TEMA 48

INGENIERÍA DEL SOFTWARE. CICLO DE DESARROLLO DEL SOFTWARE. TIPOS DE CICLOS DE DESARROLLO. METODOLOGÍAS DE DESARROLLO. CARACTERÍSTICAS DISTINTIVAS DE LAS PRINCIPALES METODOLOGÍAS DE DESARROLLO UTILIZADAS EN LA UNIÓN EUROPEA.

ÍNDICE

1. INTRODUCCIÓN

Como pudiera esperarse, las organizaciones pequeñas de proceso de datos tienden a ser relativamente informales: los proyectos de desarrollo de sistemas nacen de conversaciones entre el usuario y el administrador del proyecto (que puede ser a la vez el analista, el programador, el operario y el conserje) y el proyecto procede desde el análisis hasta el diseño e implantación sin mayor alboroto.

Sin embargo, en las organizaciones más grandes, las cosas se llevan a cabo de manera mucho más formal. La comunicación entre los usuarios, la administración y el equipo del proyecto suele ser por escrito, y todo mundo entiende que el proyecto pasará por diversas fases antes de completarse. Aun así, es sorprendente ver la gran diferencia entre las maneras en que dos administradores afrontan sus respectivos proyectos. De hecho, a menudo se deja a discreción del administrador determinar las fases y actividades de su proyecto, y cómo se llevarán a cabo.

Recientemente, sin embargo, ha empezado a cambiar el enfoque que se le da al desarrollo de sistemas. Cada vez son más las organizaciones grandes y pequeñas que están adoptando un ciclo de vida uniforme y único para sus proyectos. Esto a veces se conoce como el *plan del proyecto*, la metodología del desarrollo del sistema o, simplemente, "la forma en la que hacemos las cosas aquí". El manual del ciclo de vida del proyecto suele ser un libro tan voluminoso como el compendio de normas, que yace (usualmente sin ser leído) sobre el escritorio de todo analista y programador.

El enfoque puede ser casero o, como alternativa, pudiera ser que la organización para el desarrollo de sistemas decida comprar un paquete de administración de proyectos y ajustarlo a las necesidades de la compañía. Parece obvio que, aparte de darle empleo a las personas que crean los manuales de ciclo de vida de los proyectos (y a aquellos que escriben libros de texto al respecto), es conveniente la metodología del proyecto. ¿De qué sirve entonces tener un *ciclo de vida de un proyecto*? Existen tres **objetivos** principales:

1. Definir las actividades a llevarse a cabo en un proyecto de desarrollo de sistemas.
2. Lograr congruencia entre la multitud de proyectos de desarrollo de sistemas en una misma organización.
3. Proporcionar puntos de control y revisión administrativos de las decisiones sobre continuar o no con un proyecto.

El primer objetivo es de particular importancia en una organización grande donde constantemente está ingresando personal nuevo a los puestos de administración de proyectos. El administrador novato pudiera no tomar en cuenta o subestimar la importancia de fases clave del proyecto si se basa sólo en su intuición. De hecho, pudiera suceder que los programadores y analistas de bajo rango no entiendan dónde y cómo encajan sus esfuerzos individuales en el proyecto global, a menos que se les de una descripción adecuada de todas las fases del proyecto.

El segundo objetivo también es importante en una organización grande. Para los niveles más altos de la administración pudiera ser bastante confuso seguir la pista de cientos de proyectos diferentes, cada uno de los cuales se lleva a cabo de distinta manera. Por ejemplo, si el proyecto A define la actividad de análisis de sistemas de diferente forma que el proyecto B y el B no incluye una fase de diseño, ¿cómo puede darse cuenta el administrador de segundo o tercer nivel de cuál proyecto se está rezagando y cuál continúa según lo previsto?

El tercer objetivo de un ciclo de vida de proyecto normal se refiere a la necesidad de la administración de controlar un proyecto. En los proyectos triviales, el único punto de revisión probablemente esté al final del proyecto: ¿se concluyó a tiempo y dentro de los márgenes del presupuesto acordado? (o, más simple aún, ¿se concluyó siquiera?) ¿Y cumplió con los requisitos del usuario? Pero, para proyectos más grandes, debería contarse con varios puntos intermedios de revisión, que permitieran determinar si el proyecto se estuviera retrasando o si fueran necesarios recursos adicionales. Además, el usuario inteligente también necesitará puntos de revisión en diversas etapas del proyecto para que pueda determinar si quiere continuar financiándolo.

2. EL CICLO DE VIDA DEL PROYECTO CLÁSICO

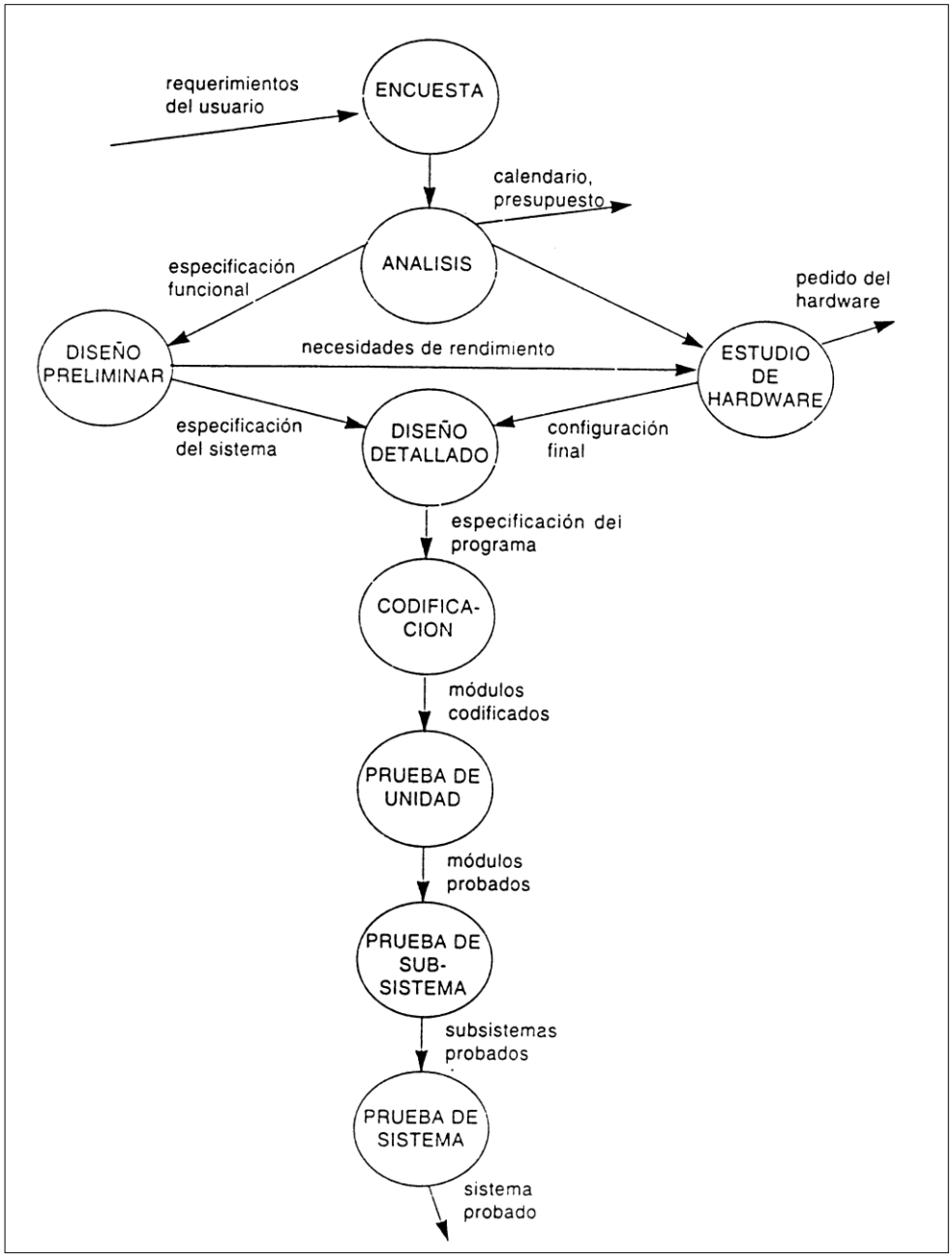
En la figura se muestra el ciclo de vida clásico o convencional. Cada proyecto atraviesa por algún tipo de análisis, diseño e implantación, aunque no se haga exactamente como se muestra en el diagrama. El ciclo de vida de proyecto utilizado, por ejemplo, en su organización, puede diferir del que se muestra en la figura en una o en todas las formas siguientes:

- Las fases de exploración y análisis pudieran juntarse en una sola (sobre todo en organizaciones en las cuales se considera factible desde el inicio cualquier cosa que quiera el usuario).

- Puede no haber fase de estudio de hardware si se cree que cualquier sistema nuevo pudiera instalarse con los ordenadores existentes sin causar mayor problema operacional.

- Las fases de diseño preliminar y de diseño de detalles pudieran juntarse en una sola llamada simplemente de diseño.

- Diversas fases de prueba pueden juntarse en una sola; de hecho, podrían incluirse con la codificación.



De aquí que el ciclo de vida del proyecto en una organización sola puede tener cinco fases o siete o doce, pero seguir siendo todavía de tipo clásico.

¿Qué es lo que realmente caracteriza el ciclo de vida de un proyecto como *clásico*? Se distinguen dos aspectos: una fuerte tendencia a la implantación ascendente del sistema y la insistencia en la progresión lineal y secuencial de una fase a la siguiente.

2.1. Implantación ascendente

El uso de la implantación ascendente es una de las grandes debilidades del ciclo de vida de los proyectos clásicos. Como se puede ver en la figura anterior, se espera que los programadores lleven a cabo primero sus pruebas modulares, luego las pruebas del subsistema, y finalmente las pruebas del sistema mismo. Este enfoque también se conoce como el ciclo de vida de cascada.

No está claro de dónde surgió originalmente este enfoque, pero pudiera haberse tomado de las líneas de montaje de las industrias manufactureras. La implantación ascendente es buena para el montaje de automóviles en línea, ¡pero sólo después de que el prototipo esté completamente libre de fallas!. Desdichadamente, muchas organizaciones que desarrollan sistemas todavía producen sistemas únicos, para lo cual el enfoque ascendente presenta un gran número de dificultades serias:

> Nada está hecho hasta que todo esté terminado. Por eso, si el proyecto se atrasa y la fecha límite cae precisamente en medio del proceso de prueba del sistema, no habrá nada que mostrarle al usuario más que una enorme pila de listados de programas, los cuales, vistos en su totalidad, no le ofrecen nada de valor.
>
> Las fallas más triviales se encuentran al comienzo del período de prueba y las más graves al final. Esto es casi una tautología: las pruebas modulares dejan al descubierto fallas relativamente simples dentro de los módulos individuales. Las pruebas del sistema, por otra parte, descubren errores grandes de interfaz entre subsistemas. La cuestión es que los errores de interfaz no son lo que el programador desea descubrir al final de un proyecto de desarrollo; tales fallas pueden obligar a la recodificación de un gran número de módulos, y pueden tener un impacto devastador sobre el calendario, justo en un momento en el cual es probable que todo el mundo esté algo cansado y molesto tras haber trabajado duro durante tantos meses.
>
> La eliminación de fallas suele ser extremadamente difícil durante las últimas etapas de prueba del sistema. Nótese que se puede distinguir entre pruebas y eliminación de fallas. La eliminación de fallas es el arte de descubrir dónde está la falla (y subsecuentemente cómo arreglarla) después de que el proceso de prueba ha determinado que la falla de hecho existe. Cuando la falla se descubre durante la fase de prueba del sistema en un proyecto ascendente, a menudo suele ser extremadamente difícil determinar cuál módulo la contiene; pudiera tratarse de cualquiera de los cientos (o miles) de módulos que se han combinado por primera vez. Localizar una falla a menudo es como hallar una aguja en un pajar.
>
> La necesidad de prueba con el ordenador aumenta exponencialmente durante las etapas finales de prueba. Para ser más específicos, el administrador del proyecto a menudo descubre que necesita una gran cantidad de horas-máquina para probar el sistema; tal vez 12 horas de labor ininterrumpida diaria. Dado que suele ser difícil obtener tanto tiempo de uso del ordenador, el proyecto suele retrasarse mucho.

2.2. Progresión secuencial

La segunda debilidad más importante del ciclo de vida de un proyecto clásico es su insistencia en que las fases se sucedan secuencialmente. Querer esto es una tendencia natural humana: deseamos decir que hemos terminado la fase de análisis del sistema y que nunca tendremos que volver a preocuparnos por ella. De hecho, muchas organizaciones formalizan esto con un ritual conocido como "congelar" la especificación o congelar el documento de diseño.

