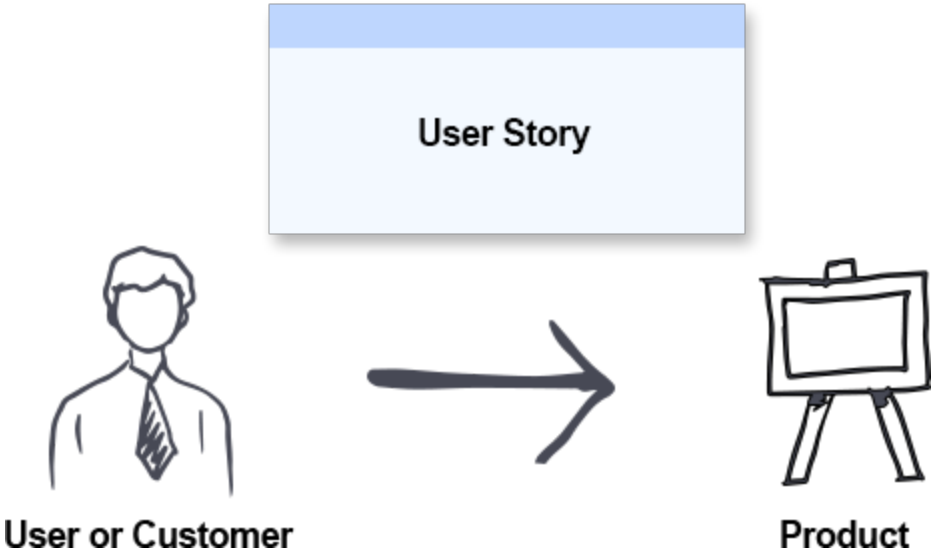
What is User Story?

visual-paradigm.com/guide/agile-software-development/what-is-user-story

In software development and product management, a user story is an informal, natural language description of one or more features of a software system. A user story is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. A user story describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement.

User stories are often recorded on index cards, on Post-it notes, or in project management software. Depending on the project, user stories may be written by various stakeholders such as clients, users, managers or development team members.



*"**User stories** are part of an agile approach that helps shift the focus from writing about requirements to talking about them. All agile **user stories** include a written sentence or two and, more importantly, a series of conversations about the desired functionality"* - Mike Cohn, a main contributor to the invention of Scrum software development methodology

Agile Software for Scrum Teams

Need an agile software solution for product backlog management? Visual Paradigm supports a powerful agile toolset that covers user story mapping, affinity estimation, sprint management, etc. It's powerful but yet easy-to-use, intuitive and, most important, AGILE.

Free Download

Why User Stories?

Requirements always change as teams and customers learn more about the system as the project progresses. It's not exactly realistic to expect project teams to work off a static requirements list and then deliver functional software months later.

With user story approach, we replace big upfront design with a "just enough" approach. User stories reduce the time spent on writing exhaustive documentation by emphasizing customer-centric conversations. Consequently, user stories allow teams to deliver quality software more quickly, which is what customers prefer. There are quite a few benefits for adopting user story approach in agile development such as:

- The simple and consistent format saves time when capturing and prioritizing requirements while remaining versatile enough to be used on large and small features alike.
- Keep yourself expressing business value by delivering a product that the client really needs
- Avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution.
- Avoid the appearance of false completeness and clarity
- Get to small enough chunks that invite negotiation and movement in the backlog
- Leave the technical functions to the architect, developers, testers, and so on

Basic Concepts of User Story

A user story is a lightweight method for quickly capturing the "who", "what" and "why" of a product requirement. In simple terms, user stories are stated ideas of requirements that express what users need. User stories are brief, with each element often containing fewer than 10 or 15 words each. User stories are "to-do" lists that help you determine the steps along the project's path. They help ensure that your process, as well as the resulting product, will meet your requirements.

A user story is defined incrementally, in three stages:

- The brief description of the need
- The conversations that happen during backlog grooming and iteration planning to solidify the details
- The tests that confirm the story's satisfactory completion

And these, although, are known as the 3C's - Card, Conversation and Confirmation. We will talk more about this later on in this user story guide.

User Stories - INVEST

The acronym INVEST helps to remember a widely accepted set of criteria, or checklist, to assess the quality of a user story. If the story fails to meet one of these criteria, the team may want to reword it, or even consider a rewrite (which often translates into physically tearing up the old story card and writing a new one).

