Ingeniería de Software, Sem: 2021-1, 3CV3, Tarea 1, 30/09/2020

Tarea 1: Pertinencia del temario

Josué David Hernández Ramírez.

Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
*jhernandezr*1605@*alumno.ipn.mx*

1 Temas

1.1 Ingeniería de Software

1.1.1 Conceptos básicos de ingeniería de software.

¿Qué es software?
Programas de cómputo y su documentación asociada: requerimientos, modelos de diseño y manuales de usuario,puede ser creado desarrollando nuevos programas, configurando sistemas de software genérico o reutilizando software existente.
¿Qué es la ingeniería de software?
Una disciplina de la Ingeniería que concierne a todos los aspectos de la producción de software, utilizando las herramientas y técnicas apropiadas para resolver el problema planteado, de acuerdo a las restricciones de desarrollo y a los recursos disponibles.
¿Qué es un proceso de software?
Un conjunto estructurado de actividades cuya meta es el desarrollo o evolución de un software.
Algunas actividades genéricas en todos los procesos de software son:

1. Especificación: qué debe hacer el software y cuáles son sus especificaciones de desarrollo.

2. Desarrollo: producción del sistema de software Validación, verificar que el software cumple con lo solicitado por el cliente.

3. Evolución: cambiar/adaptar el software a las nuevas demandas.

¿Qué es un modelo de proceso de software?
Representación formal y simplificada de un proceso de software, presentada desde una perspectiva específica.
Modelos Genéricos:

1. Cascada, separar en distintas fases de especificación y desarrollo.
2. Desarrollo Iterativo, la especificación, desarrollo y validación están interrelacionados.
3. Prototipo, un modelo sirve de prototipo para la construcción del sistema final.
4. Basado en componentes, asume que partes del sistema ya existen y se enfoca a su integración.

¿Cuáles son los costos de la ingeniería de software? El costo total de un software esta dividido aproximadamente de la siguiente forma:

- 60% costos de desarrollo
- 40% costos de pruebas
- Los costos dependen del tipo de sistema que se desarrolla y de los requerimientos del mismo tales como desempeño y confiabilidad, la distribución de los costos depende del modelo de desarrollo empleado.

¿Qué es CASE? CASE es Computer-Aided Software Engineering son programas que son usados para dar soporte automatizado a las actividades del proceso de software como:

- Las herramientas CASE son comúnmente usadas para dar soporte a los métodos de software.
- Módulos de análisis que verifican que las reglas del método se cumplan.
- Generadores de reportes que facilitan la creación de la documentación del sistema
- Generadores de código a partir del modelo del sistema.

1.1.2 Atributos y características del software

Características Para poder comprender lo que es el software (y consecuentemente la ingeniería del software), es importante examinar las características del software que lo diferencian de otras cosas que los humanos pueden construir.

1. **El software se desarrolla, no se fabrica en un sentido clásico**.

2. **El software no se estropea**.

3. **Aunque la industria tiende a ensamblar componentes, la mayoría del software se construye a medida**.

Atributos El software debe proveer la funcionalidad y desempeño requeridos por el usuario y debe ser mantenible, confiable, eficiente y aceptable.

- **Mantenible:** El software debe poder evolucionar.
- **Confiable:** No debe causar daños económicos o físicos.
- **Eficiente:** No desperdiciar recursos del sistema.
- **Aceptable:** Los usuarios deben de aceptarlo.
- Debe ser entendible, utilizable y compatible con otros sistemas.

1.1.3 Importancia y aplicación del software

Importancia del Software

Cada software desarrolla funciones específicas dentro de una diversa gama de aplicaciones, y sin duda alguna uno de los programas que mayor utilidad representa dentro de una empresa, son los denominados Sistemas de Soporte a la Decisión (DSS).

De esta manera, la toma de decisiones se convierte en una variable crítica de éxito dentro de las empresas, y es aquí donde radica la importancia de un DSS.

Aplicaciones del Software

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales.

El contenido y el determinismo de la información son factores importantes a considerar para determinar la naturaleza de una aplicación de software. El contenido se refiere al significado y a la forma de la información de entrada y salida. El determinismo de la información se refiere a la predictibilidad del orden y del tiempo de llegada de los datos.

Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales:

- **Software de sistemas:** es un conjunto de programas que han sido escritos para servir a otros programas.
  Algunos programas de sistemas (por ejemplo: compiladores, editores y utilidades de gestión de archivos) procesan estructuras de información complejas pero determinadas.

- **Software de tiempo real:** coordina, analiza, controla sucesos del mundo real conforme ocurren, se denomina de tiempo real.

