UNIDAD 1

¿QUÉ ES EL SOFTWARE?

El software es el componente lógico que poseen los sistemas de información automatizados que permiten a la computadora realizar determinadas tareas.

CICLO DE VIDA DEL DESARROLLO DE SOFTWARE

La Normativa ISO/IEC/IEEE 12207:2017 establece:
"Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente".

FASES QUE COMPONEN EL DESARROLLO DE SOFTWARE

Identificación de Requerimientos:
Se entienden las necesidades del cliente.
Análisis:
Traducción a requerimientos de la aplicación a desarrollar.
Diseño:
Transformación en especificaciones técnicas. Se da solución técnica.
Construcción:
Construcción de todos los componentes de software.
Pruebas:
Aseguramiento de la funcionalidad y el comportamiento deseado.
Aceptación:
Operación de la aplicación.

CALIDAD EN EL DESARROLLO DE SOFTWARE

Definición básica: Satisfacer las necesidades del cliente
Hay tres cualidades en el software y su información
• Funcionalidad
• Calidad (Portabilidad, Eficiencia, Usabilidad, Fiabilidad, Seguridad: Autenticidad, Integridad, Confidencialidad, Responsabilidad y No Repudio)
• Defectos
Los clientes se centran principalmente en la funcionalidad, y en menor medida, en las propiedades de la calidad y defectos. Debemos centrarnos en los Defectos.

EL COSTO DE LOS DEFECTOS

Considere los costos de estos dos escenarios.
Un individuo encuentra y corrige un error en una fase temprana del proyecto.
• Lo encontró haciendo Peer Review.
• Le tomó menos de 1 hora arreglarlo. Nadie más estuvo involucrado.
Un cliente encuentra un problema en la operación.
• Lo encontró y lo reporta a la Organización que lo desarrolló.
• Estos analizan el problema, identifican en donde se encuentra, corrigen el defecto, se compila, se vuelve a probar y lo ponen de nuevo en operación.
¿Cuál de los 2 escenarios tiene mayor costo?

ÉTICA PROFESIONAL
Conjunto de normas y valores morales que los profesionales de un determinado sector deben respetar durante el ejercicio de su profesión.
O dicho de otra manera:
Serie de comportamientos y pautas de actuación encaminadas a fomentar las buenas prácticas laborales y la armonía social.

ÉTICA PROFESIONAL: EJEMPLOS
Cumplir con la cotización o propuesta de negocio ante un cliente (Alcance, Inversión, Tecnología, Equipo de Trabajo, Tiempo de Entrega)
Cuando un cliente paga al 100% el desarrollo de un software, el código, la documentación y toda la información inherente al trabajo, le pertenece sólo a él. No podemos utilizar nada de esto (excepto la experiencia vivida) con otro cliente.
Proporcionar garantía de nuestro trabajo de desarrollo de software, sin costo para nuestro cliente, cuando los defectos son imputables a nuestro trabajo.
Etc.

UNIDAD 2
Metodología de Desarrollo de Software
Conjunto de métodos, técnicas, y herramientas que se utilizan para diseñar una solución de software.
Nos indica el qué, el cómo y con qué para encontrar una solución que satisfaga las necesidades del cliente.

Aportación de la Metodología de Desarrollo de So5ware
Organización.
Conlleva la ejecución del trabajo con disciplina ingenieril.
Con racionalidad y eficiencia.

Controlar el desarrollo del trabajo.
Esto sirve para minimizar los márgenes de errores y anticiparse a esa situación.
Ahorrar tiempo y gestionar mejor los recursos disponibles.
Recursos que pueden ser humanos, materiales y tecnológicos.
Mejor imagen ante el cliente.
Ya que siempre estará por encima de la improvisación, el uso de una metodología.

CASCADA
Paradigma mas antiguo.
Identificación de Requerimientos.
Se entienden las necesidades del cliente.
Análisis.
Traducción a requerimientos de la aplicación a desarrollar.
Diseño.
Transformación en especificaciones técnicas. Se da solución técnica.
Construcción.
Construcción de todos los componentes de software.
Pruebas.
Aseguramiento de la funcionalidad y el comportamiento deseado.
Aceptación.
Operación de la aplicación.

