UCAB UNIVERSIDAD CATÓLICA ANDRÉS BELLO

ESPECIFICACIÓN DE REQUERIMIENTOS I

Tabla de contenido

Objetivos

Revisar los enfoques utilizados al momento de especificar necesidades y *features*, a fin de satisfacer de manera efectiva las necesidades reales de los clientes, según sea el enfoque.

Reconocer cómo se identifican necesidades y *features* (características y épicas, dependiendo del enfoque) con el Diagrama de casode usoInicial, el mapa e impacto.

Introducción

Todo proceso de desarrollo comienza por describir, de manera muy precisa, qué es lo que se va a construir para, con base en ello, proponer una solución tecnológica que luego de implementarla se probará. A su vez, si tenemos una especificación de lo que se desea construir, a lo largo del proceso podemos ir verificando que se está construyendo el producto deseado o, lo que es lo mismo, el producto correcto.

Esta forma de describir lo que se espera también ha evolucionado en el tiempo. En concreto, es a partir de este siglo que se toma muy en serio esta actividad, por lo que actualmente se puede decir que contamos con dos enfoques para hacer esta especificación que siguen siendo tendencia. Uno de ellos, que podemos llamar "clásico", es el que se basa en el lenguaje de modelado UML que surge a inicios de siglo; y el otro, llamado "ágil", que surge por el paradigma metodológico de la programación extrema o *Xprograming* (XP). En este tema veremos ambos enfoques.

¿Qué se espera?

La META es entregar productos de calidad, a tiempo y dentro del presupuesto, que satisfagan las necesidades reales del cliente.

Típicamente, se nos presentan dos grandes obstáculos a la hora de precisar exactamente qué es lo que se quiere construir:

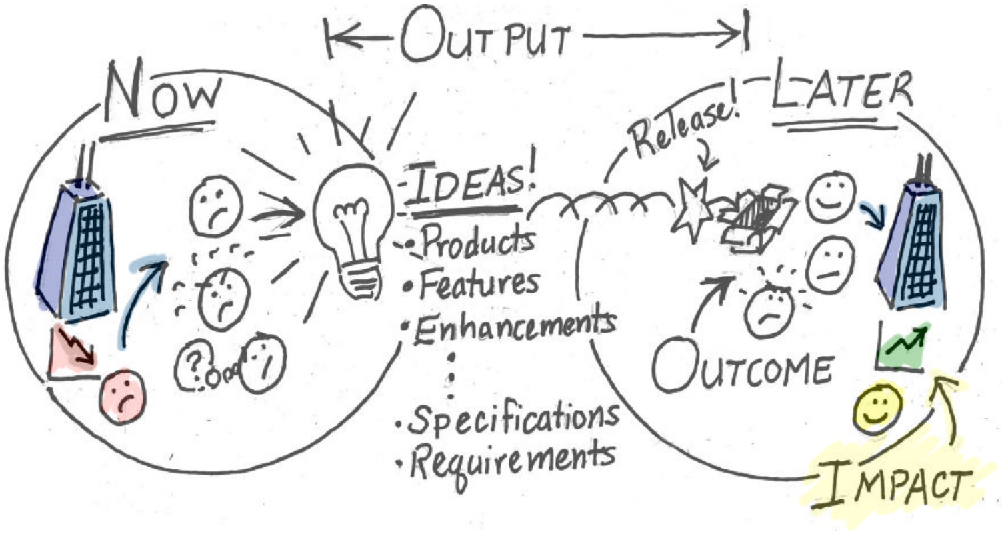
1. **Barreras de comunicación:** en un equipo de desarrollo de *software* participan cada vez más y más personas, cada una con sus intereses, con su lenguaje y con su visión de qué es lo que se requiere, por lo que se deben abordar nuevas técnicas que ayuden a superar estas barreras y así ganar experiencia dentro del marco del sistema propuesto. Estas barreras de comunicación se deben a distintas causas, entre ellas:

    - El cliente/usuario habla un lenguaje diferente al nuestro (español); habla como un contador, médico, profesor, etc.
    - A veces no tiene claridad en lo que quiere y nosotros debemos saber enfocarnos en su necesidad.

2. **Evolución e integración del sistema:** en la práctica, los proyectos se derivan de sistemas ya existentes. Por lo tanto, los analistas de sistemas deben comprender estos sistemas ya construidos, que por lo general son el resultado de una integración de componentes de varios proveedores y que, en muchísimos casos, no cuentan con una documentación adecuada o actualizada de lo que se construyó, dificultando así la incorporación de alguna nueva mejora. Tenemos entonces que adivinar cómo está estructurado este *software* para poder incorporar el nuevo componente o mejora.

