**Clase 2 – material de lectura**

**¿Cómo es el proceso de creación del software?**

El proceso de creación de software puede llegar a ser muy complejo, dependiendo de sus características y criticidad. Por ejemplo, la creación de un sistema operativo es una tarea que requiere proyecto, gestión, numerosos recursos y todo un equipo de trabajo. En el otro extremo, si se trata de un programa simple como una calculadora, este puede ser realizado por un solo programador.

Los procesos de desarrollo de software deben sistematizarse ya que en caso contrario lo más seguro es que el proyecto no logre concluir o termine sin cumplir los objetivos previstos, y con variedad de fallas inaceptables. Cuando un proyecto fracasa, rara vez es debido a fallas técnicas, la principal causa de fallos y fracasos es la falta de aplicación de una buena metodología o proceso de desarrollo.

El proceso de desarrollo puede involucrar numerosas y variadas tareas,​ desde tareas administrativas, pasando por el personal técnico y hasta la gestión y el gerenciamiento.

Podemos organizarlas de la siguiente forma:

* Especificación y análisis de requisitos
* Diseño
* Codificación
* **Pruebas / Testing**
* Instalación y paso a producción
* Mantenimiento

Nuestra tarea en este curso consistirá en conocer las diferentes fases de desarrollo, como se conforman los equipos de trabajo, como se relacionan entre sí y particularmente dedicarlos a las pruebas de funcionamiento. Para ello es necesario entender que es un requisito o requerimiento.

**Requisitos**

En la ingeniería de software, un requisito es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Establecen qué debe hacer el sistema, pero no cómo hacerlo.

Entonces, ¿qué es un requisito?

* Condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo
* Condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta.
* Una condición o capacidad que debe ser conformada por el sistema.
* Algo que el sistema debe hacer o una cualidad que el sistema debe poseer.

**Tipos de requisitos**

1. **Requisito funcional**

Un requisito funcional puede ser una descripción de lo que un sistema debe hacer. Este tipo de requisito especifica algo que el sistema entregado debe ser capaz de realizar.

1. **Requisito no funcional**

Un requisito no funcional especifica algo sobre el propio sistema, y cómo debe realizar sus funciones. Algunos ejemplos de aspectos solicitables son la disponibilidad, el testeo, el mantenimiento, la facilidad de uso, etc.

**Características de un requisito**

Los requisitos bien formulados deben satisfacer varias características. Si no lo hacen, deben ser reformulados hasta hacerlo.

* **No ambiguo:** el texto debe ser claro, preciso y tener una única interpretación posible.
* **Conciso:** debe redactarse en un lenguaje comprensible por los inversores en lugar de uno de tipo técnico y especializado, aunque aun así debe referenciar los aspectos importantes.
* **Consistente:** ningún requisito debe entrar en conflicto con otro requisito diferente, ni con parte de otro.
* **Completo:** los requisitos deben contener en sí mismos toda la información necesaria, y no remitir a otras fuentes externas que los expliquen con más detalle.
* **Alcanzable:** un requisito debe ser un objetivo realista, posible de ser alcanzado con el dinero, el tiempo y los recursos disponibles.
* **Verificable:** Se debe poder verificar con absoluta certeza, si el requisito fue satisfecho o no.

Estas características suelen ser subjetivas, es decir, no pueden ser calculadas de forma automática por ningún sistema. Es por ello, que se tiende a utilizar métricas o indicadores que sí que pueden ser calculados de forma automática y que, de algún modo, pueden contribuir a ponderar las características mencionadas.

**Roles en el desarrollo de software**

El desarrollo de software no es una actividad simple y requiere de la participación de diferentes actores, es por ello, que necesitamos definir quienes son los participantes y como se vinculan entre sí.

Un equipo de trabajo está formado por 2 o más personas que trabajan con un objetivo en común. Cada persona tendrá un rol, es decir, actividades asociadas a la tarea a desarrollar. En equipos grandes es habitual tener funciones bien diferenciadas.

Las definiciones de roles dentro de un proyecto posibilitan la estructuración de este y se convertirán en uno de los pilares para que este camino sea exitoso.

