



POINTS OF DEFECT CREATION

- SPEEDING DETECTION AND CORRECTION IN
PRODUCT DEVELOPMENT

Authors:

Shankar Krishnamoorthy | Krishna Sivaramakrishnan | Aparna Venkateshwaran

Software Product development methodologies try to improve quality by promoting the tactic of testing "early and often." When a defect is detected, a report is filed, the developer tries to reproduce and repair the problem, and then testing must verify that the modification corrected the problem and did not create any new problems. Because it doesn't prevent the same types of errors from recurring, this approach is time consuming, costly, and inefficient. Errors are introduced into the product at various stages—requirements definition, design, and coding. If we focus our efforts on eliminating defects at the points of error introduction, we can reduce the time spent on error detection and correction. This white paper discusses the best practices for error prevention that Aspire Systems has discovered in their experience of developing almost five hundred products.

- * Traditional software testing practices versus error prevention methods*
- * The value of error prevention in lowering costs*
- * Error prevention practices from the real world*

PRODUCT DEVELOPMENT

We can divide software development into two broad categories: Product Development and Application Development. There is significant difference between developing a product (eg. Yahoo! Messenger, www.salesforce.com) and an internal application (eg. employee portal, marketing information systems). Though, from a high (20000 feet) level, developing an internal application and a product would look similar from the technology and development process perspective; but, approach to product development is quite different from developing an application. Lot of effort goes towards assuring product quality and making it defect free. Cost of rework and defects are quite high in case of product management.

Several articles related to defect prevention (and cost of defects) are available all over the Internet. We, as an organization, have been adopting best practices in defects prevention at different phases of software product development life cycle. This paper discusses about identifying defect creation points as the key to defect prevention in software product engineering

DEFECT CREATION POINTS

In our experience, five major factors that influence the product development process are:

- Product vision
- Product lifetime
- End users
- Product stage (early stage, mid stage, matured stage)
- Product release dates

Clear understanding of these factors is very important for the success of a product. They also become the key points of defect creation; they drive the development methodology, processes, tools and skill sets of people. All the stakeholders must have clear understanding of these and communicate to their team.

STORY 1

Consumer internet product in customer loyalty space: It is a desktop product and directs users to a site that would help in saving few dollars in online shopping. This product looked very simple with few functionality. Development team went ahead with creating requirement specs, product UI prototype and appropriate design. Non functional specifications (mechanism of communication protocols, security, etc.) were understood right in the beginning and several POCs were built as a part of the development. This product engineering team comprised of development team and a product manager. Team was very comfortable with the work until the product was released for beta testing. The way, beta testers (sample of user cross section) used the product helped unearthing quite a bit of new requirements – needless to say all these were identified as defects.

Moral: Know your customers
Root cause: Missing requirements
Defect creation point: Missing requirements.

STORY 2

SaaS based business model: This product is based on a novel idea and has a huge business potential. It is to be used by large businesses for interfacing with various users and automating their purchase transactions between users, purchase teams and vendors. When initially the work was started, all the specs were tied to a single business (beta customer). So, the specs and design were skewed to the requirements of a single customer; Team faced several difficulties and ended up in reengineering the product to generalize it when more customers were added.

Moral: Identify more number of beta customers; they will help in generalizing the product.
Root cause: Developing a generalized product for a single customer.
Defect creation point: Insufficient exposure to product development process.

STORY 3

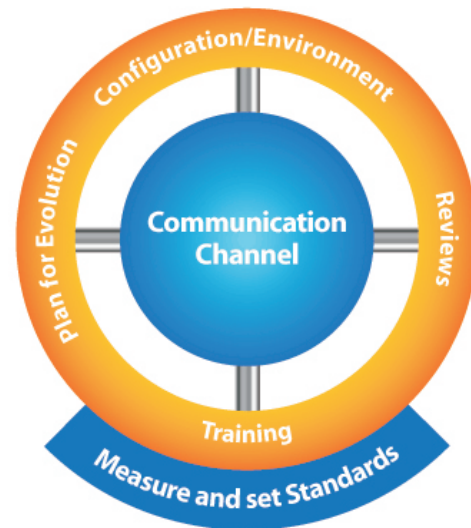
Product in industrial automation space: The marketing team required this product to be developed and released in very short duration; but, requirements were evolving. Developers never got an opportunity to put in time to find out reusable components / do a scalable, maintainable architecture. On top of this, they did not have knowledge of how the product would be used. This resulted in lot of duplication of same piece of code. Code base has increased significantly and several coding related defects emanated, obviously delaying entire product release. Good news is that this product has evolved into a product line and has almost 15-16 products.