A good user story should be - **INVEST**:

- **Independent**: Should be self-contained in a way that allows to be released without depending on one another.
- **Negotiable**: Only capture the essence of user's need, leaving room for conversation. User story should not be written like contract.
- **Valuable**: Delivers value to end user.
- **Estimable**: User stories have to able to be estimated so it can be properly prioritized and fit into sprints.
- **Small**: A user story is a small chunk of work that allows it to be completed in about 3 to 4 days.
- **Testable**: A user story has to be confirmed via pre-written acceptance criteria.

Origins of User Stories

- First mention in Kent Beck's book "Extreme Programming Explained". It was unstructured text that was quite similar to use cases with restriction in size.
- Ron Jeffries introduced the concepts of 3C's: card, conversation, confirmation in 2001
- 2003: the INVEST checklist for quickly evaluating user stories originates in an article written by Bill Wake, which also repurposed the acronym SMART (Specific, Measurable, Achievable, Relevant, Time-boxed) for tasks resulting from the technical decomposition of user stories.
- 2004: the INVEST acronym is among the techniques recommended in Mike Cohn's "User Stories applied".

How to Write User Stories?

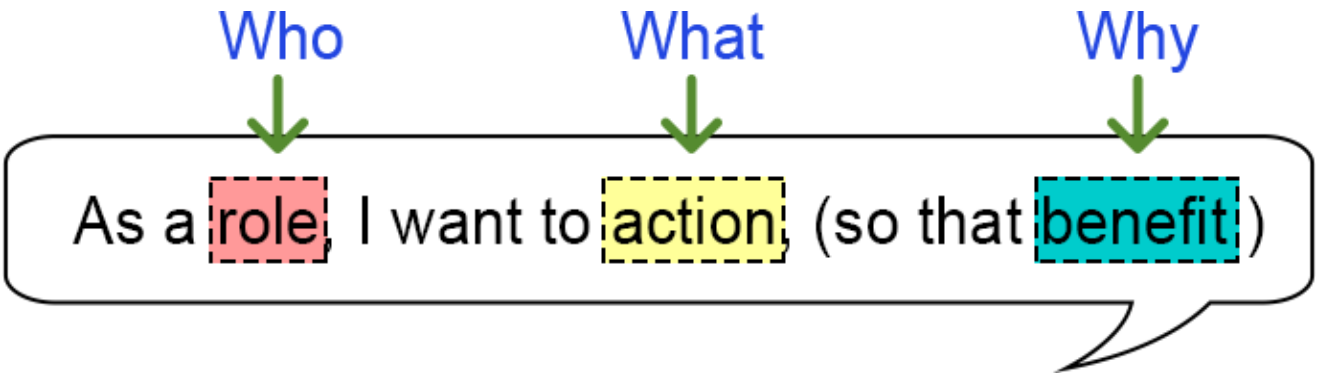
When getting started with writing user stories, a template can help ensure that you don't inadvertently start writing technical tasks:

User Story Template

User stories only capture the essential elements of a requirement:

- Who it is for?
- What it expects from the system?
- Why it is important (optional?)?

Here is a simple format of user story used by 70% of practitioners:



Role - The user should be an actual human who interacts with the system.

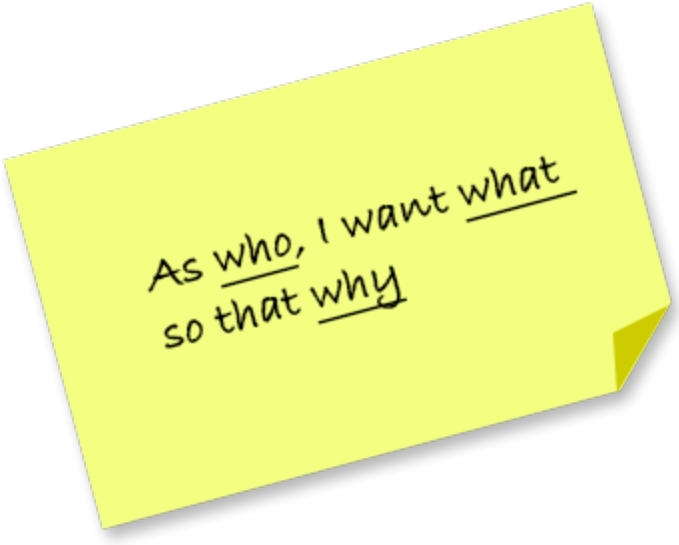
- Be as specific as possible
- The development team is NOT a user

Action - The behavior of the system should be written as an action.

