*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 1, Ver. IV (Jan.-Feb. 2017), PP 50-54*
*www.iosrjournals.org*

# Time to Value: Insights on Software Engineering Practices

Dr Zeenat Alam

## I. Introduction

Time toValue isspeed at which the company can transform product ideas into saleable product and create value. Customers are ready to pay premium price for short response time,consequently timelinesstranslatesinto revenue. The concept helps in penetrating the market for winning new customers and bringing new inventions to market fast and creating market share.

Rapid time-to-value is important for the competitive success of many companies for the following reasons.

a) Competitive advantage of getting to market sooner;
b) Premium prices early in life cycle;
c) Faster breakeven on development investment and lower financial risk;
d) Longer market life cycle; and
e) Greater overall profits and higher return on investment.

The key process requirements for shorter time-to-value are:

1) Clear understanding of customer needs , at the start of the project and stability in product requirements or specifications partnering with customer;
2) A characterized, optimized product development process;
3) A realistic project plan based on this [2] process;
4) Availability of needed resources to support the project and use of full-time, dedicated personnel;
5) Early involvement and rapid staffing build-up to support the parallel design of product and process;
6) Remove Waste in the processes :
   a) **waiting**: in queues, in transit, for the next step to begin
   b) **excess**: over-processing, scrap, inventory, tasks done to a higher standard than required
   c) **rework**: items that need to be remade or fixed, tasks that need to be done again, errors that need correcting
   d) **redundancy**: items that aren't needed at all, tasks that didn't need to be done at all
7) Prototype product development to minimize time consuming physical mock-ups and testing;
8) Design re-use and standardization to minimize the design content of a project.

## II. Key Performance Indicators

The Main measurement areas for success of TTV are as follows:


**Fig 1:** Key Performance Indicator Map

**A. Revenue**

**1. Volume of Iteration/Output**

Volume of finished products that is in saleable format, is directly proportional to the revenue growth of any firm in software world it'd be measured asthe number of releases for single customer or several customers. Economies of scale is most effective way for reaching fast moving , cost effective and profitable organization We see it happening in Fast moving consumer goods , Infrastructure projects and now in Electronic and Software Industry .

**The KPI Measurements can be done with following calculations:**

- Volume of releases per unit time [Duration varies from month/s]
- EBIT per quarter / per month
- Business ROI
  *ROI = [(Financial value – Project cost) / Project cost] x 100*
  In looking at the formula, there are two components we need to determine: Financial value and project cost.

1. It quantifies project value –the most important aspect of ROI is its ability to show business leaders dollar figures of a project's worth. ROI turns the subjective into the objective, which can often turn uncertainty into support.
2. It can build stakeholder support – Tying a dollar value to a project may help with a "go/no-go" decision. Many times, stakeholders want to see what the dollar value is to them if they are to support a particular project. Without an ROI, that is very difficult to do.

**2. Time To Market**

Organizations can measure the Time To Market Value for their goods /products /solutions using following calculations:

**a. Time to Market = Elapsed Time from Idea Conception to Delivery**

**b. Alternate Time to Market = Actual Completion Time - Budgeted Completion Time**

**c. Actual Completion Time** The Actual production time / Potential production time

**d. Average cycle time – ACT**

Time from the order to the delivery of final ordered product

e. Cycle Time Ratio

CTR – Standard Cycle Time / Real Cycle Time

**3. External Quality**

Quality of Software system is measured on reliability, performance and stability of the system. It also caches on the turnaround time for solution of bugs.

New projects need to define the Software Quality Metrics Methodology,in order to achieve planned targets .

| Software Quality Activity | Development Cycle Phasing |
|---|---|
| Establish software quality requirements | _____ |
| Identify software quality metrics | _____ |
| Implement software quality metrics | _____ |
| Analyze results of these metrics | _____ |
| Validate the metrics | _____ |

**Fig 2:** IEEE Software Quality Metrics Methodology

Quality is measured bydefects per KLOC [1000 Lines of code] which also illustrates the defect density Software problems are not, always defined as, valid defects, but rather are due to individual user and organizational learning curves. It places an enormous burden on user support during the early days of a new release. The catch here is that neither alpha testing (initial testing of a new release by the developer) nor beta testing (initial testing of a new release by advanced or experienced users) of a new release with current users identifies these problems. The purpose of a new release is to add functionality and performance to attract new users.

**B. Efficiency Costs**

**Timeliness**

The act of saving money by making a product or performing an activity in a better way defines the efficiency cost. It focuses on removing waste in processes and procedures organizational structure, Responsibilities etc ; and encourages continuous assessment and improvements .

It can be measured on following parameters:

- Timeliness
- Automation level with respect to manual effort with activities.
- Re-usability of products, tools and processes.

