Los Sistemas de Información y Las Organizaciones

11056 - Sistemas de Información I

Universidad Nacional de Luján

Departamento de Ciencias Básicas
División Computación
Área Sistemas de Información

Propiedades de los sistemas

Las propiedades de los sistemas, o propiedades emergentes según Sommerville (2002), son los atributos del sistema visto como un todo. Estas propiedades no siempre pueden predecirse, por lo general, solo pueden medirse una vez que se hayan integrado los distintos subsistemas que conformarán el sistema completo.

Clasificaremos a estas propiedades en dos tipos, propiedades funcionales y propiedades no funcionales:

- Las propiedades funcionales son inherentes al objetivo del sistema, visto como un todo dejando de lado las propiedades particulares de cada uno de los elementos que lo componen. Se describen en términos de las capacidades que tiene el sistema en términos de satisfacer necesidades de los usuarios (entendiendo por estos a todos quienes interactúan con el sistema, ya sean personas, máquinas u otros sistemas).
- Las propiedades no funcionales están relacionadas con el comportamiento del sistema en su entorno operacional. Desde este aspecto debe considerarse al sistema operando en su entorno y por lo tanto se considerarán propiedades tales como la fiabilidad, el rendimiento y la seguridad. Estas propiedades aparecen directamente relacionadas a la automatización de los procesos de una organización mediante la utilización de computadoras. Desde otra perspectiva, podríamos decir que las propiedades no funcionales restringen o condicionan la manera en que se deben llevar a cabo las propiedades funcionales.

Tanto las propiedades funcionales, como las no funcionales, de un sistema se definen en la especificación de requerimientos.

Vamos a desarrollar un ejemplo; supongamos que estamos planeando el desarrollo de un nuevo modelo de computadora personal; definiremos, entre alguna de sus propiedades funcionales, que debe tener la propiedad de poder ejecutar diferentes sistemas operativos, que permita la conexión de diversos periféricos mediante diferentes puertos de comunicación, que tenga capacidad para direccionar determinada cantidad de memoria, etc.

Entre sus propiedades no funcionales, definiremos que debe operar en un ambiente que tenga determinadas condiciones de temperatura y humedad, así como estar alimentada eléctricamente por una tensión que se mantenga dentro de determinado rango (190-230 volts). Las propiedades que definamos para este equipo de computación podrán verificarse siempre que se cumpla con las condiciones operacionales que fueron previstas en su construcción, si la temperatura del ambiente es inferior o superior a los valores preestablecidos, posiblemente, el computador comience a mostrar fallas, lo mismo ocurriría si la tensión que recibe no se encuentra dentro del rango que se ha determinado.

Algunas de las propiedades que, generalmente, se requieren de los sistemas son: la fiabilidad, el rendimiento, la usabilidad o la seguridad. Estas propiedades no solo se refieren a un atributo que deberá tener el comportamiento general del sistema, sino al comportamiento que el sistema no debería mostrar.

Un sistema seguro es aquel que no permite accesos no autorizados a sus datos pero todos sabemos que es imposible predecir todos y cada uno de los modos probables de acceso y prohibirlos en forma explícita. Debido a ello, esta propiedad se valora por defecto; sabemos que un sistema es inseguro cuando alguien ha violado su seguridad, mientras tanto, el sistema es considerado seguro y siempre que cumpla además con las medidas de seguridad que se hayan definido.

Esta particularidad se da también con otras propiedades, se las considera alcanzadas hasta tanto no se verifique su carencia.

Los Sistemas y su Entorno

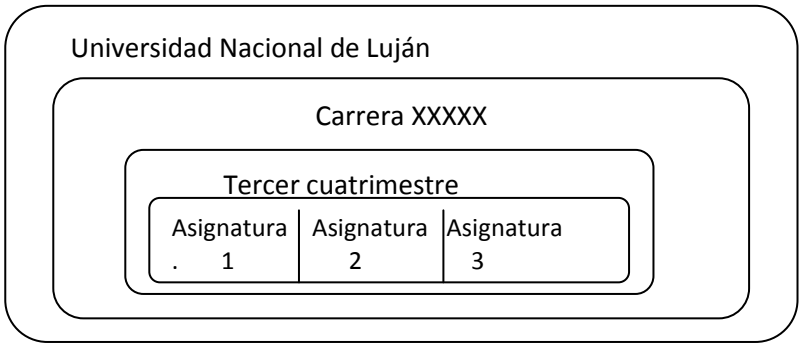
Ya hemos visto, que la Teoría General de Sistemas plantea que una de las características de los sistemas es el **ambiente:** *es todo lo que está fuera del sistema. El ambiente actúa sobre el sistema cuando nos provee insumos (entradas) o cuando recibe sus productos (salidas).*

Por otra parte, hemos definido la **Jerarquía Sistémica** asumiendo para ello que todo sistema se encuentra involucrado en un sistema de nivel superior y a su vez está compuesto por subsistemas.

Llamaremos ahora, a todo lo que se encuentra fuera del límite del sistema su **entorno (o contexto).** El entorno afecta el funcionamiento y rendimiento del sistema. Este puede verse tanto como un sistema por sí mismo o como un cierto número de sistemas que interactúan entre ellos y con nuestro sistema en estudio (aquel en el que se encuentra nuestro foco de atención).

Observemos el siguiente ejemplo:

Sistema de Educación Superior Argentino



En el mismo vemos como en el entorno de la Asignatura 1 se encuentran las restantes asignaturas correspondientes al tercer cuatrimestre de la carrera, que son aquellas que interactúan en el mismo momento en que nosotros estamos fijando nuestro foco de atención en esta asignatura. En otro momento puede ser importante observar la relación de la Asignatura 1 con las asignaturas precorrelativas y postcorrelativas. El entorno del tercer cuatrimestre de la carrera, es la Carrera XXXXX como un todo. Para la carrera, el entorno es la Universidad Nacional de Luján (UNLu), lugar donde se desarrollan las actividades curriculares de la misma; mientras que para la UNLu su entorno es el Sistema de Educación Superior Nacional.

Este ejemplo no es una visión exacta del entorno pues dependiendo del tipo de interacciones que estemos observando aparecerán unos u otros sistemas del entorno interactuando con nuestro sistema. Por otra parte, en este ejemplo se da la coincidencia de que los sistemas que identificamos en el entorno se encuentran involucrados en una jerarquía sistémica. Esto no siempre se da de este modo pues en el entorno del sistema, en el cual hemos fijado el foco de atención, puede haber sistemas interactuando que no correspondan a una misma jerarquía sistémica.

Pensemos, por ejemplo, que también se encuentra en el entorno de cada Asignatura, la oficina de alumnos de la Universidad, pues es la que determina que estudiantes cumplen con las condiciones académicas para poder tomar el curso, la comisión de plan de estudios de la carrera, pues es quien determina el régimen de correlatividades de la carrera y así podemos encontrar otros sistemas que interactúan con la asignatura pero no participan, necesariamente, de la misma jerarquía sistémica.

¿Por qué nuestro interés en el entorno? Existen dos razones por las cuales la Ingeniería de Sistemas considera importante comprender el entorno del sistema en estudio:

1. En muchos casos, el sistema ha sido diseñado para producir cambios en su entorno, por lo tanto, el funcionamiento del sistema solo puede ser evaluado por los efectos que produce en su entorno. Pensemos por ejemplo en un sistema de calefacción cuyo objetivo es mantener la temperatura ambiente cálida. Si el sistema es exitoso en cumplir su objetivo, entonces modificará las condiciones de su entorno. En otras palabras, podemos decir que el sistema es una máquina para convertir energía. Analicemos la implantación del sistema de cajeros automáticos el cual no tiene otro objeto que “hacer de interfaz” entre el sistema bancario y los usuarios del mismo. En este caso, resulta esencial la interacción entre el cajero y el usuario y entre el cajero y el banco en el cual se encuentra radicada la cuenta sobre la cual se realizan las operaciones.
2. El funcionamiento de los sistemas, y principalmente el de los sistemas abiertos, se ve afectado por los cambios en su entorno. Recordemos el ejemplo que dimos de una computadora personal la cual ve alterado su funcionamiento si las condiciones de temperatura o suministro eléctrico de su entorno se modifican más allá de los rangos de tolerancia que fueron previstos.

