El futuro digital es de todos

MinTIC

Universidad Industrial de Santander

UIS

MisiónTIC 2022

METODOLOGÍA

DE TRABAJO COLABORATIVO

2.1. Metodologías de desarrollo de software

Una Metodología de desarrollo de software, consiste en el uso de herramientas, técnicas, métodos y modelos que ayuden a desarrollar software de calidad. Cada una de ellas tiene su propio enfoque, las cuales parten de enfoques generales de las metodologías tradicionales, que luego se desarrollan en otras metodologías específicas.

2.1.1. Metodologías de desarrollo de software tradicionales

Las metodologías de desarrollo de software tradicionales definen los requisitos de forma rígida durante el inicio de los proyectos. Esto hace que **los ciclos de desarrollo sean poco flexibles y no permitan realizar cambios**. En ese sentido, estas metodologías tienen una organización del trabajo lineal, donde las etapas son consecutivas y dependientes, es decir, no se puede empezar la siguiente sin terminar la anterior. Tampoco se puede volver hacia atrás una vez se ha iniciado la siguiente etapa. Las principales metodologías tradicionales o clásicas son:

· **Waterfall (cascada)**: Metodología en la que las etapas se organizan de arriba a abajo como una cascada, diferenciadas y obedeciendo un riguroso orden. Antes de finalizar cada etapa, se debe revisar el producto para ver si está listo para pasar a la siguiente fase. Los requisitos son rígidos y no se pueden cambiar. Los resultados se observan solamente hasta que el proyecto ya esté en una etapa avanzada.

· **Prototipado**: Se construyen rápidamente prototipos para que los usuarios puedan probarlo y realizar una retroalimentación temprana. De esta forma se corrigen errores de funcionalidades y diseño o incluso incluir requerimientos que no estaban contemplados. Es un modelo iterativo que se basa en el método de prueba y error para comprender las especificidades del producto.

· **Espiral**: Es la combinación de los dos modelos anteriores, que incluye el análisis de riesgo. Se divide en cuatro etapas: planificación, análisis de riesgo, desarrollo de prototipo y evaluación del cliente. En esta metodología se van procesando las etapas en forma de espiral. Cuanto más cerca del centro se está, más avanzado está el proyecto.

· **Incremental:** El desarrollo de software en esta metodología se realiza de manera progresiva, pues en cada etapa se agrega una nueva funcionalidad. Esto permite obtener resultados más rápidos comparado con el modelo en cascada clásico. El software se puede empezar a utilizar en etapas tempranas de desarrollo, incluso antes de terminarse y, en general, es mucho más flexible que las demás metodologías.

· **Diseño rápido de aplicaciones (RAD)**: Esta metodología permite desarrollar software de alta calidad en un corto periodo de tiempo, y dentro de sus objetivos está iterar el menor número posible de veces para conseguir una aplicación completa de forma rápida. Como ventaja ofrece un desarrollo más flexible; en contraste, requiere mayor intervención de los usuarios, los costos son mucho más altos, el código es más susceptible a errores y sus funciones son limitadas debido al corto tiempo para su desarrollo.

2.1.2. Metodologías de desarrollo de software ágiles

Las metodologías ágiles se basan en la metodología incremental, en la que en cada ciclo de desarrollo se van agregando nuevas funcionalidades a la aplicación final. Sin embargo, los ciclos son mucho más cortos y rápidos, por lo que se van agregando pequeñas funcionalidades en lugar de grandes cambios.

Este tipo de metodologías permite construir equipos de trabajo independientes que comparten las novedades de los desarrollos con periodicidad. Esto permite ir construyendo y puliendo poco a poco el producto final, con la ventaja que el cliente puede ir aportando nuevos requerimientos o correcciones bajo la marcha, ya que puede comprobar cómo avanza el proyecto en tiempo real.