- **Software de gestión:** Las aplicaciones en esta área re estructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones.
  Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de ventas).

- **Software de ingeniería y científico:** está caracterizado por los algoritmos de manejo de números. Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática.

- **Software empotrado:** reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno de microondas) o suministrar una función significativa y con capacidad de control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, indicadores en el salpicadero, sistemas de frenado, etc.).

- **Sodtware de computadoras personales:** . El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimientos, gestión de bases de datos, etc.

- **Software basado en Web:** Las páginas Web buscadas por un explorador son software que incorpora instrucciones ejecutables y datos.

- **Software de inteligencia artificial:** hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo.

1.1.4 Ciclo de vida del software

Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.

1.1.5 Modelos de procesos

¿Qué es un modelo de proceso de software?
Representación formal y simplificada de un proceso de software, presentada desde una perspectiva específica.
Ejemplos de perspectiva del proceso del hardware.

- Flujo de trabajo, secuencia de actividades.
- Flujo de datos, flujo de la información.
- Rol/acción, quien realiza qué.

Modelos genéricos

1. **Lineal secuencial:** sugiere un enfoque sistemático, secuencial, para el desarrollo del software. Comprende el análisis, diseño, codificación, pruebas y mantenimiento.

2. **Cascada:** Tiene las mismas características que el modelo lineal como se había mencionado anteriormente pero en este se tiene la capacidad de regresar si se detecta un error en cualquiera de las etapas de planeación, desarrollo, diseño, pruebas y mantenimiento del software.

3. **Incremental:** El modelo incremental combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos. El software se ve como una integración de resultados sucesivos obtenidos después de cada interacción

4. **Desarrollo rápido de aplicaciones:** es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto.

   El modelo DRA es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes.

5. **Prototipos:**Comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición.

   El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

6. **Esprial:** La meta del modelo espiral del proceso de producción del software es proporcionar un marco para diseñar tales procesos.

7. **Basados en componentes:** asume que partes del sistema ya existen y se enfoca a su integración.

1.2 Proceso de gestión de proyecto

1.2.1 Ámbito del software

El ámbito del software describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad; Se evalúan las funciones descritas en la declaración del ámbito, y en algunos casos se refinan para dar más detalles antes del comienzo de la estimación.

Dado que las estimaciones del coste y de la planificación temporal están orientadas a la función, muchas veces es útil llegar a un cierto grado de descomposición. Las consideraciones de rendimiento abarcan los requisitos de tiempo de respuesta y de procesamiento.

1.2.2 Estudio de factibilidad

Los autores Putnam y Myers tratan este aspecto cuando escriben que: no todo lo imaginable es factible ni siquiera en el software.
La factibilidad del software tiene cuatro dimensiones sólidas:

- Tecnología ¿Es factible un proyecto técnicamente? ¿Está dentro del estado actual de la técnica?.

- Financiamiento ¿Es factible financieramente? ¿Puede realizarse a un coste asumible por la empresa de software y por el cliente?

- Tiempo ¿Pueden los proyectos adelantarse a los de la competencia?

- Recursos ¿La organización cuenta con los recursos suficientes para tener éxito?

La respuesta es sencilla depende de la experiencia, ya que puede que se haya hecho antes algún proyecto de este tipo o puede que no se tenga experiencia en el proyecto y por lo tanto no son fáciles.

1.2.3 Análisis de riesgo

El riesgo se mide por el grado de incertidumbre en las estimaciones cuantitativas establecidas por recursos, coste y planificación temporal. Si no se entiende bien el ámbito del proyecto o los requisitos del proyecto están sujetos a cambios, la incertidumbre y el riesgo son peligrosamente altos.

Lo que es más importante, el cliente y el panificador, deben tener presente que cualquier cambio en los requisitos del software significa inestabilidad en el coste y en la planificación temporal.

1.2.4 Recursos

La segunda tarea de la planificación del desarrollo de software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de software. Cada recurso queda especificado mediante cuatro características:

Descripción del recurso, informe de disponibilidad, fecha cronológica en la que se requiere el recurso, tiempo durante el que será aplicado el recurso.

1.2.5 Estimación

La estimación del coste y del esfuerzo del software nunca será una ciencia exacta, son demasiadas las variables humanas, técnicas, de entorno, políticas que pueden afectar al coste final del software y al esfuerzo aplicado para desarrollarlo.

1.2.6 Planificación del proyecto

La planificación temporal de un proyecto de software es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto, asignando el esfuerzo a las tareas específicas de la ingeniería del software.

La planificación temporal identifica las principales actividades de la ingeniería de software y las funciones del producto a las que se aplican, se identifican y programan las tareas del software especificas (requeridas para realizar una actividad).