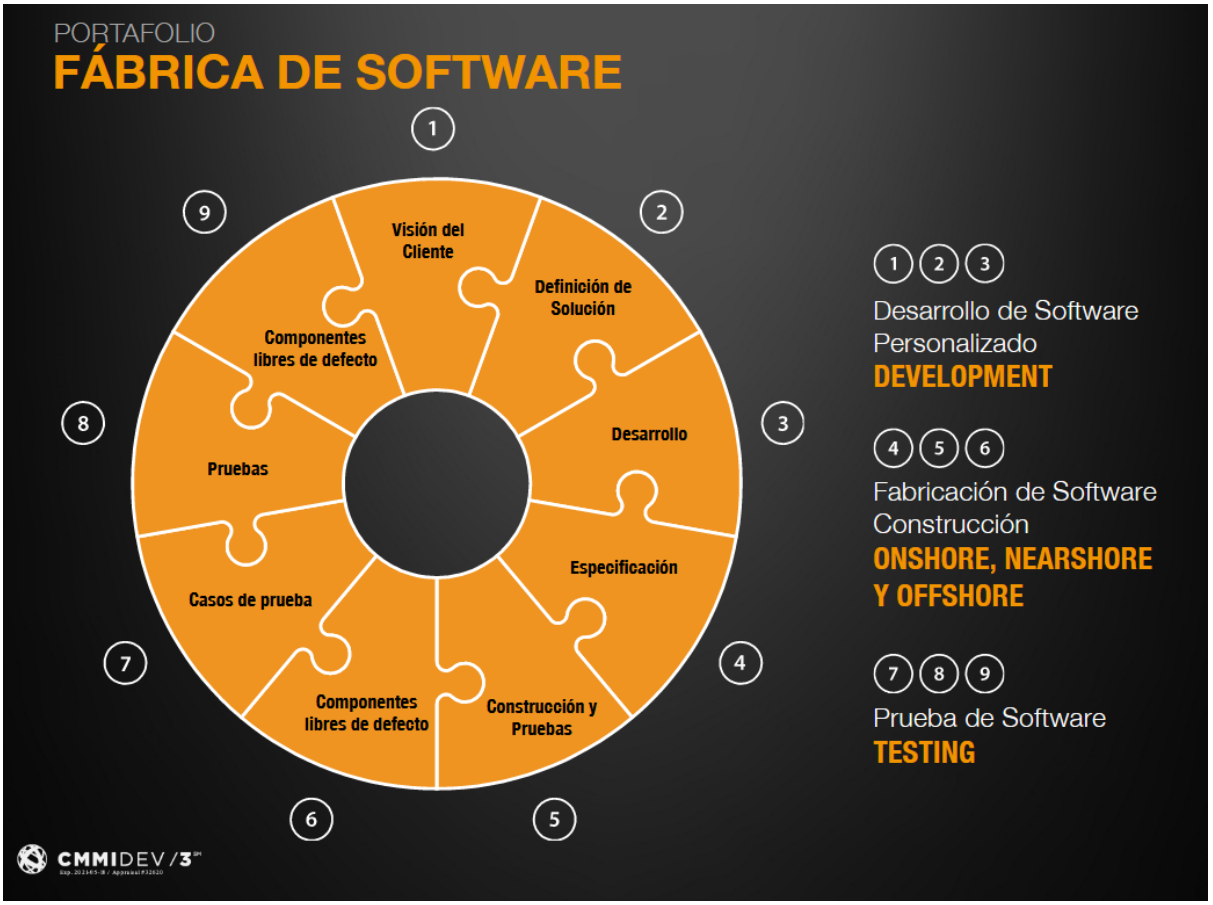
PROTOTIPOS
-> Recolección de Requerimientos -> Diseño Rápido -> Construcción Prototipo -> Evaluación de Cliente -> Refinamiento del Prototipo -> Producto de Ingeniería -> (Ciclo)

PROTOTIPOS
Se diseña solución en etapas tempranas del Proyecto.
Contempla la funcionalidad principal mediante una alta participación del usuario.
• Recolección de Requerimientos: Definición de la funcionalidad.
• Diseño Rápido: Funcionalidad medular visible.
• Construcción del Prototipo: Construcción de funcionalidad.
• Evaluación del Cliente y Refinamiento: Aprobación del Cliente.
• Producto de Ingeniería: Operación de la aplicación desarrollada.