- Usually unique for each User Story
- The "system" is implied and does not get written in the story
- Active voice, not passive voice ("I can be notified")

Benefits - The benefit should be a real-world result that is non-functional or external to the system.

- Many stories may share the same benefit statement.
- The benefit may be for other users or customers, not just for the user in the story.



Notes:

User stories are written in everyday language and describe a specific goal (what) from the perspective of an individual (who) along with the reason (why) he/she wants it.

In software development, the goal is often a new product feature, the individual is some type of end-user and the reason is the benefit that the user sees in the targeted product feature.

User Story Examples:

- As a [**customer**], I want [**shopping cart feature**] so that [**I can easily purchase items online**].
- As an [**manager**], I want to [**generate a report**] so that [**I can understand which departments need more resources**].
- As a [**customer**], I want to [**receive an SMS when the item is arrived**] so that [**I can go pick it up right away**]

In the examples above:

- **Role** represents the person, system, subsystem or any entity else who will interact with the system to be implemented to achieve a goal. He or she will gain values by interacting with the system.
- **Action** represents a user's expectation that can be accomplished through interacting with the system.
- **Benefits** represents the value behind the interaction with the system.

It is not a rule but a guideline that helps you think about a user story by considering the followings:

- The user story will bring value to someone or certain party (e.g. customers).
- The user story is fulfilling a user's need (e.g. receive an SMS when the item is arrived)
- There is a reason to support implementing this user story (e.g. customer can go pick up the item she purchased)

Detailing User Stories with 3Cs (Card, Conversation and Confirmation)

Ron Jeffries, another of the creators of XP, described what has become our favorite way to think about user stories. A User Story has three primary components, each of which begin with the letter 'C': Card, Conversation, and Confirmation to describe the three elements of a user story. Where:

Card

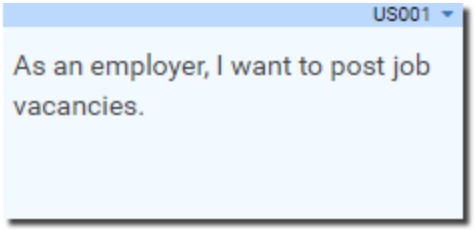
Card represents 2-3 sentences used to describe the intent of the story that can be considered as an invitation to conversation. The card serves as a memorable token, which summarizes intent and represents a more detailed requirement, whose details remain to be determined.

You don't have to have all of the Product Backlog Items written out perfectly "up front", before you bring them to the team. It acknowledges that the customer and the team will be discovering the underlying business/system needed as they are working on it. This discovery occurs through conversation and collaboration around user stories. The Card is usually follows the format similar to the one below:

As a (**role**) of the product, I can (do **action**) so that I can obtain (some **benefits** / value)

Note:

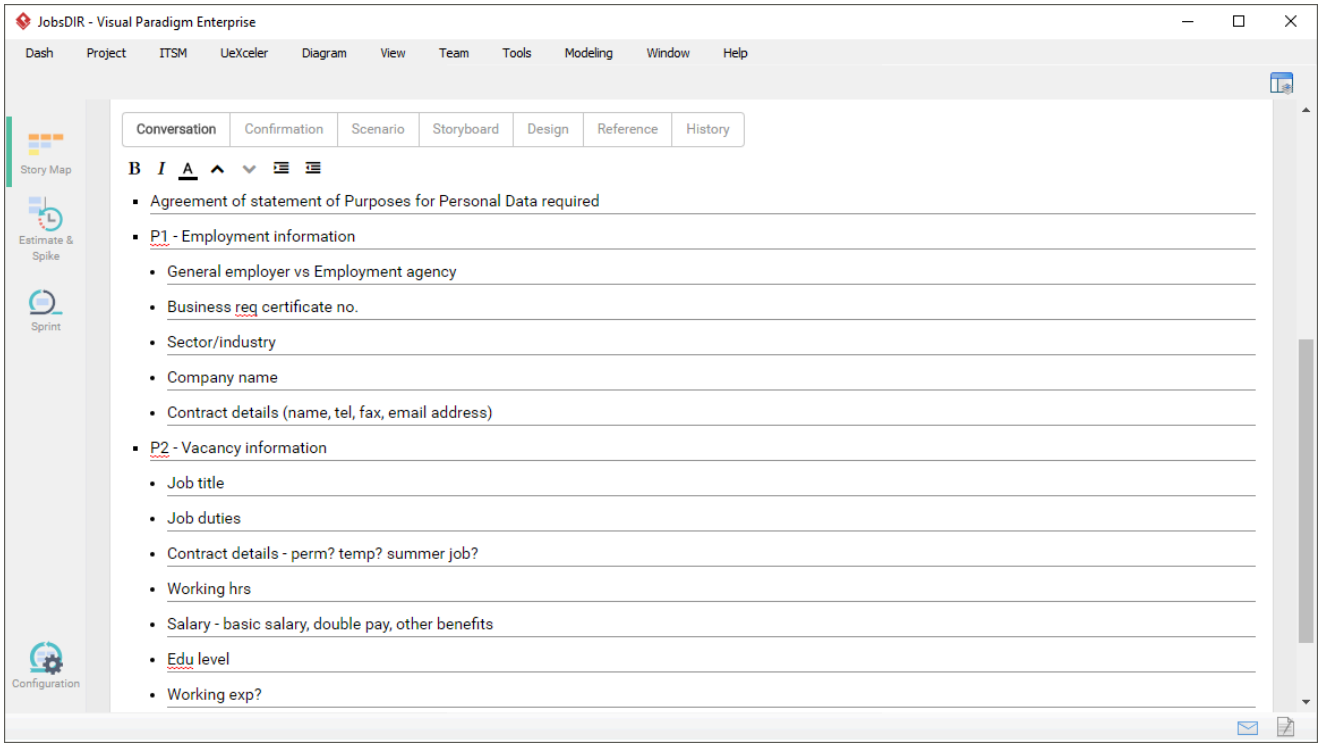
> The written text, the invitation to a conversation, must address the "**who (role)**", "**what (action)**" and "**why (benefits)**" of the story.



Conversation

Conversation represents a discussion between the target users, team, product owner, and other stakeholders, which is necessary to determine the more detailed behavior required to implement the intent. In other words, the card also represents a "promise for a conversation" about the intent.

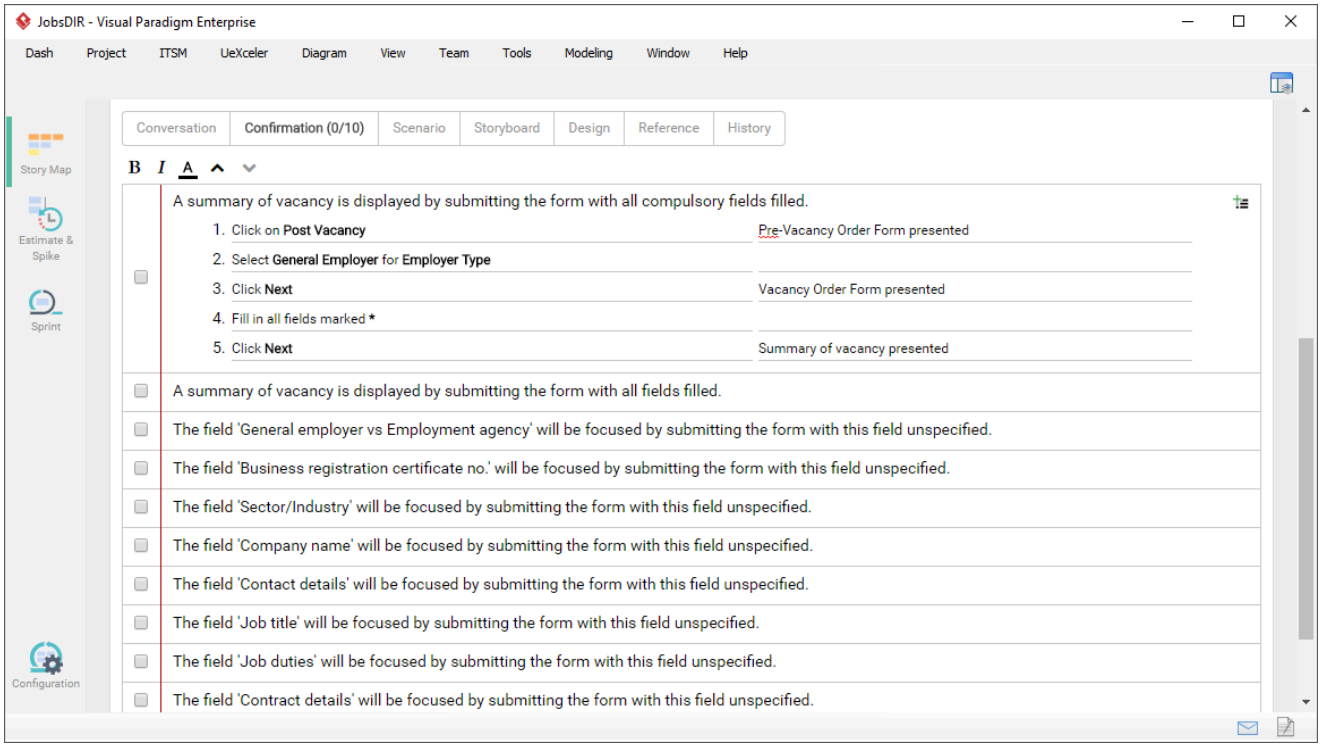
- The collaborative conversation facilitated by the Product Owner which involves all stakeholders and the team.
- The conversation is where the real value of the story lies and the written Card should be adjusted to reflect the current shared understanding of this conversation.
- This conversation is mostly verbal but most often supported by documentation and ideally automated tests of various sorts (e.g. Acceptance Tests).



