software development process and the reasons for delayed delivery

JyotsnaRupaliya, Aditya Trikha, Raunak Kumar

Abstract -Software project management is the process of planning and developing a software project accordingly following the given specifications, budget and time. Many software projects are delayed due to one or other reason. In this paper we will discuss the methodologies or measure for the progress evaluation and analyze the reasons for delay in the same context. Teams developing software project uses one of the many software development process models for planning the project and its successful implementation. Each process model is different from other in some aspect therefore, their progress evaluation technique should also be different. After analysing the methods we will try to look into the reasons for delay in project delivery due to slow progress of project and provide corrective measures for better evaluation of project progress.

I. Introduction

The development of a software requires a large amount of data. Also, there are several steps involved in the software development process like picking up a project manager, building a team of professionals to develop the software, setting a framework, creating a prototype, discussing with the clients and stakeholders, etc. There are many software process models which is followed to develop a software viz, waterfall model, Agile model, etc. In spite of following these process models, metrics and other guidelines, the software development process generally gets delayed. We try to find out and analyze the reasons for the delays in the project and suggest some measures to control the delays.

II. Keywords

Project progress, delayed delivery, agile development method, collaborative development tools, development teams, project manager, developers and project management.

III.An analysis of research papers:

PAPER I: Collaboration Tools for Global Software Engineering

By FilippoLanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaíno

In the process of doing a software project sometimes team members have to join in from different location and time. To develop better software and completing the work assigned more efficiently tools are used to make team members work together, share information and progress, and achieve the project goals collectively. The tools are essential to maintain coordination among team members and control of the entire process effectively. The paper gives a detailed description of collaborative development environment and tools used in effective development for global or co-located projects

There are seven collaborative tools as described in the paper:

- Version Control Systems - enables each team member to share software artifacts in a controlled manner. Some examples are subversion, Git, Mercurial etc.
- Trackers - are used to manage issues regarding the projects. Any issue posted on these platform has an identifier,

description and life cycle. Example: jira and bugzilla

- Build Tools – helps maintain project repositories and manage workflow using web based dash boards having status of current and past builds. Examples: Maven And CruiseControl
- Modelers – enables members to develop software artifacts, UML models and customized software processes. Examples: Artisan studio, rational software modeler, and visible analyst.
- Knowledge Centers – contains internal documents, technical references, standards, FAQs, and best practices, expert identification and skills management. Example: Eclipse help system, and KnowledgeTree.
- Communication Tools – there are many communication tools popular among software engineers such as email, mailing lists, newsgroups, Web forums, blogs, telephone and conference calls, chat, instant messaging, voice over IP, and videoconferencing. But there are more sophisticated tools such as WebEx, WorkSpace3D, IBM lotus, Domino also contains combining tools for document sharing and review, concurrent editing, and shared calendars.
- Web 2.0 Applications – very common in global and open source software projects, is a method for informal communication among team members. Examples: WordPress, Delicious, and Portland Pattern Repository.

Collaborative development environment(CDE) provides a combined workspace having a set of standardized tools useful to the entire team. These CDEs provides an environment which ease the development processes as well as increase the productivity. CDE often uses appropriate features of different collaboration tools to provide smooth environment for project development. Some popular CDEs are SourceForge, Gforge, Trac, Google Code, Assembla, Rational Team Concert, Git Hub, LaunchPad, and CodePlex.

Some activity specific collaboration tools are also available to developers. Examples are:

- Project Management: ActiveCollab and WordView are web based platform to track milestones and to maintain project calendar. These also provide the manager with the details of current activities derived from the workspace of developers.
- Requirement Engineering: These are specific tools for recording project requirements, use-cases and relationships between them. Oftendesigned to use natural language helps the developers to access and manage requirement information. Examples are: Doors, IrqA, eRequirements.
- Design: these are tools to help the developers with design related issues. The store all design related information, role definition and version control coordination. Examples are Camel, Gliffy and Creately. Gliffy can also be integrated with Jira tracking System.
- Testing: Some of the popular tools for managing the testing phase in the software development life cycle by managing test cases into test plans are TestLink, Selenium and OpenSTA.

None of the available tools or CDE can provide all the necessary functions and requirements for developing software projects. Users must identify their collaboration needs and requirements according to the project assigned and hence choose the appropriate tools or CDE. This the major reason every company uses different tools or CDE according to their own needs and culture. Effective tool support for

collaboration is a basic need for every organization with distributed resources, and strategies such as offshore development, outsourcing, or supplier networks. Software needs to be shared, and appropriate tool support is the only way to do this efficiently, consistently, and securely.