Moral: Train the team on appropriate skill sets (product knowledge, reusability, etc.)
Root cause: Team skill sets, short development time.
Defect creation point: Insufficient training, Insufficient reviews and communication gap.

PROPEL ONE

All the above storylines are some examples for defect creation points. We have analyzed different scenarios and bugs that were introduced in products over a period of time. Statistics show that close to 60-70% of human errors emanate due to a mix of three factors: communication gap, inadequate review mechanisms and inadequate training. From our experience, we have found that if any of the following has issues, it would induce defects into the product:

- Setup proper communication channel
- Setup right Environment/Configuration
- Plan your Reviews throughout the life cycle
- Train the team on required skill sets
- Plan for Evolution (of everything)



We have developed a framework based on the above principles. We call this framework as ***Propel One***. All our tools and techniques for defect prevention are around this framework. This framework consists of the following:

- Clear definition of roles and responsibilities of all stake holders
- Procedures/Guidelines for various activities
- Checklists
- Collaboration mechanism (Communication framework, Knowledge Management systems, Document management systems, Source control systems, Issue tracking tool(s), etc.)
- Status review reports / meetings – specifically analyzing metrics

All the stake holders communicate to each other through email, IM, telephone, etc. and produce lot of information. This information is spread across multiple emails, multiple documents, etc. And, it also leads to a situation where it becomes subjective on who gets what information and it is upto to the email sender to copy the mails to all the stake holders informed. Also, when the information is shared across different emails or documents, teams end up in spending lot of time in retrieving appropriate (latest, valid) information.

This problem can be addressed to a decent extent by implementing a collaboration mechanism which will help in keeping all the information in a single place. There by, subjectivity gets eliminated, duplication of efforts get reduced and message retrieval becomes easier. There are several tools available, for eg. Microsoft Sharepoint Portal Server, Windows Sharepoint Services, Groove Networks, ActiveCollab, BaseCamp, Wiki, etc.). On the source code management systems, again, there are several tools available, for eg. (Clearcase, VSS, CVS, Perforce, Subversion, etc.). These mechanisms along with personal discipline can do wonders in reducing the defects due to communication gap, configuration management, environment issues etc. Communication channel is the backbone of ***PropelOne*** framework.

Several reviews can be planned / done over the product life cycle – prototype review, requirements review, design review, etc. It is important to involve appropriate stake holders and experts for these reviews. For example, when the product requirements are under discussion, it is always helpful to

build a UI prototype and have the business user review it. It will unearth lot of use cases which might have not been captured during the requirements. Same way, when you do the design, you have to review the design against product vision, roadmap, implementation environment, security requirements, etc. Here, again, the product manager has the responsibility of involving appropriate stake holders. Often times, it is a common sense approach, but, in most of the cases, review gets skipped, which leads to defects infusion.

Overall, reviews will be very helpful in several ways – addressing missing requirements, removing gold plating of requirements, simplifying design, improving the architecture, bringing in best practices and recommendations, etc.

We apply this framework against all our service categories / different stages of the product:

- Early stage product
- Growth stage product
- Matured stage product

Each of these has variants with respect to life cycle, team skill sets, etc. Let us discuss how our **Propel One** framework addresses various defect creation points (and eliminates defects) in each of these stages.

EARLY STAGE PRODUCT

There are several challenges while developing a new product – requirements would be evolving, technology and design decisions are to be made, team need to become comfortable on the domain/technology, etc. Every phase has several defect creation points. Following table shows some of the defect creation points in early stage product and how this Propel One framework helps in closing these defect creation points.