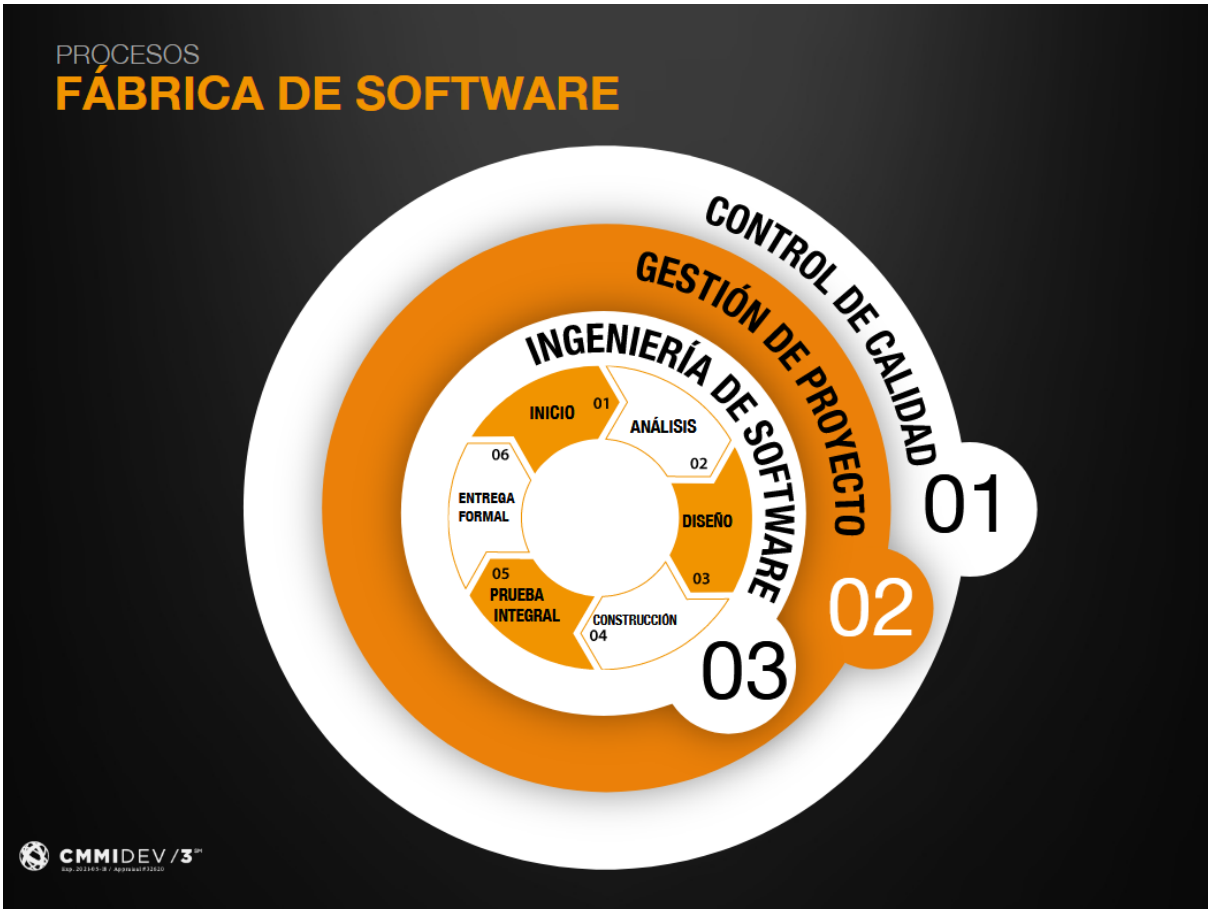
Ágil (RAD – Rapid Application Development)
-> Plan -> Design -> Develop -> Test -> Release -> Feedback -> (Ciclo)

Ágil (RAD – Rapid Application Development)
Implica el desarrollo iteraTIvo (sprint) y la construcción de ProtoTIpos.
Se desarrolla cada funcionalidad con alta parTIcipación del cliente en las decisiones y definición de comportamiento deseado de la aplicación en desarrollo.
• Plan: Determinación y priorización de las funciones.
• Diseño: Solución a cada funcionalidad.
• Desarrollo y Prueba: Construcción de componentes de soKware. Aseguramiento del comportamiento y uso adecuado de los recursos computacionales.
• Liberación y Feedback: Operación y ajustes de la aplicación.

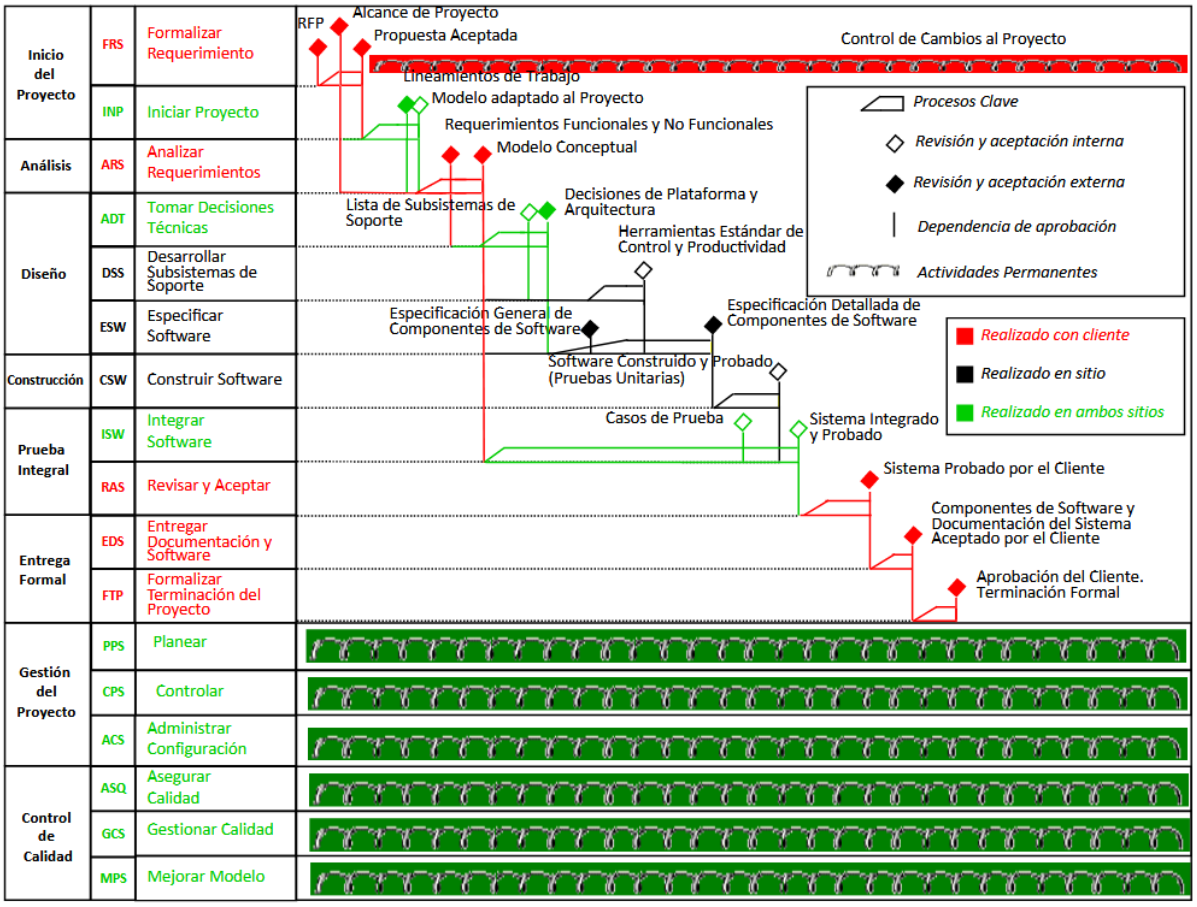
Fábrica de Software
• Servicios relacionados al Desarrollo de Software.
• Grupo de especialistas TI.
• Líneas de Producción no por Cliente, sino por roles (Analistas, Diseñadores, Constructores, etc) atendiendo diversos proyectos.