- Calendario de actividades: Designa la programación predeterminada de los trabajos para todos los recursos asignados al proyecto. Puede establecer el calendario del proyecto para indicar un periodo no laborable (como los días festivos de la organización).

- Diagrama de Gantt: Gráfica de Gantt o carta Gantt es una herramienta que permite modelar la planificación de las tareas necesarias para la realización de un proyecto, cuyo objetivo es mostrar el tiempo de dedicación

previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

- Diagrama de Pert: PERT es básicamente un método para analizar las tareas involucradas en completar un proyecto dado, especialmente el tiempo para completar cada tarea, e identificar el tiempo mínimo necesario para completar el proyecto total.

  La parte más famosa de PERT son las Redes PERT, diagramas de líneas de tiempo que se interconectan. PERT está diseñado para proyectos de gran escala, que se ejecutan de una vez, complejos y no rutinarios.

1.2.7 Supervisión y control del plan del proyecto

Al frente de este equipo se situará un director o jefe del proyecto, que será el último responsable de la coordinación del equipo de gestión, y de todos las partes involucradas en el proyecto, así como del control de las actividades, tareas, costes, uso de los recursos... Para poder llevar a cabo estas tareas, es necesario disponer de una correcta programación, así como de las herramientas adecuadas de control.

1.3 Metodologías

1.3.1 Metodologías estructuradas

Son metodologías que se basan en la descomposición de un problema en funciones.
Las metodologías estructuradas se dividen en:

- Metodologías orientadas a procesos.
- Metodologías orientadas a datos.
- Metdologías mixtas.

Merise Las metodología Merise fue desarrollada en 1977 por el ministerio de industria francés. La base de Merise comenzó en 1972 en la universidad de Aix en Provence. Esta metodología esta integrada por: Análisis, Concepción y Gestión de proyectos. La metodología esta conformada por 4 fases:

1. Estudio preliminar: Análisis de la situación. Propuesta de solución global (gestión, organización, decisiones del comité y directivo).
2. Estudio detallado: Análisis del sistema realizar estudios técnicos y presupuestos.

3. Implementación:Solución en un lenguaje de programación. Evaluación de hardware y software. Pruebas.

4. Realización y puesta en marcha: Instalación del sistema desarrollado. Organización del personal por áreas.

Yourdon Esta metodología involucra análisis, desarrollo del diseño y mejora en la medición de la calidad del diseño de software.
Gane-Sarson La métrica Gane Sarson se comienza a utilizar en 1977.
Esta métrica es el resultado de varios años de práctica en la consultoría de análisis y diseño estructurado.
Es creada por la empresa MCAUTO/IST bajo el nombre de STRADIS SDM.
Para el uso y desempeño de esta metodología se utilizan los siguientes 5 pasos:

1. Construir un modelo lógico en curso.

2. Construir un modelo lógico del nuevo sistema lo cual involucra:

   Diagramas de flujo de datos, Diccionario de datos, y especificaciones de los procesos.

   También construir un modelo de datos que exprese en 3era forma normal (3FN) los datos almacenados.

3. Diseñar físicamente la BD.

4. Crear un nuevo modelo físico del sistema.

5. Empaquetar la especificación en subsistemas.

1.3.2 Metodologías Orientadas a objetos

El modelado Orientado a Objetos es una forma de pensar y solucionar problemas usando modelos del mundo real.

Para ello se utilizan objetos que combinan estructuras de datos y comportamientos en una unidad simple (clases con atributos y métodos).

Las metodologías OO se dividen en: “revolucionarias o puras” y “sintetistas o evolutivas”.

OMT
La metodología OMT (Técnica de Modelado de Objetos) es desarrollada por James Rumbaugh y sus colaboradores de General Electric en 1991.
Es una metodología que en su momento gozo de mucho prestigio para el desarrollo de sistemas dentro de la industria.

Ingeniería de Software Orientado a Objetos
Esta metodología es creada por Ivar Jacobson en 1992, la ingeniería de software orientado a objetos por su abreviatura se conoce como OOSE.

Proceso unificado
El Proceso Unificado es un proceso de desarrollo de software definido como: "Conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software".

1.3.3 Proceso unificado racional

Para el proceso unificado de Rational o RUP tenemos que incluimos lo anteriormente visto para el proceso unificado pero se complementa lo siguiente:
El ciclo de vida de esta metodología es la misma que el proceso unificado pero sus iteraciones se realizan bajo algo conocido como disciplinas.
Las disciplinas pueden ser de dos tipos:

- Disciplinas de desarrollo que están formadas por:
  - Ingeniería de negocios.
  - Requerimientos.
  - Análisis y diseño.
  - Implementación.
  - Pruebas.
- Disciplinas de soporte.
  - Configuración y administración del cambio (versiones del proyecto).
  - Administrando horarios y recursos del proyecto.
  - Administrando ambiente de desarrollo.
  - Distribución del proyecto.