Confirmation

Confirmation represents the Acceptance Test, which is how the customer or product owner will confirm that the story has been implemented to their satisfaction. In other words, Confirmation represents the conditions of satisfaction that will be applied to determine whether or not the story fulfills the intent as well as the more detailed requirements.

- The Product Owner must confirm that the story is complete before it can be considered "done"
- The team and the Product Owner check the "doneness" of each story in light of the Team's current definition of "done"
- Specific acceptance criteria that is different from the current definition of "done" can be established for individual stories, but the current criteria must be well understood and agreed to by the Team. All associated acceptance tests should be in a passing state.



How to Identify User Story?

User stories should be identified together with the stakeholders, preferably through a face-to-face meeting. User story is a requirement discovery process instead of an upfront requirement analysis process.

In the traditional requirements capturing approaches, system analyst tries to understand customers' needs and then prepare a requirement specification for the system in detail. This is not how the user story approach works. Instead of a documentation process, the identification of user story is more like a note taking process. We list the major steps for identifying user stories as following:

1. Through the discussions with users, we listen to and understand their problems and needs
2. And then, write down their needs as user stories at the same time.
3. These user stories will become the source of requirements.
4. The details could be subsequently filled just-in-time, providing the team with a "just-enough" requirement references throughout the project development process.

Lifecycle of a User Story

In a broad sense, there are six main states for each user story throughout a software project:

Pending

Through the communication between user and project team, user stories are found. At this state, the user stories have nothing more than a short description of user's need. There is no detailed discussion of requirements, no system logic and no screen design yet. In fact, the only purpose of user story, for now, is just for reminding all parties for a future discussion of user's request written in this user story (card). It is possible that the user story will be discarded in the future.

To-do

Through a discussion between different stakeholders, the user stories to be addressed in the next few weeks are decided, and are put into a time-box called a sprint. Such user stories are said to be in the to-do state. No detailed discussion has yet been carried out in this state.

Discussing

When a user story is in the Discussing state, the end user will communicate to the development team in confirming the requirements as well as to define the acceptance criteria. Development team will write down the requirements or any decisions as conversation notes. UX specialist may create wireframes or storyboards to let user preview the proposed features in visual mock-ups, and to feel it. This process is known as user experience design (UX design).

Developing

After the requirements are clarified, the development team will design and implement the features to fulfill user's requests.

Confirming

Upon the development team has implemented a user story, the user story will be confirmed by the end user. He/she will be given access to the testing environment or a semi-complete software product (sometimes known as an alpha version) for confirming the feature. Confirmation will be performed based on the confirmation items written when detailing the user story. Until the confirmation is done, the user story is said to be in the Confirming state.

Finished

Finally, the feature is confirmed to be done, the user story is considered in the Finished state. Typically, this is the end of the user story. If user has a new requirement, either it is about a new feature, or it is an enhancement of the finished user story, the team would create a new user story for the next iteration.