PAPER II: Information Needs for Software Development Analytics

By Raymond P.L. Buse and Thomas Zimmermann

Developing a software is a vast activity using rich data with worldly metrics. Still experts(engineers) frequently fall short of instrument and methods required to grip probably valuable information assets to make decision. Data needs and also their analysis needs have been presented in the paper referred; result of survey on more than 100 developers.

Engineering a software is an activity rich in content i.e. data. Various factors of development, from coding to testing the code frameworks to debugging, they are measurable with a high degree of computerization and efficiency. Throughout their life-cycle the projects are measurable: be it the specification or be it maintenance. There are number of metrics as well as models for intricacy, sustainability or maintainability, collapse aptness and many other valuable aspects of software quality and improvement process health have been suggested. Despite this large amount of data and metrics, improvement continues to be tough to predict and unsafe to conduct.

Various factors to be considered for on time successful project are:

- Benchmarking, organization tests, technical tests, etc.
- Identifying the needs of managers when assessing alternative technologies which impact on costs, quality, and schedule
- Managers shouldn't be much busy with their everyday tasks as then they will not have much time to perform measurement activities.
- Instead of managers preferring to collect information from colleagues and they should consider empirical evidence as comparably relevant.
- Goal-oriented perspective should be used as they take into consideration goals, tasks, plans or strategies, and many different mechanisms for guiding the decision to choose particular data to be collected and evaluated.
- Rather than having a separate organization, our ultimate aim is to empower software development teams so that they can independently gain and share information and conclusion from their data without depending on a separate entity.

If the above mentioned factors are not followed strictly then it may lead to delay, or even shutdown due to excess time or even technical difficulties.

Analyzing various important tasks and key points will help revolutionize decision making and help our software development process.

Software engineering has many standards that indicate a business process that guide itself well to analytics:

- Data-rich - Logics operates most excellently when huge amounts of data are accessible for analysis.
- Labor - Software engineering is especially labor demanding or intensive. There is an order of magnitude productivity difference or gap between developers.
- Timing dependent.
- Dependent on consistency and control
- Dependent on distributed decision making
- Monitor a project
- Improve risk management and efficiency

- Predict changes and review past decisions

The landscape of artifacts and symbols or indicators are well known in software engineering, but the strong decisions they might help in are not. We classify the types of decisions for each analytical question was said to help or assist with. Some general or common themes emerged:

- Targeting Testing: Needs information on the code that changes from build to build and the result of problem or bug fixes so that we can easily map out what traits and what other code needs reexamination.
- Targeting Refactoring: Various bug reports for a certain feature area helps us decide if the feature area is mature for a refactoring
- Understanding Customers: Analysis and Observation of the process help us understand the way user is using the product
- Judging Stability: Using fault or bug / fix / regression metrics over time to know what areas have not yet become stable and how much time it might take for us to make them stable
- Targeting Training
- Targeting Inspection

We conclude that meticulous holds out useful promise for improving the efficiency of software engineers and managers aim to make efficient informed decisions. Understanding meticulous for software development requires comprehending the information requirements of development managers; what their needs are, what decisions they influence, and how those needs map to analyses.

The Future works suggested in the paper reviewed can be categorized into categories like:

- Data Collections: We should rethink on our method of collecting data. So far, mining software repositories has had a data-focused approach,
- Data privacy: Data can be very dangerous when used inappropriately,

Other categories can be Understanding user needs, User experience as well as education.

PAPER III: Factors that Affect Software Systems Development Project Outcomes: A Survey of Research

By Laurie McLeod and Stephen G. MacDonell

Since 1975 researchers have made numerous attempts to know the decisive parameters of the project success and failure so that a successful outcome was more likely with mixed results. Earlier there were attempts or perhaps hope to channelize something we did not know completely of. Since then we gained a far more detailed understanding of the development of systems; this guarantees us successwhich remains a significant challenge, however.

In the paper we found reference and use of works by many others like: reasons for systems failure. Since 1987, many other class frameworks or models in their tries to discover and form an understanding of systems success or failure. Poulymenakou& Holmes utilize a model to examine the effects of many macro and micro contextual factors on systems project processes and outcomes. Butler & Fitzgerald also focus on the effect of institutional contexts on the systems development process and systems success.