Metodologías agiles
El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil del desarrollo de software.
El manifiesto ágil consta de 4 puntos:

- Elegir primero el equipo de desarrolladores que el entorno en el que se va a desarrollar.

- Desarrollar software más funcional que una buena documentación.
- El cliente forma parte del equipo de desarrollo.
- La planificación del software es flexible y abierta para su fácil modificación.

Programación extrema
Creada por Kent Beck famoso ingeniero de software estadounidense y participante del manifiesto ágil.
Esta metodología es conocida como Extreme Programming o XP.
Es la práctica mas utilizada para la creación de software cuando se dispone de un proyecto y equipos pequeños y el plazo de entrega es demasiado limitado.

SCRUM
Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle.
Está especialmente indicada para proyectos con un rápido cambio de requisitos.
El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días.
El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.
Se requiere una reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Otras metodologías

- Crystal Methodologies.
- Dynamic Systems Development Method (DSDM).
- Adaptative Software

1.4 Calidad y normas de calidad

1.4.1 Conceptos de la calidad

- Calidad.
  - Calidad de diseño.
  - Calidad de concordancia.
- Control de calidad.
- Garantía de calidad.
- Coste de calidad.

1.4.2 Calidad de sistemas de información

Los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad.
La ingeniería de software marca las pautas que marcan la calidad en el software.

1.4.3 Calidad del producto de software

La calidad en el desarrollo y el mantenimiento del software se ha convertido en uno de los principales objetivos estratégicos de las organizaciones.

1.4.4 Modelos y normas de calidad

Los sistemas de tecnologías de la información desempeñan un papel crítico en la práctica totalidad de las empresas.
En el desarrollo de software como otros servicios se necesita supervisión constante por parte de profesionales para mantener actualizado y en condiciones de funcionamiento optimas dicho servicio.

- **ISO 25000:** La familia de normas 25000 establece un modelo de calidad para el producto software además de definir la evaluación de la calidad del producto.

- **ISO 9000:** La familia de normas ISO 9000 evalúa Sistemas de Gestión de la calidad eficaz por sus siglas SGCE.
  El objetivo de los SGCE es dirigir y controlar una organización con respecto a la calidad, logrando la realización de todas las actividades y obtener resultados planificados.

- **IEE Std. 1061-1998:** es un estándar para una metodología de métricas de calidad del software.
  Proporciona una metodología para establecer requisitos de calidad e identificar, implementar y validar medidas de calidad del proceso y del producto software.

- **ISO/IEC 15939:** El estándar ISO/IEC 15939 identifica las actividades y las tareas necesarias para aplicar con éxito la medición del software dentro de un proyecto o una estructura de medición organizacional.

1.5 Modelos de madurez

1.5.1 Proceso de Software Personal (PSP)

PSP es una metodología que se concentra en las prácticas de trabajo de los ingenieros del proyecto de manera individual.
La calidad de un software aumenta ya que PSP obliga a que cada uno de los desarrolladores del sistema haga un análisis individual de las tareas y código que le toca realizar en torno a la calidad.

1.5.2 Proceso de Software de Equipo (TSP)

Es una metodología que permite crear software de calidad cuando se tiene un equipo desarrollador generando un entorno donde el trabajo de equipo sea efectivo, normal y natural.
Se basa en procesos estructurados que indican que debe de hacer el equipo en cada fase de desarrollo del proyecto.

1.5.3 Modelo de Capacidad de Madurez (CMM)

Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso (KPA - Key Process Area).

1.5.4 Modelo de Capacidad de Madurez Integrado(CMMI)

CMMI es una metodología formada por:

- Systems engineering (SE).
- Software engineering (SW).
- Integrated product and process development (IPPD).
- Supplier sourcing (SS) .

1.5.5 MoProSoft

El Modelo de Procesos de Software fue desarrollado a solicitud de la Secretaría de Economía para servir de base a la Norma Mexicana para la Industria de Desarrollo y Mantenimiento de Software.

1.6 Temas selectos

1.6.1 Herramientas CASE

Son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de software.

1.6.2 Ingeniería Web

La ingeniería web estudia el desarrollo y la creación de aplicaciones web. Este tipo de aplicaciones tienen atributos diferentes al software general. Existen algunas actividades importantes y que forman parte el proceso, estas son: formulación, planificación análisis, modelización, generación de páginas, test y evaluación del cliente.