Following table shows a sample of pain points and how PropelOne addresses these pain points in early stage product development:

Pain Points	Root Cause	How Propel One framework addresses this pain point?
Product I wanted is not the one I got	Missing requirements, communication gap between stake holders	Reviews, collaboration mechanism
Mis-interpretation of requirements	Communication gap between stake holders and no reviews, insufficient exposure to developers on product usage and domain.	Train the team on the domain
Not user friendly	Insufficient reviews	Reviews
Product is not stable	Insufficient reviews, inadequate design (and could be any other reasons)	Reviews, train the team on design principles, etc.

GROWTH STAGE PRODUCT

Product vision guides the product evolution. Customers would have newer wish lists. So, a product will have to go from 2.0 to 3.0, 4.0, etc. with newer features. Development team has to constantly implement these wish lists. In this case, there are several defect creation points – new functionality implementation, impact analysis, backward compatibility and extending the architecture (based on requirements).

Following table shows the pain points and how *PropelOne* addresses them:

Pain Points	Root Cause	How Propel One framework addresses this pain point?
Features are not customizable	Missing requirements (analysis of customer requirements and priorities), communication gap between stake holders (product team and customer team)	Reviews, collaboration mechanism
Performance is degrading	Inadequate design, inadequate reviews (or testing) of design, coding, functionalities, etc.	Plan for reviews
Adoption to new technologies and business models, new markets		Collaboration mechanism
Product roadmap and customization requirements are conflicting		Collaboration mechanism
“Difficult to maintain” code / Uncontrolled code segments	Inadequate reviews, insufficient training to the team	Plan for reviews and training

MATURED STAGED PRODUCT

All the products will go through a maintenance period. Depending on the nature of the product, customers acquired over a period of time, technology and size of the product, maintenance period varies among products. Typical tasks in this phase would be feature enhancements, bug fixes, etc.

The table below explains possible defect creation points and associated solutions during maintenance mode.

Pain Points	Root Cause	How Propel One framework addresses this pain point?
Maintenance is taking more to fix issues / release patches	Insufficient Training, Difficulty in understanding the impact of a fix on the product, etc	Reviews, collaboration mechanism (for knowledge management and continuity), train the team on product, architecture, critical modules of the product, etc
Managing multi-code base (issue fixed in one code base needs to be propagated to other code bases).	Insufficient understanding of product roadmap and development process, insufficient reviews which resulted in multi-code base.	Plan for reviews and training
Compatibility Issues (OS, App Server Versions, SDK Versions)	Configuration management issues	Implement proper configuration management plan

Propel One framework helps in disciplining the development process. It insists on the following:

- Continuously Collect data
- Perform causal analysis to identify the root cause of defects and suggest preventive actions
- Initiate an action team to implement the preventive actions
- Conduct knowledge sharing meetings to increase awareness of quality issues specific to each development stage and
- Constantly monitor the improvements

We have been incorporating Propel One framework for the past one year and we have experienced good results (improved customer satisfaction) in our overall rework / defects injected in several of the product development initiatives that we have participated.

LESSONS LEARNT/SUMMARY

- Although Defect Prevention is considered an SEI/CMMI Level-5 KPA, we found that a strong Level-3 organization, with Propel One framework, can build an effective Defect Prevention Process.
- Train the team so that they bring in personal discipline in their work. Due to the nature of product development, requirements and human errors would be the major points of defects creation. An effective training plan would help in reducing these errors significantly.
- Plan for reviews throughout the development lifecycle.

We consider Defect Prevention as the process of improving quality and productivity by preventing the injection of defects into a product. This Propel One framework has been successfully implemented in a variety of software products within Aspire and it is continuously improving.

Aspire Systems (India) Pvt. Ltd.
 1/D-1, SIPCOT IT PARK, Siruseri, Chennai - 603 103
 Tel : +91-44-67404000. Fax: +91-44-67404234
 e-mail : info-india@aspiresys.com