MANTEMA: una metodología para mantenimiento de software

Tesis doctoral defendida en del Departamento de Informática de la Universidad de Castilla-La Mancha el 18 de julio de 2000

Autor: Macario Polo Usaola

Director: Dr. D. Mario Piattini Velthuis

“Es bueno tener en costumbre algunos vicios como pueden ser fumar, comer cerdo, beber alguna sobrecopa o no hacer gimnasia, para que si algún día cae uno enfermo tenga el médico algo que prohibir y uno sane. Pero si uno es todo virtud, en cayendo enfermo morirá, por impotencia de mejora”.

(Luis Landero, en *Juegos de la Edad Tardía,*

Premio Nacional de Literatura 1990).

ÍNDICES

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

RESUMEN

El mantenimiento es la etapa más costosa del ciclo de vida software y, sin embargo, hace sólo pocos años que ha comenzado a recibir atención por parte de la comunidad investigadora. Por lo general, las soluciones planteadas para paliar el problema del mantenimiento se han enfocado sobre todo en proporcionar ayuda para algunas de las tareas que implica toda modificación del software, como la ingeniería inversa, la reingeniería y la reestructuración, habiendo dejado a un lado las soluciones metodológicas, mucho más completas.

En esta tesis se presenta una metodología para la gestión integral del proceso de mantenimiento del software que incluye:

1. Un modelo del proceso de mantenimiento, en el que se entiende el proceso como una secuencia de actividades y tareas.
2. Un conjunto de métricas, tanto de producto como de proceso, que pueden ser utilizadas durante el mantenimiento.
3. Varias técnicas, susceptibles de ser empleadas en determinados momentos del proceso.
4. Un marco para definir la estructura de las organizaciones que intervienen en el proceso de mantenimiento.
5. Una herramienta para gestionar el mantenimiento conforme al modelo del proceso.

Esta metodología se ha desarrollado en el marco de los proyectos de investigación MANTEMA (iniciativa ATYCA), MÁNTICA (CICYT 1FD97-0168) y MANTIS (CICYT 1FD97-1608) usando el método Investigación en acción, y se ha validado en entornos reales mediante la colaboración con varias empresas, entre ellas Atos ODS, S.A.

1 INTRODUCCIÓN

La elección de la cita que casi comienza esta tesis no ha sido casual: en efecto, el software en ejecución adolece de muchos *vicios* que, *afortunadamente* para las empresas de servicios informáticos, vienen produciendo cada año más y más trabajo. Desde el punto de vista del mantenimiento, tales "imperfecciones" tienen su origen en el propio software cuando de lo que se trata es de corregir defectos; en los usuarios, cuando solicitan la adición a los sistemas de nuevas funcionalidades; y en el ambiente tecnológico cambiante, cuando se pretende adaptar un sistema a un nuevo entorno.

Por otro lado, y asimilando las leyes de Lehman (1980) con el último fragmento de la cita, el software que no evoluciona es software que no se utiliza.

Este capítulo se organiza de la siguiente forma: en la sección 1.1 explicamos los diferentes tipos de mantenimiento; en la 1.2 hablamos de sus costes, resaltando su importancia dentro del ciclo de vida. Dedicamos la sección 1.3 a explicar algunas de las causas que hacen de éste un proceso tan costoso, mientras que en la 1.4 hablamos de algunas de las soluciones propuestas. En la sección 1.5 presentamos los proyectos de investigación en los que se enmarca la tesis. En la 1.6 explicamos el concepto de "metodología" manejado durante el trabajo, previamente a la exposición, en la sección 1.7, de los objetivos y la hipótesis de partida. Por último, mostramos la organización del resto de la tesis en la sección 1.8.

1.1. TIPOS DE MANTENIMIENTO.

Según la terminología ANSI-IEEE (IEEE, 1990), el mantenimiento de software es "la modificación de un producto software después de su entrega al cliente o usuario para corregir defectos, para mejorar el rendimiento u otras propiedades deseables, o para adaptarlo a un cambio de entorno". Un estándar posterior distingue explícitamente los siguientes tipos de mantenimiento (IEEE, 1992b):

1) *Correctivo*, que es la modificación de un producto software después de su entrega para corregir fallos descubiertos.
2) *De emergencia*, que es un mantenimiento correctivo sin planificación previa que se realiza para mantener el sistema en funcionamiento.
3) *Perfectivo*, que es la modificación de un producto software después de su entrega para mejorar el rendimiento o la mantenibilidad. Algunas referencias, sin embargo, utilizan el término mantenimiento *preventivo* para distinguir y hacer referencia a la modificación del software destinada a la mejora de su mantenibilidad (ISO/IEC, 1995; Pressman, 1998).
4) *Adaptativo*, que es la modificación de un producto software después de su entrega para conservarlo operativo en un entorno cambiado o cambiante.