Organizing User Stories with Story Map

User stories is a useful way to build a better product backlog, one that is user-centric and describes software requirements in a practical, actionable way. But user stories on their own do not reveal the whole picture that can clue you in on the larger journey the user goes through from the moment that load an app until they reach their final goal.

A user story map can help us to arrange user stories into a manageable model for plan, understand and organize the functionality of the system systematically. By manipulating the structure of the map, we can identify holes and omissions in your backlog and interrelating the user stories in a meaning structure; helping plan holistic releases effectively that deliver value to users and business with each release. User story map allows you to add a second dimension to your backlog. Here are a few reasons you should consider using this technique:

- It allows you to see the big picture in your backlog.
- It gives you a better tool for making decisions about grooming and prioritizing your backlog.
- It promotes silent brainstorming and a collaborative approach to generating your user stories.
- It encourages an iterative development approach where your early deliveries validate your architecture and solution.
- It is a great visual alternative to traditional project plans.
- It is a useful model for discussing and managing scope.
- Allows you to visualize dimensional planning and real options for your project/product.

User Story Map Template

Story mapping is a top-down approach of requirement gathering and is represented as a tree. Story mapping starts from user activities. A user activity should achieved a particular goals. And to complete an activity, users needs to perform the associated tasks. And these tasks can be transformed into epics and user stories for software development. Typically, user story map consists of 3 levels: User Activities / User Tasks / User Stories. For enterprise scale projects, perhaps a 4 levels structure may be more appropriate by introduce Epics in the third level.

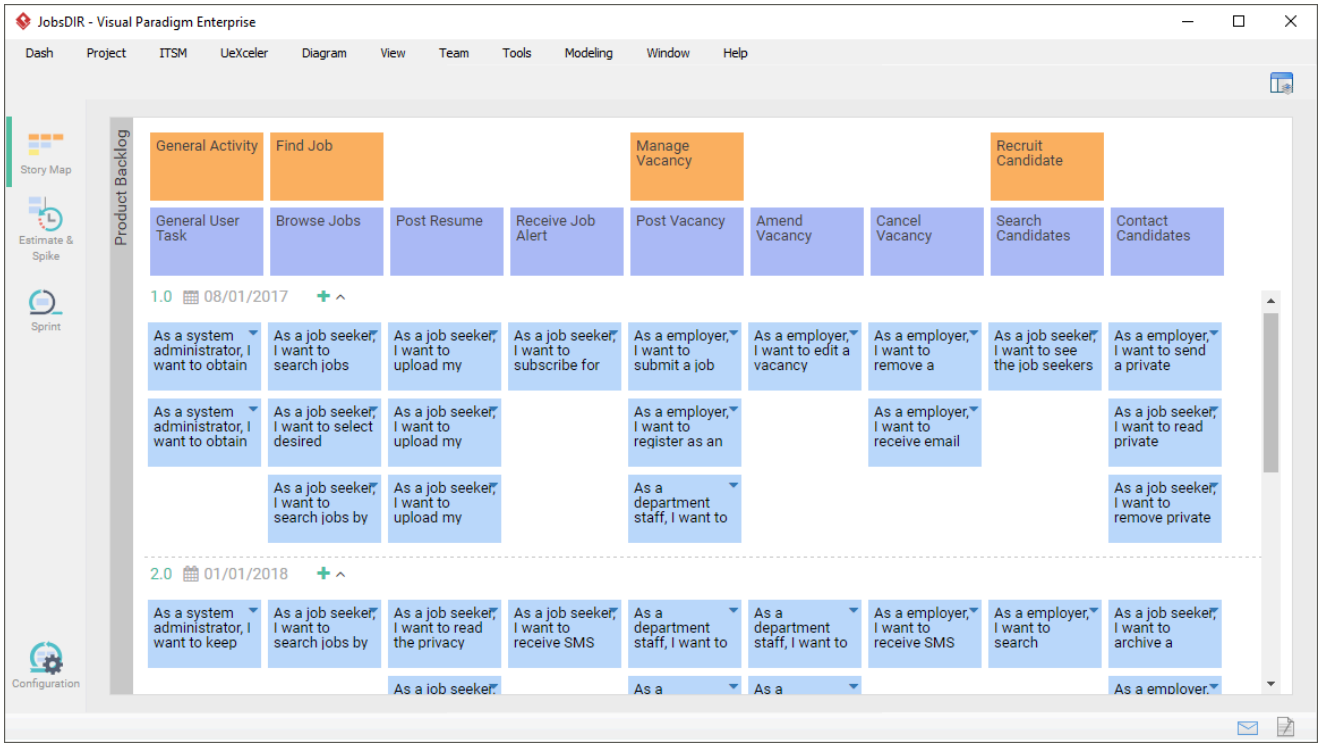
User Activities - They are laid out in the second column. This are major objectives that the system must support, with tangible business outcome. The entire row forms the backbone.