**Entonces, ¿cuáles son los roles dentro de un equipo?**

* **Project manager / Líder de proyecto:** se ocupa de coordinar el equipo y busca asegurar que los demás participantes cumplan con las tareas asignadas. Debe tener una visión integral del proyecto, saber cómo se vinculas los diferentes módulos y administrar los tiempos para evitar retrasos.
* **Analista funcional:** se ocupa de entrevistar y explicitar las necesidades del cliente en lo que llamaremos los *requerimientos* para poder definir lo que se espera del sistema.
* **Diseñador:** a partir del trabajo del analista funcional genera el diseño del sistema y *prototipo.*
* **Desarrollador:** toma los *requerimientos* y el *prototipo* y lo convierte en un programa funcional en un lenguaje de programación específico.
* **Tester / probador:** se ocupa de asegurar la calidad del producto ya que verifica el funcionamiento y lo contrasta con los requerimientos del cliente y el prototipo.

Adentrándonos en el corazón de este curso, tomaremos **uno de los roles** del equipo de trabajo e iremos conociendo sus tareas y las habilidades necesarias.

**¿Cuáles son las tareas que realiza un tester?**

* Reunirse con usuarios del sistema para comprender el alcance de los proyectos.
* Trabajar con desarrolladores de software y equipos de soporte.
* Identificar las necesidades del negocio.
* Planificar proyectos.
* Supervisar aplicaciones y sistemas de software.
* Llevar a cabo pruebas de estrés, pruebas de rendimiento, pruebas funcionales y pruebas de escalabilidad.
* Escribir y ejecutar scripts de prueba.
* Realizar pruebas manuales y automatizadas.
* Pruebas en diferentes entornos, incluyendo web y móvil.
* Escribir informes de fallos.
* Llevar a cabo la planificación de recursos.
* Revisar la documentación.
* Trabajar para cumplir los con plazos departamentales y de proyectos.
* Proporcionar garantía de calidad.
* Proporcionar información objetiva a los equipos de proyectos de desarrollo de software.
* Detectar potenciales fallos.
* Pruebas de diseño para mitigar el riesgo.
* Presentar los resultados a los equipos de desarrollo de software y al cliente.
* Trabajar en múltiples proyectos a la vez.
* Análisis de documentación.

**¿Qué habilidades necesita un tester?**

* Ser curioso
* Ser minucioso en el detalle
* Ser imaginativo
* Ser un buen comunicador

El área de calidad de software es multidisciplinar, por lo que podemos tener una carrera orientada hacia las pruebas manuales, hacia el análisis de pruebas o hacia la gestión de equipos. También existen otros ámbitos en los cuales las pruebas son muy importantes, por ejemplo, las pruebas de rendimiento, las pruebas de seguridad o las pruebas de automatización. El testing está evolucionando mucho en los últimos años y es una profesión que está en auge. Se están creando cada vez más y más necesidades sobre las pruebas.

**Concepto de testing o pruebas de software**

Cuando pensamos en el concepto de testing, pueden surgir varias interpretaciones, veamos algunas de ellas:

* Metodología para encontrar defectos en el software.
* Proceso utilizado para medir la calidad de una aplicación.
* Proceso de verificación del correcto funcionamiento de un sistema.

El testing es el proceso que permite verificar la calidad de un producto de software, permitiendo identificar posibles fallos en la implementación, calidad o usabilidad de un programa.

El proceso de testing no solo es realizar una serie de pruebas en forma secuencial, es decir, una tras otra, sino que también este proceso incluye:

* *actividades de planificación* de las pruebas,
* la *preparación específica* de la prueba y
* la *evaluación* de los productos de software para determinar si cumple o no con los requerimientos.

El testing puede probar la presencia de errores pero no la ausencia de estos. Durante la planificación, lo que haremos es determinar que pruebas vamos a realizar.

Luego, durante la ejecución vamos a correr esas pruebas, pero ¿qué sucede si existe alguna funcionalidad que no estamos teniendo en cuenta? Es por ello que **solo** podremos probar todo aquello que hayamos definido adecuadamente.

**Fundamentos del testing**

1. **Concepto de error, defecto y fallo**

* **Error:** es una acción realizada por una persona que produce un resultado incorrecto. El error es una equivocación del desarrollador o del analista. Un error puede llevarnos a generar uno o más *defectos.*
* **Defecto (defect):** desperfecto que se encuentra en un componente o sistema y puede causar que el sistema falle en su funcionamiento.
* **Fallo (failure):** es la exposición física o visible de un defecto. Un ejemplo es cuando nuestra pantalla se pone de color “azul” por un fallo del sistema.

