USER STORIES & ACCEPTANCE CRITERIA

Material producido por la Universidad Peruana de Ciencias Aplicadas
UPC, 2021.

CONTENIDO

1. OBJETIVO DE APRENDIZAJE

Al finalizar la unidad de aprendizaje, el estudiante planifica experimentos para validar requisitos de software especificados, aplicando métodos y frameworks de actualidad para procesos de software.

2. INTRODUCCIÓN

Comunicación

En un mundo ideal, las personas se entenderían instantáneamente y nada sería motivo de confusión entre ellas.

Pero en el mundo real se necesita **comunicar con claridad**.

Acceptance criteria
(Criterios de aceptación)

En software development, **acceptance criteria** ayuda a establecer de forma apropiada las expectativas del cliente para un producto.

"Quiero que mi app sea sorprendente y popular entre toda la gente que sea posible."

- ¿Claro? ¿En serio?

Utilizamos User Stories y acceptance criteria para asegurarnos de tener las descripciones claras de cómo los usuarios finales utilizarán una app y cómo los equipos deberían abordar las tareas relacionadas.

Las User Stories nos permite explicar:

- Los roles de los usuarios en un sistema
- Sus actividades deseadas
- Qué pretenden lograr completando satisfactoriamente cada User Story.

Agile approach

Ya conocemos el formato:

"As a (type of user), **I want to** (perform some action) **so that I** (can achieve some goal/result/value)."

Para los equipos ágiles, User Stories son el primer método para identificar necesidades de los usuarios.

¿Cómo nos aseguramos de que las User Stories se han completado correctamente y conforme a las demandas de los clientes?

Acceptance criteria

3. ACCEPTANCE CRITERIA (CRITERIOS DE ACEPTACIÓN)

- Son una lista formalizada de condiciones que permiten asegurar que todos los escenarios se han tomado en cuenta y se ha completado una User Story.
- Criterios expresados de forma concisa por escrito, ayudan al equipo a evitar ambigüedad sobre las necesidades de los clientes y evitan problemas de comunicación.
- Permiten establecer la visión de un producto, incluso ayudan en el proceso de desarrollo.

3.1 UTILIDAD

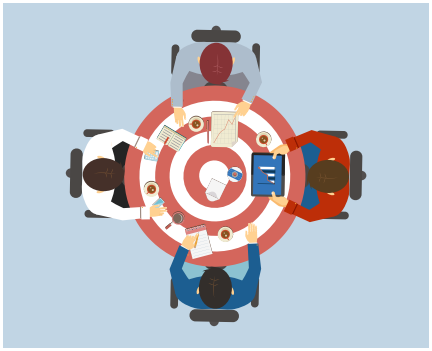
Definir alcance	Ayudan al equipo a definir los límites de una user story. Permiten confirmar cuándo la aplicación funciona como se desea, lo que significa que la user story se ha completado.
Lograr concenso	Tener acceptance criteria sincroniza el equipo de desarrollo con el cliente. El equipo sabe con exactitud qué condiciones se deberían cumplir, a la vez que el cliente sabe qué esperar de la app.
Servir de base para tests	Son la piedra angular de pruebas positivas y negativas que buscan verificar si el sistema trabaja como se espera.
Permitir planificación y estimaciones precisas	Los escenarios de acceptance criteria facilitan que las user stories se dividan en tareas que se puedan estimar y planificar.

Who writes

- Puede ser el cliente o el equipo de desarrollo.
- Criterios escritos por el product owner (cliente) son revisados por un miembro del equipo de desarrollo, para asegurarse que los criterios están especificados con claridad y que no hay restricciones técnicas o inconsistencias desde la perspectiva del desarrollo.
- Este flujo aplica a colaboraciones donde el product owner tiene experiencia en proyectos de desarrollo de software y tiene conocimiento sobre documentación de proyectos.
- Cuando los criterios son escritos por el equipo de desarrollo, entonces un analista de requisitos, project manager o especialista de QA debería encargarse de esta tarea.