**C. Flexibility Cost**

An expense that is easily altered or avoided by the firm bearing the cost. Flexible expenses are costs that may be manipulated in amount or eliminated by not engaging in the activity that incurred the expense.

Includes the

## III. Factors to Create Project Success

**Survey was conducted in 2016 for 50 projects across 20companies working on enterprise and digital solutions, with intend to derive key success factors for projects and their weightages.**

The questionnaire was prepared with regressive characteristic to identify the main reasons for project failures in Waterfall SDLC model. Some of the key highlights from the response were

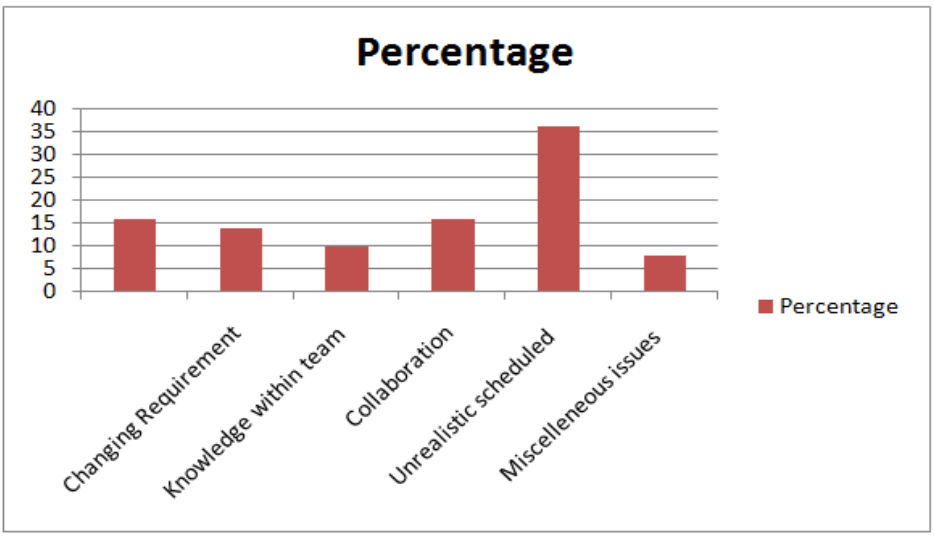


**Fig 2: Reason for Failure of Projects**

**Factors of Project Success and Their Reasons:**

**1. Modular Products. Development Approach**

Reduce time to write and maintain code. The Solution needs to be

- Modularized based on functional scope. It gives feasibility to put modules together and create solution as per customer need.
- Code needs to be simple and direct, It should be pleasantly graceful.
- Easy to understand, maintain and enhance. , by other developers.
- It is important that there are Coding guideline developed for the meaningful names for abbreviation used for variables and functions.
- Also Continuous reviews for Refactor duplicate code, and also to Refactor Long Methods for shorter methods.
- Comments in the program should be to explain "Why" and not "What".
- Avoid Complexity where issue shows deeper problem. Avoid Comments to explain code

**2. Assimilated Configuration Management**

In Assimilated configuration management (CM), development teams apply CM at the solution level, they must consider production configuration issues between their solutions with outlook of the external Interfaces with other vendor system. This can be a major change for some developers because they're often used to thinking about configuration management only in terms of the solution they are currently working on and the portfolio testing scope. In a Devops environment, developers need to understand, and manage, the full range of dependencies for their product. Integrated configuration management enables operations staff to understand the potential impact of a new release, thereby making it easy to decide when to allow the new release to occur.

**3. Integrated Change Management**

Integrated change management is the process of ensuring successful and meaningful evolution of the Application and infrastructure together to better support the overall Integrated Solution. This is delicate at a project -team level because many technologies, and even versions of similar technologies, will be used in the development of a single solution. Since DevOps brings the enterprise-level issues associated with operations into the mix, an integrated change management strategy can be far more complex, due to the need to consider a large number of solutions running and interacting in production simultaneously. With integrated change management, development teams must work closely with operations teams to understand the implications of any technology changes at an organization level. This approach depends on the earlier practices of active stakeholder participation, integrated configuration management, and automated testing.

**4. Continuous Integration**

Continuous integration is mechanism for developing andvalidating software of a project, through automated regression testing and code analysis whenever updated code is checked into the version control system. CI enables developers to develop a high-quality working solution safely in small, regular steps by providing immediate feedback on code defects.

Success of the phase depends on:

- Strong Infrastructure or environment to test the setup
- Code quality and alignment with functional scope.

**5: Implement test automation and test data provisioning tooling**

Test automation is more than just automated testing; it's the ability to take code and data and run standard testing routines to ensure the quality of the code, the data, and the overall solution. With DevOps, testing must be continuous. The ability to toss code and data into the process means you need to place the code into a sandbox, assign test data to the application, and run hundreds — or thousands — of tests that, when completed, will automatically promote the code down the DevOps process, or return it back to the developers for rework.