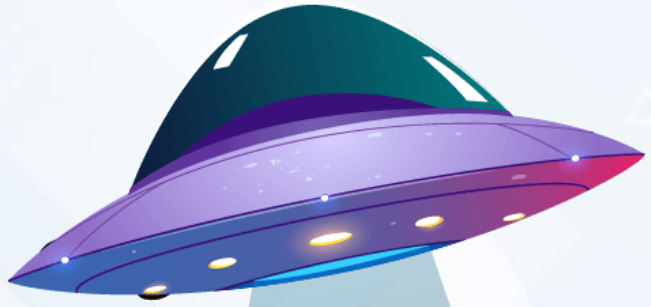
Las principales metodologías ágiles son:

- **Kanban**: Inventada por la empresa de automóviles Toyota. Consiste en dividir las tareas en actividades sencillas, para luego organizarlas en un tablero de trabajo según su estado: pendientes, en curso y finalizadas. Esta metodología permite de forma muy visual identificar las tareas prioritarias para luego definir un flujo de trabajo que permita incrementar el valor del producto.

· **Scrum**: Es una metodología incremental que al igual que Kanban, divide los requisitos y tareas. Sin embargo se itera sobre bloques de tiempos cortos para conseguir un resultado completo en cada iteración. En el siguiente capítulo se abordará esta metodología en detalle.

· **Lean**: Está diseñado para que equipos de desarrollo pequeños, pero muy capacitados, elaboren cualquier tarea en poco tiempo. Esta metodología se centra en las personas, donde su aprendizaje, las reacciones rápidas y potenciamiento del equipo, son fundamentales para el éxito del proyecto. El tiempo y los costos quedan en un segundo plano.

· **Programación extrema (XP)**: Esta metodología tiene como objetivo la creación de un buen ambiente de trabajo en equipo y un desarrollo con una retroalimentación constante del cliente. El trabajo se basa en los conceptos de: diseño sencillo, refactorización y codificación con estándares, testing, propiedad colectiva del código, programación por parejas, integración continua, entregas semanales e integridad con el cliente, cliente in situ, entregas frecuentes y planificación.



2.2. Introducción a SCRUM

La metodología SCRUM se recomienda para proyectos de alta complejidad, que manejan incertidumbre debido a los entornos cambiantes o en los cuales intervienen todos los integrantes de un equipo. Las funcionalidades del prototipo a desarrollar contienen principalmente requisitos funcionales o historias de usuario que van consignadas en la Pila del Producto(en inglés Product Backlog).

Ver Glosario SCRUM Microsoft Word - 20200311 GLOSARIO SCRUM.docx (scrumcolombia.org)

2.2.1. Características claves de SCRUM

Comunicación: La comunicación entre los miembros del equipo es esencial y está presente en todo momento.

Tiempo: En Scrum está dividido en iteraciones, más conocidas como Sprint. El cual es un periodo de corta duración, que va entre 2 y 4 semanas, según lo decida el equipo. Cada sprint está diseñado para terminarlo dentro de las fechas estipuladas y se espera que al finalizarlo, se cuente con un prototipo funcional o un producto potencialmente entregable.

Entregables: Los sprints se caracterizan por estar enfocados a entregables, es decir que todo lo que se realiza en él es entregable: tareas, funcionales, productos, prototipos, etc. De manera que si una tarea se pospone para el próximo sprint siempre habrá algo que entregar al cliente.

2.2.2. Roles del desarrollo de software en SCRUM

En SCRUM se asigna a todos los que intervienen un rol claro y definido. Uno de los principales es el Scrum Master, que se encarga de supervisar que el trabajo y el seguimiento de la metodología se lleve a cabo correctamente por parte de todo el equipo.

A continuación se detallan los roles:

- **Product Owner.** Es el encargado de liderar el proyecto, desde la perspectiva del cliente, y será la persona encargada de supervisar que se lleve a cabo el proyecto de tal forma que cumpla las expectativas del cliente.

- **Scrum Master.** Es el líder de cada una de las reuniones, ayuda en la resolución de los problemas, jugando un rol de "facilitador" para minimizar los obstáculos. El Scrum Master debe ser una persona con conocimientos técnicos de los lenguajes de programación, bajo los cuales se llevará a cabo el proyecto; de lo contrario, no tendría como ayudar a solucionar problemas.

- **Scrum Team.** Es el equipo de desarrollo y núcleo de la metodología Scrum. Se encarga de la codificación del software y de cumplir los objetivos o metas propuestas por el Product Owner.