El reto es:

Según nos señala Patton (2014), se quiere cambiar una realidad (ver figura) que, a partir de un conjunto de IDEAS (puede ser un producto o mejoras sobre él), se logre una solución que tenga un impacto que "haga feliz" al promotor/organización/cliente:



Fuente: Out Put. Patton (2014)

Entonces, el reto es minimizar el tiempo para implementar estas mejoras y maximizar el impacto (positivo, por supuesto).

La parte más difícil de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requisitos técnicos detallados, incluyendo todas las interfaces con gente, máquinas y otros sistemas. Se resalta que en todo el proceso esta parte del trabajo tiene gran impacto, ya que afecta el sistema si está mal hecha. También es la más difícil de corregir posteriormente.

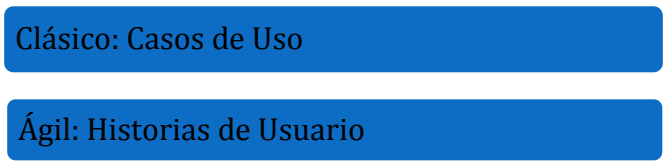
Entonces, la tarea más importante que el ingeniero de *software* hace para el cliente es la extracción iterativa y el refinamiento de los requisitos del producto.

Extracción iterativa, es decir, por iteraciones. Ya sabemos lo que es eso. Pero además debe ser ágil. Es decir, no vamos a identificar lo que se necesita y precisar esto en requisitos, detallándolos todos a la vez, sino por iteraciones (trozos/partes) que le aporten valor al promotor/usuario.

El amplio espectro de tareas, técnicas y métodos que llevan a la comprensión, identificación o elicitación de los requisitos, se le conoce como Ingeniería de requerimientos.

Su gestión es una de las mejores prácticas del proceso de desarrollo.

Como te señalé en la introducción, nos vamos a mover en dos mundos para especificar estos requisitos:



Fuente: elaboración Propia

Lo primero que debemos precisar con el promotor/cliente es la NECESIDAD.

Necesidades

Siempre hay una necesidad. Estas son la expresión de lo que el negocio/cliente requiere, y esto puede ser indispensable para su progreso. Por tanto, no se crean, existen.

Cuando el cliente tiene una necesidad, el papel de la Ingeniería de *software* es precisar si se pueden transformar en oportunidades de automatización, producir satisfacción (de productos y/o servicios) y garantizar la calidad de dichos productos o servicios.

Ejemplos de necesidades:

- Actualmente, muchos negocios de comida, con tal de seguir operando, necesitan hacer entregas a domicilio; para ello necesitan no solo un motorizado con gasolina, sino también que las personas accedan a sus menús, precios y los contacten.
- Otros negocios que manejan muchos materiales necesitan controlar el ingreso/egreso de estos materiales en sus almacenes.

- Muchos de nosotros necesitamos acceso a los medicamentos, y como no podemos ir de farmacia en farmacia, es necesario saber en cuál de ellas los están vendiendo.
- Otros lo que necesitan es botar al empleado. Ninguna aplicación hará que trabaje eficientemente.

Nuestro trabajo es:

Identificar la necesidad, y verificar que una aplicación o mejora de una de ellas logre satisfacer su necesidad.

Sin embargo, ino todas las necesidades se resuelven con un *software* o una automatización!

Cuando vemos entonces que la necesidad se resuelve con una posible funcionalidad o mejorando un *software*, estamos hablando de una característica, de una épica o de un entregable.

Característica/Épica

Una característica (*feature*) es una capacidad del sistema que satisface directamente la necesidad de un involucrado. Es lo que se estila llamar "un módulo", palabra en obsolescencia que pertenece a la década de los 80 del siglo pasado.

Tenemos entonces que la característica es una "funcionalidad" amplia que debe cumplir un *software*.

Ejemplos:

- El sistema debe proveer información de tendencias para ayudar al gerente a evaluar la situación del proyecto
- El sistema debe permitir al cliente transferir fondos entre cuentas
- La interfaz de usuario debe proveer ayudas sensibles al contexto
- Debe estar disponible los 365 días del año.

Una épica (en el enfoque ágil) es un conjunto de trabajo grande que puede dividirse en tareas específicas (denominadas "historias de usuario"), en función de

las necesidades o solicitudes de los clientes o usuarios finales. También se le llama entregable.