User Tasks - Each of the user activities is broke down into a set of related user tasks called narrative flow. The entire row forms the walking skeleton)

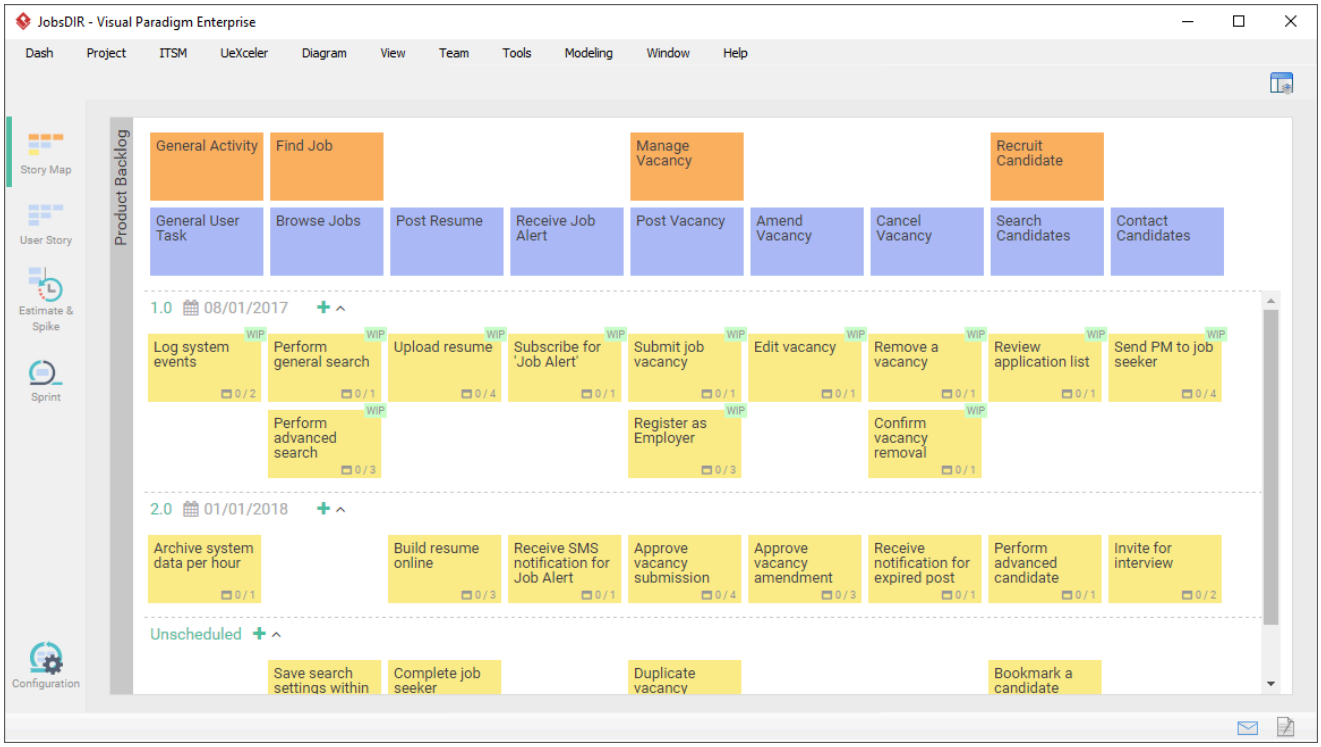
Epics / user stories - Each of the user tasks is broken down into Epics / User Stories underneath directly the user task that the feature realizes. Depending on the complexity of your projects, your team may choose the 3 or 4 level of story map which is more appropriate to you as mentioned above.

Visual Paradigm Story Map supports both 3 and 4 levels of complexity for you to cope wide variety type of projects.