Ya dimos una idea de que hay distintos entornos y que los identificamos en función del tipo de interacción que estamos observando. Hay entornos del tipo físicos, espacios en los que el sistema debe operar; entornos organizacionales, como el que hemos ejemplificado para una asignatura; entornos económicos y entornos sociales.

Si trabajamos en un sistema de información sin comprender adecuadamente el entorno organizacional en el cual se deberá desempeñar el sistema, es altamente probable que no cumpla con las necesidades propias de la organización y por consiguiente sea rechazado por los usuarios y los directivos de la organización.

Por este motivo, cuando trabajemos con sistemas de información en entornos organizacionales, tendremos que considerar los factores humanos y organizacionales que se

derivan del entorno del sistema y que pueden afectar su diseño. La actividad a llevar a cabo, es verificar si con nuestra intervención o propuesta estamos provocando:

- *Cambios en el proceso*. Si el sistema requiere cambiar los procesos en el entorno, será necesario implementar, con los usuarios del mismo, un fuerte programa de capacitación. Si los cambios son significativos, o implican que la gente pierda su trabajo, existe el peligro de que haya resistencia al sistema. Pensemos que, producto de la implantación del sistema de cajeros automáticos, al mismo tiempo en que se amplió la banda horaria en que los usuarios pueden hacer sus operaciones bancarias, los bancos tienen contratado menos del 50% del personal que necesitarían para atender las cajas si no se hubiera implementado este sistema.
- *Cambios en el trabajo*. Si el sistema inhabilita a los usuarios en su entorno o provoca que se cambie la forma de trabajar, seguramente se resistirán a la introducción del sistema en la organización. Pensemos que cuando se implementa en una organización un sistema que cambia la forma de trabajar; quienes se vean involucrados en el cambio van a resistirse por una simple cuestión de costumbre.
- *Cambios organizacionales.* Es posible que el sistema cambie la estructura de poder de una organización. Si se han automatizado la mayoría de los procesos organizacionales, aquellos que tengan mayor domino del sistema y sepan como operarlo comienzan a tener mayor poder político. Por otra parte la posibilidad de "sociabilizar" la información de la organización hará perder poder a quienes eran considerados "los dueños" de la información.

Considerar estos factores es, generalmente, un aspecto crítico para determinar si un sistema cumple o no de manera adecuada los objetivos que se le han asignado. También debemos señalar que predecir los efectos de estos factores sobre los sistemas es una tarea muy difícil para quienes no tienen experiencia en estudios sociales o culturales, más aún si no tienen experiencia en implantación y puesta en marcha de sistemas computarizados.

Existen numerosos trabajos que estudian los efectos de los sistemas en las organizaciones, Sommerville(2002) propone la lectura de un trabajo de Ackroyd y Harper(1992), *Information Technology and Practical Police Work*. Nosotros proponemos la lectura y discusión del siguiente artículo, con el objetivo de identificar qué tipo de cambios (en los procesos, en el trabajo u organizacionales) fueron tenidos en cuenta o dejados de lado por el equipo de sistemas que tuvo a su cargo el proyecto:

El servicio de ambulancias de Londres

El Servicio de Ambulancias de Londres (SAL) maneja el tráfico de ambulancias en la cuidad de Londres, incluyendo el centro y gran parte de la zona conurbana. Esto cubre un área de cerca de 600 millas cuadradas, cerca de 5000 pacientes por día en 750 vehículos. El SAL recibe cerca de 2000 llamadas por día, incluyendo más de 1300 llamadas de emergencia. El sistema que se discutirá aquí es un sistema despachador [1] auxiliado por computadora (DAC). Tal sistema DAC tiene la siguiente funcionalidad:

- *Maneja las llamadas que recibe, aceptando y verificando los detalles del incidente incluyendo la localización del mismo;*

[1] El término despachador se utiliza en este artículo como "derivador de ambulancias"

- *Determina cual ambulancia debe ser enviada;*
- *Maneja la movilización de la ambulancia y comunica los detalles del incidente a la ambulancia;*
- *Toma en cuenta la administración de los recursos de las ambulancias, en particular la posición actual del vehículo para minimizar los tiempos de respuesta.*

Un sistema DAC completo es algo complejo. En el pánico, alguien podría llamar y decir que el accidente ha sucedido enfrente de Foyle, asumiendo que cualquiera conoce donde está localizada esta librería. Un componente de este sistema es una guía telefónica extensiva que incluye la identificación de los teléfonos públicos para ayudar a resolver este tipo de problema. Pensemos que este sistema fue desarrollado en una época donde los celulares no estaban difundidos y por lo tanto las llamadas a emergencias se hacían desde teléfonos fijos. Siendo así, con una guía reversa, es decir, "dado el número que llama obtener la ubicación", ayudaba a resolver el problema en cuestión. El sistema DAC también contiene un sistema de radio, terminales móviles en las ambulancias y un sistema de localización automática de vehículos. El proyecto DAC del Servicio de Ambulancias de Londres fue iniciado en otoño de 1990. La terminación se contempló para enero de 1992. A esa fecha, sin embargo, el software estaba todavía lejos de estar terminado. En los primeros nueve meses de 1992, el sistema fue instalado pieza por pieza sobre diferentes divisiones del SAL, pero nunca fue estable. El 26 y 27 de octubre de 1992, hubo serios problemas con el sistema y se decidió revertirlo a un modo de operación semiautomático. El 4 de noviembre de 1992 el sistema se vino abajo. La Autoridad Regional de Salud estableció un Equipo Examinador para investigar las fallas y la historia que había llevado a ellas.

Ellos acabaron con un reporte de 80 páginas, el cual se lee como una novela de suspenso. Aquí se subrayarán algunos de los puntos tratados en el reporte.

Ningún servicio de emergencia había intentado llegar tan lejos. El plan fue mover todo el proceso manual – en el cual se llenaban formularios y se transportaban de un empleado al siguiente vía una cinta transportadora – a una automatización completa, en un solo salto. El esquema fue muy ambicioso. Parece ser que los participantes no tenían muy en mente todos los riesgos que estaban tomando.

Mucho antes de que el proyecto comenzara, ya se había cuestionado a la firma consultora administrativa que se había contratado para que trabajara en la selección del contratista. Esta empresa había sugerido que una solución completa podría costar 1.5M de libras y tomar 19 meses. También habían indicado en su informe que si no se encontraba una solución empaquetada (producto ya construido), los costos estimados se incrementarían significativamente. De todas maneras, se escogió una solución no empaquetada (a medida), y solo se tuvieron en cuenta los aspectos económicos de este informe, o eso es lo que parece.