(Fábrica de Software)



(Fábrica de software)



(Foto tabla proyecto)

FRS - Formalizar Requerimiento

Objetivos: Tener claramente definidas las necesidades del cliente antes de comenzar a trabajar en el
proyecto y acordar todos los compromisos con los involucrados en el proyecto.

1. Generar el equipo de persecución del negocio
2. Generar el plan para la elaboración de la propuesta
3. Generar el pre-análisis de los requerimientos
4. Definir las caracterísFcas de calidad del cliente
5. Establecer la solución tecnológica
6. Estimar el proyecto de soFTware
7. Generar el plan general del proyecto
8. Integrar la propuesta de negocio y presentar la propuesta de negocio

INP - Iniciar Proyecto

Objetivos: Formalizar el inicio del proyecto con el cliente (y con el equipo de desarrollo), estableciendo
los lineamientos de trabajo y los recursos con los que inicia el proyecto, y generar el plan detallado del proyecto que de visibilidad y viabilidad.

1. Llevar a cabo junta de presentación del proyecto.
2. Formalizar el inicio del proyecto.
3. Asignar recursos al proyecto y registrar al equipo de trabajo.
4. Adaptar el modelo que se utilizará para llevar a cabo el proyecto de software.

5. Detallar las características de calidad del proyecto.
6. Determinar lineamientos de trabajo.
7. Planear las actividades de arranque de administración de la configuración
8. Definir el plan detallado de trabajo
9. Notificar el plan de trabajo.
10. Generar respaldo inicial

ARS - Analizar Requerimientos
Objetivos: Recopilar, organizar y modelar una solución a los requerimientos del cliente para desarrollar
una aplicación computacional.
1. Detallar los requerimientos funcionales de la aplicación (Procesos de negocio, Diccionario de datos, Interfases de Usuario, Etc.)
2. Detallar los requerimientos no funcionales de la aplicación (Políticas de operación y medio ambiente, Volumen de datos, Tiempos de respuesta, Etc.)
3. Generar el documento del modelo conceptual (Objetivos, Requerimientos, Implantación)

ADT - Tomar Decisiones Técnicas
Objetivos: Detallar los componentes tecnológicos que están siendo usados a lo largo del proyecto.
1. Identificar Arquitectura Computacional (Procesos, Distribución de información y equipo tecnológico, Enlaces de comunicación y Políticas generales, etc.)
2. Identificar Plataforma Tecnológica (Software y Hardware para la operación y el desarrollo del proyecto, Etc.)
3. Generar documento con los subsistemas de soporte a desarrollar.
4. Generar documento de decisiones técnicas.

DSS – Desarrollar Subsistemas de Soporte
Objetivos: Generar la infraestructura necesaria para permitir que la etapa de construcción del software
cumpla con las expectativas funcionales y técnicas del cliente.
1. Adaptar plantillas para diseño computacional y casos de prueba.
2. Adaptar herramientas para construir el software.
3. Establecer estándares de codificación.
4. Desarrollar lógicas de conducta predefinida, que permitan tener un mejor rendimiento en la construcción del software.
5. Desarrollar lineamientos de trabajo para los miembros del equipo de desarrollo.