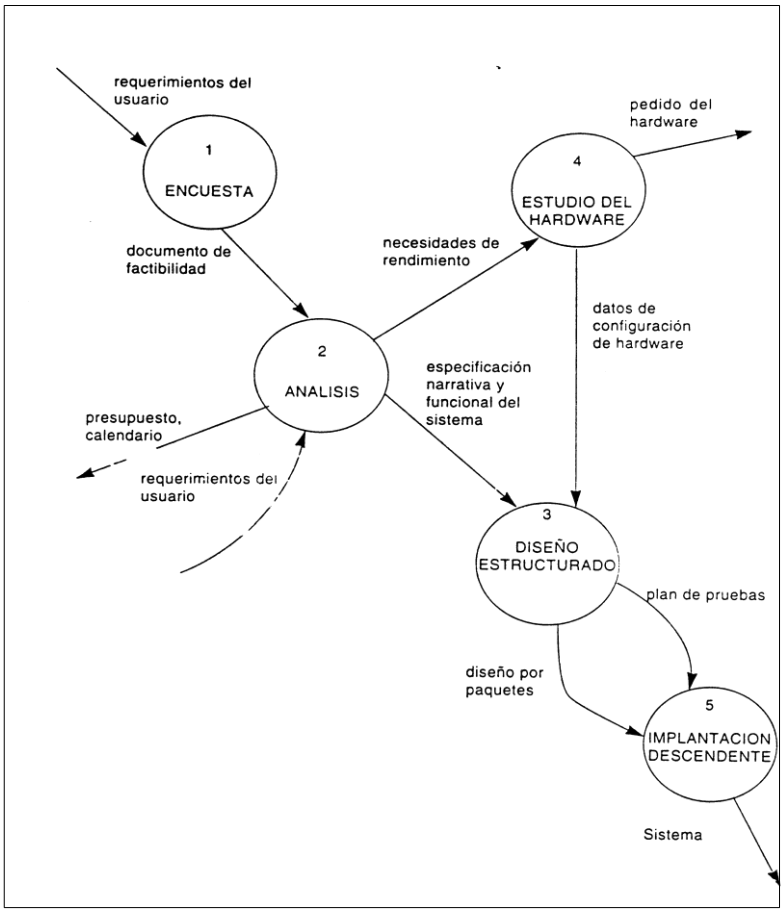
El único problema que trae consigo este deseo de progreso ordenado es que no es nada realista. En lo particular, el enfoque secuencial no permite el tratamiento de fenómenos reales como los relacionados con el personal, la política de la compañía, o la economía. Por ejemplo, la persona que hizo el trabajo, el analista o el diseñador, pudieron haber cometido un error y haber elaborado un producto con fallas. De hecho, como humanos, rara vez atinamos a hacer bien un trabajo al primer intento, pero se suelen hacer repetidas mejoras del trabajo imperfecto. También pudiera ser que la persona que revisa el trabajo o, como caso particular, el usuario que revisa el trabajo del analista del sistema pudiera haber cometido un error. O tal vez el encargado de llevar a cabo la labor asociada con cada fase no haya tenido tiempo suficiente para terminar, pero no quiera admitirlo. Esta es una manera amable de decir que, en la mayoría de los proyectos complejos, la labor de análisis, de diseño y de prueba concluye cuando alguien decide que se ha agotado el tiempo, no cuando se quisiera.

3. EL CICLO DE VIDA SEMIESTRUCTURADO

Los comentarios de la sección anterior pueden hacer que parezca que la mayoría de las organizaciones de proceso de datos todavía viven en la Edad Media. De hecho, esto es injusto: no todas las organizaciones utilizan el ciclo de vida clásico. Desde fines de los años 70 y principios de los 80, ha crecido la tendencia a reconocer al diseño estructurado, la programación estructurada y la implantación descendente como parte del ciclo de vida del proyecto. Este reconocimiento ha llevado al ciclo de vida del proyecto semiestructurado que se muestra en la figura. Se muestran dos detalles obvios no presentes en el enfoque clásico:

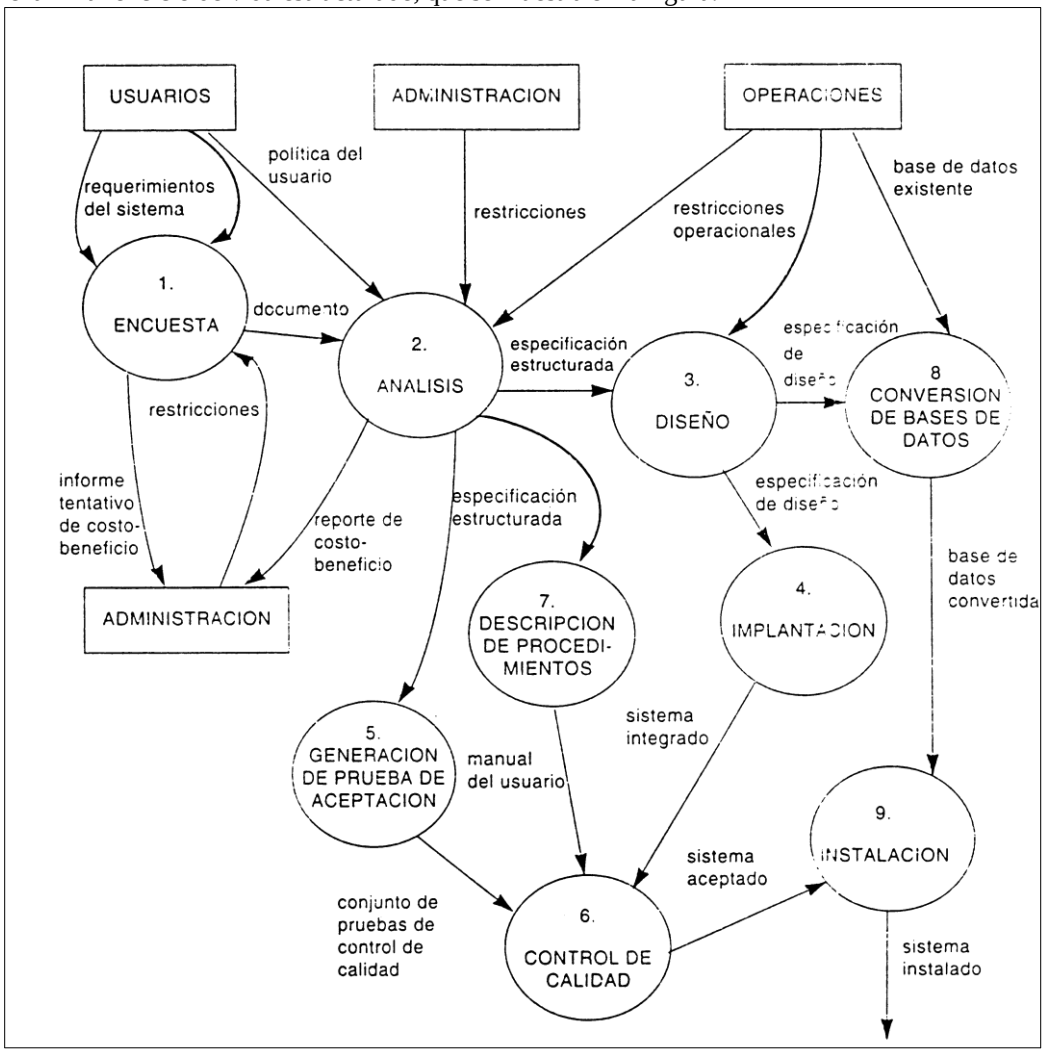
1. La secuencia ascendente de codificación, la prueba de módulos y prueba del sistema se reemplazan por una implantación de arriba hacia abajo, que es un enfoque en el cual los módulos de alto nivel se codifican y prueban primero, seguidos por los de bajo nivel, más detallados. También hay fuertes indicios de que la programación estructurada debe usarse como método para codificar el sistema.

