Un software es un conjunto de programas, procedimientos, documentacion y datos, y existen 3 tipos de software.

El software de base, es el conjunto de instrucciones que permiten el manejo de la computadora. Sirve para controlar e interactuar con el hardware. También se lo conoce como Software de Aplicación. Por ejemplo, los Sistemas Operativos.

Utilitarios. Es un software/programa o conjunto de instrucciones destinado a tarea de mantenimiento., soporte para la construcción de programas, entre otras tareas en general.

Aplicación o app. Realiza un conjunto de funciones, tareas o actividades coordinadas para beneficio del usuario. Como puede ser un procesador de texto, hojas de cálculo etc.

La Ingeniería de Software consiste en distintas disciplinas:

Requerimiento -> Análisis y Diseño -> Construcción de prueba -> Despliegue.

Planificación de proyecto -> Monitoreo y control de proyecto ->

Gestión de configuración de Software -> Aseguramiento de calidad -> Métricas.

El algoritmo de un desarrollo de softwares es el siguiente:

Se necesita de un personal -> Se lo incorpora al proyecto -> El proyecto se adapta -> Automatizado con herramientas -> Se obtiene el producto.

Definición de proceso de un software:

El proceso es la serie de pasos ejecutados para un proceso dado. Al hablar de un software podemos ejemplificar de la siguiente forma los procesos de un algoritmo:

Primer paso -> Definir en que consiste, que función o que funciones debe cumplir el programa.
Segundo paso -> Determinar cantidad y tipos de datos de entrada/salida.
Tercer paso -> Realizar un diagrama de flujo para desarrollar el algoritmo.
Cuarto paso -> Realizar el lote re pruebas o prueba de escritorio.
Quinto paso -> Si todo va bien, pasar el algoritmo a código.

Podemos definir una serie larga de procesos y subprocesos que se realizan antes incluso de realizar un programa en un lenguaje determinado. Pero eso llevaría mas de una hoja. Sin embargo, creo que queda claro que cada uno de estos pasos mencionados, representan un algoritmo que permiten realizar tareas específicas de forma ordenada.

Definido (Inspirados en las líneas de producción):

Si un algoritmo, se ejecuta una y otra vez y da los mismos resultados, quiere decir que el algoritmo está bien planteado. Caso contrario hay que verificar cual es la falla.

Existen los procesos empíricos los cuales pueden obtener resultados diferentes.

Los softwares tienen ciclos de vida: Secuencial, iterativo/Incremental y Recursivo.

Entre los modelos de procesos o ciclos de vida se encuentran:

Cascada, que puede tener subproyectos o no y consiste en Requerimiento -> Análisis y Diseño -> Implementación -> Prueba -> Operacion y Mantenimiento.

Incrementa: Análisis -> Diseño -> Codificación -> Pruebas, y este mismo proceso se realiza varias veces, o en distintas versiones de forma incremental. Versión 1, Versión 2 y Versión 3.

Mientras que el ITERATIVO hace las mismas pasadas en simultáneo.

La Filosofía Ágil es una ideología con un conjunto definido de principios que guía el desarrollo de un producto. (Scrum, Dsdm, Xp, Fdd, Crystal).

La filosofía Ágil tiene un manifiesto de 12 pasos:

1 - Satisfacer al Cliente con entregas frecuentes y tempranas ->2- Cambios de requerimientos ->3- Release frecuentes (2*7/4*7) -> 4- Técnicos y no Técnicos juntos -> 5- Individuos Motivados ->6- Medio de comunicación Cara a Cara ->7- Métrica de Progreso(Software Funcionando) -> 8- Ritmo de desarrollo sostenible -> 9 Atención continua a la excelencia técnica -> 10- Simplicidad: Maximización del trabajo no hecho -> 11 - Arquitecturas, diseños y requerimientos emergentes -> Intervalos regulares (Evaluación del desempeño por parte dle equipo).

El SCRUM conlleva valores como el respeto , focalización, compromiso , respeto y apertura. El scrum es Empírico, requiere de colaboración, auto-organización, buen manejo de tiempos y prioridades.
Los roles del scrum se basan en el responsable (product owner) quien demanda el desarrollo de un producto al scrum-master, y el se lo pasa al equipo, el equipo lo desarrolla y deja listo el producto.

El Product Owner, también tene responsabilidades como Gestionar la economía, participa en la planificación, cuida el product blocking, define el criterio de aceptación y verifica si no se cumple, colabora con el equipo, colabora con los stakeholders.

Las responsabilidades del Scrum Master son: Coching; Liderar el servicio ; Autoridad sobre el proceso; Escudo de interferencia; Remueve impedimentos; Agente de cambio.

Y las del Equipo son: Auto-organiado; Funcionalmente transversal; Habilidades en forma de T; Buena comunicación; Tamaño Correcto; Focalizado y Comprometido; Larga vida; Trabajo en clima de paz sostenible; Tamaño Adecuado; Actitud de Mosquetero.