Como resumen, un error puede generar uno o más defecto y un defecto va a generar un fallo.

Un error puede deberse a dos posibles causas: al error humano o a causas ambientales (fallas en el disco, fluctuación en el suministro de electricidad entre otros).

1. **Concepto de requerimiento y calidad**

* **Requerimiento:** se trata de describir un atributo funcional del sistema que se considera obligatorio, es decir, que un sistema no será aprobado por el cliente si no cumple con los requisitos o requerimientos definidos.

Por ejemplo, si el cliente especificó que un documento puede ser guardado con su nombre original o puede generarse una copia con la opción de guardar como.

Si yo genero un sistema que solo permite guardar sin especificar el nombre y la ubicación de mi archivo, ese requerimiento no se verá aprobado.

* **Calidad:** es el grado en que un componente de un sistema satisface los requerimientos especificados y las expectativas del cliente.
* **Calidad del software:** es la suma de todos los atributos que se refieren a la capacidad de satisfacer los requerimientos planteados.

**¿Qué son los atributos de la calidad?**

Los atributos pueden estar organizados en dos grupos: *funcionales y no funcionales*.

Dentro de los *atributos funcionales* encontramos dos características sumamente importantes: correctitud y completitud.

La correctitud implica que la funcionalidad satisface correctamente los atributos requeridos. Mientras que la completitud significa que la funcionalidad del software satisface todos los requisitos que el cliente ha pedido.

Dentro de los *atributos no funcionales* encontramos muchos más atributos: fiabilidad (confiable en el tiempo), usabilidad (facilidad de uso / uso intuitivo), portabilidad (fácil de instalar / desinstalar y fácil de transferir a otro entorno de trabajo), eficiencia (requiere de un mínimo de recursos para realizar la tarea específica) y mantenibilidad (medida del esfuerzo que se requiere para realizar cambios en mi sistema).

Estos atributos son más difíciles de lograr porque no están perfectamente definidos o bien establecidos.

**¿Por qué son necesarias las pruebas?**

A lo largo del tiempo, el desarrollo de los lenguajes de programación ha ido creciendo en forma notable. Como así también, las técnicas y estrategias para el trabajo en equipo y la estructuración de sistemas, esto ha posibilitado asegurar la calidad del software.

Durante el desarrollo de diferentes tipos de sistemas o productos de software encontramos ejemplos que si bien pueden parecer casos extremos, permiten demostrar por qué es necesario realizar el proceso de testing.

A lo largo de la historia espacial o armamentística, encontramos problemas en el trabajo de los desarrolladores que generaron grandes gastos de dinero adicional y pudieron causar desastres naturales, como así también en la medicina nuclear, donde por errores humanos se vio afectada la salud de otras personas (control de radiación por ejemplo).

*En síntesis:*

* El software es probado por personas y las personas nos equivocamos
* A medida que se incrementa la presión en las entregas, muchas veces, no alcanza el tiempo para verificar el funcionamiento.
* Es una forma de demostrar que el software no tiene fallas y hace lo que tiene que hacer.
* Por último, por una cuestión de costos, es más barato encontrar los fallos en el momento de desarrollo que cuando está funcionando en el cliente.

**Objetivos de las pruebas**

* Adquirir conocimiento sobre el funcionamiento del sistema
* Confirmación de la funcionalidad
* Generación de información sobre posibles riesgos antes de ser entrega al usuario
* Generar confianza a partir de la validación del funcionamiento

**Duración del proceso de testing**

Es imposible revisar y chequear todas las opciones, por lo tanto, podemos preguntarnos ¿cuánto testing es necesario?

Para ello es necesario evaluar riesgos (de quien es la responsabilidad en caso de fallos) y prioridades y depende de cada proyecto.

Las pruebas de software (software testing) son técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a los actores involucrados. Es una actividad más en el proceso de control de calidad.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.

Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo.

El objetivo de las pruebas es presentar información sobre la calidad del producto a las personas responsables de este. Las pruebas de calidad presentan los siguientes objetivos: encontrar defectos o bugs, aumentar la confianza en el nivel de calidad, facilitar información para la toma de decisiones, evitar la aparición de defectos.

Teniendo esta afirmación en mente, la información que puede ser requerida es de lo más variada. Esto hace que el proceso de testing sea completamente dependiente del contexto en el que se desarrolla.

El ambiente ideal de las pruebas es aquel que es independiente del desarrollo del software, de esta manera se logra objetividad en las pruebas.