1.6.3 Reingeniería

Reingeniería es la revisión fundamental y el rediseño radical de procesos para alcanzar mejoras espectaculares en medidas críticas y contemporáneas de rendimiento, tales como costos, calidad, servicio y rapidez.

Procesos del negocio
La Reingeniería de procesos es la búsqueda e implementación de cambios radicales en el proceso de negocios para lograr un avance significativo.

Del software
Es modificar un producto de software, o de ciertos componentes, utilizando técnicas de Ingeniería Inversa y, para la etapa de reconstrucción, herramientas de Ingeniería Directa, de tal manera que se facilite el cambio y el mantenimiento, reutilización, comprensión o evaluación.

Restructuración
Ingeniería inversa
La ingeniería inversa del software es el proceso de análisis de un programa con el fin de crear una representación de programa con un nivel de abstracción más elevado que el código fuente.

Ingeniería directa
La ingeniería directa, no solamente recupera la información de diseño de un software ya existente, sino que, además, utiliza esta información para alterar o reconstruir el sistema existente en un esfuerzo por mejorar su calidad global.

2 Sobre los temas

La mayoría de los temas que se abordan en el curso de esta unidad de aprendizaje me parecen bien, ya sea porque estan vigentes en la actualidad o por saber la historia de lo que tenemos hoy en día, pero, por otro lado, hay algunos temas que considero que se pueden enfatizar, actualizar o dejar de impartir.

En la primera unidad debería de enfatizar ejemplos de uso de cada modelo de procesos, relacionarlos a un ejemplo práctico, ya sea hipotético o real.

En la segunda unidad me parece que todo esta bien, pero profundizar en el apartEn la unidad tresado de métricas y supervisión y control del plan de proyecto, pero, tambien agregar contenido sobre como dirigir un proyecto, juntar todo y dar de ser posible métricas o métodos de dirección o consejos de como hacerlo.

En la unidad tres, me parece que hay un nuevo paradigma de programación que es la reactiva y esta se usa en lenguajes de programación como JavaScript y Dart entre otros lenguajes de programación empezando a dejar "obsoleta" POO, en dado caso que haya metodologías se agradecería ver algo con ese tema y sobre las que ya estan en el plan, dar mas preferencia a las metodologías ágiles y ver cuales son las que mas se usan al día de hoy y las que se seguirán usando con las nuevas tecnologías.

En la unidad 4 hay una norma que ya no se usa que es la ISO 15939 puesto que quedó obsoleta y agregar algo sobre seguridad informática ya que cada vez va tomando mas relevancia, en este caso creo la ISO 27001.

3 Sobre los libros del curso

Como podemos observar en la tabla 1 la mayoría de los libros siguen disponibles en formato digital, sin embargo la mayoría son del 2005 o anteriores haciéndolos desactualizados a los requisitos que pide la industria hoy en día.

Libro	Año	¿Aun disponible?
Medición y estimación del software: Técnicas y Métodos para mejorar la calidad y la productividad.	2008	Digital
PSP: A Self-Improvement Process for Software Engineers.	2005	Físico y digital
Manual de UML.	2006	Físico y digital
UML 2	2006	Físico y digital
El día a día en los proyectos Software.	2010	Físico y digital
Análisis y diseño de aplicaciones informáticas de gestión. Una perspectiva de Ingeniería del Software.	2004	Físico y digital
Calidad de Sistemas Informáticos.	2007	Digital
Ingeniería del software: Un enfoque Práctico.	2008	Digital
Métodos Ágiles.	2009	Físico y digital
Análisis y diseño orientado a objetos con UML y el proceso unificado.	2005	Digital
Ingeniería de Software	2005	Físico y digital
Análisis de sistemas: diseño y métodos	2008	Físico y digital

Table 1: Tabla de bibliografía en el temario

4 Sugerencia de temas y bibliografía para este curso

Resumiendo sobre sugerencias me permito sugerir la siguiente lista de libros que han sido publicados en fechas más recientes y que tienen buenas reseñas:

- Introducción a la Ingeniería en Sistemas Computacionales y al diseño orientado a objetos, Lopéz Takeyas Bruno, 2019
- Scrum: The Art of Doing Twice the Work in Half the Time. Jeff Sutherland, 2014.
- Kanban in Action, Marcus Hammarberg y Joakin Sundén, 2014.

Y de temas que se pueden agregar:

- Ejemplos reales o hipotpeticos sobre modelos de procesos.
- Programación reactiva.

- Metodología sobre programación reactiva.
- Metodologías ágiles.
- Normas sobre seguridad informática.