Product to Employee Engagements: Best Practices

Samrat Dutta
M.Tech, International Institute of Information Technology, Electronics City, Bangalore
Software Engineer, IBM Storage Labs, Pune
samrat.dutta@iiitb.org

ABSTRACT

The term 'work environment' is very subjective. People have different opinions over the same thing, some may be positive or some may be negative. At the end, every body wants work satisfaction, fun and improvement over the work and get recognized. To do this many software companies adopt several techniques. This paper collectively presents some practices ranging from the very basic to the best practices through which to address the above issues and their technical aspects.

INTRODUCTION

A software company enforces many software development life cycles in their projects, including requirement analysis, development, testing and maintenance, to name a few. But among these processes, there can be new engagements between products and employees. Many new innovative ways are followed in several companies to introduce product to employee engagements, but on an experimental basis and not regularly. This paper will introduce a few software engineering practices, which if used as a pattern on a regular basis by software firms, will result in new ways of achieving targets and meeting goals. The patterns described here range across all software engineering processes but mainly focus on development, testing, innovation and feature enhancements that can be used in any kind of project.

Every company has their own work culture and a lot of activities are done to maintain it. That could be team building activity or some outdoor activity. These activities build team bonding and have fun. But, software firms does these kind of activities to engage their employees to the projects and products. Either fun or work is what is adapted in most companies, but a lack of something that involves all these aspects like fun, team bonding, innovation and talent showcase may in turn generate excellent outputs for the project. This will also create healthy competition between employees to do work innovatively and efficiently.

While the product to employee best practices can be brought about in many ways, this paper will portray each practice as a day, where that day can be focused on that activity itself. There are few points which qualify each practice to focus on one day. A day of activity, including team bonding, innovation, healthy talent competition, also product enhancement/development, is affordable by any company. If we portray these activities across a phase or for weeks or months, then the enthusiasm might vanish. Also dedicating a complete day for each activity will induce a sense of festive mood, where employees might prepare for these and hence better productivity. Also, a complete team, product stakeholders and in fact everyone in the company can be involved. Higher management can also be included for introduction and prize distribution for competition winners. Since it is a day, it can be repetitive over a calendar year, thus maintaining a pattern that can be followed by companies. Also it will remove the monotonous work schedule and act as a rejuvenating factor for the employees. And above all, because a complete day is dedicated for each activity, even if a company has several projects at different locations, it is easily possible to accommodate all of them.

Debug day, blitz day, innovation day, feature enhancement day, know your product and critical analysis day are the patterns that will be explored in this paper. Each day will answer the following questions: What problem it tries to address, What is the context in which the problem arises, What is the solution, Which other ideas look like solution but not applicable along with real industry experience and examples wherever applicable.

DEBUG DAY

Consider a scenario where you are involved in a software development team and parallel to you is your colleague who is working in the test team. Both of you are working on the same component of the software product, where you are developing and he/she is testing the product. So, both of you start by understanding the component and then you develop, he tests and raises bugs, and you fix those bugs. And this process continues in a loop until a new requirement comes up or when either of you is assigned a different component. But largely, the process remains the same. For few months or maybe in the initial years, this is interesting, but sometimes when you work on the same thing over and over again, it sometimes become *monotonous*. Same applies for the test engineer, who understands the components, designs test-cases around it, tests it and then report bugs and does this day-in day-out.

If technically thought, quality assurance (QA) engineers are, more often than not, limited only to writing test cases, reporting issues and verification of the fixed issues. Most of them are mostly not aware of deep technicalities like how processes work internally and how data flows. QA is very much dependent on the work of the development teams. Similarly, a developer often overlooks bugs in his own code. At times, a QA engineer thinks of a scenario in a different but valid perspective, which might miss the brains of the developers. How would it be if a day is dedicated for debugging various bugs, where the developers and testers sit together and work collaboratively towards resolving issues?

Debug day is an effort to introduce one full day where developers and testers sit together for debugging critical and non-critical issues. This is an effort where they can team up to resolve issues and track potential root causes of the reported bugs. Since it is the QA engineers who mainly find bugs, they are more comfortable to know the scenario, background, pre-requisites and the environment where the bug was found. This can save a lot of the developers' time. This will also give a different perspective to think and find bugs from the QA front wherein they can come up with some steps or test-cases that will help find hidden bugs which may arise at the later stage of the product.

Debug day is an attempt to make QA engineers smarter and put the development and QA folks on the same pace. Mostly on a debug day several critical bugs, customer issues (bugs reported by customer) are simulated in the product and participants are asked to find the root cause. It is held as a competition where QA engineers team up with developers to debug issues. At the end of the competition, the team who resolves maximum bugs is declared as the winner.