**El proceso de testing**

La ejecución de pruebas es sólo una parte del proceso de las pruebas. Dependiendo del enfoque seleccionado el proceso de pruebas se realizará en diferentes puntos del proceso de desarrollo. Es decir, además del proceso de desarrollo, tenemos un proceso para el testing.

Si bien tenemos diferentes fases en este proceso, estas fases pueden superponerse y la fase de control se realiza en todas las fases para poder saber en cada momento en qué estado nos encontramos.

1. **Planificación y control**

El control de las pruebas es una actividad continua que influye en la planificación de las pruebas. El plan de pruebas maestro (master test plan) será modificado de acuerdo a la información adquirida a partir del control de prueba.

Para saber el estado del proceso de pruebas comparamos el progreso logrado con respecto al plan de pruebas. Se inician medidas correctivas y esto permite tomar decisiones.

Tareas del Control de Pruebas

- Medir y analizar los resultados de las pruebas: De esta manera tendremos conocimiento de la cantidad, el tipo y la importancia de los defectos que hemos encontrados.

- Monitorear y documentar el progreso: Esta tarea nos sirve para determinar cuántas pruebas se completaron, cuáles fueron los resultados, y qué riesgos se encontraron para evaluar.

- Brindar información de las pruebas a personas interesadas: Estos reportes les sirven a las personas interesadas para saber el estado de las pruebas y así poder tomar las acciones necesarias.

- Iniciar acciones correctivas dependiendo de lo que se necesite corregir: Dependiendo de lo que necesitemos corregir, es posible ajustar los criterios de salida para los defectos detectados, o priorizar defectos bloqueantes.

- Tomar decisiones que definimos si seguimos o paramos las pruebas.

Tareas de la Planificación de Pruebas

-Determinar el alcance y riesgos: Nos preguntamos si vamos a probar un software completo, un componente, o algún otro producto.

-Identificar los objetivos de las pruebas y los criterios de salida de pruebas: Qué es lo más importante en este caso? Prevenir defectos, verificar que el software cumple con los requerimientos, medir la calidad?

-Determinar el enfoque: Todas estas preguntas nos permiten definir el enfoque de nuestras pruebas. Cómo ejecutaremos las pruebas, qué técnicas usaremos, qué se probará y cuán extensamente, es decir la cobertura de pruebas, quienes van a participar en el equipo de pruebas y durante cuánto tiempo?

-Implementar la estrategia de pruebas, y planificación del período de tiempo para el desarrollo de las actividades a seguir.

-Adquirir recursos necesarios para las pruebas como personas, computadoras, software, entorno de pruebas, presupuesto de pruebas, fechas.

-Selección de condiciones de entrada y de salida: Una condición de entrada será, por ejemplo, que el entorno de pruebas está listo y estable, que las herramientas necesarias están instaladas, el equipo de pruebas está completo. Los criterios de salida pueden ser: que todos los casos de pruebas diseñados sean ejecutados, que se hayan ejecutado pruebas de regresión, que no existan defectos bloqueantes, que los defectos de criticidad mediana no superen un determinado número. Los criterios de salida son condiciones acordadas con la gente involucrada, para que el proceso sea considerado formalmente concluido.

Lo que se concluya en estas tareas se plasma en documentos que crearemos también en esta fase y que usaremos en todo el proceso. Los documentos son los siguientes:

-El documento de Plan de pruebas describe alcance, enfoque, recursos y calendario de las actividades de pruebas previstas.

-El documento de Estrategias de prueba describe a alto nivel los niveles de prueba a realizar.

-El documento de Enfoque de pruebas incluye análisis de riesgo, técnicas de diseño de pruebas a aplicar, criterios de salida y tipos de prueba a ejecutar.

1. **Análisis y diseño**

Las tareas más importantes:

- Revisión de elementos: Se revisan los elementos básicos para las pruebas (requerimientos, arquitectura, especificaciones de diseño, interfaces). Utilizamos estas bases para poder comenzar con el diseño de pruebas.

- Testeabilidad: Evaluación de si los elementos básicos del punto anterior pueden generar casos de prueba.

- Condiciones de Pruebas: Identificación y revisión de las condiciones de las pruebas basándonos en los puntos anteriores: Esto nos brinda una lista a alto nivel de lo que nos interesa probar.