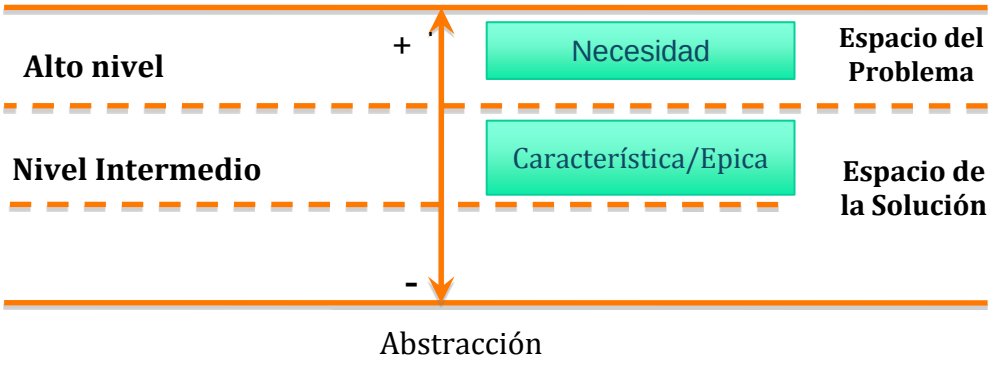
Como puedes ver, característica y épica o entregable, es lo mismo en ambos enfoques (clásico y ágil). Solo cambia el nombre.

Cuando se están identificando requisitos, nosotros debemos aprender a hablar como habla el cliente; usar su mismo lenguaje para poder entender qué es lo que necesita. Es por ello por lo que desarrollar *software* es un reto: cada cliente tiene un lenguaje diferente.

Luego de tener claro qué es lo que necesita el promotor/cliente, se le proponen (de manera colaborativa con el resto del equipo y los clientes) una o varias características/épicas que "resuelven" esa necesidad.

Esta es la razón por la que la característica/épica es una descripción en lenguaje no muy técnico y con poco detalle de la (s) posible (s) solución (es). Se dice que la necesidad pertenece al "espacio del problema", porque está escrito en el lenguaje del cliente. Por otro lado, también se dice que la característica/épica pertenece al "espacio de la solución" porque se empieza a escribir en el lenguaje común.

En realidad, estos son dos niveles de abstracción:



Fuente: elaboración propia

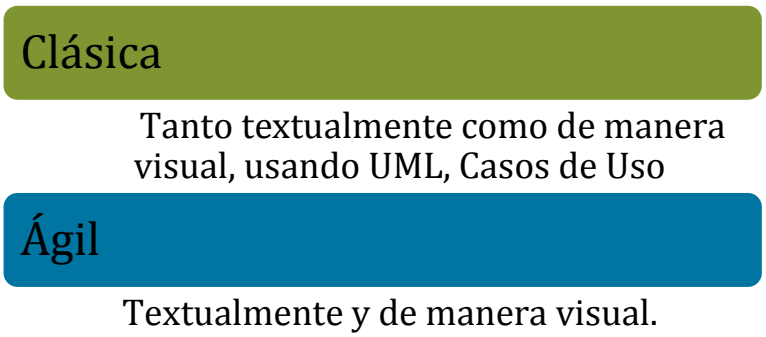
Revisemos otros ejemplos:

- **Característica/épica 1:** gestionar los productos de un conjunto de compañías, es decir, registrarlos, modificar sus datos, eliminarlo o consultarlo.
- **Característica/épica 2:** gestionar las compañías, es decir, registrarlas, modificar sus datos, eliminarlas o consultarlas.

Recuerda que algunos autores llaman a las épicas "entregables".

Ya podemos ir especificando lo que hasta ahora hemos identificado: la mal llamada "documentación" no se hace al final, sino a medida que vamos avanzando, para así demostrar que estamos llegando a acuerdos, o para analizar lo que hayamos encontrado con los otros miembros del equipo.