La licitación obtuvo propuestas de 35 compañías. Las especificaciones y el calendario de ejecución fueron discutidos con esas compañías. El calendario propuesto fue de 11 meses. Y aunque muchos proveedores plantearon cuestionamientos sobre lo ajustado del plan, les fue dicho que eso no era negociable. Solo 17 proveedores entregaron propuestas completas. Se seleccionó la oferta de menor costo, cerca de 1M de libras. Esta oferta fue de cerca de ***700,000 libras más barata que la siguiente oferta más barata****. Nadie se cuestionó esta enorme diferencia. La propuesta seleccionada superficialmente sugería que la compañía ya tenía experiencia en diseño de sistemas para servicios de emergencia. Esto no era mentira: ellos ya habían desarrollado*

sistemas administrativos para tales servicios pero el sistema SAL fue mucho más grande que cualquier proyecto que ellos hubiesen manejado antes.
El sistema propuesto impactaba significativamente en la forma en que la tripulación de las ambulancias llevaba a cabo su trabajo. Este debía ser por lo tanto el aspecto fundamental a tener en cuenta en la implantación del proyecto. Si la tripulación no presionaba los botones correctos, en la forma correcta, en el tiempo preciso y en el orden correcto, podría resultar un caos. Aún así, hubo muy poco involucramiento de los usuarios durante el proceso de ingeniería de requerimientos.
El sistema DAC proyectado operaría automáticamente y movilizaría los recursos óptimos (según su criterio) para cualquier incidente.
Esto modificaba muchas de las prácticas de trabajo presentes, las cuales la administración consideraba anticuadas y de poco interés para el SAL. Por ejemplo, el nuevo sistema localizaría los recursos disponibles más cercanos de la estación de origen. Con esto, podría ocurrir el siguiente escenario:

- *La tripulación de John tiene que ir a un accidente a unas pocas millas al este de su base.*
- *Una vez allí, son dirigidos a un hospital a unas pocas millas más al este para dejar al paciente.*
- *Si llegan otras llamadas y parece que John es el más cercano, a él se le ordena viajar unas pocas millas más al este.*
- *Y así sucesivamente.*

De esta forma, la tripulación podría operar cada vez más lejos de su base y en un territorio que no les es familiar (una de las características de Londres es que hay dos regiones y los que viven en una de ellas desconocen la otra). Con esto, se pierde tiempo por tomar caminos equivocados, o mas aún por parar para preguntar por las direcciones. Además, al finalizar su jornada tendría que viajar más para volver a su base. A la tripulación no le gustó este aspecto del nuevo sistema.
El nuevo sistema tampoco tomaba en cuenta la flexibilidad de las estaciones de emergencia locales, al decidir cuales recursos asignaba. En el nuevo esquema, la administración de recursos fue centralizada completamente y manejada por el sistema. De esta forma, supongamos que John corría hacia donde las ambulancias estaban estacionadas y la computadora había ordenado tomar el coche número 5. John estaba apurado y posiblemente podría no reconocer el coche número 5, o quizás éste está estacionado detrás de otro coche y era muy difícil sacarlo. De manera que John podía perder su paciencia y decidir tomar el coche número 4. Esto significaba un problema: el sistema asignaría el vehículo número 4 a otro accidente y este ya no se encontraría en el estacionamiento. Por otro lado, el coche 5 no se movía, por lo tanto el sistema podría ordenar a otra ambulancia concurrir al accidente y de esta forma dos ambulancias irían al mismo lado.
La gente responsable de estos requerimientos tenía mal juicio o ingenuidad sobre lo que los sistemas computarizados podrían brindar en las prácticas de los humanos. Las computadoras están aquí para ayudar a la gente a hacer su trabajo, y no viceversa. Cuando los sistemas generan resistencia en quienes lo deben operar están condenados al fracaso.
La caída del sistema el 4 de noviembre de 1992 fue causada por un error de programación menor. Unas tres semanas antes, un programador que había estado trabajando en una parte del sistema olvidó remover una pequeña parte del código del programa. El código en sí mismo no era dañino. Sin embargo, este ocupaba una pequeña cantidad de memoria cada vez que el sistema generaba una movilización de un vehículo. Esta memoria no era liberada y el sistema funcionaba sin interrupción durante las 24 horas del día. Después de tres semanas, se consumió toda la memoria y el sistema se cayó.

El proyecto del SAL no falló debido a ese error de programación. Este solo fue la última gota que derramó el vaso. La previsión del proyecto fue demasiado rígida. La administración tanto del Servicio de Ambulancias de Londres como del contratista o empresa contratada tenían muy poca o ninguna experiencia en desarrollo de proyectos de software de este tamaño y complejidad. Fueron demasiado optimistas en su análisis de riesgos.
Asumieron que toda la gente podría interactuar con el sistema y podrían hacerlo de la forma correcta, todo el tiempo. La administración decidió sobre la funcionalidad del sistema, sin realizar ninguna consulta con quienes serían sus usuarios primarios. Cualquier proyecto de tales características está condenado a fallar, desde el mismísimo primer día.

De manera ideal, podemos decir que todo el conocimiento relevante del entorno deberá incluirse en la especificación del sistema, de manera tal que pueda ser tenido en cuenta por los diseñadores del sistema. Los diseñadores del sistema solo deberán hacer suposiciones sobre el entorno basándose en otros sistemas comparables y en el sentido común.

Estructura Organizativa y Sistemas de Información

El análisis y diseño de sistemas de información no es una actividad que pueda desarrollarse desvinculada del contexto mayor con el que interactúa: la estructura organizativa sobre la cual se implantan los sistemas.

Este vínculo indisoluble hace que debamos estudiar tanto las técnicas de análisis de sistemas como las de análisis estructural. El análisis estructural es el estudio de la forma en que se dividen las tareas en una organización, la asignación de funciones y responsabilidades entre sus miembros y las relaciones jerárquicas entre los mismos. El propósito de este tipo de análisis es verificar la estructura de la organización para adecuarla a los objetivos corporativos y al mantenimiento adecuado del control interno.

Desde el punto de vista sistémico, una organización puede verse como un sistema de mayor jerarquía compuesto por subsistemas que deben mantenerse interrelacionados entre sí. Las diversas funciones que se cumplen en la organización se integran a través de tecnología informática que permite formar un todo organizacional efectivo.

Para efectuar un correcto análisis de los requerimientos de información y para desarrollar sistemas de información que satisfagan esos requerimientos, es fundamental comprender el concepto de organización como un todo integrado. Lardent(2001) plantea que al construir un sistema de información es necesario interpretar la relación entrada/salida para crear las interfaces que relacionan un sistema con otro; una salida de información del subsistema ventas se convierte en entrada del subsistema de producción (para programar la producción); una salida del sistema de producción se convierte en una entrada del subsistema ventas (para conocer la disponibilidad). De esta manera debería poder describirse todas las relaciones que existen entre los subsistemas de la organización encontrando la dependencia funcional que existe entre un subsistema y otro.

Operaciones de una empresa

Las operaciones de Comprar, Pagar, Producir, Vender y Cobrar resultan insustituibles y solo puede estar ausente la tercera, en el caso de empresas que se dediquen a la compra / venta de productos terminados. A estas operaciones las denominaremos operaciones primarias. Estas operaciones primarias son las que sostienen el cumplimiento de los objetivos de la organización.

De todas maneras, también existen operaciones que tienen por finalidad mejorar el desempeño de las operaciones primarias o implementar controles para el logro más eficiente de las operaciones primarias. A estas operaciones se las denomina operaciones secundarias, las más destacadas son:

- Administración general (contabilidad, asistencia, liquidación de haberes, etc)
- Financieras (bancos, flujo de caja, créditos, cobranzas, etc)
- Conservación de activos (reparaciones, inversiones en bienes, mantenimiento de vehículos, etc)

Estas operaciones pueden ser vistas como algunos de los subsistemas que se pueden identificar en la empresa.