2. El diseño clásico se reemplaza por el diseño estructurado, que es un enfoque de diseño formal de sistemas tratado en textos tales como [Yourdon y Constantine, 1989] y [Page-Jones, 1988].



4. EL CICLO DE VIDA ESTRUCTURADO DEL PROYECTO

Ahora que ya hemos visto los ciclos de vida del proyecto clásico y semiestructurado, estamos listos para examinar el ciclo de vida estructurado, que se muestra en la figura.



Examinaremos brevemente las nueve actividades y los tres terminadores del ciclo de vida del proyecto, como se muestra en la figura. Los terminadores son los usuarios, los administradores y el personal de operaciones son individuos o grupos que proporcionan las entradas al equipo del proyecto, y son los beneficiados finales del sistema. Ellos interactúan con las nueve actividades que se muestran en la figura. En las siguientes secciones se resume cada una de las actividades.

4.1. Actividad 1: La Encuesta

Esta actividad también se conoce como el estudio de factibilidad o como el estudio inicial de negocios. Por lo común, empieza cuando el usuario solicita que una o más partes de su sistema se automaticen. Los principales **objetivos** de la encuesta son los siguientes:

> *Identificar a los usuarios responsables y crear un "campo de actividad" inicial del sistema.* Esto puede comprender la conducción de una serie de entrevistas para determinar qué usuarios estarán comprendidos en (o serán afectados por) el proyecto propuesto. Pudiera también implicar el desarrollo de un diagrama inicial de contexto, que es un diagrama de flujo de datos sencillo en el cual se representa el sistema completo con un solo proceso.

Identificar las deficiencias actuales en el ambiente del usuario. Esto en general comprenderá la lista de funciones que hacen falta o que se están llevando a cabo insatisfactoriamente en el sistema actual. Por ejemplo, esto pudiera incluir declaraciones como las siguientes:

- El hardware del sistema actual no es confiable y el vendedor se acaba de declarar en quiebra.

- El software del sistema actual no se puede mantener, y no podemos ya contratar programadores de mantenimiento dispuestos a darle mantenimiento en el lenguaje que originalmente se utilizó para desarrollarlo.

- El tiempo de respuesta del sistema telefónico de pedidos actual es tan malo que muchos clientes cuelgan frustrados antes de hacer su pedido.

- El sistema actual no es capaz de producir los informes requeridos por la modificación a los impuestos decretada el año anterior.

- El sistema actual no es capaz de recibir los informes sobre límites de crédito del departamento de contabilidad, y no puede producir los informes de promedio de volumen de pedidos que requiere el departamento de mercadotecnia.

Establecer metas y objetivos para un sistema nuevo. Esto puede ser también una simple lista narrativa que contenga las funciones existentes que deben reimplantarse, las nuevas que necesitan añadirse y los criterios de desempeño del nuevo sistema.

Determinar si es factible automatizar el sistema y de ser así, sugerir escenarios aceptables. Esto implicará algunas estimaciones bastante rudimentarias y aproximadas del costo y el tiempo necesarios para construir un sistema nuevo y los beneficios que se derivarán de ello; también involucrará dos o más escenarios (por ejemplo, el escenario con una computadora grande, el de procesamiento distribuido, etc.). Aunque a estas alturas la administración y los usuarios usualmente querrán una estimación precisa y detallada, el analista tendrá mucha suerte si logra determinar el tiempo, los recursos y los costos con un error menor del 50% en esta etapa tan temprana del proyecto.

Preparar el esquema que se usará para guiar el resto del proyecto. Este esquema incluirá toda la información que se lista anteriormente, además de identificar al administrador responsable del proyecto. También pudiera describir los detalles del ciclo de vida que seguirá el resto del proyecto.

En general, la encuesta ocupa sólo del 5 al 10 por ciento del tiempo y los recursos de todo el proyecto, y para los proyectos pequeños y sencillos pudiera ni siquiera ser una actividad formal. Sin embargo, aún cuando no consuma mucho del tiempo y de los recursos del proyecto, es una actividad verdaderamente importante: al final de la encuesta, la administración pudiera decidir cancelar el proyecto si no parece atractivo desde el punto de vista de costo-beneficio.

Como analista, usted podrá o no estar involucrado en la encuesta; pudiera ser que antes de que siquiera se haya enterado del proyecto, el usuario y los niveles apropiados de la administración ya la hayan hecho. Sin embargo, en proyectos grandes y complejos, la encuesta requiere trabajo tan detallado que a menudo el usuario solicitará la colaboración del analista lo más pronto posible.

4.2. Actividad 2: El Análisis de Sistemas

El propósito principal de la actividad de análisis es transformar sus dos entradas principales, las políticas del usuario y el esquema del proyecto, en una especificación estructurada. Esto implica modelar el ambiente del usuario con diagramas de flujo de datos, diagramas de entidad-relación, diagramas de transición de estado y demás herramientas.

El proceso paso a paso del análisis de sistemas (es decir, las subactividades de la actividad de análisis de la figura) implica el desarrollo de un modelo ambiental y el desarrollo de un modelo de comportamiento. Estos dos modelos se combinan para formar el modelo esencial, que representa una

descripción formal de lo que el nuevo sistema debe hacer, independientemente de la naturaleza de la tecnología que se use para cubrir los requerimientos.

Además del modelo del sistema que describe los requerimientos del usuario, generalmente se prepara un conjunto de presupuestos y cálculos de costos y beneficios más precisos y detallados al final de la actividad de análisis.

4.3. Actividad 3: El Diseño

La actividad de diseño se dedica a asignar porciones de la especificación (también conocida como modelo esencial) a procesadores adecuados (sean máquinas o humanos) y a labores apropiadas (o tareas, particiones, etc.) dentro de cada procesador. Dentro de cada labor, la actividad de diseño se dedica a la creación de una jerarquía apropiada de módulos de programas y de interfaces entre ellos para implantar la especificación creada en la actividad 2. Además, la actividad de diseño se ocupa de la transformación de modelos de datos de entidad-relación en un diseño de base de datos.

Parte de esta actividad le interesará como analista: el desarrollo de algo conocido como el *modelo de implantación del usuario*. Este modelo describe los asuntos relacionados con la implantación que le importan al usuario al grado de que no se los quiere confiar a los diseñadores y programadores. Los asuntos principales que suelen preocupar al usuario son aquellos relacionados con la especificación de la frontera humano-máquina y la especificación de la interfaz hombre-máquina. Esa frontera separa las partes del modelo esencial que llevará a cabo una persona (como actividad manual), de las partes que se implantarán en un o más ordenadores. De manera similar, la interfaz hombre-máquina es una descripción del formato y de la secuencia de entradas que los usuarios proporcionan al ordenador (por ejemplo, el diseño de pantallas y el diálogo en línea entre el usuario y el ordenador), además del formato y la secuencia de salidas -o productos- que el ordenador proporciona al usuario.