ESW – Especificar Software
Objetivos: Detallar cada uno de los componentes de software especificados.
1. Generar la secuencia de cada uno de los procesos computacionales.
2. Desarrollar los diagramas de flujo de cada proceso.
3. Especificar de manera general y a detalle, cada uno de los componentes de software a construir (Programas, Rutinas, Procedimientos, Objetos, Etc.)
4. Especificar Base de Datos.

CSW – Construir Software

Objetivos: Construir y probar unitariamente cada uno de los componentes de software construidos.
1. Construir componentes de software.
2. Probar componentes de software (Prueba Unitaria)
3. Ubicar a cada uno de los componentes construidos, bajo la Administración de la Configuración.
4. Generar la documentación de cada componente de software construido.
Ver ejemplo de Construcción de Software
ISW – Integrar Software
Objetivos: Agrupar a los componentes de software que satisfacen en conjunto un requerimiento
funcional, con significado y utilidad para el usuario.
1. Unir los componentes de software que permitan demostrar al usuario, resultados significativos (Módulos, Subsistemas, Etc.)
2. Desarrollar plan de pruebas.
3. Elaborar todos los casos de prueba.
4. Desarrollar los procedimientos de prueba.
5. Alimentar la Base de Datos con datos de prueba.
6. Probar los componentes de software integrado (Prueba Integral ejecutada por el equipo de desarrollo.
7. Generar los manuales de instalación y de operación del software desarrollado.

RAS – Revisar y Aceptar
Objetivos: Conseguir del usuario la aprobación del software desarrollado.
1. Desarrollar plan de pruebas con el usuario.
2. Elaborar todos los casos de prueba que cubran las expectativas del usuario.
3. Desarrollar los procedimientos de prueba con visto bueno del usuario.
4. Alimentar la Base de Datos con datos del usuario.
5. Entrenar al usuario en el uso del software.
6. Probar los componentes de software (Prueba ejecutada por el usuario).
7. Generar los manuales de soporte a la operación.
8. Obtener del usuario la aprobación del software.

EDS – Entregar Documentación y Software
Objetivos: Colocar componentes de software en el ambiente tecnológico acordado con el usuario.
1. Instalar el software en el ambiente tecnológico acordado con el usuario.
2. Entregar al usuario la documentación y los componentes de software.

FTP – Formalizar Terminación del Proyecto
Objetivos: Formalizar la terminación del proyecto, asegurando que el producto final cubra fielmente las
necesidades del usuario y que satisface totalmente sus expectativas.
1. Formular un documento en el que se certifica la entrega del producto final y que el usuario firma conciente de que cumple con sus necesidades y expectativas.
2. Centralizar y almacenar los documentos significativos generados durante el desarrollo del proyecto.

PPS – Planear
Objetivos: Asegurar que todos los involucrados en el desarrollo, conozcan su responsabilidad, para
poder cumplir con los objetivos del proyecto y establecer el tamaño, duración y costo del mismo.
1. Realizar estimaciones de duración, gente involucrada, tanto para cada actividad, como para cada producto a entregar.
2. Estimar recursos de cómputo y recursos físicos necesarios.
3. Establecer el plan de trabajo de las actividades a realizar y orientado al equipo de desarrollo.
4. Establecer el plan calendario y entrega de productos para el usuario, y ponerlo visible al equipo de trabajo.
5. Elaborar organigrama del proyecto y afín a los planes presentados.
6. Identificar los factores críticos de éxito y de riesgo para el proyecto.

CPS – Controlar
Objetivos: Establecer los mecanismos que permitan vigilar el desarrollo adecuado del proyecto y
asegurar que los productos del mismo cumplan con las expectativas del usuario.
1. Dar seguimiento al plan de trabajo (actividades y productos)
2. Dar seguimiento al plan de solución de desviaciones.
3. Reportar periódicamente al usuario, del avance y desviaciones del proyecto.
4. Ajustar el plan de trabajo en caso de ser necesario.
5. Modificar la organización del equipo de trabajo si es necesario.
6. Dar seguimiento a los factores críticos de éxito y de riesgo.
7. Detectar nuevos requerimientos.
8. Dar seguimiento a las métricas del proyecto.
9. Analizar métricas.
10. Realizar auditorias aleatorias para verificar que los productos y actividades se encuentran
realmente en el estatus reportado.

ACS – Administrar Configuración
Objetivos: Establecer y mantener la integridad de los productos del proyecto de software, durante todo
el ciclo de vida del proyecto.
1. Definir y comunicar políticas para cambiar componentes del proyecto.
2. Definir ambientes (Trabajo individual, Integración, Producción) mediante alguna estructura (Directorios, Bibliotecas, Etc.) reconocible para el proyecto.
3. Realizar un análisis de impacto en el proyecto, cuando ocurren cambios a los requerimientos del usuario.
4. Controlar las versiones del software y sus correspondientes especificaciones.
5. Custodiar y liberar componentes de software.