Control de operaciones

Se debe procurar que la información que brinda la contabilidad esté exenta de errores o deformaciones. Si los controles no son eficaces podrían deslizarse errores o fraudes. La función administrativa del control es la medida y la corrección del desempeño de las actividades de los subordinados para asegurar que, los objetivos y planes de la empresa diseñados para conseguir los objetivos, se están llevando a cabo.

Las operaciones básicas por el hecho de ser frecuentes y repetitivas permiten ser normalizadas. La normalización consiste en programas que indican la forma más adecuada de realizar las operaciones, y se materializa en los manuales de procedimientos o de trámites y en los circuitos administrativos o **cursogramas**. Los manuales de procedimientos establecen en detalle cuáles son las operaciones y en que secuencia se deben presentar para llevar a cabo una determinada operación. Al mejorar el programa, lo que corresponde es la optimización del procedimiento y la posterior comunicación de las instrucciones al personal afectado.

El control interno es el encargado de garantizar que las operaciones que reflejan los registros hablan de la verdadera situación del negocio, que se minimicen las posibilidades de errores y fraude, y que se respeten los procedimientos que se han implantado a través de manuales de instrucciones.

El propósito es**:**

- la prevención de errores y fraudes en la ejecución de las operaciones,
- garantizar las registraciones que habrán de servir de fuente para los informes y estados provenientes de la contabilidad,
- procurar detectar cualquier desvío sobre los procedimientos estipulados y
- en general ejercer un control efectivo sobre todos los aspectos del negocio.

La base del control interno es la **división del trabajo**, una operación básica que puede ser subdividida en varias partes, nunca debe ser realizada por una sola persona. Para que esa complementación pueda llevarse a cabo será necesario programar la operación básica instruyendo y capacitando al personal para su ejecución. Será factible establecer un **control por oposición** o automático entre las personas. Al haber mayor control sobre el desarrollo de las operaciones, se intensifica el control sobre las registraciones y los datos que pueden extraerse de los registros serán más confiables.

Es necesario que se instrumente un sistema de información entre los distintos participantes, para que cada uno conozca cuándo le corresponde actuar, cuáles son las operaciones que ya se han cumplido y cuál fue el resultado de las mismas, y a la vez pueda indicar cuando concluye su parte y cuál es la información que produjo. Esta información se materializa en comprobantes básicos y en los registros previstos.

Un sistema de control interno no puede garantizar totalmente la detección de errores, fraudes o desvíos de los procedimientos administrativos, ni tampoco la exactitud de los informes. En algunos casos no resulta posible aplicar el control interno en forma estricta, por resultar antieconómico en estos casos; se suele aplicar un control directo más intenso. (Ej. Arqueo físico de inventario en un hipermercado)

Un ejemplo de norma de Control Interno es determinar que los cheques que se libran para el pago a proveedores deber ser firmados por dos personas de la organización que trabajen en distintos sectores. Por ejemplo alguien de pagos y alguien de administración. De esta manera, la segunda firma sirve de control respecto de que el monto que se está pagando se corresponde con los comprobantes que dan sustento al trámite.

Automatización de los sistemas organizacionales

El estudio de los sistemas de información en una organización, generalmente, se lleva a cabo por los siguientes motivos:

- Identificar el origen de problemas de control, performance, eficiencia, seguridad, etc. con la finalidad de proponer soluciones y mejoras.
- Automatización de actividades repetitivas que se están ejecutando en forma manual.
- Mejorar el desempeño del sistema actual en términos de performance, control de operaciones, información que produce, etc.
- Automatización de los circuitos de información, con la finalidad de mejorar los procesos de negocio y disminuir los tiempos muertos.

Estos procesos de automatización han ido evolucionando de la misma manera que lo hizo el hardware y el software, acompañado por una baja de los costos de estos elementos que resultan esenciales para la automatización.

En términos de la pirámide organizacional, una organización debe, necesariamente, comenzar por la automatización de sus sistemas transaccionales u operacionales para ir evolucionando hacia los niveles superiores de la pirámide mediante sistemas de apoyo a las decisiones hasta llegar a sistemas de información ejecutiva. Este proceso de automatización hace que los sistemas van disminuyendo a cantidad de información que administran para pasar a administrar conocimiento organizacional.

Ingeniería de Sistemas

La ingeniería de Sistemas es una actividad interdisciplinaria, que conjuga equipos de personas con diferente formación y bases de conocimiento. Estos equipos interdisciplinarios son imprescindibles debido al amplio conocimiento necesario para poder considerar todos los aspectos, cuando se toman decisiones relativas al diseño de sistemas.

Las diferentes disciplinas de la ingeniería deben negociar para decidir qué funcionalidad debe proporcionarse. En general no existe una única decisión al respecto, sino un rango de alternativas posibles.

Fases del Proceso de la Ingeniería de Sistemas

El Proceso de Ingeniería de sistemas consta de las siguientes fases: Definición de requerimientos, diseño del sistema, desarrollo del subsistema, integración del sistema, instalación del sistema, evolución del sistema, y desmantelamiento del sistema.

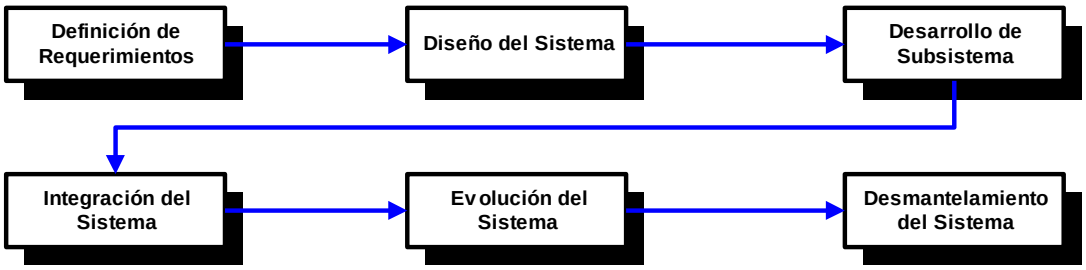


Figura 1: Fases de un Proceso de Ingeniería de Sistemas

Definición de requerimientos del sistema

La actividad de definición de requerimientos del sistema pretende descubrir los requerimientos completos del sistema propuesto. Esta fase usualmente se concentra en tres tipos de requerimientos: requerimientos funcionales abstractos, propiedades del sistema y características que no debe mostrar el sistema.

- **Requerimientos funcionales abstractos:** son las funciones básicas que el sistema debe proporcionar definidas en un nivel abstracto. La especificación detallada de los mismos tiene lugar en el nivel de subsistemas.
- **Propiedades del sistema:** estas son propiedades emergentes no funcionales del sistema. Incluyen propiedades como la disponibilidad, la protección, el rendimiento, etc. Las mismas afectan los requerimientos para todos los subsistemas.
- **Características que no deben mostrar los sistemas:** algunas veces tiene igual importancia especificar lo que el sistema debe y no debe hacer.

Una parte importante de la fase de definición de requerimientos es establecer un conjunto completo de objetivos que el sistema debe cumplir. Este no necesariamente debe expresarse en términos de funcionalidad del sistema, pero debe definirse el porqué se construye el sistema para un entorno particular.

Una dificultad fundamental al establecer los requerimientos del sistema es que los problemas para los cuales se construyen los sistemas complejos son normalmente "problemas traviesos", entendiéndose por ello, a problemas complejos, donde hay demasiadas entidades relacionadas, que no existe una especificación definitiva del problema. La verdadera naturaleza de este emerge solo cuando se desarrolla una solución.

Un ejemplo extremo de un "problema travieso" es la previsión de terremotos, ya que nadie puede predecir en forma precisa, dónde será, en qué momento ocurrirá y qué consecuencias traerá aparejadas. Por lo tanto, no es posible especificar completamente cómo actuar ante un terremoto. El problema solo puede abordarse una vez que ha pasado.

