4BA
BUSINESS ANALYSIS

USER STORY

A short summary

Authors:
Paulina Grębska and Radosław Grębski

Translation: Monika Mrozek

Partners:

What is a User Story?

A User Story is an distinct term related to the agile methodology of software development. When we think of Scrum we usually associate it with User Story and the other way around.

What is a User Story? Certainly, **it is a popular and widely applied technique used to document the requirements**. In a short and concise way, it describes what a given type of a user (actor) wants to do and why. In other words, it revolves around the value which is to be delivered.

User Story components

Statement of Value

Sometimes, a User Story is understood narrowly, only as its core part called "Statement of Value". This is a statement describing what value is being delivered when a particular story is implemented.

Usually a „Statement of value" is based on one of the few available templates. Probably the most popular is:

"As a <who?> I need to <what?>, so that <why?>."

This gives us a structure of 3 main elements:

ACTOR <who?> – Indicates a particular user type (designated actor interacting with the system, for example, an Administrator, a Customer, a Supplier, etc.). Usually, various user groups may be identified based on different reasons for using the system, having different levels of permissions, etc.

ACTION <what?> – A description of what a designated actor aims to achieve or do in the system, a description of performed activity.

VALUE <why?> – Specifies the reason behind the implementation, describing the delivered value. Unfortunately, this is the element prone to errors as it is often represented improperly or at times even skipped. This, in my view, poses a serious issue as the business value to be delivered is lost from sight while it should be at the centre of our attention.

Title

Treated as optional. It is usually a short, one sentenced formulation of a User Story's goal.

Conversation

A User Story should be treated merely as an introduction to further discussion with the project team and other stakeholders, a basis for further consideration and modeling. This element should not be treated as a visible, written down fragment of text. It should rather underline that each story requires a wider conversation.

Acceptance criteriaria

The acceptance criteria are there to help clarify a User Story and define its scope. These are the conditions that should be met so it will be possible to verify if the required value has been in fact delivered.

Is User Story useful in every case?

A User Story is a tool to describe in a short and concise manner a need of a particular user. It may be additionally clarified by adding the acceptance criteria. Then, a User Story should be subject to further discussion and consideration aimed at finding appropriate solution. Sounds good, right? So good in fact that this technique if often misused.

I've heard opinions that Scrum does not recognize the documentation, or the only allowed form of it is a User Story. Of course, this view is wrong.

I've experienced the following situation in one of the teams I used to work with as a Business Analyst. When we reached a moment when the system developed by my team was to integrate with a system developed by another team, I asked the other Business Analyst to share the documentation so I could learn their part, its functionalities, available interface, etc. I received a list of 153 User Stories related to the development of their part... Of course, the User Stories were developed over the period of several years. Features created later modified the previous implementations which were also described in User Stories.

As one can imagine, creating a consistent vision of the system based on the available documentation seemed like an extremely uneasy task. This is a clear example that a User Story, while usually very useful, should not be used for every purpose.

When a User Story should BE USED?

Whenever you need to store requirements on a short term basis and where there is a need to emphasize the value being delivered. Other than that, User Stories are used as an introduction to further discussion in order to find a common understanding and continue to work on creating a solution model.

Additionally, due to their specific nature allowing to divide the work into smaller, contained functionalities or quality requirements, they may be also used to:

- → Discussing priorities
- → Work estimation
- → Planning the product development process and its versioning
- → High level requirements mapping
- → Designing acceptance testing
- → Tracking and reporting the work progress

When a User Story should NOT BE USED?

First of all, a User Story should not be used as a long term storage of requirements. They prove to be a poor base if you want to create a documentation used to develop and maintain a product. Mostly because basing only on User Stories without any additional documentation, it is very difficult to recreate current state of the system.

It is important to remember, that a User Story alone, without any accompanying context is also usually not enough for the development team to work effectively. They require broader scope to increase their level of understanding of the implemented change, the goal and impact on the individual system components. **All these could be provided by employing other techniques and different forms of documentation.**

Traits of a good User Story

I often come across User Stories written on the go, without a thorough analysis, only aimed at meeting requirements coming from the template adopted by the organization. Obviously, this often leads to serious negative consequences. If the only goal of the author is to fit the User Story into a template ('cause they told me to' or 'cause this is our standard'), it's not worth the effort. It is probably better to describe the requirements in another form, for example, using a natural language.