4.4. Actividad 4: Implantación

Esta actividad incluye la codificación y la integración de módulos en un esqueleto progresivamente más completo del sistema final. Por eso, la actividad 4 incluye tanto programación estructurada como implantación descendente.

Como podrá imaginar, el analista típicamente no se verá involucrado en esta actividad, aunque hay algunos proyectos (y organizaciones) donde el análisis, el diseño y la implantación de sistemas los hace la misma persona.

4.5. Actividad 5: Generación de Pruebas de Aceptación

La especificación estructurada debe contener toda la información necesaria para definir un sistema que sea aceptable desde el punto de vista del usuario. Por eso, una vez generada la especificación, puede comenzar la actividad de producir un conjunto de casos de prueba de aceptación desde la especificación estructurada.

Dado que el desarrollo de las pruebas de aceptación puede suceder al mismo tiempo que las actividades de diseño e implantación, pudiera ser que al analista le sea asignada esta labor al término del desarrollo del modelo esencial en la actividad 2.

4.6. Actividad 6: Garantía de Calidad

La garantía de calidad también se conoce como la prueba final o la prueba de aceptación. Esta actividad requiere como entradas los datos de la prueba de aceptación generada en la actividad 5 y el sistema integrado producido en la actividad 4.

El analista pudiera estar involucrado con la actividad de garantía de calidad, pero por lo regular no lo está. Pueden tomar la responsabilidad uno o más miembros de la organización usuaria, o pudiera

llevarla a cabo un grupo independiente de prueba o un departamento de control de calidad. Consecuentemente, no se discutirá con más detalle la función de garantía de calidad.

Nótese, por cierto, que algunas personas le llaman a esta actividad "control de calidad" en lugar de "garantía de calidad". Sin importar la terminología, se necesita una actividad que verifique que el sistema tenga un nivel apropiado de calidad; le hemos llamado garantía de calidad en este tema. Nótese también que es importante llevar a cabo actividades de garantía de calidad en cada una de las actividades anteriores para asegurar que se hayan realizado con un nivel apropiado de calidad. Por eso, se esperaría que esto se haga durante toda la actividad de análisis, diseño y programación para asegurar que el analista esté desarrollando especificaciones de alta calidad, que el diseñador esté produciendo diseños de alta calidad y que el programador este escribiendo códigos de alta calidad. La actividad de garantía de calidad que se menciona aquí es simplemente la prueba final de la calidad del sistema.

4.7. Actividad 7: Descripción del Procedimiento

Nos debe preocupar el desarrollo de un sistema completo: no sólo de la porción automatizada, sino también de la parte que llevarán a cabo las personas. Por ello, una de las actividades importantes a realizar es la generación de una descripción formal de las partes del sistema que se harán en forma manual, lo mismo que la descripción de cómo interactuarán los usuarios con la parte automatizada del nuevo sistema. El resultado de la actividad 7 es un manual para el usuario.

Como podrá imaginar, esta también es una actividad en la que pudiera verse involucrado como analista.

4.8. Actividad 8: Conversión de Bases de Datos

En algunos proyectos, la conversión de bases de datos involucraba más trabajo (y más planeación estratégica) que el desarrollo de programas de ordenador para el nuevo sistema. En otros casos, pudiera no haber existido una base de datos que convertir. En el caso general, esta actividad requiere como entrada la base de datos actual del usuario, al igual que la especificación del diseño producida por medio de la actividad 3.

Según sea de la naturaleza del proyecto, el analista podría tener que ver con la actividad de conversión de la base de datos. Sin embargo no discutiremos esta actividad con mayor detalle en este tema.

4.9. Actividad 9: Instalación

La actividad final, desde luego, es la instalación; sus entradas son el manual del usuario producido en la actividad 7, la base de datos convertida que se creó con actividad 8 y el sistema aceptado producido por la actividad 6. En algunos casos, sin embargo, la instalación pudiera significar simplemente un cambio de la noche a la mañana al nuevo sistema, sin mayor alboroto; en otros casos, la instalación pudiera ser un proceso gradual, en el que un grupo tras otro de usuarios van recibiendo manuales y entrenamiento y comenzando a usar el nuevo sistema.

4.10. Resumen del Ciclo de Vida del Proyecto Estructurado

Es importante ver la figura como lo que es: un diagrama de flujo de datos. No es un diagrama de flujo; nada implica que toda la actividad N debe concluir antes de comenzar la actividad N + 1. Por el contrario, la red de flujos de datos que conectan las actividades hace ver con claridad que pudieran estarse llevando a cabo diversas actividades paralelamente.

Debido a este aspecto no secuencial, usamos la palabra actividad en el ciclo de vida del proyecto estructurado en lugar de "fase", que es más convencional. El término fase tradicionalmente se refiere a un período particular en un proyecto en el cual se estaba desarrollando una, y sólo una, actividad.

Hay otra cosa que debe recalcarse acerca del uso de un diagrama de flujo de datos para describir el ciclo de vida del proyecto: un diagrama de flujo de datos clásico, como el que se muestra en la figura, no muestra en forma explícita la retroalimentación, ni el control. Prácticamente todas las actividades de la figura pueden y suelen producir información que puede llevar a modificaciones adecuadas de una o más de las actividades precedentes. De aquí que la actividad de diseño puede producir información que acaso cambie algunas de las decisiones de costo-beneficio en la actividad de análisis; de hecho, el conocimiento que se obtiene a partir de la actividad de diseño pudiera incluso llevar a revisar decisiones anteriores acerca de la factibilidad básica del proyecto.

Más aún, en casos extremos, ciertos eventos que pudieran darse en cualquier actividad pueden causar que todo el proyecto termine repentinamente. Las entradas de la administración se muestran sólo para la actividad de análisis pues ésta es la única que requiere datos de la administración; sin embargo, se supone, que la administración ejerce control sobre todas las actividades.

5. METODOLOGÍAS DE DESARROLLO

Generalmente se entiende por metodología, refiriéndonos a cualquier ámbito o trabajo, a un sistema ordenado de proceder para la obtención de un fin.

Si nos centramos en el entorno informático, es decir, en la producción o desarrollo de sistemas informáticos, es evidente que el uso de una metodología en este proceso aporta unas ventajas que hacen aconsejable su uso.

El problema en el momento actual es elegir una de las disponibles en el mercado con el suficiente conocimiento de causa.