Diseño del sistema

En el diseño, la idea se centra en proporcionar la funcionalidad del sistema a través de sus diferentes componentes. Las actividades que se realizan en este proceso son: dividir requerimientos, identificar subsistemas, asignar requerimientos a los subsistemas, especificar la funcionalidad de los subsistemas, y definir las interfaces de los subsistemas.

- **Dividir requerimientos:** los requerimientos se analizan y recolectan en grupos relacionados, existiendo normalmente varias opciones posibles de división.
- **Identificar subsistemas:** se identifican los subsistemas que individual o colectivamente pueden cumplir con los requerimientos. Los grupos de requerimientos normalmente están relacionados con los subsistemas, de forma tal que esta actividad y la de partición de requerimientos se ven disminuidas.
- **Asignar requerimientos a los subsistemas:** los requerimientos son asignados a los subsistemas. Debería ser un paso directo si la partición de requerimientos se utiliza para la identificación de subsistemas. En la práctica las limitaciones de los subsistemas comerciales implican que los requerimientos deben modificarse.
- **Especificar la funcionalidad de los subsistemas:** se deben enumerar las funciones específicas asignadas a cada subsistema. Esto puede verse como parte de la fase de diseño o bien como parte de la actividad de especificación de requerimientos del sistema, si el mismo es un sistema de software.
- **Definir las interfaces del subsistema:** esto comprende definir las interfaces necesarias y requeridas por cada subsistema. Una vez que las interfaces se han definido, es posible el desarrollo paralelo de los subsistemas.

En este proceso de diseño existe un compromiso de retroalimentación e iteración de una etapa a la otra.

Para la mayoría de los sistemas existen muchos diseños posibles que se pueden desarrollar. Estos cubren un amplio rango de soluciones con combinaciones diferentes de hardware, software y operaciones humanas. La solución elegida para el desarrollo futuro deberá ser la solución técnica más apropiada que cumpla los requerimientos. Sin embargo en muchos casos las intervenciones organizacionales y políticas influyen en la elección de la solución.

Desarrollo de los subsistemas

Durante el desarrollo de los subsistemas se implementan los que se hayan identificado durante el diseño del sistema. Ocasionalmente, el proceso de desarrollo construirá todos los

subsistemas desde sus inicios. Sin embargo, algunos de ellos son comerciales, los cuales se compran para integrarse en el sistema. Normalmente es mucho más barato comprar productos existentes que desarrollar componentes de propósito específico.

Integración del sistema

Este proceso consiste en tomar subsistemas desarrollados de forma independiente y juntarlos para crear un sistema completo. La integración se puede llevar a cabo utilizando un enfoque del tipo "*big bang*" que consiste en integrar todos los sistemas al mismo tiempo. Sin embargo, por razones tanto técnicas como de administración, el mejor enfoque es un proceso de integración creciente, donde los sistemas se integran uno a uno.

Es el enfoque más apropiado por dos razones:

- Por lo general, es imposible calendarizar todos los desarrollos de los diferentes subsistemas de tal forma que estos se completen al mismo tiempo.
- La integración creciente reduce el costo en la localización de errores.

Instalación del sistema

Durante la instalación del sistema, este se ubica en el entorno en el cual se pretende que opere. Aunque puede parecer un proceso sencillo, suelen surgir muchos problemas que implican que la instalación de un sistema complejo pueda tardar meses o años.

Algunos de los problemas que habitualmente surgen son:

- El entorno en el cual el sistema se instala no es el mismo que el supuesto por los desarrolladores. Cuando es un sistema de software, puede utilizar funciones provistas para una versión específica del sistema operativo.
- Los usuarios potenciales del sistema pueden ser hostiles a su introducción, ya que puede reducir su responsabilidad y el número de empleos en la organización.
- Un nuevo sistema puede convivir con uno existente hasta que la organización esté satisfecha con el funcionamiento del nuevo sistema. Esto provoca problemas de instalación particulares si los sistemas no son completamente independientes sino que comparten algunos componentes. Puede ser imposible instalar un nuevo sistema sin desmontar el anterior.
- Puede haber muchos problemas en la instalación física, debido a que no exista espacio suficiente en los conductos de los cables de la red, o que se necesite aire acondicionado, o que el edificio sea histórico y por lo tanto las modificaciones edilicias estén totalmente prohibidas.

Operación del sistema

Una vez que el sistema ha sido instalado, se pone en operación. Operar el sistema implica entrenar a los usuarios y cambiar el proceso normal de trabajo, para que la utilización del nuevo sistema sea efectiva. Pueden detectarse en esta etapa errores, ya que el sistema puede cumplir con las especificaciones, pero no obstante eso, sus funciones pueden no cumplir con las necesidades reales. A menudo es difícil operar un nuevo sistema con sistemas existentes, debido a que la transferencia de datos puede dificultarse, o bien**,** si las interfaces a las que los usuarios están acostumbrados son diferentes, esto puede ocasionar más cantidad de errores de operación.

Evolución del sistema

Todos los sistemas precisan ir evolucionando con el tiempo, para corregir errores en los requerimientos originales y para cumplir con nuevos requerimientos.

A continuación mencionaremos algunas razones por las cuales la evolución del sistema es costosa:

- Los cambios propuestos deben analizarse cuidadosamente, y deben ser aprobados por un grupo de personas.
- Debido a que los subsistemas nunca son completamente independientes, el cambio en alguno de ellos, puede afectar el funcionamiento del resto. Lo que nos conduce a tener que cambiarlos a estos también.

Desmantelamiento del sistema

El desmantelamiento del sistema, implica poner fuera de servicio dicho sistema una vez que su vida útil termina. Algunas veces esto es directo, sin embargo algunos sistemas pueden contener materiales que son potencialmente peligrosos para el entorno. Pensemos en el desmantelamiento de alguno de los sistemas de una central nuclear.

Generalmente, la organización desea conservar los datos del sistema que se está desmantelando, para ello estos deben convertirse para ser utilizados en otros sistemas.

Procesos del Software

El proceso del software es un conjunto de actividades y resultados asociados que conducen a la creación de un producto de software. Puede consistir en el desarrollo de software desde cero, o la ampliación y modificación de sistemas existentes.

Los procesos de software son complejos, y como todo proceso intelectual se basa en el juicio humano. Los intentos para automatizar estos procesos han tenido un éxito limitado. Las herramientas CASE[2] pueden ayudar a algunas actividades del proceso. No existe un proceso ideal. Aunque existen muchos procesos diferentes de software, todos tienen actividades fundamentales que les son comunes, como ser:

- **Especificación del software:** se debe definir la funcionalidad del software y las restricciones en sus operaciones.
- **Diseño e implementación del software:** se debe producir software que cumpla con su especificación.
- **Validación del software:** se debe validar el software para asegurar que hace lo que el cliente quiere que haga.
- **Evolución del software:** el software debe evolucionar para cumplir los cambios en las necesidades del usuario.

[2] CASE (*Computer Aided/Assisted Software/System Engeneering)*, herramientas y metodologías que soportan un enfoque de ingeniería en el desarrollo de software para todas las fases de este proceso.

Modelos de Procesos del Software

Un modelo del proceso de software es una representación abstracta de la manera en que se lleva a cabo la actividad de construir un producto de software. Cada modelo representa un proceso desde una perspectiva particular, por lo que solo provee información parcial acerca del mismo.

Vamos a introducir varios modelos de procesos, muy generales (muchas veces se los conoce como paradigmas de proceso). No son descripciones definitivas de los procesos de software, más bien son abstracciones útiles que se pueden utilizar para explicar diferentes enfoques para desarrollar software.