This also brings forward a very healthy learning experience among co-workers. In order to deliver the understanding of the technical details of the product to the QA engineers, potential knowledge sharing sessions can be held, but this often ends up on a very theoretical note. Similarly, developers often miss out the functional tricks or tests that QA engineers so easily find. Implementing the knowledge gained in an actual environment is what actually helps both the employees and in turn, the product. As an added advantage, the developers now, to some extent, know how the testers test the components, the loop-hole that they track. So, now, they can build their component keeping these points in mind. Similarly, the QA engineers now know how the developers write their code, how much level of unit testing and validation and verification checks they put in their code. Having known these points, it will always help them go deeper in search of defects.

The competition is just to empower the QA engineers and the developers with added specialty than their regular job. The scenarios kept for debugging might or might not include real unsolved issues or bugs on production builds which are still open and not yet resolved. It can always be held on defects which are already resolved.
Debug day can be introduced as one day concept, hence can be formed as a repetitive pattern over a particular time of the calendar or specific to product delivery.
Many companies including IBM, EMC2, Fiberlink, etc. already implement concepts like debug day among particular teams. But then, this is not an industry-wide practice that is followed. This can tremendously help to make the product quality better and increase the organizational resources by improving the skill set of the employees.

BLITZ DAY

Have you ever thought, how would you test the complete product, if it is given to you for a day? No processes to follow, no time constraints, no component specific testing, no writing test cases and formalize the procedures. Just plain and simple test the product and you are allowed to tear apart the product if that is a step in one of your tests. If you are a QA and often feel, give me that component and see how I break it, or if you are a developer and feel, although you are developing the product, but would you be satisfied if you were the customer of this product, then its time you should introduce a Bug blitz day in your organization.

Bug blitz is a terminology used to describe the process of hunting bugs. It may include each and every team member, be it developer, QA engineers, architects, L2 and L3 support and any member who might or might not be owning a component but is a part of the product. Apart from the traditional bugs, there are often many hidden bugs and bugs that occur only once or twice that are very difficult to reproduce. Since development, product and test are well aware of the functionality of the overall product, at times, due to their routine checks, many potential hidden bugs are overlooked. This is where a third eye is always very important for finding such bugs in the complete work flow. Doing this in an early stage of the product release gives insight to many such severe potential bugs. No product can be left till the end when the customer finds the bug and reports it. It is never sure how the customer is going to use the product. Since it is not always possible to call or give the customer for testing a product and use their feedback, it is better to have team members act as end users.

During a feature development, other than stakeholder (developers, product engineers and QA) most other team members do not know the feature so deeply. They are the best people who can act as the end users to test the feature. They will be provided with user manuals and product documentation and then, by exploring those documents, they will start testing the product for functionality, load, stress, performance, usability, user interfaces and even the documentation. Another way that is followed in many software companies is the customer feedback for beta versions. Although it is not wrong, but it is a long process and consumes a lot of time and requires regular follow up on the customer front. This can also result in a delay to deliver a feature to all the customers. Also, if customers have to give feedback, it is often the case that each customer uses the product according to its environment and need, which can vary from one customer to another. Including all such features into the plan and satisfying all the customer needs only diverges the core goal of the product. The customer can be involved after a functionality test is rigorously performed and at least one such bug-blitz has already been done. Then it reduces potential bugs to a substantial amount. After this process is done, then the customer can be given the beta product for feedback.

Blitz can also be introduced in the form of an event or competition where teams are formed and each team contains one stakeholder from each group. The team finding the most bugs based on the complexity of the bugs, winners can be decided. Bug-blitz is followed in many software companies with extremely high results. It can go with almost any software life cycle model.

INNOVATION DAY

A common notion in big software firms is THINK. But are you really THINKing as a part of your job? And if you are thinking, how much of it is for achieving a task that you are stuck at and how much of it is actually for innovating something new? To some extent everyone is an innovator, just that how much and what he does with these random thoughts. Most of us keep it in our minds and forget after a few days, but people who really work behind those ideas are the real innovators. Often companies play a big role in this. There is so much mechanical work that people often lose their creative instincts. Consider if your company provides a day for all the employees to THINK and invent something, anything. Would you be excited?