How to write a User Story so it is both effective and useful? Pay attention to its quality! And in order to help you focus on the quality, use the INVEST technique.

I for Independent

And so the User Stories should be independent from each other. This translates into the rule that it is possible to understand and implement them autonomously. They should also add value when analyzed as a **standalone element.**

Of course, this does not mean that User Stories should not be related in any way. Just as the mutual relations may appear between the requirements (described in any form), so it may happen that the User Stories are linked. These relations may, for example, indicate that one User Story must be implemented before another, or when you implement one of them first, the implementation of the other turns out to be much easier. It's all about making your User Story a specific, self-contained logical unit.

You may compare it to building a house: laying foundations and erecting walls. Of course, there is a certain relation between these tasks and one activity must be completed prior to the other one. However, they are examples of the standalone, logically distinguishable stages of construction process.

An good example of a wrong use of the User Story technique would be presenting the radiator fitting and the central heating pipe works as two separate stories. These elements do not add value on their own.

Examples of an incorrect Statement of Value

As a client, I need to have a database prepared so that in future, I will be able to store all my orders.

As a client, I need to have a fully tested functionality where products can be ordered, so that I can avoid issues with the ordering process.

Common mistakes:

- Creating separate User Stories concerning the technical elements of the solution (for example, having the database schema and elements of the logic described separately)
- Creating separate User Stories concerning different stages of the software development process (e.g. analysis, code implementation and testing stages)

N for **Negotiable**

A User Story should be negotiable. In other words, it should leave enough room for a discussion and negotiation of the possible ways of delivering a solution. It is incorrect to create User Stories that are detail-heavy and too technical as you already impose a specific solution on your development team. This kills creativity and puts you at risk of missing a potentially more viable solution.

Examples of an incorrect Statement of Value

As a client, I want to have separate fields for the street name, building number, flat number, town, postal code and to save the address by clicking the Save address button, so that all my data are stored in the database.

As a client I want to receive an email with a PDF file listing the article's catalog number, its price and the name, so that I can receive a confirmation of the transaction.

Common mistakes:

- → Defining too many details of the solution in a User Story
- → Leaving no space for the development team's creativity

V for **Valuable**

A well written User Story expresses the value which is to be delivered through its implementation. In fact, the very structure of a User Story compels the analyst to indicate 'Who?' 'What?' and 'Why?' The 'Why?' here is extremely important as it represents the clue of the underlying business need and the value which is being delivered. If, however, the implementation of a given User Story does not bring any value, why bother? It only leads to increased implementation cost without any explicit gain and therefore represents a waste.

Examples of an incorrect Statement of Value

As a client, I want to be able to save the standard delivery address.
As a staff member, I want to be able to view the orders.

Common mistakes:

→ The value delivered to the stakeholders is missing

E for *Estimable*

A properly specified User Story must be subject to reasonable estimation. This means that it must be possible to determine the effort required for its implementation. It is obviously not about a precise estimation in working days, or worse even, in hours. It's about a relative estimation, which is based on relative units (such as Story Points) and is based on the previous experience of the development team. The estimation process is much easier when there are clear and unambiguous acceptance criteria specified.

Examples of an incorrect Statement of Value

Common mistakes:

→ The business goal and the required scope of the User Story are not clearly communicated

→ The User Story was never discussed with the development team

S for *Sized Appropriately*

The 'smaller' the User Story, the better it is. This may be a acceptable general rule – of course within reasonable limits. Such reasonable size limits (being a consequence of relative estimation process) may be specified as the ability to deliver it within one iteration (Sprint). If, according to the estimation, it is clear that a given User Story cannot be delivered within one iteration, it should be split into smaller pieces.

Examples of an incorrect Statement of Value

Common mistakes:

- → Making a User Story too big to be implemented within one iteration

T for *Testable*

Just like any other requirement and its definition, also a requirement specified within a User Story must be testable. Simply put, this means that a way must exist to objectively verify if a given User Story have been successfully implemented and delivered the required value to a particular stakeholder. In other words, one must be able to design and conduct a test aimed at verifying the original requirements in an unambiguous way.