Los modelos de procesos que presentamos son:

- **El modelo de cascada:** toma las actividades fundamentales del proceso de especificación, desarrollo, validación, y evolución y los representa como fases separadas del proceso.
- **Desarrollo evolutivo:** este enfoque entrelaza las actividades de especificación, desarrollo y validación. Se desarrolla un primer sistema con las especificaciones abstractas y luego se refina con la ayuda del cliente, para producir un software que satisfaga sus necesidades.
- **Desarrollo formal de sistemas:** este enfoque se basa en la producción de una especificación matemática formal del sistema y en la transformación de esta especificación utilizando métodos matemáticos, para construir un programa.
- **Desarrollo basado en la reutilización:** este enfoque se basa en la existencia de un número significativo de componentes reutilizables.

El modelo de "cascada"

Este modelo se derivó de otros procesos de ingeniería. Debido a la cascada de una fase a otra, este modelo se conoce como "modelo de cascada" o como ciclo de vida del software. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo.

- **Análisis y Definición de Requerimientos:** los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios.
- **Diseño de sistemas y de software:** el proceso de diseño de sistemas divide los requerimientos en sistemas de hardware o de software.
- **Implementación y prueba de unidades:** durante esta etapa, el diseño de software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.
- **Integración y prueba del sistema:** los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software.

- **Operación y mantenimiento:** el sistema se instala y se pone en uso práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

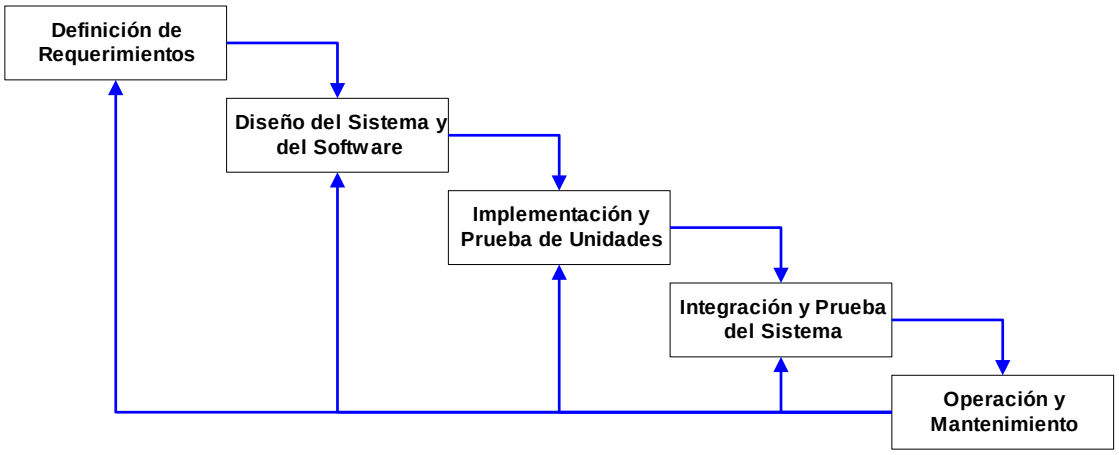


Figura 2: El Ciclo de vida del software

El resultado de cada fase es uno o más documentos aprobados. Es necesario terminar cada fase para iniciar la siguiente. El proceso del software no es un modelo lineal simple, sino que implica una serie de iteraciones de las actividades de desarrollo. Después de un número reducido de iteraciones, es normal congelar partes del desarrollo, como la especificación y continuar con las siguientes etapas.

Durante la fase final del ciclo de vida (operación y mantenimiento) el software se pone en funcionamiento, se detectan errores y omisiones de diseño y programación y se identifica la necesidad de una nueva funcionalidad. Por lo tanto el sistema, debe evolucionar para mantenerse útil. Estos cambios implican repetir algunas o todas las etapas previas del proceso.

El problema con el modelo en cascada es su inflexibilidad, al dividir el proyecto en estas etapas, es difícil responder a los cambios en los requerimientos del cliente, ya que los compromisos se adquirieron en las etapas iniciales.

Es recomendable utilizar este tipo de modelo solo cuando los requerimientos se comprenden en su totalidad y en el caso de sistemas o subsistemas simples.

Desarrollo Evolutivo

Se basa en la idea de desarrollar una implementación inicial, exponiéndola a las sugerencias y comentarios de los usuarios, para de esta forma ir refinándola a través de diferentes versiones hasta llegar a una versión definitiva. Más que tener actividades separadas de especificación, desarrollo y validación, se llevan a cabo concurrentemente a lo largo del proceso.

Existen dos tipos de desarrollo evolutivo:

- **Desarrollo exploratorio:** el objetivo es trabajar con el cliente, a efectos de descubrir sus requerimientos. El desarrollo comienza con las partes que más se comprenden y se le van agregando nuevos atributos acorde con las propuestas del cliente.
- **Prototipos desechables**: el prototipo se centra en experimentar con aquellas partes de los requerimientos del cliente que no se comprenden del todo.

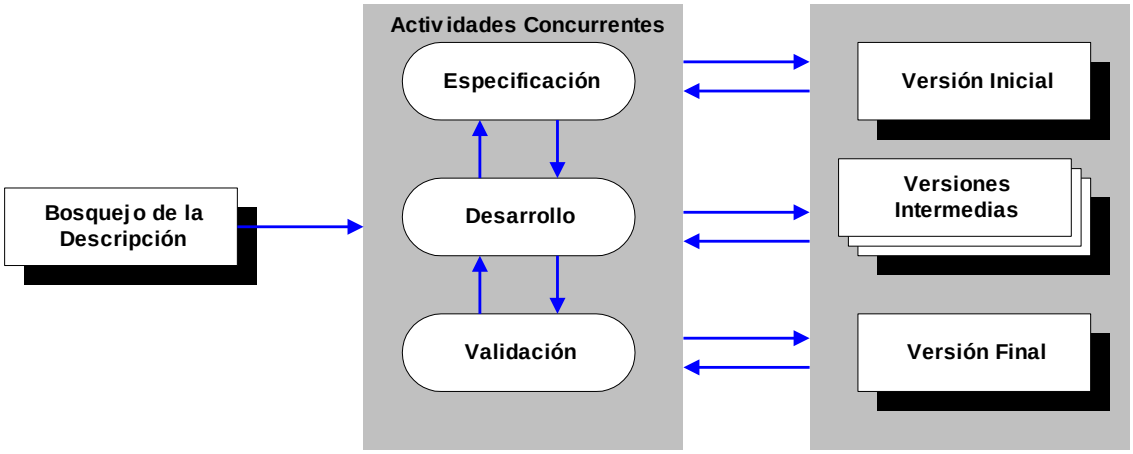


Figura 3 : Desarrollo Evolutivo

La ventaja de un proceso del software que se basa en un enfoque evolutivo, es que la especificación se puede desarrollar en forma creciente. Sin embargo, desde el punto de vista de la ingeniería y de la administración, hay tres problemas:

- **El proceso no es visible:** los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, es muy costoso producir documentos que reflejen cada versión.
- **A menudo los sistemas tienen una estructura deficiente:** los cambios continuos tienden a corromper la estructura del software, lo que hace que incorporar cambios resulte una tarea difícil y costosa.
- **Se requieren herramientas y técnicas especiales:** éstas permiten un desarrollo rápido, y muchas veces pocas personas tienen habilidades para utilizarlas.

Para sistemas pequeños a medianos (hasta 500.000 líneas de código), el enfoque evolutivo resulta ser el mejor. Sin embargo para sistemas grandes, se recomienda un proceso mixto, que incorpore lo mejor del modelo de cascada y las mejores características del modelo evolutivo.