We find from various aspects of Software System Development that consideration needs to be given to the various people, the individuals as well as the group, who are engaged and a part of or are interested in the projects of software systems development. Their actions,characteristics, relationships and interactions help inshaping the development

path and project outcomes in several ways, hence we see that there is an understanding of their responsibilities and actions during system development is also necessary.

The study used to jump to conclusion and develop patterns had focused on software systems project outcomes or factors influencing software systems development (rather than system evaluation or systems in use). The study provided empirical data on systems project outcomes or factors influencing systems development.

Various Influential Factors for the development of The Software were considered. Consideration is given to the potential influence of interaction between individuals or groups on systems development and its outcomes.

- Developers - technical abilities, other capabilities, experience; interpersonal, communication skills; domain knowledge; motivation and trustworthiness; and norms, values and beliefs. Significant variation in skills and capabilities of developers can influence development productivity and hence project outcomes.
- Users - users may affect the outcome as they may have different expectations of the system being developed;different attitude towards and involvement with the system; and different specific characteristics that may affect their ability to utilize the system.
- Top Management - The top management support, commitment as well as understanding in the software systems development literature as important in determining the outcome of a project.
- External Agents - Challenges with external consultants or contractors could include the nature of the contract and contractual issues; lack of understanding or misinterpretation of organizational requirements by consultants; lack of control over the actions of external consultants; poor product quality and poor service; communication problems between consultants and users etc. affect the development of a software.

Influential factors-
Project content like project team and the social interaction also affects the software development, Project Characteristics, Project Scope, Goals and Objectives, Resources and Technology along with other influential factors of software development process like Requirements Determination, Project Management, Use of Set Method, Participation of users, Training of user ; Management of Changes, as well as influential factors of institutional context like Organizational Properties, Environmental Conditions, and other affect the Development of Software to a large extent.

The categorization of work works as a useful analytical device for classifying and compounding the empirical literature on parameters affecting software systems development. It can be utilized for equipping analytical abstractions, since the four factors of the framework are conceptually more manageable than having to deal with the eighteen individual factors identified by our review. When greater detail is required, consideration can be given to the individual factors within each dimension.

From the various points outlined in the paper we infer that work in this field in times to come needs to take many trends into consideration. Given the significant modifications seen over the time, it looks considerable to expect more changes in software systems development and procurement in the future.

PAPER IV: Decision Making in Agile Development: A Focus Group Study of Decisions & Obstacles
By MeghannDrury , Kieran Conboy , Ken Power

Most of the companies and software development teams employeesAgile life-cycle methods these days. Agile methodology helps the teams to answer unpredictability through incremental, iterative work sequences i.e. sprints. It is an alternative to traditional development models. Agile software development(ASD) teams are often involved in decision making that are crucial towards the success or failure of the project. Decision making is divided into four groups in ASD Sprint Planning, Execution as well as Review and Retrospective. In ASD team delivers the work in short cycles called sprints which requires in more frequent, short-term decisions.

- Sprint planning: marks the start of each activity in the sprints, planning for the work to be done in sprint is decide in this phase.
- Sprint Execution: development aided testing of product is done in this phase. This marks the period between planning and review.
- Sprint Review: involves the whole team, stakeholders and other related authorities. Includes demo of the developed product from the execution phase and comparing the deliverable with the specified requirements.
- Sprint Retrospective: it is phase to look back at team performance and find out areas of improvement and inspect the working of team.

A research survey was organized by the authors in a group of 43 software engineers who uses agile method for their project development and data was collected on following topics:

- Importance of decision making for agile teams
- Decisions participants make(4 agile periods)
- Perceptions of participant about the obstacles to decision making during the four periods of the sprint cycle
- Issues related to decision making (to find out any other obstacles)

Analysis from the survey gives the following points over the decisions made in the ASD:

- Decisions Made in Each Agile Period -
    - Sprint Planning
        - Decide sprint goals and scope
        - Decide priorities within the sprint
        - Decide members of team
        - Decide capacity for team members
        - Decide each team member's role
        - Decide task estimates
        - Decide definition of when a sprint is completed
        - Decide the approach to deliver the goals
    - Sprint Execution
        - Decide whether sprint scope should be changed or reprioritize tasks
        - Define when feature is completed
        - Decide the interface design
        - Decide ways implement functionality
        - Decide time commit code X
        - Decide test cases