Examples of an incorrect Statement of Value

As a user, I want to generate the report swiftly, so that the entire process takes the minimum amount of time.
As a client, I want all bugs removed from the system.

Common mistakes:

- → Unclear and ambiguous (thus impossible to verify) requirements
- → No or vague acceptance criteria

How to split when User Story is too big?

One of the features of the correctly written User Story is when it is Sized Appropriately. Its scope should be limited so it fits into one iteration. If this is unrealistic, it should be treated as an Epic and subsequently broken down into smaller chunks.

The main objective when making the division is to have the newly created User Stories deliver value to the stakeholders independently from each other. It's also good to adhere to other features forming the INVEST rule when making the division plan. It is advisable to cut User Stories vertically, meaning to follow the functionality lines so they may be presented independently to the users.

The so called horizontal split, where the User Story follows the consecutive implementation steps (for example, preparing data structure, business logic implementation and user interface design) should best be avoided. Even worse idea is splitting the User Story following the lines of the software development process, so for example, having a separate User Story for creating a design, other for implementation and another one for testing.

Below are several examples on how to make the cut when your User Story turns out to be too big (impossible to implement in one iteration). Of course, there are many different ways to correctly split a given User Story, the only limitation being the level of creativity of the Business Analyst.

Following the process steps

If the process consists of many steps, each of them may be described by a separate User Story. The simple example may be processing an order in an online shop. It may include elements like adding items to the cart, selecting payment method, selecting shipping method, reviewing the order summary and finally placing the order. The entire process may be described as one general User Story.

Example

As a client, I want to be able to place an order in an online store so that I will receive selected items.

- It is possible to browse the products available in the online store
- It is possible to indicate the products to be ordered and their quantity
- It is possible to configure the shipping method
- It is possible to make a secure payment
- It is possible to review the order summary before it is placed

This User Story is very general and it involves a wide range of functionalities. It can be, however, successfully split into smaller stages following the process steps by defining more detailed criteria when required.

After splitting

As a customer, I want to browse products available in the online store, so that I will find out about the offer:

- It is possible to browse the products available in the online store
- The price of each product is available

As a customer, I want to add products to the cart, so that I will place the order:

- It is possible to indicate products to be ordered and their quantity
- It is possible to indicate the quantity of each product
- The information on my current order value is available as I shop

As a customer, I want to select the payment method, so that I can pay using the one I prefer:

- It is possible to select a payment method from several methods available
- At least the traditional bank transfer and an online transfer are available
- The information on the selected payment method is available

As a customer, I want to configure shipping, so that I can receive my order in the preferred way:

- It is possible to add the shipping address
- It is possible to select a preferred shipment method from several available
- An information on the cost of each shipment method is available

As a customer, I want to review the order details before it is placed, to that I can verify if it is correct

- It is possible to review the order details (the list of selected products along with their quantity, shipping information)

As a customer, I want to confirm and pay for my order, so that I can receive my items.

- It is possible to receive an order confirmation
- It is possible to pay with one of the preferred payment methods

Following the acceptance criteria

User Story may contain multiple acceptance criteria. It is then reasonable to split it according to the particular subsets of these criteria.

Continuing in an online store scenery, let's imagine a User Story related to placing an order and including multiple acceptance criteria. These criteria may be, for example, ordering multiple items, ability to edit items in the cart, option to cancel the order ('empty cart' feature), displaying cart's current value, etc. Then it is possible to design the User Stories based on excluding particular acceptance criteria.

Example

As a client, I want to place an order, so that I can order specific products.

- It is possible to order multiple products within one order
- It is possible to display the current list of products in the cart
- It is possible to check what is the current cart value
- If a given product is on a promotion, an information is available about the promotion value
- It is possible to edit the products in cart (adding or removing specific products)
- It is possible to cancel the order (emptying the cart)

After splitting

As a customer, I want to manage the order items, so that I am able to indicate products I want to order.

- It is possible to order multiple products within one order
- It is possible to display the current list of products in the cart
- It is possible to edit the products in the cart (adding or removing the specific products)
- It is possible to cancel the order (emptying the cart)

As a customer, I want to have access to the current order value and the promotions, so that I can control and optimize my total order value.