Desarrollo Formal de sistemas

Este enfoque, tiene algo en común con el modelo de cascada, pero el proceso de desarrollo se basa en la transformación matemática formal de una especificación del sistema en un programa ejecutable.

La diferencias entre este modelo y el Cascada son :

1. La especificación de requerimientos de software se refinan en una especificación formal detallada que se expresa en notación matemática.

2. Los procesos de desarrollo de diseño, implementación y prueba de unidades se reemplazan con un proceso de desarrollo de transformaciones donde la especificación formal inicial, se refina sucesivamente a través de una serie de transformaciones, hasta llegar a un programa (figura 5).

En el proceso de transformación, la representación matemática formal del sistema se convierte de forma sistemática en una representación más detallada del sistema, pero matemáticamente correcta. Cada paso agrega detalle hasta que la especificación formal se convierte en un programa equivalente.

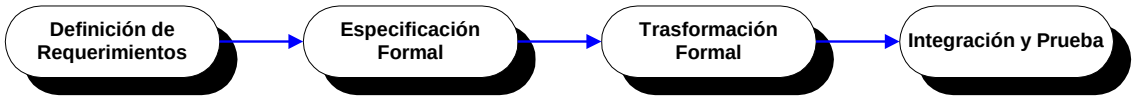


Figura 4 : Desarrollo Formal de Sistemas

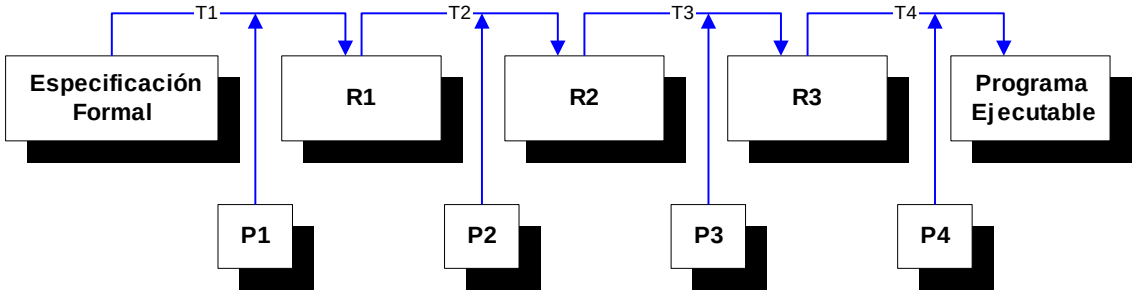


Figura 5 : Transformación Formal

La ventaja del enfoque por transformaciones, comparado con la demostración de que un programa cumpla su especificación, es que la distancia entre cada transformación es menor que la distancia entre la especificación y el programa. Un enfoque por transformaciones compuesto por una serie de pasos pequeños es más adecuado. Sin embargo, se requieren ciertas habilidades para saber qué transformación aplicar.

Fuera de los dominios especializados, los procesos que se basan en transformaciones formales no se utilizan ampliamente. Requieren de un experto y en realidad para la mayoría de los sistemas, este proceso no ofrece ventajas significativas sobre los demás enfoques, en aspectos como costos y calidad.

Desarrollo orientado a la reutilización

En la mayoría de los proyectos de software existe algo de reutilización de software. Por lo general, se buscan códigos similares, se modifican acorde a lo solicitado y se incorporan al sistema. Esta reutilización informal es independiente del proceso genérico que se utilice, sin embargo en los últimos años ha surgido un enfoque de desarrollo de software que se basa en la reutilización (ingeniería de software basada en componentes).

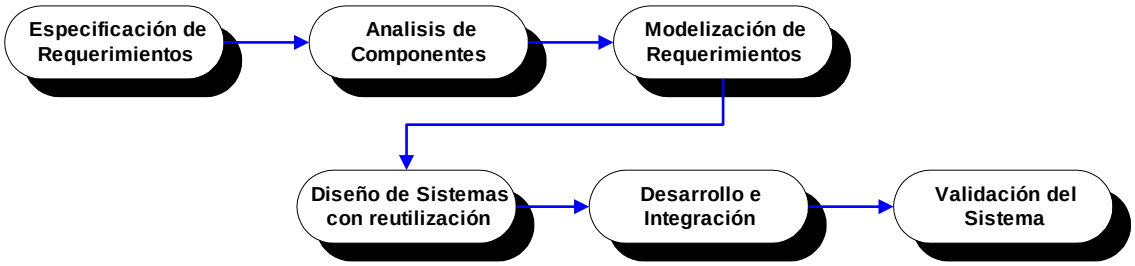


Figura 6 : Desarrollo Orientado a la Reutilización

Existe un modelo de procesos genéricos para el desarrollo basado en la reutilización, que consiste en: especificación de requerimientos, análisis de componentes, modificación de requerimientos, diseño de sistemas con reutilización, desarrollo e integración y validación del sistema. Las etapas terminales (tanto la de especificación como la de validación) son comparables con otros procesos, en cambio las etapas intermedias son diferentes.

- **Especificación de componentes:** dada una especificación de requerimientos, se buscan los componentes para implementar esta especificación.
- **Modificación de los requerimientos:** los requerimientos se analizan utilizando información acerca de los componentes que se van descubriendo. Entonces estos componentes se modifican para reflejar los que están disponibles.
- **Diseño de sistema con reutilización:** en esta fase se diseña o reutiliza un marco de trabajo para el sistema. Se tienen en cuenta los componentes que se utilizan y se organiza un marco de trabajo que se ajuste a ellos.
- **Desarrollo e integración:** para crear el sistema, el software que no se puede comprar, se desarrolla y los componentes se integran. En este modelo, la integración de sistemas es parte del proceso de desarrollo, más que una actividad separada.

El modelo orientado a la reutilización tiene como ventaja, la reducción del tiempo de desarrollo, los costos y por ende los riesgos. Por lo general también conduce a una entrega más rápida del sistema.

Iteración de Procesos

Todos los modelos de procesos anteriores tienen ventajas y desventajas. Para sistemas muy grandes existe la necesidad de combinar los diferentes enfoques, por lo cual se debe utilizar un modelo híbrido. A medida que cambian los requerimientos es necesario iterar ciertas partes del proceso, es decir repetir varias veces algunas actividades.

Presentaremos dos modelos híbridos que ayudan a los diferentes enfoques y fueron desarrollados para apoyar la iteración de procesos. Estos son:

- **El desarrollo incremental:** se divide el proceso en una serie de incrementos, los cuales se desarrollan uno a uno.

- **El desarrollo en espiral:** en el que el desarrollo gira en espiral hacia afuera, comenzando con un esbozo inicial y culminando con el desarrollo final del sistema.

La esencia de los procesos iterativos es que, la especificación se desarrolla junto con el software, esto puede crear conflictos en algunas organizaciones, donde la especificación completa del sistema es parte de un contrato que se realiza para desarrollar el sistema. Esto implicaría un nuevo tipo de contrato, difícil de incorporar en clientes grandes o instituciones gubernamentales.

Desarrollo incremental

El desarrollo incremental es un enfoque intermedio que combina las ventajas del modelo de desarrollo en cascada y el enfoque del desarrollo evolutivo.

Surgió como una forma de reducir la repetición del trabajo en el proceso de desarrollo y proporcionar a los clientes algunas oportunidades para retrasar las decisiones en los requerimientos detallados hasta que adquieran cierta experiencia con el sistema.