El Scrum se caracteriza por: Tener roles (Scrum Master - Product Owner - Scrum Team) -> Realiza ceremonias/reuniones (Las cuales consisten en: Sprint Planning; Daily Scrumm; Sprint Review; Sprint Retrospective; Story Time(Opcional) Y por último estan los Artefactos: Product Blocking; Sprint Backlog; Version del producto.

La parte más compleja del desarrollo de software es determinar ¿qué construír?, ya que para esto hace falta tener en cuenta los requerimientos para emprender el proyecto.

En requerimientos tenemos 2 clases. Requerimientos funcionales, Son aquellos relacionados al comportamiento funcional de los componentes del software, deben ser específicas en términos de entrada procesos y salidas. Y deben poseer una vista dinámica (control, tiempo de funciones de principio a fin y su comportamiento en situaciones excepcionales).

Los requerimientos no funcionales son cruciales para el diseño y desarrollo del sistema de información. Se definen como consideraciones o restricciones asociadas a un servicio del sistema, se los llama también requerimientos de calidad o no comportamentales y pueden ser tan críticos com los funcionales.

Los problemas más comúnes que se encuentran son provenientes de una mala organización y comunicación, que va desde la solicitud del usuario, pasando por lo que entiende el lider del proyecto, lo que diseña el analista de sistemas, el enfoque del programador, la recomendación del consultor externo, la falta/escasa de documentación del proyecto, la implantación en producción, y así hasta llegar a lo que el usuario necesitaba.

Para evitar este tipo de problemas hace falta una buena comunicación (preferentemente cara a cara), organización, la verificación en la detección de características esperadas del software

para evitar errores, controlar el número de requerimientos entre otras y tener en cuenta que los requerimientos pueden cambiar.

Para esto se puede aplicar "las tres C" Card, Conversation, Confirmation. Que se basa en dejar en claro que es lo que se espera que realice la aplicación, es decir, se deja en claro cuál es el requerimiento del producto (en este caso el propósito del software a desarrollar). A esto se lo denomina Historias de Usuario.
Las historias de Usuario deben ser independientes de forma tal que no se superpongan en funcionalidades y que puedan planificarse y desarrollarse en cualquier orden.
Negociable. Una buena historia de usuario es negociable. No es un contrato explícito que se deba entregar todo o nada.
Valuable. Debe tener valor para el cliente y su negocio.
Estibale. Debe poder asignarse un indicador de peso a esa historia de usuario.
Small. Debe ser pequeña en tamaño de forma tal que pueda entregarse en tiempo y forma, según indica lo establecido (iteración).
Testeable. Poder demostrar que una historia fue implementada.

Para elaborar un proyecto, primero se debe comunicar de forma clara lo que el usuario solicita, luego de establecer con claridad los requisitos o los requerimientos, se pasa a la planificación. La planificación debe contar con características como: ¿Qué es lo que podemos hacer? ¿Cuándo lo hacemos? ¿cómo lo hacemos? ¿Quién lo va a hacer?

Una vez definido este punto, se pasa a definir el alcance. Es decir, a evaluar todas las características que pueden incluirse en el proyecto. Sin embargo, una vez evaluadas, luego pasa a definirse el alcance del proyecto (una definición más realista y concisa).
Se define todo el trabajo Y solo el trabajo que debe hacerse para entregar el producto o servicio con todas las características y funciones especificadas.
La planificación tiene como objetivo reducir riesgos, reducir incertidumbre, soportar la toma de decisiones, generar confianza y transmitir información.
Sin embargo, las planificaciones suelen tener fallos, que son relevantes a la hora de realizar el proyecto. Por lo que Scrum, también propone los "Principios de planificación Ágil". Los cuales consisten en: Trabajar como equipo, realizar iteraciones cortas, inspeccionar y adaptar, entregar algo que el cliente pueda usar en cada iteración, para a fin de cuentas, entregar "valor de negocio", "no software", haciendo referencia a entregar un producto funcional, que cumple con los requerimientos, que se realizó mediante la planificación de forma ordenada, y que se entregó en tiempo y forma.

Estimaciones.

Estimar es el proceso de encontrar una aproximación sobre una medida, que se ha de valorar con algún propósito.
El resultado del proceso es utilizado incluso si los datos de entrada estuvieran inestables, fueran inciertos o estuvieran incompletos.
Las estimaciones tienen asociadas un valor de probabilidades dado que no se realizan estimaciones en universos de certezas.
Cuando una estimación resulta ser incorrecta, se denomina "sobreestimar”, si la estimación superó el resultado real.
"Sobreestimar" si la estimación fue un valor inferior respecto al resultado real.

Para evitar errores de estimación, es necesario la información sea clara y precisa, tanto acerca del software como de la empresa que realizará el proyecto Y que no haya caos en el proyecto.

Para que haya una estimación, tiene que haber encargados de realizar un juicio experto. Es cuando un grupo de personas son informadas y tratan de adivinar lo que costará tanto el desarrollo en esfuerzo como en duración.

La analogía consiste en comparar las especificaciones (tamaño ('si es mayor o menor'), complejidad ('<>'), cantidad de usuarios (usuarios++; >complejidad ;). También se destacan otros factores como: Sistema operativo, hardware y personal del proyecto.