- It is possible to check what is the current cart value
- If a given product is on promotion, an information is available about the promotion value

Following the scope of data

Making a cut following the data scope makes sense when working on User Stories involving processing complex structures or vast amount of data. Let's consider displaying a user profile in an online store. It may include user's name, basic personal or contact details (name and surname, email address, phone number), multiple shipping addresses, invoice details, preferred payment and shipping methods, etc.

If the User Story related to displaying all available data in the user interface turns out to be too big for one iteration, try to group these data and create a set of independent User Stories, each related to a particular data group.

Example

As a client, I want to be able to view my user profile, so that I can check if my personal details are correct.

- → It is possible to display the user name
- → It is possible display the user’s contact details
- → It is possible to display the address list of a given user
- → It is possible to display the user’s payment settings
- → It is possible to display the user’s preferred shipping method
- → It is possible to display the user’s preferred way of delivery

After splitting

As a customer, I want to view the contact details in the user account, so that I can check if they are correct.

- → It is possible to display the user name
- → It is possible to display the user’s contact details

As a customer, I want to view the address details and payment settings in the user’s profile, so that I can check if they are correct and up to date.

- → It is possible to display the address list of a given user
- → It is possible to display the user’s payment settings

As a customer, I want to view the preferences I have previously set, so that I can check if they are correct.

- → It is possible to display the user’s preferred shipping method
- → It is possible to display the user’s preferred way of delivery

Following operation types

Typical operation set that may be performed on a given data scope is the so called ‘CRUD’ operations set (Create, Read, Update, Delete). If the operations required in your User Story may be grouped according to the ‘CRUD’ set, then the particular type of operations may be separated into independent User Stories. If, for example, an online store user is required to provide a shipping address (or multiple shipping addresses), it is possible to contain it a separate User Story related to the address handling: displaying, editing, deleting, etc.

Example

As a client, I want to be able to manage my delivery addresses, so that I can order products to my home address, office address or my family or friends addresses.

- → It is possible to link multiple delivery addresses to one user account
- → It is possible to display all saved addresses
- → It is possible to add a new delivery addresses
- → It is possible to remove previously saved address from the list
- → It is possible to modify saved addresses

After splitting

As a customer, I want to be able to add multiple delivery addresses, so that I can order products to my home address, office address or my family or friends addresses.

- → It is possible to link multiple delivery addresses to one user account
- → It is possible to display all saved addresses
- → It is possible to add a new delivery addresses

As a client, I want to be able to manage my delivery addresses, so that I can update them if anything changes.

- → It is possible to remove previously saved address from the list
- → It is possible to modify previously saved addresses

Following non-functional requirements

Non-functional requirements do not seem to be a particularly fortunate parameter to become a basis for User Story split. There are many examples to support this argument, however, there are also many cases where it can be done successfully. To illustrate it, let's analyze the performance requirements. Assuming that a certain report should be generated in less than 5 seconds (under the given circumstances and when processing a given amount of data). In this case, one User Story may be dedicated to the report generating process and the other one should aim at process optimization.

Another example may be the availability of user interface in multiple languages or possibility to use the application on different operating systems (e.g. Android, iOS). Each interface language or support for different operating system may be placed in separate User Story.

Note: Placing the non-functional requirements into a separate User Story sometimes may prove to be risky. When the non-functional requirements are treated independently, the likelihood they will never be implemented increases.

Example

As a store manager, I want to be able to generate a report on orders placed in a given time period, so that I can optimize my stock levels and manage the storage.

→ It is possible to generate an order report which includes the following data: the catalog number of the product, its name, number of items sold and the measurement units.
→ It is possible to define the report's date range
→ The generation of the report takes no longer than 5 seconds

After splitting

As a store manager, I want to be able to generate a report on orders placed in a given time period, so that I can optimize my stock levels and manage the storage.

→ It is possible to generate an order report which includes the following data: the catalog number of the product, its name, number of items sold and the measurement units
→ It is possible to define the report's date range

As a store manager, I want to minimize the time of the report generation process, so that it does not impact the effectiveness of my work.