En un proceso de desarrollo incremental los clientes identificarán, de forma general y somera, los servicios que proveerá el sistema. Clasificándolos según su importancia. Entonces se definen varios incrementos, los cuales proporcionan cada uno cierta funcionalidad. La asignación de servicios a un incremento depende de la prioridad del mismo. Los servicios de prioridad más alta son los que se entregan primero. Los requerimientos para los servicios que se van a entregar en el primer incremento se definen en detalle y se desarrollan utilizando el proceso más adecuado. Durante el desarrollo se analizan los requerimientos para los incrementos posteriores, pero no se aceptan cambios en los requerimientos para el incremento en cuestión. Tan pronto como se completan los nuevos incrementos, se integran a los existentes, de forma tal que la funcionalidad del sistema mejora con cada incremento.

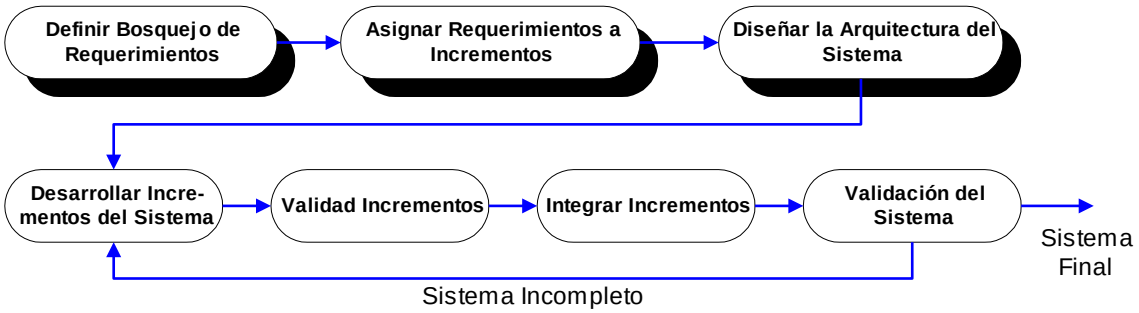


Figura 7 : Desarrollo Incremental

El proceso de desarrollo incremental tiene varias ventajas:

- Los clientes no tienen que esperar a que el sistema completo se entregue para sacar provecho de él.
- Pueden utilizar los incrementos iniciales a modo de prototipo, para adquirir experiencia, para los requerimientos posteriores.
- A los servicios más importantes del sistema son a los que se les hacen más pruebas, ya que son los primeros que se entregan y los posteriores se integran

a ellos, por lo tanto disminuye la cantidad de errores en el software, en las partes más importantes.

- Existe un bajo riesgo de fallar en el proyecto total.

Existen algunos problemas en el desarrollo incremental: los incrementos deben ser lo suficientemente pequeños y cada uno debe entregar alguna funcionalidad al sistema. Puesto que los requerimientos no se definen en detalle hasta que un incremento se implementa, es difícil identificar los recursos comunes que requieren todos los incrementos.

Recientemente se ha desarrollado una evolución de este enfoque incremental denominado "programación extrema". Se basa en el desarrollo y la entrega de incrementos de funcionalidad muy pequeños, en la participación del cliente en el proceso y en el mejoramiento constante del código.

Desarrollo en espiral

El modelo en espiral del proceso del software, fue originalmente propuesto por Böehm (1998). Representa el proceso de software como una espiral. Así, el ciclo más interno podría referirse a la factibilidad del sistema, el siguiente ciclo a la definición de requerimientos del sistema, el siguiente al diseño y así sucesivamente.

Cada ciclo de la espiral se divide en 4 sectores:

- **Definición de objetivos:** se definen los objetivos específicos. Se identifican los riesgos del proyecto, y dependiendo de los mismos, se planean estrategias alternativas.
- **Evaluación y reducción de riesgos:** se lleva a cabo un análisis detallado para cada uno de los riesgos del proyecto. Se definen los pasos para reducir dichos riesgos.
- **Desarrollo y validación:** después de la evaluación de riesgos, se elige un modelo para el desarrollo del sistema.
- **Planeación:** el proyecto se revisa y se decide si se debe continuar o no con otro ciclo de la espiral.

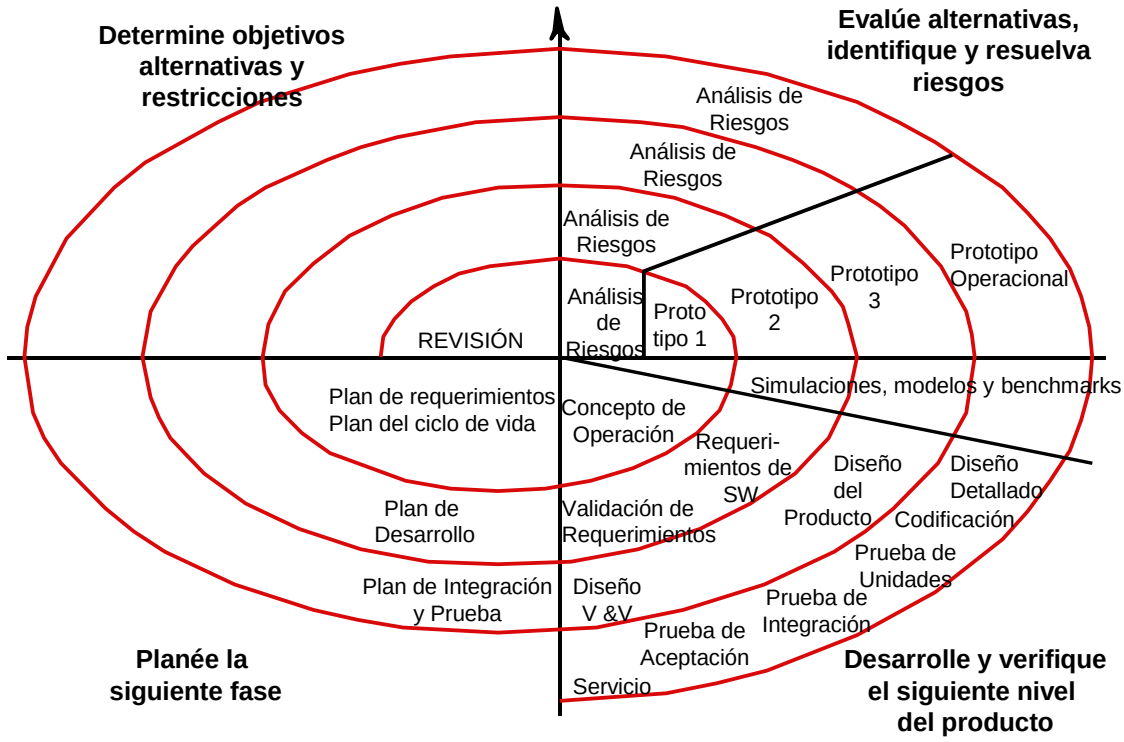


Figura 8 : Modelo en espiral de Böehm para el desarrollo del software (© 1998 IEEE)

La diferencia importante entre el modelo en espiral y los otros modelos, es que en el primero se considera el riesgo.

En el modelo en espiral no existen fases fijas como la de especificación o diseño. Este modelo puede contener otros modelos. La construcción de prototipos se utiliza para resolver las dudas en los requerimientos y así reducir el riesgo, por ejemplo.

Bibliografía

- Ingeniería de Software de Ian Sommerville. Editorial Addison Wesley (2002). Ingeniería de sistemas basados en computadora: pp. 20 – 25
- Sistemas de Información para la Gestión Empresaria, Planeamiento, Tecnología y Calidad de Alberto R. Lardent. Editorial Prentice Hall (2001). Estructura organizativa y sistemas de información: pp. 79 – 119
- Ingeniería de Software de Ian Sommerville. Editorial Addison Wesley (2002). El proceso de la ingeniería de sistemas : pp 29-37, Procesos del software : pp. 43 - 55