Una primera observación nos lleva a una separación entre metodologías **públicas**, o sea, aquéllas cuya utilización no lleva al usuario al pago de ninguna cantidad a las entidades que la crearon y **privadas**, aquéllas desarrolladas por entidades del mismo tipo y que, por tanto, basan sus beneficios en el cobro de licencias de uso como si de cualquier otro producto software se tratara.

Dentro de las **metodologías públicas**, se pueden distinguir tres corrientes actualmente:

- La francesa, que dio como fruto la metodología **Merise,** potenciada por la administración francesa a partir del año 1977.

- La inglesa, también impulsada por la administración en Gran Bretaña y que dio lugar, a partir de 1981, al **SSADM**.

- La americana, basada en las teorías de **Edward Yourdon** y que tiene algunas variantes aportadas por otros autores como **Demarco, Gane y James Martin.**

El resto de las metodologías existentes, tanto públicas como privadas, deben considerarse como adaptaciones, más o menos mejoradas, de las citadas anteriormente.

También hay que considerar que los objetivos que persiguen todas ellas son parecidos, por tanto, es evidente que para hacer un desarrollo estructurado y ordenado de una aplicación, los caminos seguidos, o sea, las fases y su cronología, no pueden ser muy diferentes.

Uno de los posibles enfoques de un libro sobre metodología, sería definir una propia y divulgarla como la metodología ideal.

Pero el futuro del uso de las metodologías pasa por su soporte en herramientas informatizadas, de las englobadas actualmente bajo el nombre de **Case.** Por esto la tendencia general es que, cada vez menos, cada cual haga la guerra por su cuenta. Afortunadamente, van pasando los tiempos en que cada casa fabricante seguía su línea y se imponen líneas comunes a la hora de buscar estándares en casi todos los terrenos.

En el ámbito europeo, todos trabajan en la elaboración del **Eurométodo,** proyecto impulsado por la Comunidad Europea en un intento de unificar las tendencias actuales sobre metodología.

No parece buen momento, por tanto, para aportar nuevas vías y sí parece conveniente, en un momento de explosión de las herramientas Case en el mercado, clarificar ideas y documentar técnicas preparándonos para el nuevo entorno de desarrollo futuro.

Veremos algunas de las metodologías más extendidas actualmente.

Dentro de las metodologías europeas, las más consolidadas son la francesa **Merise** y la inglesa **SSADM**. Ambas nacieron y se desarrollaron al amparo de los respectivos gobiernos con la intención de extenderlas en sus administraciones públicas.

Al hablar de metodologías americanas, alrededor de la de **Yourdon** hay multitud de colaboraciones que aportan variantes y opciones para determinadas etapas del ciclo de vida o para determinado tipo de proceso.

De cualquier forma, a través de la descripción de estas tres metodologías, quedan representadas la mayoría de las técnicas y las tendencias actuales en cuanto a las diferentes etapas del ciclo de vida.

5.1. Merise

Surge en Francia a partir del año 1977, tras una petición del Ministerio de Industria, como un intento de definir una metodología a emplear en la Administración Pública para el desarrollo y diseño de sistemas informáticos.

Esta petición se realiza en un momento en que la gran diversidad de lenguajes y formalismos empleados para la representación de sistemas informáticos hacen necesario un esfuerzo de unificación de criterios y métodos. En un momento en que hay que tratar de eliminar los problemas derivados de la rotación de personal.

Los principios generales en que se apoya **Merise** son:

- Desglose del desarrollo en etapas.

- Definición de los documentos estándar de cada una.

- Uso del modelo **entidad/relación y** sus formalismos para la representación de datos.

- Uso de las **redes de Petri** para la representación de procesos y tratamientos.

- Definición de grupos de trabajo y reparto de las responsabilidades y funciones a lo largo del desarrollo.

- Especificación del reparto de tareas y tratamientos entre los usuarios y el ordenador.

- Definición de los flujos de información entre las unidades del sistema.

El sistema se contempla desde diferentes niveles de abstracción y esto da lugar a una descripción del mismo a tres niveles: conceptual, lógico u organizativo y físico.

En la fase de concepción se trabaja básicamente sobre dos elementos: *datos* y *tratamientos*. La descripción de los datos reflejará la información existente en el entorno y las relaciones entre ellos. La representación de los tratamientos reflejará los procesos a realizar con los datos así como su secuencia en el tiempo.

Con la descripción de estos dos elementos habremos conseguido reflejar tanto el contenido del sistema como su funcionamiento.

Se podría establecer el siguiente cuadro en cuanto a los diferentes niveles, tanto de decisión como de descripción de datos y tratamientos:

Niveles de decisión	**Nivel de descripción**	
	Tratamientos	**Datos**
Conceptual	Conceptual	Conceptual
Organizativo	Organizativo	Lógico
Técnico	Operativo	Físico

Veremos en primer lugar lo que significan estos niveles y cual es su contenido. Empecemos por los tratamientos.

* En cuanto a la **descripción de tratamientos,** el **nivel conceptual** consiste en la descripción del **qué** hay que hacer, es decir, en la descripción en términos de operaciones y resultados de la gestión que debe resolver el sistema independientemente de quién sea el que la realice, hombre o máquina, y de qué modo.

En el **nivel organizativo,** se desglosan las operaciones descritas a escala conceptual en procedimientos funcionales, es decir, en tareas realizadas sucesivamente en un puesto de trabajo. A este nivel se concreta ya **quién, cuándo y dónde** se han de realizar estas tareas.

En el **nivel operativo** se responde a la pregunta de **cómo** hay que hacer las cosas. Se detallan, tanto para procedimientos automatizados como para los manuales, las normas para realizarlos correctamente.

Veamos a continuación como se estructura la descripción de los datos desde diferentes niveles de abstracción.

* En la **descripción de datos** contemplamos en primer lugar un **nivel conceptual,** en el que se observa la información del sistema en términos de objetos o entidades, se describen sus propiedades, la información de cada uno de ellos y las relaciones entre los mismos. Este modelo conceptual de datos es, en principio, bastante estable a lo largo del ciclo de vida del sistema.

En el **nivel lógico** se traduce el modelo conceptual en agrupaciones o estructuras lógicas de datos para su tratamiento por el sistema. Todavía este nuevo modelo debe ser independiente de la opción técnica a elegir en cuanto a soporte software del mismo.

En el nivel más bajo de la descripción de datos, es decir, en el **nivel físico,** se concreta ya cual va a ser la estructura final de los datos de acuerdo al sistema gestor elegido (base de datos, tipos de ficheros, etc.). Una vez conocido esto se podrán hacer optimizaciones del modelo lógico para mejorar rendimientos.

En estos dos procesos, descripción de datos y de tratamientos, se hacen necesarias validaciones para comprobar la coherencia entre unos y otros. En este sentido se podrían estructurar los pasos a seguir en el siguiente diagrama que propone un estudio paralelo de datos y tratamientos.

Por otra parte, **Merise** como todas las metodologías, propone una serie de *etapas en el desarrollo de un proyecto*:

- Estudio preliminar.
- Estudio detallado.
- Realización.
- Puesta en marcha o implantación.