→ The generation of the report takes no longer than 5 seconds

Naturally, there are many other good ideas on how to divide a User Story when it is too big. The ones described in this section represent only a few of them and do not exhaust available options.

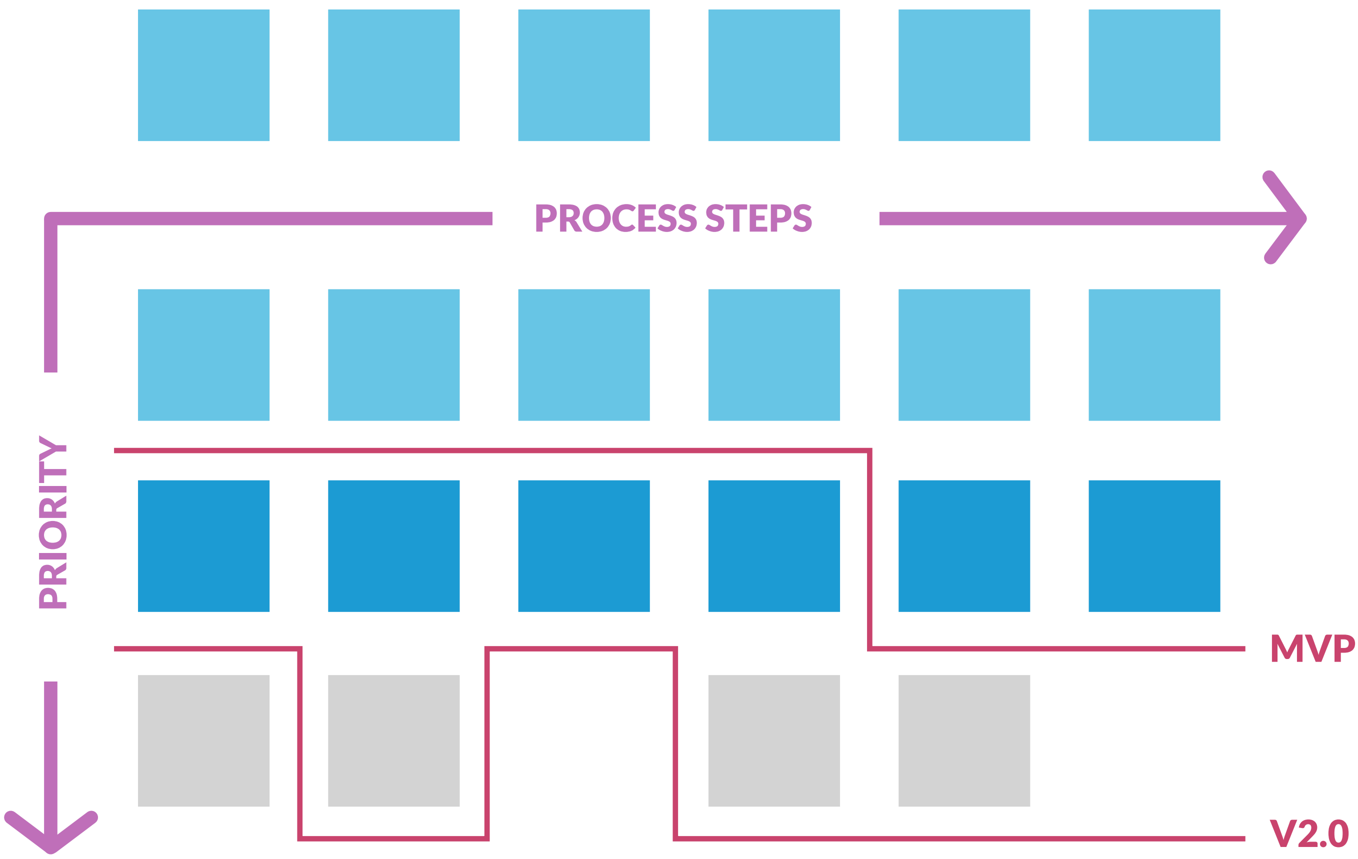
Story Mapping

One of the key downsides of a User story is that is does not provide a wider context. It is rather focused on one selected requirement. A typical User Story describes the needs of the particular stakeholders along with the value assigned to these needs. Unfortunately, this is not enough for the development teams to work efficiently. And most certainly, it is not enough to understand the general goal of the project and the need that it stems from. **For this reason, when representing the requirements in form of a User Story, it's worth to support your efforts also with other techniques and forms of documentation.** One of such techniques is a Story Mapping.