When

- Deberían especificarse antes y nunca después que el desarrollo ha comenzado. De esta manera, el equipo y el product owner estarían de acuerdo sobre la minima cantidad de entregas que satisfacen los requisitos del product owner.



How to write

Tipos:

- **Rules-oriented**. En forma de lista.
- **Scenario-oriented**. En forma de escenarios que ilustran cada criterio.

4. SCENARIOS

4.1 SCENARIO-ORIENTED APPROACH

Ayuda a captar de forma rápida los requisitos, contemplar varios casos de uso y además utilizar los escenarios para pruebas de aceptación manuales y automatizadas.

- Es el approach preferido por equipos ágiles.
- Template común: Given / When / Then

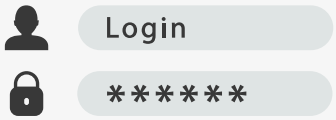
4.2 GIVEN / WHEN / THEN

- Derivado de Behavior-Driven Development (BDD).
- Formato conveniente para humanos (dado que está escrito en un estilo causa y efecto), a la vez que se adapta a herramientas de pruebas automatizadas como Cucumber y Rspec.

4.3 EJEMPLOS

Como usuario no autenticado, deseo iniciar sesión en el sitio, para acceder a mi perfil personal.

Scenario: usuario inicia sesión con credenciales válidas.



Dado que el usuario no está autenticado
Y el usuario ingresa a la página Sign-in
Cuando el usuario coloca en los campos “Username” y “Password” sus credenciales de autenticación
Y el usuario hace click en el botón Sign-in
Entonces el sistema asigna una sesión al usuario.

Como usuario del website, deseo poder buscar en el sitio, para poder encontrar información necesaria.

Scenario: usuario visualiza productos.

Dado que el usuario cuenta con el rol de usuario registrado o invitado

Cuando el usuario abre la página “Productos”

Entonces el sistema muestra una lista de todos los productos

Y el sistema muestra la sección “Buscar” en la esquina superior derecho de la pantalla.

Scenario: usuario busca un elemento por nombre.

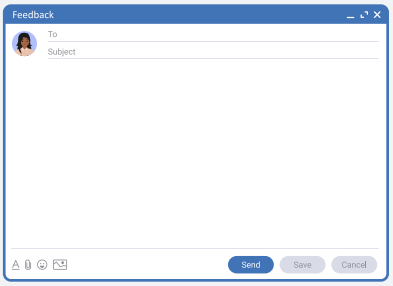
Dado que el sistema muestra la sección “Buscar” en la esquina superior derecho de la pantalla

Y el usuario llena el campo “Buscar” con el nombre de un ítem existente en la lista de productos

Cuando el usuario ordena “Aplicar”

Entonces el sistema muestra los productos en la sección “Resultados de búsqueda” con nombres de productos que concuerdan con el nombre de producto ingresado

Y el sistema muestra el número de resultados de búsqueda en la parte superior de la sección “Resultados de búsqueda”.

Como usuario del website, deseo poder proporcionar feedback, para que mi opinión sea considerada por los dueños del website en futuras actualizaciones del mismo.	**Scenario:** usuario visualiza formulario "Brindar feedback".
	**Dado que** el usuario cuenta con el rol de usuario autenticado o invitado **Cuando** el usuario abre la página "Feedback" **Entonces** el sistema muestra el formulario "Brindar feedback" conteniendo los campos "Email", "Nombre" y "Comentario" los cuales son requeridos.

Scenario: usuario envía el formulario de feedback con datos válidos.

Dado que el sistema muestra el formulario "Brindar feedback" conteniendo los campos "Email", "Nombre" y "Comentario" los cuales son requeridos.
Y el usuario coloca una dirección de email válida en el campo "Email"
Y el usuario coloca su nombre en el campo "Nombre"
Y el usuario coloca su comentario en el campo "Comentario"
Cuando el usuario ordena "Enviar feedback"
Entonces el sistema envía mi feedback
Y el sistema muestra el mensaje "Su feedback fue enviado con éxito"
Y el sistema limpia los campos del formulario "Brindar feedback".