Podemos preguntarnos: ¿qué tenemos más o menos claro? Y la respuesta es "las necesidades" y las "características/épicas". Podemos entonces especificarlas: ¿cómo lo haremos? Hay dos rutas:

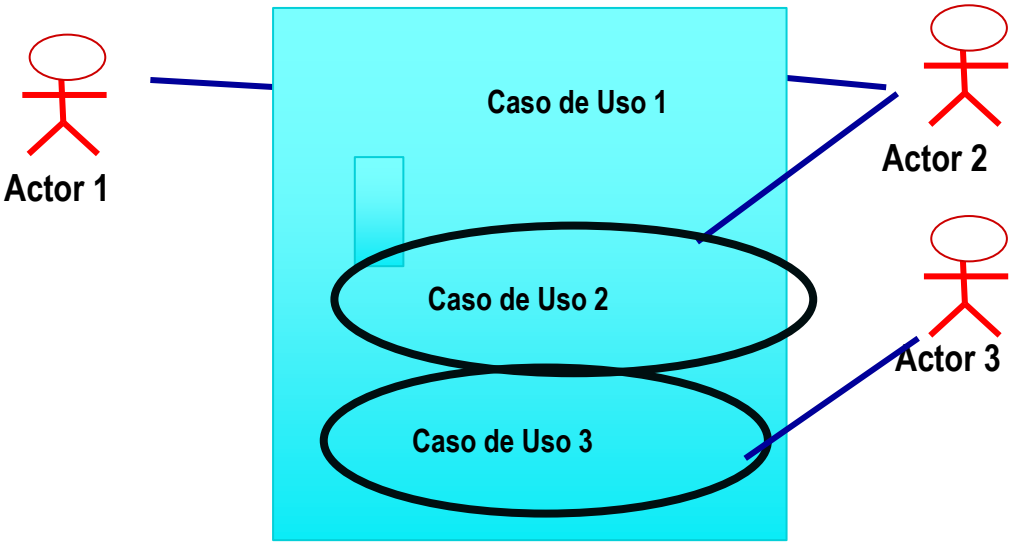


Fuente: elaboración propia

Especificación de características

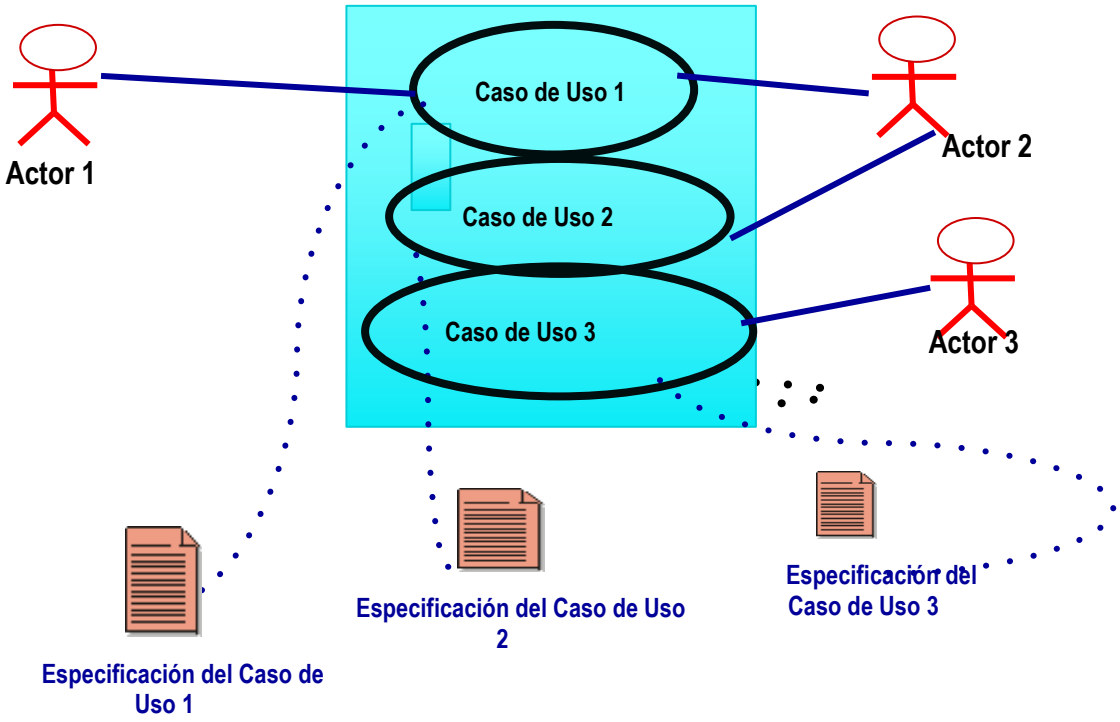
En la ruta clásica, la especificación de las características se hace modelando estas características con UML. En concreto, con el Modelo de Casos de Uso. Este modelo está conformado por tres elementos:

- **El Diagrama de Casos de Uso:**



Fuente: elaboración propia

- **Las especificaciones de los Casos de Uso:**



Fuente: elaboración propia

- **Una lista de actores vs. Casos de Uso:**

Modelo de Casos de Uso
- **Lista de todos los actores**
- **Lista de todos los Casos de Uso**

Fuente: elaboración propia

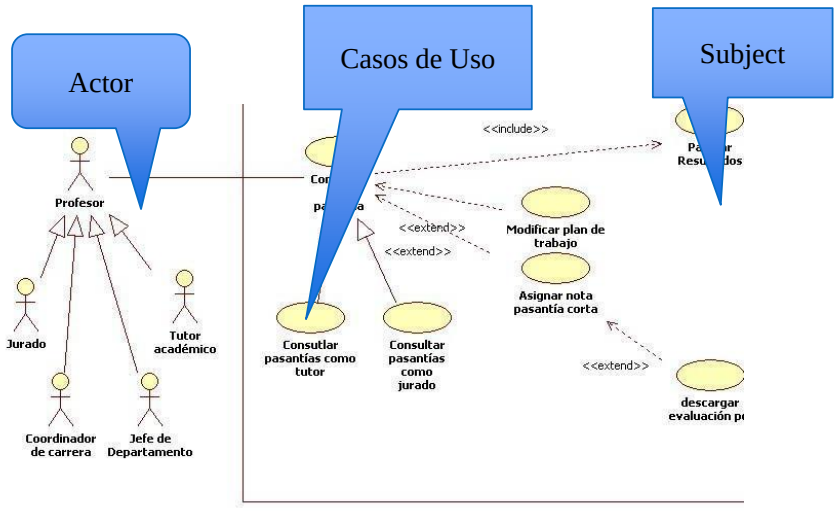
Para especificar las características, solo vamos a usar el elemento 1 del Modelo de Casos de Uso: el Diagrama de Casos de Uso. ¡Y en una versión tosca, de borrador! Por eso es mejor llamarlo inicial. Sin embargo, es excelente para llegar a un acuerdo con el cliente y los otros miembros del equipo de desarrollo.

Para elaborar un Diagrama de casode usoinicial, se necesita:

- Encontrar Actores y Casos de Uso:
  - Casos de Uso: son las características.
  - Actores: aquellos que usarán/participarán en las características.

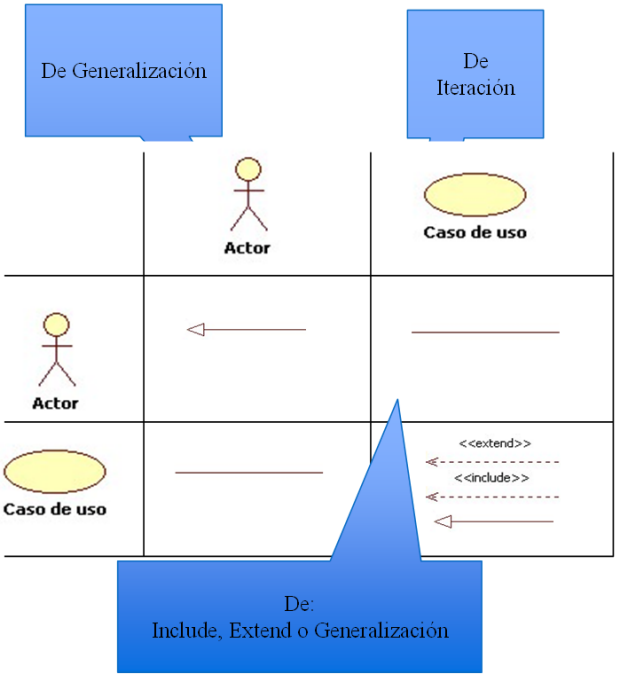
Un Diagrama de Casos de Uso, contiene, a su vez, tres elementos:

- Actores
- Casos de Uso
- *Subject*



Fuente: elaboración propia

Entre estos tres elementos del diagrama, se establecen relaciones:



Fuente: elaboración propia

Para construir el Diagrama de caso de uso inicial, simplemente:

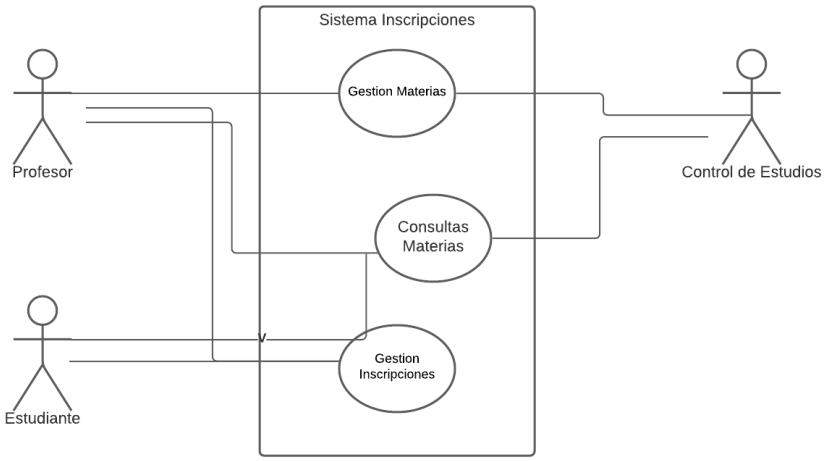
- Se toman las características identificadas y cada uno será un caso de uso.
- Se identifican los actores y se establecen las relaciones de iteración.

Por ejemplo:

Asumamos que la aplicación es un sistema de inscripciones de materias. Tendríamos dos actores: los profesores y los estudiantes. Las características identificadas son tres:

- Gestionar materias (el registro de materias se hará sobre la BD de Control de Estudios, por lo que debe interoperar con el sistema de control de estudios).
- Gestionar las inscripciones.
- Consultar materias (esta última debe interoperar con Control de Estudios, que es donde están registradas las materias).

Nuestro Diagrama de caso de uso inicial queda entonces así (ver figura).



Fuente: elaboración propia

Observa lo siguiente:

- Control de Estudios, que es otro sistema, se considera un actor, ya que "actor" es todo lo que interopera con nuestra app.
- Los actores que no son personas/unidades organizativas/cargos se colocan del lado derecho. El resto, del lado izquierdo.
- Para este diagrama (Caso de uso inicial) se coloca una relación de iteración por cada actor que interactúe con cada caso de uso.

Especificación de Épica/Entregable

Se elabora el mapa de impacto, el cual es un método de planificación estratégica gráfico. Con él se decide qué características incorporar en un producto.

Todas las características identificadas deben tener un impacto directo en el logro de ese objetivo y una razón clara de cómo lo harán.

Tenemos el mapa de impacto que fue presentado al mundo por Gojko Adzic (2012) en su libro titulado *Impact Mapping*.

Consta de cuatro elementos:

- **Meta:** que se desea alcanzar con esa característica

- **Personas:** quienes jugarán un rol en el uso de esta característica
- **Impacto:** comportamiento específico deseado de esa persona con esa característica y que le aporta valor al negocio/promotor
- **Entregable:** que "usarán" para lograr ese impacto.

Nótese que, con este método de especificación, se "desmenuza" la épica un poco más.

Meta	Personas	Impacto	Épica/Entregable
Materias	Profesor	Hace más eficiente su trabajo	Gestionar las Materias (registrarlas, consultarlas y modificarlas)
Catálogo	Estudiante	Revisar cómodamente la oferta académica	Consultar Materia
	Profesor	Consultar eficientemente la oferta académica	Consultar Materia
Inscripciones	Estudiante	Inscribirse eficientemente	Registrar Inscripción
	Profesor	Hace más eficiente su trabajo	Consultar Inscripciones

Fuente: elaboración propia

Si comparamos ambos enfoques:

Diagrama Casos de Uso Inicial	Mapa de Impacto
Características	Épica/Entregables
Actores	Personas
Subject	------
Relaciones	-------
--------	Impacto

Fuente: elaboración propia

Cierre

- Para lograr un *software* de calidad, todo el equipo de desarrollo debe conocer con precisión qué se va a construir.
- Un recorrido mental de esta actividad (identificar lo que se desea construir) comienza siempre con la identificación de la necesidad.
- Esta necesidad se puede resolver con la implementación de una mejora, o la construcción de un *software* que contenga grandes funcionalidades (*features*).
- La especificación de estas grandes funcionalidades se puede hacer bajo dos enfoques: clásico o ágil.
- El clásico utiliza el Diagrama de caso de uso inicial y el ágil, el mapa de impacto.

Referencias

Adzic, J. (2012). *Impact Mapping*. Provoking Thoughts Limited. https://www.impactmapping.org/assets/impact_mapping_20121001_sample.pdf

Patton, J. (2014). *User Stories Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly Inc.

Referencias de las imágenes

Patton, J. (2014). Out Put [Imagen]. Recuperado de *User Stories Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly Inc.