**6: Perform acceptance tests for each deployment tooling**

Part of the testing process should define the acceptance tests that will be a part of each deployment, including levels of acceptance for the infrastructure, applications, data, and even the test suite that you'll use. For the tool set selected, those charged with DevOps testing processes should to spend time defining the acceptance tests, and ensuring that the tests meet with the acceptance criteria selected.These tests may be changed at any time by development or operations. And as applications evolve over time, you'll need to bake new requirements into the software, which in turn should be tested against these new requirements. For example, you might need to test changes to compliance issues associated with protecting certain types of data, or performance issues to ensure that the enterprise meets service-level agreements.

**7: Ensure continuous feedback between the teams to spot gaps, issues, and inefficiencies**

Feedback Loops to automate communication between tests that spot issues, and tests that process needs to be supported by your chosen tool. The right tool must identify the issue using either manual or automated mechanisms, and tag the issue with the artifact so the developers or operators understand what occurred, why it occurred, and where it occurred.

**8. Integrated Deployment Planning**

From the point of view of development teams, deployment planning has always required interaction with an organization's operations staff; in some cases, via liaison specialists within operations typically called release engineers. Experienced development teams will do such planning continuously throughout construction with active stakeholder participation from development, operations, and support groups. When you adopt a DevOps strategy, you quickly realize the need to take a cross-team approach to deployment planning due to the

need for operations staff to work with all of your development teams. Deployment to Integrated System test environment, UAT or Performance labs or Pre-production and production . Furthermore, to support continuous deployment, release engineers will need to increase the number of release slots available to agile teams that are disciplined enough to continuously and consistently meet the quality requirements for release.

**9**. **Continuous Deployment**

Continuous deployment extends the practice of continuous integration. With continuous deployment, when your integration is successful in one sandbox, your changes are automatically promoted to the next sandbox, and integration is automatically started there. This automatic promotion continues until the point where any changes must be verified by a person, typically at the transition point between development and operations. Continuous deployment enables development teams to reduce the time between a new feature being identified and being deployed into production. It enables the business to be more responsive. However, continuous deployment increases operational risk by increasing the potential for defects to be introduced into production when development teams aren't sufficiently disciplined. Successful continuous deployment in an enterprise environment requires all the practices described earlier.

**10. Production Support**

In enterprise environments, most application development teams are working on new releases of a solution that already exists in production. Not only will they be working on the new release, they will also have the responsibility of addressing serious production problems. The development team will often be referred to as "level three support" for the application because they will be the third (and last) team to be involved with fixing critical production problems. Although the need to do level three production support is common, with the exception of Kanban and Disciplined Agile Delivery (DAD), many agile methods only address this effort in passing. An important side effect of this practice is that it gives developers an appreciation of the kinds of things that occur in production, providing them with learning opportunities to improve the way that they design solutions in the first place.

**11. Application Monitoring**

As the name suggests, this is the operational practice of monitoring running solutions and applications once they are in production. Technology infrastructure platforms such as operating systems, application servers, and communication services often provide monitoring capabilities that can be leveraged by monitoring tools (such as Microsoft Management Console, IBM Tivoli Monitoring, and jManage). However, for monitoring application-specific functionality, such as what user interface (UI) features are being used by given types of users, instrumentation that is compliant with your organization's monitoring infrastructure will need to be built into the applications. Development teams need to be aware of this operational requirement or, better yet, have access to a framework that makes it straightforward to provide such instrumentation.

**12. Automated Dashboards**

The practice of using automated dashboards is business intelligence (BI) for IT. There are two aspects to this, development intelligence and operational intelligence. Development intelligence requires the use of development tools that are instrumented to generate metrics; for example-Configuration management (CM) tools already record who checked in what and when they did it.Continuous integration tools could similarly record when a build occurred, how many tests ran, how long the tests ran, whether the build was successful, how many tests were successful, and so on. This sort of raw data can then be analyzed and displayed in automated dashboards. Operational intelligence is an aspect of application monitoring discussed previously. With automated dashboards, an organization's overall metrics overhead can be dramatically reduced (although not completely eliminated because not everything can be automated).

Automated dashboards provide real-time insight to an organization's governance teams.

## IV. Conclusion

Time To value derives its power from simplicity, modularity, smart model of working and continuous improvement/innovation which help shorten the development lifecycle and deliver quality and value faster to the customer.. Ttv KPI’s will help to measure theeffectiveness with respect to verticals – of Revenue, Efficiency cost and Flexibility Cost. Engaging andharnessing powerto bring Customer Life value, with long term relationship.

## Reference

[1]. Managing the Global Supply Chain by TajeSkjott-Larsen.
[2]. Concurrent Engineering Techniques and Advancement by C.T Leondes .