**BA/DA TRAINING**

**Assignment (July30, 2019)**

**SHAFALI GUPTA**

1. **Explain AS IS TO BE.**

To implement a continual improvement initiative, it is essential to understand what an 'As Is To Be' is and how it is crucial to successful product improvement. Put simply, if you want to improve a product feature, you first have to know its current state (as is) and then you need to model the improved future state (to be). This is crucial to continual improvement because

As Is Process, Maps Will Help to:

* Capture an accurate visual picture of your product feature
* Visualize process breakdowns and areas of risk

To Be Process Maps Will Help to:

* Model the impact of any future changes before you make them.

**Defining the AS-IS process:**

The As Is process defines the present state of a current feature in any given product or software. The goal of analyzing the current state of a product/feature is to find out which components could be improved.

It is important to bear in mind that this As Is analysis will only show you what can be improved, but not necessarily how. Mapping the As Is process will only reveal how your product is working today.

**Defining the TO-BE process:**

Typically, a To Be process defined as the future state of a product or feature. It is the ideal state to define how the product should work, and mapping the To Be processes will structurally clarify how you can get there only then will you be able to see what changes are necessary in order for your product to reach their future To Be state.

**Why AS IS TO BE matters?**

Not every product requires an in-depth analysis of As Is To Be. Here are a few example scenarios to show you when analyzing your As Is processes is particularly necessary:

* It is known that issues with the current state exist. Customers could have reported these issues.
* Your system has customers or business users confused about the correct steps to take in order to use the feature/software.
* You are interested in automating your present functionality.
* You want to move from a paper-based to a mobile process mapping solution

**How to Create an As Is and To Be Process Cycle:**

In order to first create an As Is To Be analysis and then continuously create improvements (As Is To Be improvement cycle) you need to complete the following 10 steps:

* Define your business objectives.
* Involve stakeholders so that you can capture your As Is processes ACCURATELY.
* Define and align process activities of both As Is and To Be processes.
* Analyze gaps and deficits of your As Is process activities.
* Compare your As Is and To Be process maps.
* Determine what should be changed in the As Is process in order to get to your finished To Be map

Make sure to document the envisioned results.

* Estimate and thoroughly analyze business risks within your To Be process model.
* Implement your To Be process so that it now becomes your new As Is process.
* Analyze the effectiveness of your new As Is process (start from the beginning) and create another To Be map from the new process data.

1. **INVEST**

The INVEST criteria for agile software projects is the way to define characteristics of a good quality Product Backlog Item. Below are the characteristics of Invest Criteria.

1. **Independent:** User stories are the lowest level of functional decomposition. They are expressed in the canonical form of ‘as a <user>, I want some <product feature>, so that I get some <business value>. As much as possible, care should be taken to avoid introducing dependencies between stories. Dependencies between stories lead to prioritization and planning problems. For example, suppose the customer has selected as high priority a story that is dependent on a story that is a low priority. Dependencies between stories can also make estimation much harder than it needs to be.
2. **Negotiable:** User stories are not detailed specification. They are not a precise guide for how we are going to build the product, or even what we are going to build. User stories should be high level enough that we can fine-tune both the business implementation and the technical implementation of the requirement. This attribute gives us room to inspect and adapt as we learn about the emerging product.
3. **Valuable:** User stories are written in the language of the business. They represent an increment of product that can be demonstrated to a potential customer of the system. This attribute can be contrasted with a technical task that might have value, but is irrelevant to the business user of the system. The best way to ensure that each story is valuable to the customer or users is to have the customer write the stories.
4. **Estimable:** This attribute implies that the team has enough knowledge about how the user story will be developed (and tested) that they can provide a high-level estimate for how long the user story will take to implement. As the user story is about to be pulled into the sprint, the user story must be well known enough that the team can provide a detailed estimate at the task and hour level.
5. **Small:** Some stories can be too big, or small. Story size does matter because if stories are too large or too small you cannot use them in planning. Epics are difficult to work with because they frequently contain multiple stories.

For example, in a travel reservation system, "A user can plan a vacation" is an epic. Planning a vacation is important functionality for a travel reservation system but there are many tasks involved in doing so. The epic should be split into smaller stories. The ultimate determination of whether a story is appropriately sized is based on the team, its capabilities, and the technologies in use.

1. **Testable:** Testability of a user story is defined up-front before we pull the user story into the sprint. This means that we know the conditions of acceptance before we start working on the requirement. We know what it will take to satisfy our customer’s need and how we will verify that those needs have been met
2. **PARETO**

The Pareto principle, better known as the 80/20 rule, is quite common and can be applied in almost every field of life. The rule maintains that 20% of efforts give 80% of the result, and the remaining 80% give only 20% of the outcome.



In software testing, the Pareto principle also plays a significant role. The 80/20 rule allows the business to appreciate and understand the risks of software implementation and, as a result, to avoid unforeseen expenses and ensure the smooth work of the product during the latter stages of developing.

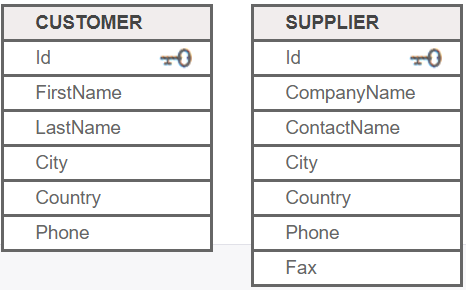
The code quality of the software could profoundly affect further successful product growth. Applying the 80/20 rule in practice, it appears that 80 percent of errors and crashes come from 20 percent of the most frequent bugs. So that, it is essential to place this 20% in a high priority, ensuring their timely elimination

To protect your project from unexpected issues, identify the most frequent bugs and where they come from. Working on the causes that have the highest effect on the product’s functionality, you will focus your team’s efforts in the right direction, thereby saving your energy (as well as time and money) to solve the business objectives of the project.

1. **UNION & UNION ALL**

**Union:** Union combines the result of two queries or two SELECT queries. The column data type in both queries should match. Union gives the combine result from two queries and it does not include duplicates.

For e.g. below is the schema of two tables Customer and supplier





**Union All:** Union all also provides the combine result of 2 select queries but it does include the duplicate values as well.

For e.g.

|  |
| --- |
| SELECT City FROM Customers UNION ALL SELECT City FROM Suppliers ORDER BY City; |