- Diseño de Casos: tanto los positivos (que dan muestra de la funcionalidad en sí) como los negativos (comprueban situaciones en las que hay tratamiento de errores).

- Entorno: Esta actividad trata sobre la puesta a punto del entorno de pruebas: disponibilidad del entorno, administración de usuarios, carga de datos.

- Herramientas: Seleccionar, proveer e instalar las herramientas de pruebas, procesos, procedimientos y responsabilidades.

1. **Implementación y Ejecución**

En esta etapa realizamos la ejecución de las pruebas, manual o mediante el uso de alguna herramienta:

- Se finalizan, implementan y priorizan los casos de prueba y los procedimientos de prueba.

- Trazabilidad: Se verifica que se pueda realizar la trazabilidad entre los elementos básicos de prueba y los casos de prueba.

- Ejecución: Se ejecutan los casos de manera automática o manual.

- Registro de los resultados: Una vez que hemos ejecutado, ya sea manual o automáticamente, los casos de pruebas, entonces podemos registrar los resultados obtenidos, esto incluye: identidades y versiones del software, las herramientas de pruebas utilizados, los productos de soporte de pruebas y también incluye comparar los resultados reales con los resultados que eran esperados.

- Comparación: Se comparan los resultados obtenidos con los resultados esperados.

- Informe: Se informan discrepancias en un defecto para que sea arreglado.

- Repetición: Se repiten las pruebas para confirmar que el defecto ha sido corregido (re testing). También incluye realizar pruebas de regresión para asegurarnos que los cambios nos han introducido nuevos defectos (**testing de regresión**)

1. **Evaluación de criterio de salida y reportes**

Los resultados de las pruebas son evaluados contra los objetivos definidos.

Tareas:

- Comparación: Comparar los registros de prueba contra los criterios de salida especificados en la fase de planificación de prueba. Es decir, evaluamos la evidencia que se tiene de las pruebas que se ejecutaron, fallas reportadas, solucionadas o pendientes para así confirmar si se completaron los criterios de salida que dan fin a las pruebas.

- Evaluación: Con resultados de la tarea anterior, se evalúa si se necesitan ejecutar más pruebas o si el criterio de salida debe modificarse. Es decir, aquí definimos si ejecutamos más pruebas en caso de que no se hayan ejecutad todos los casos de prueba diseñados; definimos si continuamos con la ejecución de las pruebas si es que no se alcanzó la cobertura deseada; o definimos que necesitamos diseñar más casos de prueba en caso de que nuevos riesgos se han descubierto.

- Reporte: Lo que hacemos aquí es preparar un resumen de las pruebas ejecutadas para que las partes interesadas estén al tanto de nuestro trabajo y de nuestro avance.

1. **Actividades de Cierre - Test closure**

Tareas:

- Recolección: Recolectar información sobre las pruebas completadas.

- Verificación: Verificamos que las pruebas y la documentación acordadas hayan sido entregadas al cliente según lo definido en el plan de pruebas.

- Documentación: Se deben documentar todos los casos de pruebas corridos, todos los resultados, todos los bugs que se hayan encontrados, vamos a documentar hasta la aceptación del sistema.

- Analizar lecciones aprendidas para futuros proyectos. Entre ellos, mejoras a los procesos del ciclo de desarrollo de software. También verificaremos dónde hubo problemas y el número de fallas para realizar mejoras de diseño, ejecución y revisión de las pruebas.

**El proceso de desarrollo de software**

El proceso de desarrollo puede involucrar numerosas y variadas tareas, desde lo administrativo, pasando por lo técnico y hasta la gestión:

* Análisis de requisitos
* Diseño
* Codificación
* Pruebas (unitarias y de integración)
* Instalación y paso a producción
* Mantenimiento

Para cada una de las fases o etapas listadas, existen sub-etapas (o tareas). El modelo de proceso o **modelo de ciclo de vida** utilizado para el desarrollo, define el orden de las tareas o actividades involucradas,​ también define la coordinación entre ellas, y su enlace y realimentación. Entre los más conocidos se puede mencionar: modelo en cascada o secuencial, modelo espiral, modelo iterativo incremental.

Los diferentes modelos de **ciclo de vida** de desarrollo necesitan diferentes enfoques hacia la prueba. Los modelos de desarrollo software son utilizados para el desarrollo de software incluyendo las actividades del proceso de pruebas. Las pruebas no existen de manera aislada, están relacionadas con las actividades de desarrollo de software.