ASQ – Asegurar Calidad
Objetivos: Proporcionar una adecuada visibilidad sobre el modelo utilizado en el proyecto de software y
sobre los productos que se están desarrollando.

1. Generar un plan de calidad que contemple las revisiones e inspecciones a llevar a cabo a lo largo del desarrollo del proyecto, así como la identificación de las personas responsables de su ejecución.
2. Revisar que cada uno de los productos desarrollados cumplan con los procedimientos y estándares establecidos.
3. Inspeccionar cada uno de los productos identificados como milestones dentro del modelo, para asegurar que estén claros, completos y correctos.
4. Registrar los resultados de las revisiones e inspecciones.
5. Actuar en consecuencia con los resultados de las revisiones e inspecciones.

GCS – Gestionar Calidad
Objetivos: Desarrollar una disciplina que cuantifique la calidad de los productos del proyecto, en base a
los criterios de calidad definidos para cada uno de ellos.
1. Definir, monitorear y revisar los criterios de calidad (Manual de Calidad) para cada producto del proyecto.
2. Medir, analizar y comparar la calidad de los productos del proyecto contra los criterios de calidad predefinidos.
3. Registrar los resultados de las mediciones de calidad para cada producto del proyecto de software.

MPS – Mejorar Modelo
Objetivos: Mejorar continuamente el modelo utilizado en el proyecto de software, buscando con esto
incrementar la productividad, reducir el tiempo para desarrollar los productos y mejorar la calidad del software.
1. Definir y aplicar un plan de mejora al modelo utilizado en el proyecto de software.
2. Manejar las propuestas de mejora al modelo, mediante un procedimiento documentado.
3. Considerar a los miembros del equipo del proyecto, dentro del programa de mejora.
4. Instalar las mejoras al modelo sobre una base experimental para determinar sus beneficios y eficacia antes de ser puestas en la práctica normal.
5. Registrar las actividades que mejoran el modelo utilizado en el proyecto de software.

UNIDAD 3

PARADIGMA ÁGIL
Implica el desarrollo iterativo (sprint) y la construcción de Prototipos. Se desarrolla cada funcionalidad con alta participación del cliente en las decisiones y definición de comportamiento deseado de la aplicación en desarrollo. • Plan: Determinación y priorización de las funciones. • Diseño: Solución a cada funcionalidad. • Desarrollo y Prueba: Construcción de componentes de software. Aseguramiento del comportamiento y uso adecuado de los recursos computacionales. • Liberación y Feedback: Operación y ajustes de la aplicación.

SCRUM
• Marco de Trabajo Liviano. Basado en Pensamiento Lean.

• Se especifica lo primordial.
• Toma en cuenta el empirismo
• Experiencia.
• Toma de Decisiones.
• Evidencia.

PILARES EMPÍRICOS TRANSPARENCIA
• Visibilidad para facilitar la inspección.
INSPECCIÓN
• Validación con frecuencia para llevar a la adaptación.
ADAPTACIÓN
• Ajustes para evitar desviaciones sobre lo que el Cliente espera recibir.

VALORES DEL SCRUM
Coraje
Respeto
Franqueza
Foco
Compromiso

SPRINT

• Periodo de 2 a 4 semanas (depende de la complejidad)
• Pequeña carrera para tener listas y probadas todas las funciones (historias de Usuario) que el Scrum Team se comprometió a entregar en esas semanas.
• Recomiendo actividades de 4 horas en general para desarrollar funcionalidad.

SCRUM TEAM
Cliente: Recibe un producto qué satisface sus necesidades
Developers: Adaptan el plan para lograr el objetivo del sprint , crean un plan para desarrollar el sprint (Sprint Backlog), Desarrollan el sprint, Responsables de lograr el incremento de valor.
Product Owner: Ordena el Trabajo, Product Backlog, Responsable de que el trabajo se realice.
Scrum Master: Fomenta el uso e implementación de Scrum, Ayuda al Scrum Team en la mejora de sus prácticas.

SCRUM PROCESS
Vision (Cliente) -> User Stories -> Product Backlog (Product Owner) -> Sprint Planning -> Selected Product Backlog -> Sprint Planning (Developers) -> Sprint Backlog -> Sprint (2 - 4 week sprint + 24 hours Daily Scrum) -> New Functionality (Cliente) -> Sprint Review -> Retrospective -> Product Backlog (Se conecta con el primer Product Backlog)

Eventos es sprint: Sprint Retrospective, Sprint planning, Daily scrum, Sprint Review.

SPRINT PLANNING
Se abordan los siguientes temas: • ¿Porqué es valioso este Sprint? • Se define el objetivo. • ¿Qué se puede hacer en este Sprint? • En conjunto con el Product Owner, los Developers

seleccionan elementos del Product Backlog para incluirlos en el Sprint Actual. • ¿Cómo se realizaría el trabajo elegido? • Los Developers planifican el trabajo necesario para crear un Increment Valor que cumpla con la definición de terminado. Sprint Backlog • Objetivo Sprint + Elementos del Product Backlog seleccionados para Sprint + Plan. 8 horas para un Sprint de 1 mes..