- Sprint Review
  - Decide if delivered product meets user requirements
  - Decide whether to continue with the project
  - Decide acceptance the sprint content
  - Decide goals and defects be scheduled for next sprint, if not completed
- Sprint Retrospective
  - Decide improvements for the next sprint
  - Decide achieved goals for next sprint
  - Decide reasons is goals are not achieved
  - Decide priorities for future sprints
  - Decide factors that will influence team success
  - Decide criteria to measure team metrics

- Obstacles to Decision Making in Agile Teams
  - People are Unwilling to Commit to a Decision
  - Conflicting Priorities
  - Inconsistent Resource Availability During Sprint
  - Decisions are Not Implemented
  - Lack of Ownership
  - Lack of Empowerment

Lack of clarity of requirements and adding of new requirements during the project makes it difficult for the team to deliver project on time. Decision making is a very important process for the successful delivery on time, if the decision making process goes wrong it may lead to project delay or even to the failure of the project. Conflicting priorities are always bad to the health of project and also make it difficult for agile teams to focus on decisions for planning.

IV.Survey organized

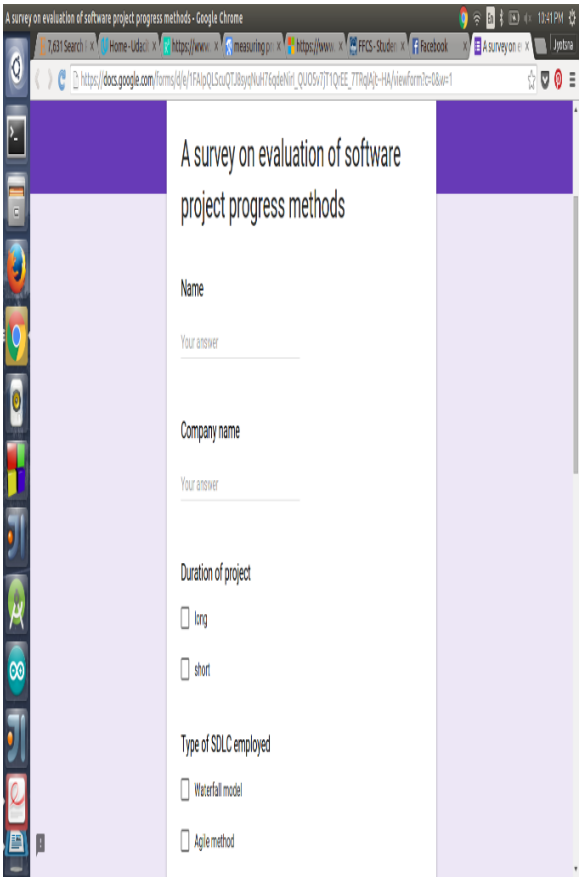

Image 1

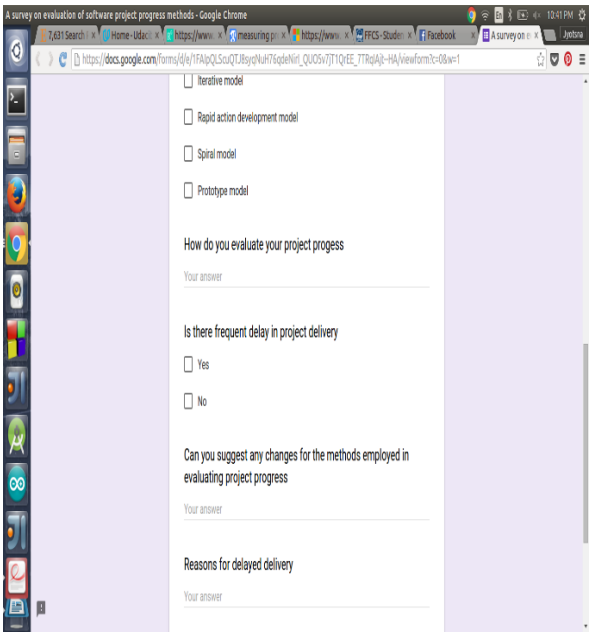

Image 2

V. Data collected

Data is collected from software engineers practicing in various companies namely Akamai, Furlenco, TCS, Wipro, Infosys, Flipkart, Morgan, HP, Fedora, NTT data, Smartek21, Dell and American Express. The results obtained are as follows:

- 77.2% projects are of long duration in comparison to 22.7% of short term duration
- Type of software development life cycle incorporated
    - Waterfall – followed by 8% of software engineers
    - Agile - followed by 52% of software engineers
    - Iterative - followed by 24% of software engineers
    - Prototype - followed by 4% of software engineers
    - Spiral - followed by 12% of software engineers
- A major portion almost 45% of software projects are delayed. There were many points recorded as reasons for delay in delivery
    - Changes in the requirements for a running sprint causes a lot of delay. A more optimistic planning at the start of projects cause delays in the delivery.
    - Unforeseen changes in requirements during the development phase of the project.
    - Resource unavailability
    - More time gets spent on designing the system than actual implementation.
    - Problem in applying what customer wants. Dependency in other teams
    - Miscommunication and lack of communication between onshore and offshore teams which results in work getting duplicated of left out.
    - Over planning
    - Multiple layers of management trying to manage same project in their own ways.
    - Blame games between teams.
    - Environmental issue.
    - Unclear requirements
    - Changing architectures and requirements at mid stages
- Methods for evaluating project progress used are following:
    - Kanban board made on asana
    - SCRUM meetings
    - JIRA Tracking
    - Number of modules completed as compared to what was planned.
    - Bugzilla for tracking the project progress
    - Following a release plan bringing in new iterations
    - Project demos and feedback evaluation.

- Metrics and reports
- Completing sprints
- By defining goals in terms of features to be completed at the onset of every sprint and then evaluating the progress at the end of every sprint.
- Features to be completed are assigned to different sprints which helps in getting an overall sense of completeness over the lifetime of the project.

- Some improvements were also suggested for better development of project smoothly without conflicts and timely delivery:
  - Project progress evolution should also take into action the changes in requirements as the project progresses and not just measure the progress against the initial set of requirements.
  - more flexibility instead of pressure
  - Freezing the requirements early.
  - Better communication between team members.
  - Progress should be evaluated with the completeness of requirements and specific modules
  - Clear understanding of requirements and requirement freeze are the most critical factors for successful delivery
  - Regular feedback and evaluation. Cross check with the project stakeholders, and market demands.

VI.Measures To Prevent Software Development Delay

- Choosing project manager and such personnel that meets the requirements of the software project.
- Setting realistic time schedules for delivering the project. This will provide a better understanding between the employees and the project manager and will also give the employees the required time to complete the modules of the project without any bug.
- The project manager should oversee the timely execution of the project. For this, he/she can hold meetings and get together regularly.
- A thorough must be undergone with the client before starting the project, in order to understand the business requirements.

VII. An analysis of data

A detailed analysis of collected data suggests that a very large portion of projects gets delayed due to various reasons which mainly include changing requirements and improper methodologies to evaluate project progress and improper communication to and between teams. Implementing and conceiving new modules in a software project can cause a significant delay in the delivery of the project. Spending more time in the elaboration and detailing of the project, rather than implementing its necessary modules can cause drastic delays. Unrealistic time schedules for completion for software project cause delay in project or the modules will not be implemented properly or there can be bugs which causes further delay. Poor design decisions are also a contributing factor for delay which have to be addressed during the development phase.

VIII.A discussion of possible future work

A lot of improvement can be done in the field of methodologies for measuring software project progress and also for to get rid of the delayed delivery problem on such a large scale. Still

today there is no complete environment which can all the different collaborative tools for keeping track of progress. A whole new Collaborative development environment(CDE) can be developed which consists the best features of each collaborative tool and gives the best CDE that facilitates the software development process as well as software engineers and managers.

As to the issue of delayed delivery following changes can be introduced to eliminate this problem:

- Choosing project manager and development team that meets the requirements of the project.
- Setting realistic time schedules for delivering the project.
- A thorough business analysis requirement with the client before starting the project, in order to understand the business requirements and freezing the requirements before starting the implementation phase.
- Timely review meetings during the projects and better communication among the team
- Avoiding or finding proper solution to conflicting decisions.

References

[1] FilippoLanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaíno, "Collaboration Tools for Global Software Engineering" IEEE Software Volume: 27, Issue: 2, March-April 2010

[2] Raymond P.L. Buse and Thomas Zimmermann , "Information Needs for Software Development Analytics", IEEE Press Piscataway, NJ, USA 2012

[3] Laurie McLeod and Stephen G. MacDonell, "Factors that Affect Software Systems Development Project Outcomes: A Survey of Research", ACM Computing Surveys (CSUR) Volume 43 Issue 4, October 2011

[4] MeghannDrury , Kieran Conboy , Ken Power, "Decision Making in Agile Development: A Focus Group Study of Decisions & Obstacles", 2011 Agile Conference

[5] https://www.projectmanagement-training.net/

[6]https://torbjornzetterlund.com/software-projects-get-delayed-fail/