Innovation day can be thought of as a day when employees are allowed to THINK and do one such thing that they always wanted to do. That can be anything, from writing some application to writing a book, generating graphs or any idea with a proof-of-concept (PoC). This gives employees a refreshing break from what they were doing on the daily basis and let them think about new things that can be done. Some of the ideas could be the next big feature of the product. Everyone is talented and has the potential to come up with something new, but only due to lack of platform, these kinds of talents get suppressed. Often situations arise when people get bored of doing the same thing and possibly they feel some cool feature could be added to the product, but due to work pressure, time constraints and lack of follow-ups they are not able to achieve. What an employee is working on might not be the area that s/he is interested in working, there can be some people who are very good at some other stuff. Innovation day will cement a platform for talent exhibition where employees can showcase their talent and get awarded. From organization perspective, people can be used for different task also according to their skill set. Innovation day can be organized within the organization for one complete working day, where people will work on what they think and what they want to do. It will be a complete flow from idea to prototype in one day, as applicable. This can then be showcased in front of a panel and the best idea will win.

Sometimes employees go up to their leads and managers with innovative ideas, followed by creating a proof of concept, but at times the leads and managers might not have that futuristic approach and often such ideas get discarded due to numerous follow-ups. There are people who are very shy and are not bold enough to speak out their idea to a higher management. Innovative day will fill these loop holes. This is very useful from both the organization and employees perspective. Companies can get many innovative ideas, some of that might as well be incorporated in existing products or which can be funded to start a new project. Also through innovative day, they are getting to know the mind-set of the employees that can always be utilized. From the employee perspective, they are feeling content and satisfied that, what they wanted to do, is finally getting to doing it and getting awarded for it.

Companies like Google still call themselves a start-up and many other companies like to maintain a start-up culture where employees are allowed to do anything along with their regular job. These are efforts by which, companies try to assert the innovative and originative mindset among employees. But again, few companies follow these and are not a regular approach. The innovative day can be introduced as a pattern that companies can adopt it in their environment. This is not goal based, hence can be repeated monthly or yearly or half-yearly, as suitable to the companies.

FEATURE ENHANCEMENT DAY

Often, while working on a component or product, we feel some enhancements can be made to it, some feature here, some user interface there, some color combination, some process. Don't we? But since products are architect-ed by senior management and architects, our notion of that change seldom gets a call. That may be due to its unimportance, extra baggage or may be anything. Everyone wants to see their direct contribution to the product that they are working on.

A product is always grows with enhancements, but these enhancements are introduced mostly by senior product managers and product directors, based on market condition, user requirements, etc. We often hear in software industries, that due to lack of schedule, some feature could not be enhanced and is postponed for future. And when the future comes, those features might or might not get picked up for implementation.

Feasible enhancement day, as the name suggests will mainly focus on major enhancements in the product which might include feature enhancement, user interface enhancement, usability enhancements and anything that might come in as an enhancement to the product. This can be done in a group, consisting of product engineer, developer and QA. Sometimes enhancements come in as employees work on the product, but those are not easy to track. This will be a collaborative planned effort towards enhancing the product. In agile practice, where development and testing happens in a very fast pace, there are some feature/bugs/enhancement that are often postponed for future releases. From the pile of such issues, some could be really efficient and might attract end users more. To address such problems, feasibility day is an effort solely targeted towards this goal.

Feasibility enhancement day can be organized by forming teams who will file enhancements to the product. When all the teams come up with their enhancement ideas, the best idea will be awarded. This in turn can be decided by a panel of architects and directors. Once the enhancements are in line, then the teams can sit together and decide which issues to pick and then parallel development and testing will be done for this enhancement. Enhancements can always come from developers, testers or anyone involved in the product, but often these enhancements are not tracked properly and due to time constraints often move into later releases. This will reduce the piled up issues and give feature and component enhancements a serious look. It must always be kept in mind that without substantial enhancements, a product becomes obsolete.

A scenario might easily occur when a fresh graduate is working on a cloud project in his company, and suppose he has expertise in that domain because he has done a cloud research project in his graduation. But most of the times fresher graduates are not entertained with direct inputs to products architecture. Now, if they come with an excellent idea to incorporate in the product based on his past experience, it is most likely to get shadowed. The feasibility enhancement day will bring out these areas to the front.

KNOW YOUR PRODUCT

Every software product or project contains numerous factors, components, parts, features, etc. Do you know each and everything of the product that you are working on? Don't worry, most of us don't know. This is not a fault or problem as such, but if an opportunity is given where everyone can have complete end to end product knowledge, then it will be really useful. In a software development process, it is very difficult for anyone to know all the components, features, functionality and behavior of a product and to whom it caters to, specifically.