DAILY SCRUM
• Evento de 15 minutos al día para Developers.
• Misma hora, mismo lugar.
• Revisan el progreso hacia el objetivo del Sprint.

SPRINT REVIEW
• Sesión de Trabajo.
• El producto se presenta a los interesados clave y se discute el progreso hacia el objetivo del producto.
• Inspeccionan el resultado del Sprint.
• Determinan futuras adaptaciones.
• 4 horas para un Sprint de 1 mes.

SPRINT RETROSPECTIVE
• Lecciones Aprendidas que permitan incluir Propuestas de Mejora para el siguiente Sprint.
• 2 horas para un Sprint de 1 mes.

Artefactos:
Increment: Definición de terminado
Product Backlog: Objetivo del producto
Sprint Backlog: Objetivo del sprint

PRODUCT BACKLOG
• Lista de elementos ordenados (priorizados) que se necesitan desarrollar para obtener un Producto.
• Ejemplo: Reservación, Check In, Check Out, Facturación.
• Refinar el Product Backlog en elementos más pequeños y precisos .
• El objetivo del Producto describe un estado futuro que sirve para que el Scrum Team planifique.

SPRINT BACKLOG
• Se compone de:
• Objetivo del Sprint (Porqué)
• Elementos del Product Backlog seleccionados para el Sprint (Qué)
• Ejemplo: Check Out (Registro de Salida, Generación de Estado de Cuenta, Ajustes al Estado de Cuenta, etc.)
• Plan de Acción para entregar el Increment (Cómo)
• El objetivo del Sprint el único propósito del Sprint.

INCREMENT • Peldaño concreto hacia el Objetivo del Producto. • Debe estar con el estatus de Terminado (Autorizado) • Terminado es el estado que se obtiene cuando cumple con las medidas requeridas por el Cliente.

ENTREVISTA A USUARIOS

¿QUÉ ES UNA ENTREVISTA EN LOS PROYECTOS?
Proceso iterativo de obtención, organización y verificación de definiciones precisas de necesidades que debe satisfacer el sistema computacional a desarrollar.

ACCIONES DIVERSAS EN LA ENTREVISTA CON USUARIOS
Obtener información
Verificar requerimientos
Organizar requerimientos

SECUENCIA DEL PROCESO DE LA ENTREVISTA
1. Definir objetivo de la entrevista.
2. Preparar información para la entrevista.
3. Realizar la entrevista con el usuario definidor para revisar, retroalimentar y corregir la información.
4. Concluir con el usuario definidor acerca de decisiones tomadas durante la entrevista.
5. Documentar las conclusiones de la entrevista en una Minuta de Junta.
6. Incorporar la información obtenida al modelo correspondiente.
7. Verificar la consistencia entre la información nueva y la información existente.
8. Identificar la información faltante dentro de los modelos del sistema para preparar la siguiente entrevista.

Técnica de Estimación y Planeación:
Actividad presente desde el inicio y a lo largo del desarrollo de software.
Propuesta de Negocio. Administración de Requerimientos.
Proceso disciplinado, Documentado, Justificado.
Experiencia y Exposición.

OBJETIVO
Aprender una técnica para estimar y planear Alcance, Esfuerzo, Fecha de Entrega, Costo y Recursos Humanos involucrados en un proyecto de desarrollo de software (llave en mano o a la medida)

TÉCNICA DE ESTIMACIÓN Y PLANEACIÓN
1.- Precisar alcance funcional o Productos. n Requerimientos funcionales.
2.- Determinar modelo de desarrollo de software. n Cascada, Espiral, Incremental, Iteraciones
3.- Establecer categorías de componentes de software. n Tecnología. Horas a aplicar por Categoría..
4.- Calcular horas de construcción. n Alcance. Interpretación hacia componentes de software.

5.- Calcular Horas a aplicar en proyecto. n Modelo de desarrollo de software. Ponderar procesos del modelo. n Calculamos el esfuerzo (horas a aplicar) y el Tiempo calendario.
6.- Determinar Recursos humanos.
7.- Calcular Costo del proyecto. n Opciones: (1) Tarifa promedio. (2) Tarifas por etapas. Dólares.

1- Precisar alcance funcional o Productos
2.- Determinar modelo de desarrollo de software
3.- Establecer categorías de componentes de software
4.- Calcular horas de construcción
5.- Calcular Horas a aplicar en proyecto
6.- Determinar Recursos humanos
7.- Calcular Costo del proyecto