5. BEHAVIOR-DRIVEN DEVELOPMENT (BDD)

- Es un proceso de desarrollo de software que coloca en primer lugar al comportamiento.
- Un behavior (comportamiento) es cómo un feature que opera dentro de un escenario bien definido de entradas, acciones y resultados.
- Los comportamientos se identifican utilizando specification by example.
- Behavior specs se convierten en los requisitos, los criterios de aceptación y las pruebas de aceptación.

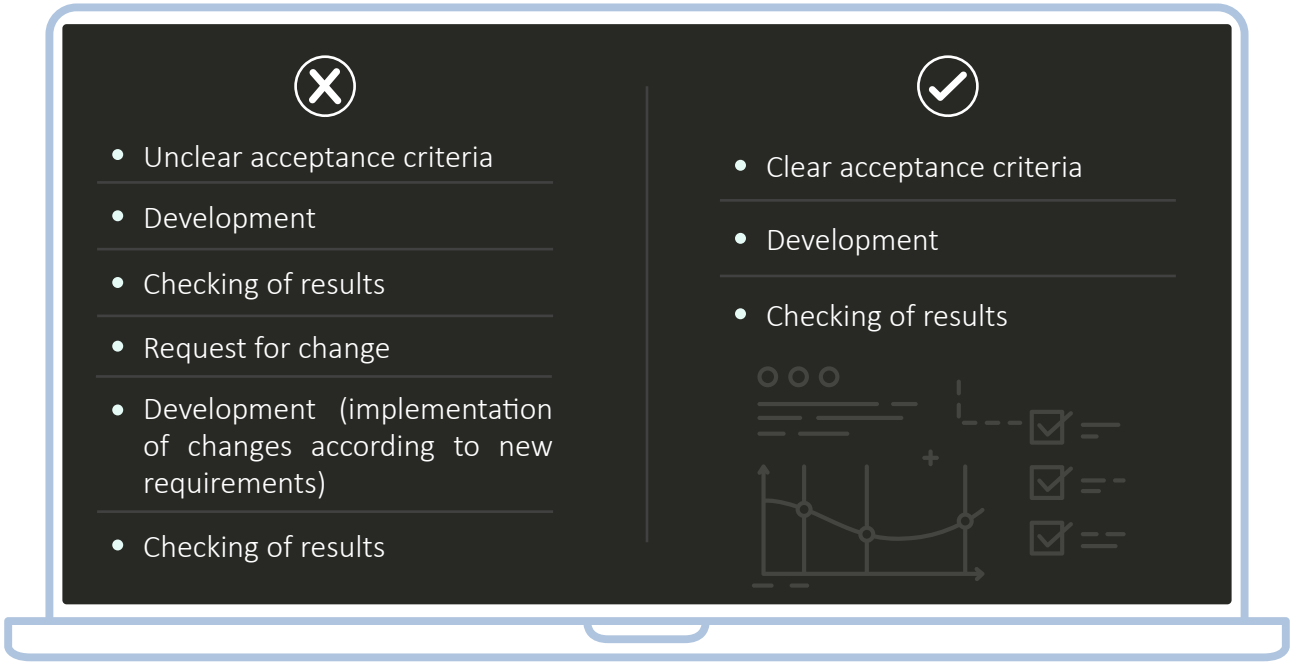
5.1 BDD TEST FRAMEWORKS

- Test frameworks también pueden automatizar specs – specs declarativos y que deberían cubrir unidades de cobertura para comportamientos únicos del producto.
- Los BDD test frameworks más conocidos son Cucumber y sus derivados, que escriben specs el lenguaje Gherkin, con el estilo "Given-When-Then".