Estimación relativa:
Las personas no saben estimar en términos absolutos. Ser bueno comparando cosas. Comparar es generalmente más rápido. Se obtiene una mejor dinámica grupal y se emplea mejor el tiempo de análisis de las stories.

Una herramienta de estimación es el "Poker Planning", y sus características son:

Estar basada en consenso.
Opinión experta
Discusión intensa
Dimensionamiento relativo
Influencia de estimaciones Históricas.

A las Estimaciones se les añade un valor, que varía del 0 al 100. 0 es cuando no sabe bien qué tipo de producto quiere, y para eso se puede evaluar preguntando a quien contrata a la empresa para desarrollar el software , que es lo que quiere, y se lo va guiando , por eso en un punto anterior se mencionaba que debe ser "negociable".
100, es cuando ya piden cosas imposibles y es mejor ni arrancar.

Como última etapa se encuentra el TESTING y las PRUEBAS DE SOFTWARE. O mejor dicho, las pruebas de software, el testing es una parte de la misma.

El testing , contrariamente a todo lo que se mencionó anteriormente en cuanto a valores de éxito, y no éxito, en este caso, cuando hay errores, hablamos de un "testing exitoso".

En el testing encontramos el "testing funcional" y el "no funcional", que es la prueba de "cómo funciona el sistema".
Dentro de las pruebas encontramos las : Performance Testings, Pruebas de Cargas, Pruebas de Stress, Pruebas de Usabilidad, de Mantenimiento, fiabilidad y portabilidad. Se realizan a traves de niveles (Unidad, integración, Sistema, Aceptación).

Entre los niveles se encuentra la prueba de integración. Consiste en aislar las partes de un sistema, y testear el funcionamiento. Puede ser de arriba hacia abajo (top-down), o de forma inversa (bottom-up). Se priorizan los módulos críticos. Los puntos clave son: Conectar de a poco las partes más complejas, minimizar la necesidad de programas auxiliares.

Pruebas de sistema: Es la prueba realizada cuando la aplicacion está funcionando como un todo, también conocida como prueba de construcción final. Determina si el sistema en su globalidad opera satisfactoriamente (recuperación de fallas, seguridad y protección, stress, performance, etc.). El entorno de prueba, debe corresponder al entorno de producción tanto como sea posible para reducir al mínimo el riesgo de incidentes debido al ambiente específicamente y que no se encontraron en las pruebas. Por ultimo deben investigar requerimientos funcionales y no funcionales del sistema.

Prueba de aceptación:
El usuario prueba la aplicación y determina si se ajusta o no a sus necesidades.
La meta en las pruebas de aceptación es el de establecer confianza en el sistema, las partes del sistema o las características específicas y no funcionales del sistema. Comprende tanto la prueba realizada por el usuario en ambiente de laboratorio (pruebas alfa), como la prueba en ambientes de trabajo reales (pruebas beta).

Planificación -> Análisis y Diseño -> Ejecución -> Evaluación y Reportes son el proceso de prueba del Software.

Administración y configuración de Software:

Su propósito es establecer y mantener la integridad de los productos del proyecto de software a lo largo del ciclo de vida del mismo. Involucra para la configuración: Identificarla en un

momento dado - > controlar sistemáticamente sus cambios -> mantener su integridad y origen

Integridad del Producto consiste en, principalmente, satisfacer las necesidades del cliente, ser fácilmente rastreable durante su ciclo de vida, satisfacer criterios de performance y cumplir con expectativas de costo.

Repositorios.

Un repositorio contiene los ítems de configuración (Ci's).
Mantiene la historia de cada (Ci) con sus atributos y relaciones.
Usado para ser evaluaciones de impacto de los cambios propuestos.
Pueden ser una o varias bases de datos.

Los ítems de configuración (CI), son los artefactos que conforman parte del producto, o del proyecto. Pueden sufrir cambios, pueden ser compartidos entre los miembros de equipo y es necesario saber su estado y evolución.

Entre los CI existentes, algunos ejemplos son: Plan de CM, Propuesta de Cambio, Visión, 10 Riesgos Principales, Plan de desarrollo, Prototipo de Interface entre otros...

Versión: Una versión es el punto de vista de la evolución (1, 2,3 ,4).
El control de versiones refiere a la evolución de un único ítem de configuración (ic), o de cada ic por separado.
Cuando una versión de un IC (o de la configuración simplemente) evoluciona por separado, hablamos de una variante.
Las variantes representan configuraciones alternativas.
Un producto de Software puede adoptar distintas configuraciones según el lugar donde se instale. Dependiendo del SO + la máquina (es decir la plataforma), que la va a soportar. O de las funciones que vaya a realizar o no.

Una configuración que fue formalmente revisada y sobre la que se ha llegado a un acuerdo, puede servir como base para posteriores desarrollos, y puede cambiarse solo a través de un procedimiento formal de control de cambios. También permiten ir atrás en el tiempo y reproducir un entorno de desarrollo en un momento dado del proyecto.