Esto es lo que compone el ciclo de vida del sistema.

Otro punto contemplado por **Merise** es la descripción de los diferentes grupos de trabajo, es decir, el personal implicado en el desarrollo del sistema a todos los niveles. Así, habrá:

- **Comité director** que indicará los objetivos a cubrir por el sistema en las primeras fases de concepción.

- **Comité de usuarios** que aportará el punto de vista de los futuros utilizadores del mismo.

- **Grupo de desarrollo** o personal técnico encargado de satisfacer esos requerimientos a través del sistema informático.

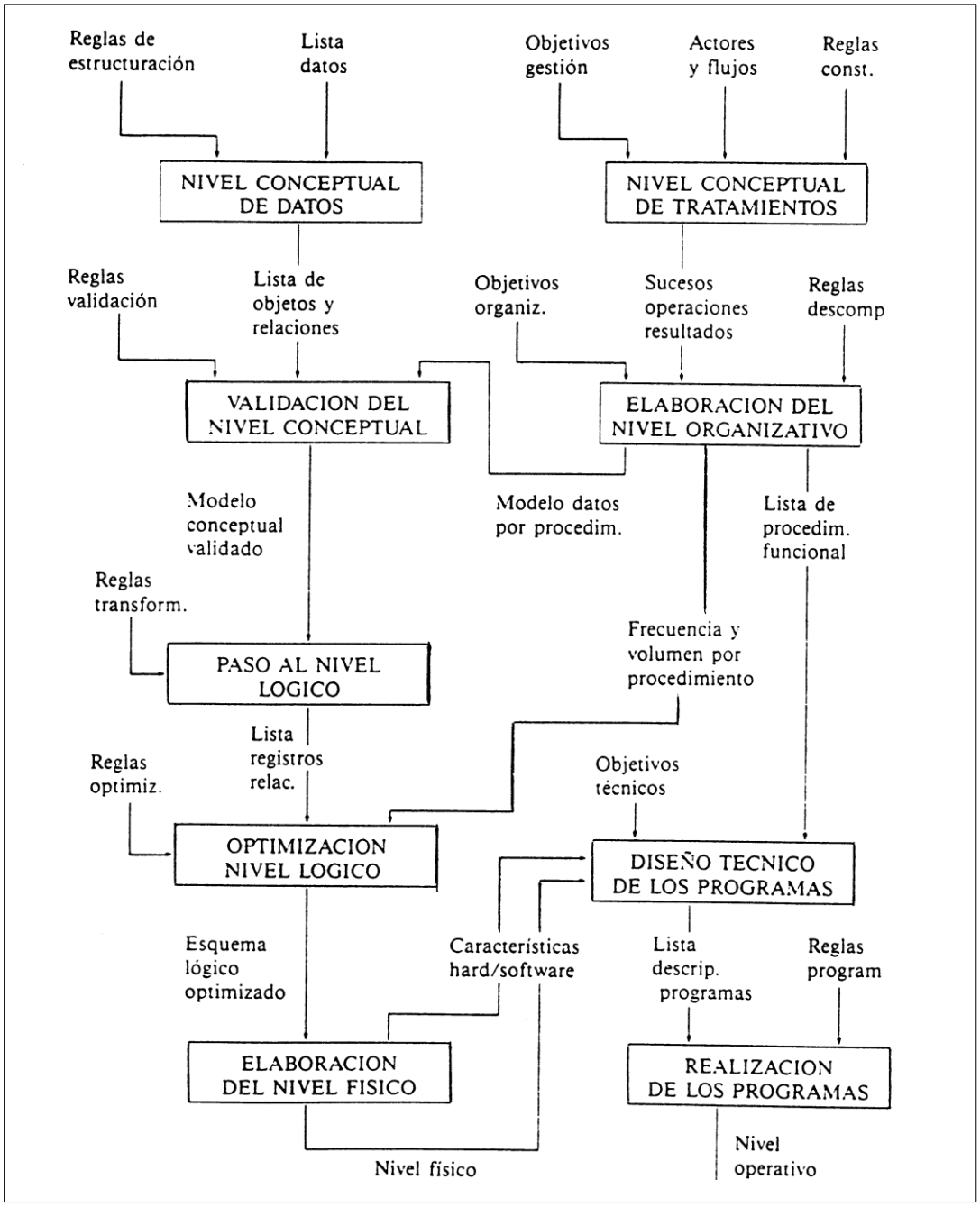


Diagrama: Niveles de descripción de datos y tratamientos

5.2. SSADM

Las siglas de esta metodología corresponden a las iniciales de Structured System Analysis and Design Method.

Nace en el Reino Unido a petición del Gobierno inglés con la intención de ser un sistema de desarrollo de aplicaciones informáticas para toda la administración pública de Gran Bretaña.

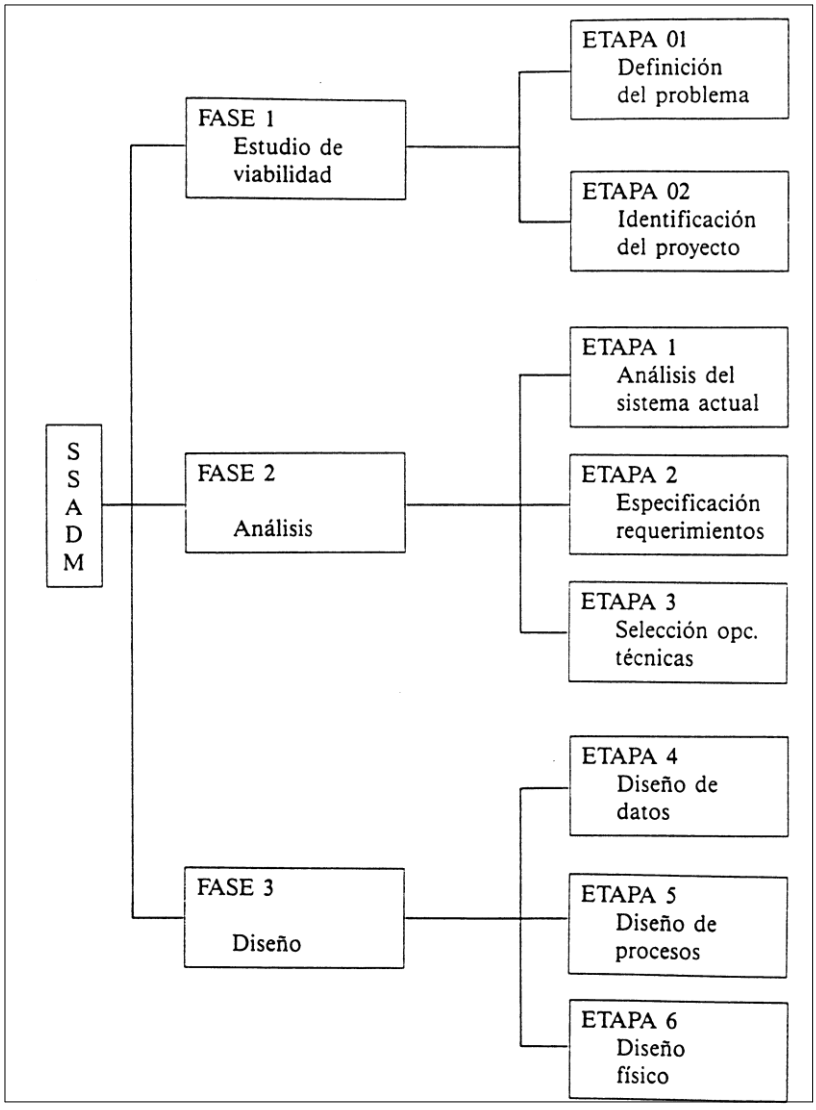
Es aceptada en 1981 en que se presenta la primera versión de la misma. Se hace de uso obligatorio en toda la administración en 1983, aunque se siguen sacando sucesivas versiones hasta el modelo actual que van incorporando mejoras o contemplando fases del ciclo de vida no cubiertas en principio.