Language	Framework	Type
C	Catch	In-line Spec
C++	Igloo	In-line Spec
C# and .NET	Concordion LightBDD NBehave x NSpec SpecFlow * xBehave.net	In-line Spec In-line Gherkin Separated semi-Gherkin In-line Spec Separated Gherkin In-line Gherkin
Golang	Ginkgo	In-line Spec
Java and JVM	Cucumber-JVM * JBehave JDave x JGiven * Scalatest Spock	Separated Gherkin Separated semi-Gherkin In-line Spec In-line Gherkin In-line Spec In-line Spec
JavaScript	Cucumber.js * Yadda Jasmine Mocha Vows	Separated Gherkin Separated semi-Gherkin In-line Spec In-line Spec In-line Spec
Perl	Test::BDD::Cucumber	Separated Gherkin
PHP	Behat Codeception *	Separated Gherkin Separated or In-line
Python	behave * freshen x lettuce pyspecs pytest-bdd * radish	Separated Gherkin Separated Gherkin Separated semi-Gherkin In-line Spec Separated semi-Gherkin Separated Gherkin-plus
Ruby	Cucumber * RSpec Spinach	Separated Gherkin In-line Spec Separated Gherki
Swift / Objective C	Quick	In-line Spec

6. TIPS & BEST PRACTICES

6.1 LA IMPORTANCIA DE LA CLARIDAD



6.2 TIPS

- **Mantén tus criterios bien definidos**, de tal forma que cualquier miembro del equipo del proyecto entienda la idea que estás tratando de transmitir.
- **Mantén los criterios realistas y alcanzables**. Define la mínima pieza de funcionalidad que seas capaz de entregar y mantente firme en ella. No trates de describir todos los detalles, pues podría terminar saturando tu backlog y quedar enterrado entre tantas tareas pequeñas.

- **Coordina con todos los stakeholders** de tal forma que los acceptance criteria estén basados en el consenso.
- **Crea criterios medibles** que permitan estimar adecuadamente tiempos de desarrollo, permanecer dentro del presupuesto y restricciones de tiempo.
- **Considera utilizar checklists** que permitan ver qué user stories están cubiertos con acceptance criteria.

6.3 BEST PRACTICES

Proper behavior

- Error típico: escribir sin un pensamiento orientado al comportamiento (procedure-driven en vez de behavior-driven).

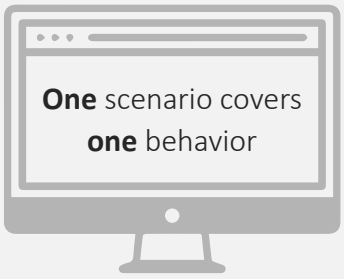
- Evita los procedure-driven criteria, que son a menudo imperativos y trazan una ruta para el sistema que cubre múltiples comportamientos, lo cual resulta en extensión innecesaria, retrasa el análisis de fallas, incrementa costos de mantenimiento y crea confusión.

Bad example! Do not copy

- **Feature:** Google Searching
- **Scenario:** Google Image search shows pictures
- **Given** the user opens a web browser
- **And** the user navigates to **"https://www.google.com/"**
- **When** the user enters **"panda"** into the search bar
- **Then** links related to **"panda"** are shown on the results page
- **When** the user clicks on the **"Images"** link at the top of the results page
- **Then** images related to **"panda"** are shown on the results page

The right way

- **Feature:** Google Searching
- **Scenario:** Search from the search bar
- **Given** a web browser is at the Google home page
- **When** the user enters **"panda"** into the search bar
- **Then** links related to **"panda"** are shown on the results page
- **Scenario:** Image search
- **Given** Google search results for **"panda"** are shown
- **When** the user clicks on the **"Images"** link at the top of the results page
- **Then** images related to **"panda"** are shown on the results page

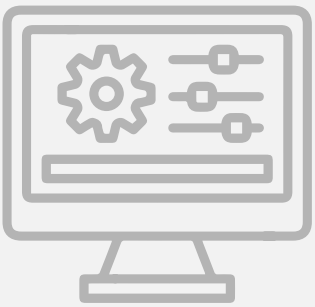


Phrasing steps

- Cómo escribir un paso es importante. Un paso escrito de forma deficiente es difícil que se pueda reutilizar.
- **Escribe todos los pasos en tercera persona**. Solo usa tercera persona siempre.
- **Escribe los pasos como una frase que denote acción con sujeto y predicado**. Las frases parciales generan ambigüedad e incluso se podrían reutilizar de forma incorrecta.
- Los pasos "Given" se utilizan para establecer un estado inicial, no para denotar un comportamiento.