Today most of the software firms follow agile development where the release cycle is very short. There is a major time-constraint, due to which, it is very difficult to take a plunge into other components for gaining deep knowledge, apart from one's own component. Also most resources are assigned features that span across many releases. This is mainly done so that software engineers develop a rich understanding of that feature. It is expected that each resource has a complete cognition of at least one component. This is one of the ways by which companies try to create domain experts. No one is expected to know everything. But, this does not negate the point that if employees have accomplished product understanding to some extent, it can be used at any stage and single point of dependency reduces to a large extent.

Know your product is an effort to educate employee about the same, so that they are not bogged down under one feature of the product but can contribute, in any way, to the overall product. Their understanding about the product can be utilized for the betterment of the product. For example, suppose a developer is writing an internal module that is used to communicate between two processes. Now suppose this module is going to be included in a banking application, then the developer should have handled the banking application precision in his code for the internal module. Or suppose there is another person who is developing the storage of all communication channels in the banking application. Then the two internal modules, i.e. the communication system and the storage system to store communication channels, will definitely interact with each other when both are integrated inside the banking application. So it is very important for both the components stakeholders to be aware of the other component's functionality. The same applies to product engineers, QA engineers, support and maintenance teams as well. Know your product increases the awareness about the product, who are working on what components, where each component is going to be implemented and the complete work flow. This helps in writing codes and designing the product efficiently and testing efficiently, that can give more scenarios where things should be taken care of, while developing and testing. Cross-component interaction is an important aspect of developing flawless and feature-rich product.

Know your product does not limit to only knowing the basic essentials of the product or project. It can be anything related to the product that can be leveraged into profitable outcomes. Apart from comprehending the complete product, knowing a product might also include knowing competitors, market trends, target customers, business planning and future plans, and anything that even remotely links to the product.

While Know you product kind of activities are often an ongoing practice, but it can still be formed as an event. It can be conducted as a questionnaire where set of questions will be asked to the people in groups or individually. Prizes could be associated with the winning party, this creates a healthy environment for learning what they are working on. The second thing that can be done is to create a competition where individual/team has to show the best possible use case for the feature. This may lead to some new things that are not discovered or explored yet, may be customers are using and organization does not know about it.

A similar approach is also followed in many companies, where all the stakeholders sit together and discuss various points, possible use cases and targets. This approach is also useful at times, where timeline based data is mostly discussed. When a healthy competition is introduced, it is a whole new ball game. It is useful for increasing awareness about the product and the stand point at any given time during the entire software cycle. This instills confidence among the people to develop and to test.

CRITICAL ANALYSIS DAY

Have you ever critically analyzed the product that you are working on? Critics are often the ones' who discloses loop holes and ambiguities of a product and mostly provide true and honest feedback about the product. Although, this is done at a very early stage of any product life cycle and mostly done by architects and program directors and managers, but critical analysis of a product will enhance the employees' knowledge about the product, its competitors and market trends. Often these product to product comparisons are available as white papers. But that is just an overview. No one knows a product better than who are involved in the product life cycle. So it is better when all the stakeholders spend some hours to do some ground work about the product. After a customer, the members who developed the product often become its main critics. But this is a good thing. Any product should always be critically analyzed before releasing in the market. Many online businesses use product reviews and critical analysis tools that allow customers to rate items which they have purchased and these ratings are then displayed with those products which help other customers decide based on the review. Critical analysis of a product might not always include the complete product. It can be broken down into components and then teams developing and testing those components can do a critical analysis of that component. When all the critical analysis of all the features, components and functionality is finished, it can then be collaborated and integrated into one document which can then be taken up as a work item.

CONCLUSION

There can be many ways by which a software company can introduce efficiency in the work place combined with fun. This paper attempted to bring forward six such points that are in some way or another, being used in many software companies. These kind of activities can improve the productivity of any product and also indulge employees towards their products with an underlying fun quotient. Organizing all these events will only take 6 days and each event can be planned in a span of 2-3 months. These will engage employees and also act as a festival where employees will prepare for such events. It helps to bring the mood and all the related stake holders can be included. Even someone important can be invited, so that employees are more interested. All these adds to the technical activity in a more fun way.

REFERENCES

[1] Frederick Brooks, (1975), *The Mythical Man-Month,* Addison-Wesley, Available: https://archive.org/details/mythicalmanmonth00fred

[2] Robert L Read, (2003), *How to be a Programmer: A Short, Comprehensive, and Personal Summary,* Available: [Online] http://samizdat.mines.edu/howto/HowToBeAProgrammer.html