What is Story Mapping?

In a nut shell, it is arranging the User Stories (or alternatively, the requirements captured in other form) into the process steps. We therefore need to outline a process, the particular steps it consist of and assign different User Stories to each step. It is also possible to allocate priority to each Story. This way it is easy to determine the scope for different versions of the product.

Story Mapping Scheme

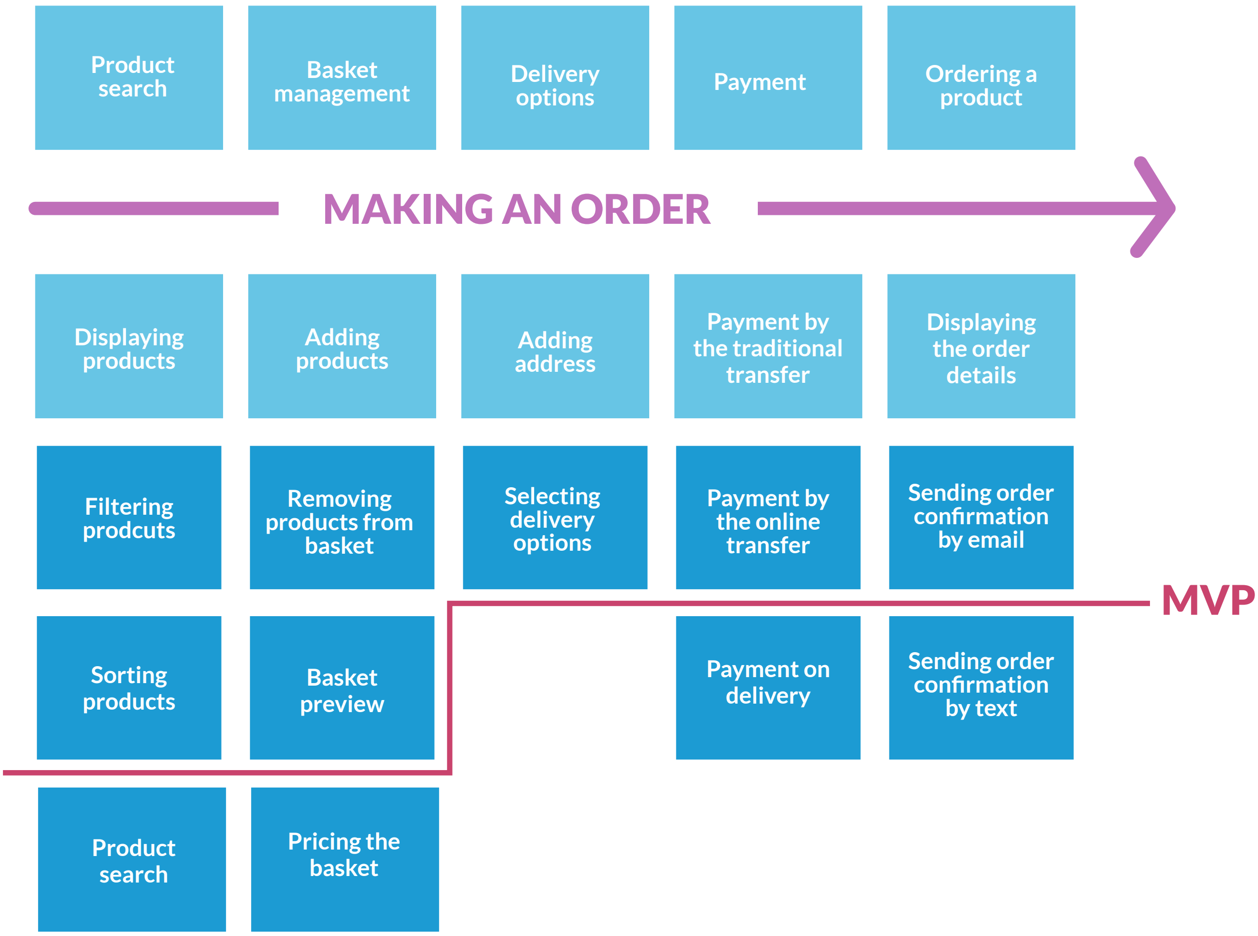


There are two axes in the above scheme: horizontal and vertical. The horizontal axis indicates the course of the process. Above the axis are the individual process steps which create an integral and consistent picture of the process. The vertical axis indicates the priority. The lower a given User Story is placed, the lower its priority. The red lines frame the scope of the particular product versions. The MVP(Minimum Viable Product), is the minimum scope of the product, delivering value to the users and justified from the business perspective. The second red line, indicates the scope to be released in the next product version.

Story Mapping – specific example

Let's consider using the Story Mapping technique on the real life example – ordering products in an online store. The example is simple to allow us to focus on presenting the techniques rather than untangling the complex process.

Tip: When creating your Story Map, use the titles of your User Stories rather than the full Statement of Value.



This example process consists of several consecutive steps. They are visible above the horizontal axis outlining the direction of the process. Several User Stories were allocated to each step in the process. They were ordered according to their priority. The more important ones were placed higher up, while the less important are on the lower side of the scheme. We can also see the line indicating the MVP which represents the functionality (requirements) scope which will make it to the first, basic product version. Naturally, we could go on and plan the subsequent versions including the remaining requirements.

Story Mapping as a workshop

The main advantage of the Story Mapping is that it allows to communicate and discuss the wider context. We do not focus on a perspective of a particular User Story but rather see the entire process or event the whole solution which we are building. Thanks to this top-down approach, we can map an individual User Story within the bigger picture. **This approach also feels the most relevant. Only when we gain a wider perspective and truly understand our goal and business need, can the development teams work out the optimal solution.**

The Story Map may be also used as a form of documentation: it consists of the process steps, related User Stories and their assigned priorities. In my opinion, however, this is not the optimal usage of the entire potential of Story Mapping. Its greatest advantage is that it can be used to reach common understanding between the stakeholders. The understanding may involve different stages: the understanding of the business need, the solution that responds to it as well as its scope.