3 Level Story Map (User Activates > User Tasks > User Stories)

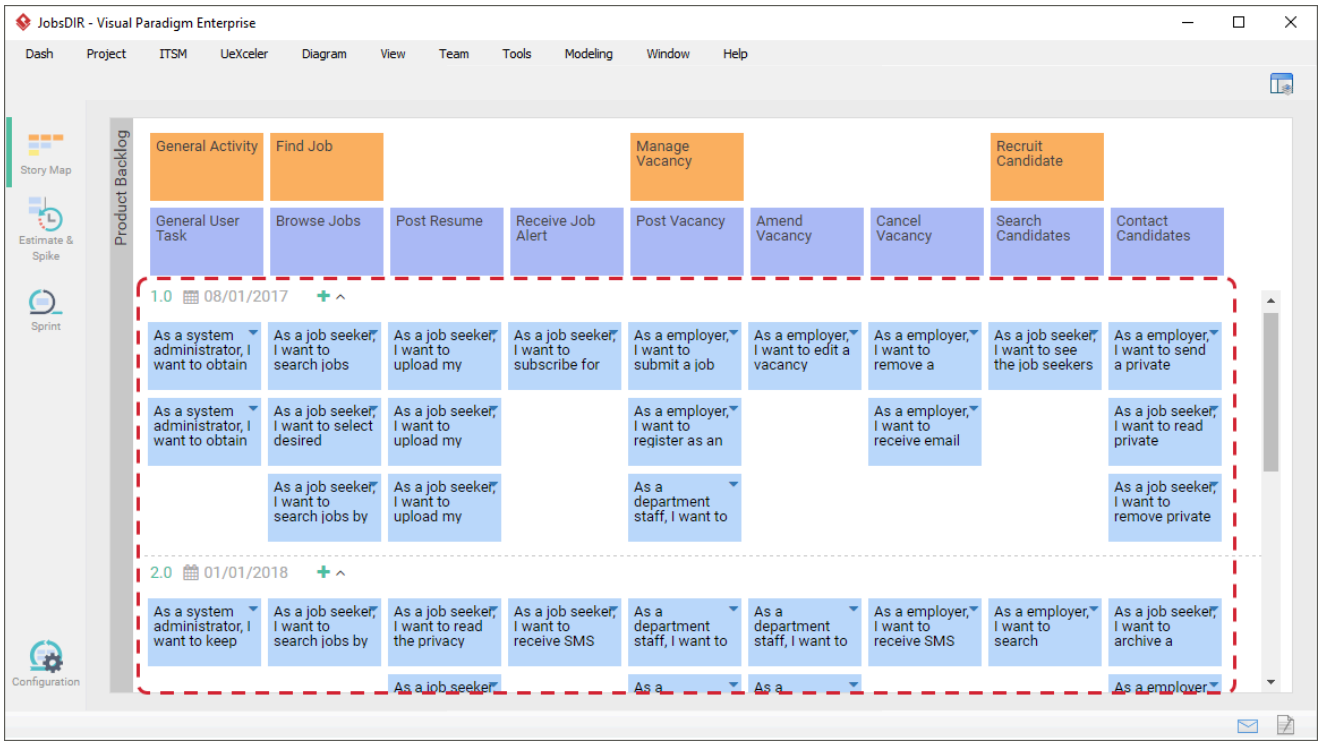


4 Level Story Map (User Activates > User Tasks > Epics > User Stories)



Planning for Releases

Use a separator to identify slices of tasks that users might use your software for to reach their goals. The smallest number of tasks that allow your specific target users to reach their goal compose a viable product release as shown in the Figure below:



If you want to develop a story map like this one, please check Visual Paradigm's [story mapping tool](#).

How to Use User Story Effectively?

Just like many other software development methodologies, if you apply user story properly in your software project you will be able to produce a quality software system plus to win the trust and satisfaction from customers. Here are some points that you need to keep in mind when using user story:

- Keep the description of user story short.
- Think from end user's point of view when writing a user story.
- Confirmation items must be defined before you start the development
- Estimate the user story before implementation to ensure the workload of your team is under control.
- Requirements are found with the end users, not by the end user or just by the development team. Keeping a good relationship with the end users will be a win-win situation for both parties.
- Communication is always important in understanding what the end user wants.

Related Links

1. Professional agile software tool with story mapping, affinity estimation and more