Bad example! Do not copy

- **Feature:** Google Searching
- **Scenario:** Google search result page elements
- **Given** the user navigates to the Google home page
- **When** the user entered **"panda"** at the search bar
- **Then** the results page shows links related to **"panda"**
- **And** image links for **"panda"**
- **And** video links for **"panda"**
- **Feature:** Google Searching
- **Scenario:** Simple Google search
- **Given** the user navigates to the Google home page
- **When** the user entered **"panda"** at the search bar
- **Then** links related to **"panda"** will be shown on the results page



The right way

- **Feature:** Google Searching
- **Scenario:** Simple Google search
- **Given** a web browser is at the Google home page
- **When** the user enters **"panda"** into the search bar
- **Then** links related to **"panda"** are shown on the results page

"Given" should always use present or present perfect tense, and **"When"** and **"Then"** should always use present tense.

One liners

- **El título es como la cara de un escenario.** Los títulos son tan importantes como los pasos, son lo primero que la gente lee. Debes comunicar de forma concisa de lo que trata el comportamiento.
- **Los buenos títulos deberían ser líneas cortas.** Una sola oración debería bastar para capturar la intención del comportamiento.

 # Bad example! Do not copy

The user can log into the app, navigate to the profile page, and see their full name, address, phone number, email, and username.

 # The right way

The profile page displays the user's personal info.

Conjunction and disjunction

- **Evita palabras conjuntivas como "and", "or" y "but"**. Las conjunciones normalmente implican que se hará más de una cosa. Mantén cada escenario enfocado en un solo comportamiento principal.
- **Evita otras conjunciones como "because", "since" y "so".** Las frases que empiezan con esas palabras a menudo dan una explicación del porqué existe ese escenario, pero los escenarios deben enfocarse en qué o en qué consiste el escenario. El porqué se puede deducir de los pasos.

 # Bad example! Do not copy

The user can request an insurance quote from the big "Get-A-Quote" button on the home page or from the "Insurance Policies" page.

 # The right way

Two scenarios:

The user requests an insurance quote from the "Get-A-Quote" button on the home page / The user requests an insurance quote from the "Insurance policies" page.

Or

Scenario outline:

The user requests an insurance quote.

 # Bad example! Do not copy

The last five search phrases are saved so that the user can rerun them from the history page.

 # The right way

The history page saves the last five search phrases.

Avoid assertion language

- **Evita palabras “verify”, “assert” o “should”** en títulos de escenarios. Esas palabras ponen mas énfasis en la aseveración más que en el comportamiento, además se torna un título repetitivo.

 # Bad example! Do not copy

Verify the user can change their address on the profile page.

Assert that a stock quote is displayed in green text when its value is higher than its previous closing value.

The goodbye page should be displayed after a successful logout.

 # The right way

Profile page address change.

A stock quote has green text when its value is higher than its previous closing value.

Logout displays the goodbye page.

7. CONCLUSIONES

- Realizando acceptance criteria se alcanza un consenso entre las partes interesadas para la planificación, desarrollo y logro de objetivos.
- Durante el desarrollo, se establecen scenarios que permite detallar posibles interacciones del usuario con el software.
- El BDD permite desarrollar, probar y pensar el software desde la perspectiva del usuario desde un modelo de causa- efecto del comportamiento.

REFERENCIAS

Para profundizar:

- https://martinfowler.com/bliki/GivenWhenThen.html
- https://automationpanda.com/tag/bdd/
- https://cucumber.io
- https://www.mountaingoatsoftware.com/blog/the-two-ways-to-add-detail-to-user-stories

Autor: Ángel Augusto Vasquez Nuñez