When you organize a Story Mapping workshop and invite the right stakeholders (both 'business' and 'technical') you get a unique opportunity to exchange information, have substantive discussions and a chance of reaching the common understanding. All these elements are extremely important when it comes to estimating the work the solution will require and reaching an effective cooperation during the implementation stage. Story Mapping workshops are a great opportunity to have a discussion on your priorities and splitting the product into versions.

Weak points of Story Mapping

We have already discussed the advantages of Story Mapping. Of course, just as any other method, the Story Mapping also has certain downsides and limitations. The first limitation is that it may be successfully applied mostly where processes are involved.

After all, it's clue is to map the process into the individual steps. In case of the systems, where defining a clear processes is not possible, we can still use the Story Mapping technique to group the requirements according to certain characteristics or functionalities. This, however, is less convenient and the technique loses its biggest value. When there is no process, the horizontal axis on the map becomes redundant.

Another limitation of the Story Mapping is also related to processes. In case of the complex and multi-branched processes, specifying the Story Map can prove to be very difficult. When handling such projects, it may be worth applying the simplified version of Story Mapping and describe the detailed sections in another form (e.g. appropriate diagram).

Another limitation that should be mentioned is the lack of information about the mutual relations between the requirements. For business or technical reasons, some requirements must be implemented before others. Sometimes, the implementation of one requirement may also substantially impact the implementation cost of another. These dependencies are not recognized in the Story Maps and must be analyzed in some other form.

Summary

A User Story is a useful and favored technique used to document the requirements. It works out great when stipulating the stakeholders requirements. **However, as it has so many advantages, it is often misused.** In many cases it would be more beneficial to use other techniques too. Either as a supporting solution or instead of the User Story. Depending on the situation, these other techniques may be: various types of models (diagrams), interface mockups, Use Case descriptions or information captured in the natural language.

When planning to use the User Story technique, it is always worth to query whether this is the right technique for the given circumstances. If so, then ensure the User Stories you create for your project are of the best quality.

References

Agile Extension to the BABOK® Guide. Version 2. 2017.
A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide). Version 3. 2015