SPRINT PLANNING MEETING

Lleven a cabo una reunión como equipo y establezcan:
1. Objetivo (el qué): se define el objetivo del sprint, definiendo los elementos del Product Backlog que se han seleccionado para trabajar (Sprint Backlog). El Product Owner, es quien presenta el conjunto de características que le gustaría ver completadas en el sprint.
2. Desglose de tareas (el cómo): el equipo determina las tareas necesarias para implementar las características del objetivo.
3. Designación de actividades: durante el sprint planning meeting, también deben quedar claras las asignaciones de cada miembro del equipo.
4. El plan de Sprint: se definen los tiempos de cada tarea, y se acuerda el plan de sprint.
5. Conclusión y compromiso: un buen momento para dar por cerrada la reunión es cuando los miembros del equipo pueden describir el objetivo del sprint y cómo comenzarán a trabajar hacia ese objetivo. Se trata del compromiso del equipo de que está de acuerdo con el tiempo para completar todas las funciones solicitadas y que las comprende profundamente.
6. Evidencia 4. Elabora un Sprint Backlog. Revisa en Blackboard los requisitos de la tarea. Sprint Backlog • Objetivo Sprint + Elementos del Product Backlog seleccionados para Sprint + Plan. 8 horas para un Sprint de 1 mes.

HISTORIAS DE USUARIO
Frase que especifica de manera sencilla un requerimiento desde la perspectiva del propio cliente o usuario final; describe lo que la aplicación va a realizar. Las historias de usuario deben incluir un actor, un objetivo, un contexto, una motivación y una forma de validarla.

MEDICIÓN DEL PROGRESO
El desarrollo e implementación corre por cuenta de los Developers y se debe considerar el progreso de su construcción, bajo los siguientes estatus: Planeada. Cuando las historias han sido seleccionadas para formar parte de un Sprint. Serían parte del Sprint Backlog. Desarrollo. Cuando los Developers las están construyendo. Revisión. Cuando se está en revisión con el cliente y ésta se está ajustando por observaciones. Terminada. Toma este estatus final, sólo si el cliente está satisfecho y da su aprobación de manera explícita; en este momento se dice que la Historia de Usuario proporciona valor (Increment)

PROTOTIPO
• Identifica y entiende en Blackboard el ejemplo de Prototipo.
• Revísenlo con el Profesor y despejen cualquier duda.
• Evidencia 6. Elabora un Prototipo para el Sprint seleccionado. Revisa en Blackboard los requisitos de la tarea.

COMO COMBINAR SCRUM CON XP
Scrum y XP (Programación eXtrema) pueden combinarse de forma práctica.
Scrum se enfoca en las prácticas de organización y gestión, mientras que XP se centra más en las prácticas de programación.
Esa es la razón de que funcionen tan bien juntas: tratan de áreas diferentes y se complementan entre ellas.

CONSTRUCCIÓN DE SOFTWARE
Traducción del diseño a expresiones propias del lenguaje a utilizar en el programa en desarrollo.
Comúnmente para la traducción utilizamos estándares de codificación.
Objetivos de los estándares de codificación:
Facilitar la traducción del diseño a codificación.
Evitar pensar en cosas superfluas.
Limitar el uso del lenguaje.
Pensar en procesos elementales, no en instrucciones.
Establecer el formato de programas.
Establecer nomenclatura de programas.

FORMATO PARA ESTÁNDAR DE CODIFICACIÓN
• Identifica y entiende en Blackboard el formato para escribir estándares de codificación.
• Revísenlo con el Profesor y despejen cualquier duda.

TESTING
En las pruebas se busca que:
• El software haga lo que se supone debe de hacer.
Las pruebas se realizan en varios niveles durante el proceso de desarrollo.
• Prueba Unitaria.
• Pruebas Integrales.
• Pruebas técnicas (volumen, seguridad)
• Pruebas de Aceptación con el Cliente.

Prueba unitaria
Realizar Listado de Condiciones • Desarrollar Casos y Plan de Prueba

LIstado de condiciones
Elaborar Listado de Condiciones (Check List) detallado
• Considerar condiciones normales de acuerdo a los requerimientos a satisfacer y la lógica desarrollada.
• Ejemplo: formato de reporte, tablas cargadas en memoria, alta de un registro cuando el registro ya existe, etc.

• Excluir del check-list las condiciones anormales:
• Son difíciles de probar y menos importantes que la funcionalidad del programa. • Verificar que el check-list esté completo.
• El check-list especifica QUE QUEREMOS PROBAR.
• Para elaborar el check-list pueden tomarse como base:
• Especificación del programa. historias de usuario y prototipo.
• Imagen del reporte o pantalla • Conocimiento de la lógica interna del programa
• Condiciones extremas (primera vez, último registro, etc.)

DESARROLLAR CASOS Y PLAN DE PRUEBA
Definir los casos de prueba que se ejecutarán
• Situación de archivos, condiciones a probar, variantes de la prueba a ejecutar, etc.
• Diseñar los datos requeridos para las pruebas
• Los datos deben ser pensados: ¿qué quiero probar con este dato? ¿qué errores puedo detectar?).
• Determinar los resultados esperados de la prueba
• No necesariamente calcular todos los detalles pero si tener una buena idea de qué esperamos que haga el programa.
• Verificar que el conjunto de los casos de prueba cumplan con el check-list previamente establecido.
• Definir plan de prueba
• Qué se hará en secuencia y qué se hará en paralelo.