- **Cliente.** El cliente a diferencia de otras metodologías, juega un papel esencial en SCRUM, pues tiene la capacidad de influir en el proceso debido a que conoce la lógica del negocio. El cliente puede proponer nuevas ideas y hacer comentarios de retroalimentación.

2.3. Fases de SCRUM

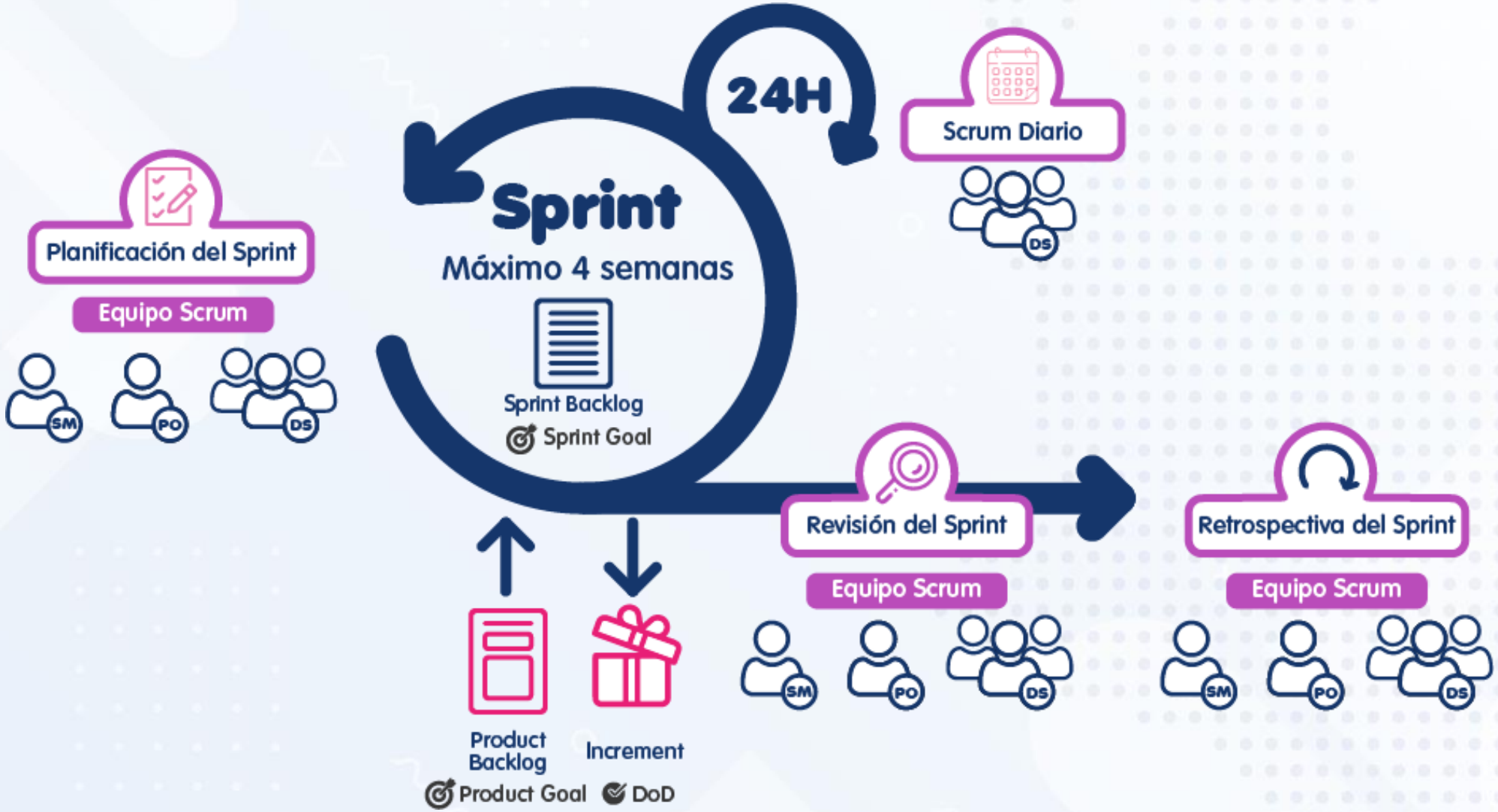
Sprint Planning. Es el primer paso de la metodología SCRUM, y consiste en planificar las acciones que se realizarán en cada Sprint:

- Unidad de tiempo: La duración del sprint, entre 1 y 4 semanas y se realiza de forma consecutiva
- Cantidad: Se deciden cuántos sprints se llevarán a cabo en el proyecto.
- Estimación de cargas de trabajo: Los miembros estiman el peso o carga de cada una de las historias de usuario y la velocidad para realizar las tareas, asignándoles a cada una un valor. Esta estimación es un poco subjetiva y cada equipo la define según la dificultad, tiempo que tarda en desarrollarse, incertidumbre de información, entre otras. La idea es tener una aproximación clara para conocer el alcance de cada acción, pero normalmente se realizan valoraciones en unidades de dificultad en vez de en horas estimadas de trabajo.

Daily Scrum Meeting. Son reuniones diarias cortas donde se supervisa el estado del Sprint. En estas reuniones de no más de 15 minutos, se analizan los problemas que no permiten el avance del equipo, se fijan los objetivos del día y se organiza la coordinación entre los miembros.

Sprint Review. Una vez finalizado el sprint, se presentan los resultados obtenidos en la iteración. También se analiza si se han cumplido todos los objetivos trazados. También se establecen, si fuesen necesarios, los cambios a realizar en el próximo Sprint.

Retrospective. Después del Sprint Review, los miembros del equipo se reúnen en un espacio de retroalimentación para comentar el trabajo realizado, analizando qué se hizo bien y qué se debería mejorar en las próximas fases. También se estima si los valores asignados en las historias fueron correctos y si se ajustaron a las necesidades del proyecto.



Fuente: [Scrum: el pasado y el futuro | Netmind]

2.4. Historias de usuario

Una historia de usuario(HU) describe una funcionalidad que será útil para el usuario del sistema, la forma como interactúa con el sistema y cómo llevar a cabo su implementación, las pruebas y verificación de la misma.

Las historias de usuario no suelen tener el nivel de detalle de una especificación de requisitos, pues si tiene demasiada información, o incluso ocupa varias hojas, entonces NO es una historia de usuario.

En ese orden de ideas, una HU debería ser pequeña, de tal forma que pueda ser desarrollada en una semana. En tan reducido espacio no pueden contener el diseño, las pruebas, normativa, etc. Ni tampoco detalles para codificar; sin embargo, se llegan a soluciones intermedias que contengan diseños(Mocks), criterios de aceptación, entre otros.

Las historias de usuario suelen expresarse con una frase simple con la siguiente estructura:

"Como [perfil], [quiero] [para]."

"Como [perfil del usuario]": Suele contestarse con la pregunta ¿para quién desarrollamos esto? Se busca más que un puesto o un nombre, buscamos el perfil de la persona. Ejm. **"como asesor de ventas"**

“Quiere”: Se describe la intención, no las funciones que usa el usuario. En el “quiere” contestamos a las preguntas: ¿Qué es lo que están intentando lograr realmente?

Esta descripción debería realizarse con independencia de las implementaciones; ya que si se describe algún elemento de la interfaz de usuario(botones, tecnologías…) y no el objetivo del usuario, se está cometiendo un error.

“Para”: Se responde a las preguntas ¿Cuál es el beneficio general que intentan lograr? ¿Cuál es el gran problema que debe resolverse?

Por ejemplo, las historias de usuario pueden tener este aspecto:

Como gestor de agenda del restaurante, quiero poder registrar las reservas, para poder estimar la ocupación o disponibilidad de las mesas.

Conocer más sobre historias de usuario: Historias de usuario | Ejemplos y plantilla | Atlassian Historias de usuario | Ejemplos y plantilla | Atlassian

Material adicional

Adicionalmente, revisa:

Guía rápida de SCRUM

Conocer más de metodologías ágiles

Metodologías del Desarrollo de Software

Metodologías de desarrollo de software