La metodología consiste en una estructuración de los pasos a seguir en el desarrollo de un proyecto informático en las fases iniciales del ciclo de vida del mismo y en la descripción de unas técnicas y formalismos sobre las que se basan los trabajos a realizar en cada fase.

Así, según un carácter puramente jerárquico, podríamos distinguir:

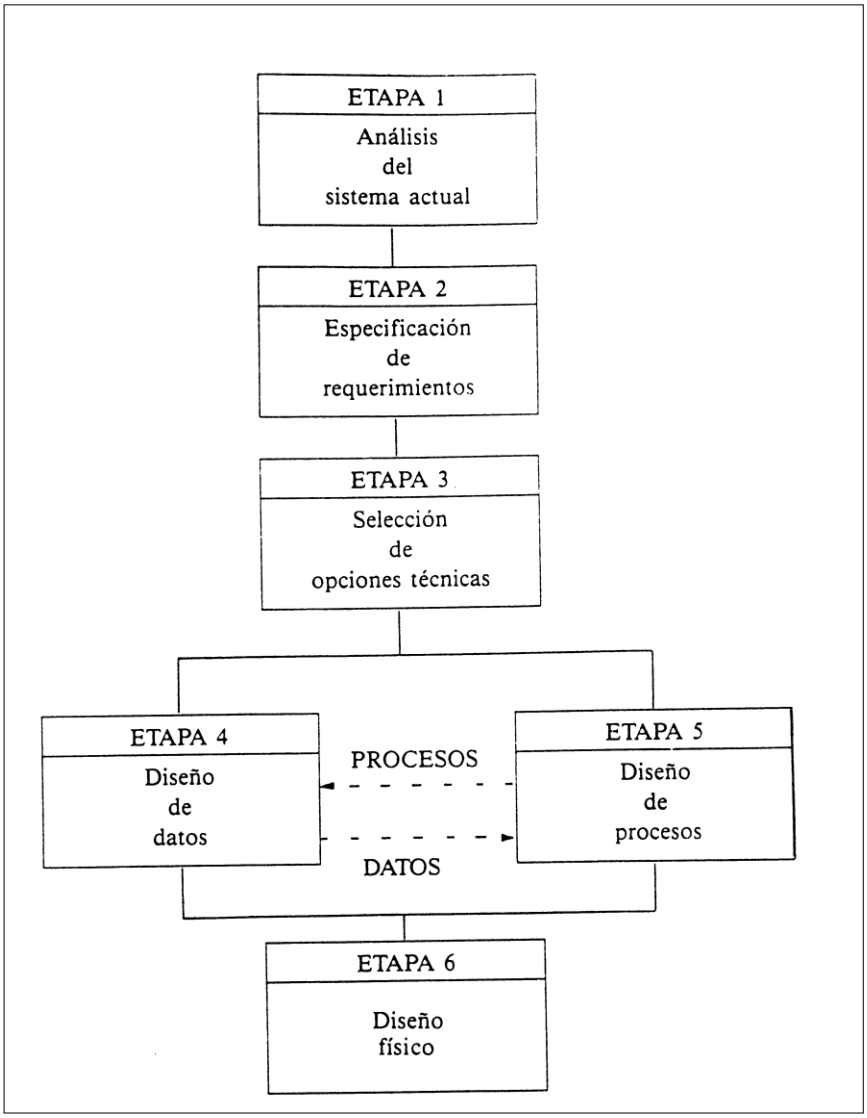
FASES ETAPAS PASOS TAREAS

El método contempla, en principio, las tres primeras fases del desarrollo (estudio de viabilidad, análisis y diseño), divididas a su vez en una serie de etapas:



Fases y etapas de SSADM

Cronológicamente, las seis etapas, una vez comprobada la viabilidad del proyecto, se desarrollarán en el siguiente orden:



Orden de ejecución de las etapas

Entre las técnicas utilizadas en esta metodología están:

- **Diagramas de Flujo de Datos** (**DFD**), que son una forma de representación de los flujos de información en el interior del sistema contemplado y entre el sistema y el exterior, es decir, sus relaciones con otros.

- **Estructura Lógica de Datos** (**LDS**). Mediante la representación de las entidades del sistema y las relaciones entre ellas. Para ello se utilizarán los formalismos y teorías del modelo entidad/relación.

- **Historia en la Vida de la Entidad** (**ELH**). Representa la descripción de cómo las entidades descritas son afectadas por diferentes sucesos que ocurren en el sistema.

- **Tercera Forma Normal** (**TNF**), en la descripción de datos. Es un método matemático para la definición de datos que ayuda a evitar inconsistencias y ambigüedades en la estructura de los mismos.

5.3. Yourdon

Es el representante de la corriente metodológica más importante de Estados Unidos, aunque hay numerosos autores que aportan variantes, matices y formalismos de representación al método de Yourdon.

A lo largo de sus obras Yourdon describe técnicas para la realización de análisis estructurado de sistemas basado principalmente en los siguientes conceptos:

- Diagramas de flujo de datos para la representación de procesos.

- Diagramas de transición de estados para la representación estructurada de las funciones a realizar en los procesos.

- Modelo Entidad/Relación para la representación conceptual de datos.

- Diccionario de datos como base o soporte de información del sistema.

- Diagramas o mapas de estructura para la representación modular de los procesos y las variables intercambiadas entre ellos.

- Especificaciones de programas basadas en lenguaje estructurado y tablas de decisión.

6. BIBLIOGRAFÍA

Pressman, Roger S.
Ingeniería del Software
Mc Graw-Hill, 1993

López, Antonio
Metodologías de Desarrollo
Ra-ma, 1990