El desconocimiento de las actividades de mantenimiento puede inducir a minusvalorar su importancia; de hecho, se tiende a asociar las actividades de mantenimiento únicamente con la corrección de errores. Sin embargo, varios autores (McKee, 1984; Frazer, 1992; Basili et al.,

1996) muestran cómo los mayores porcentajes de esfuerzo de mantenimiento se dedican a perfectivo (véase la Figura 1, tomada de Frazer).

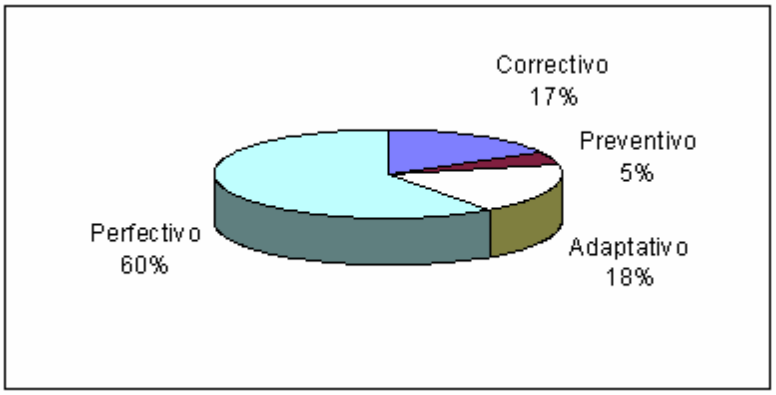


Figura 1. Costes relativos de cada tipo de mantenimiento

1.2. COSTES DEL PROCESO DE MANTENIMIENTO.

El mantenimiento es la etapa más costosa del ciclo de vida software. En efecto, Schach (1992) muestra que el coste de mantenimiento de un producto software a lo largo de toda su vida útil supone más del doble que los costes de su desarrollo. Card y Glass (1990) sitúan el porcentaje de costes entre el 67 y el 90% respecto de los costes totales del ciclo de vida. Los costes de mantenimiento del software, además, han venido mostrando una tendencia creciente en las últimas décadas, como se observa en la Tabla 1, y no hay razones para pensar que vaya a invertirse. Del mismo modo, entornos y tecnologías novedosos requerirán grandes esfuerzos de mantenimiento para conservarlos operativos: Brereton et al. (1999), por ejemplo, afirman que "ya que el mantenimiento de software es un importante problema y ya que los documentos de hipertexto comparten muchas de sus características -estructura, proceso de desarrollo y valor económico-, el mantenimiento de documentos de hipertexto llegará a ser probablemente un problema grave que requiere acción inmediata"; según Lear (2000), se está dedicando un gran esfuerzo a la modificación de grandes aplicaciones "heredadas", escritas en Cobol, para permitir su integración con servicios permitidos por las tecnologías actuales, como el comercio electrónico.

Referencia	Fechas	% Mantenimiento
Pressman (1993)	años 70	35%-40%
Lientz y Swanson (1980)	1976	60%
Pigoski, (1996)	1980-1984	55%
Pressman (1993)	Años 80	60%
Rock-Evans y Hales (1990)	1987	67%
Schach (1990)	1987	67%
Pigoski (1996)	1985-1989	75%
Frazer (1992)	1990	80%
Pressman (1993)	Años 90 (prev.)	90%

Tabla 1 Evolución de los costes de mantenimiento.

Existen organizaciones que dedican al mantenimiento un porcentaje de recursos próximo al 100%, efecto conocido como "barrera de mantenimiento", que imposibilita, obviamente, la ejecución de nuevos desarrollos. Las necesidades de mantenimiento, además, se incrementan a medida que se produce más software (Hanna, 1993), y la producción de éste ha tenido, desde sus inicios, una tendencia siempre creciente (Figura 2). Por otro lado, autores como Lehman (1980) afirman que "los grandes programas no llegan nunca a completarse y están en constante evolución", dato que confirman, por un lado, el mismo autor casi veinte años después (Lehman et al., 1998) y, por otro, el hecho de que, como ya hemos mencionado, algo más del 60% de las modificaciones que se realizan en el software se refieren a mantenimiento perfectivo.

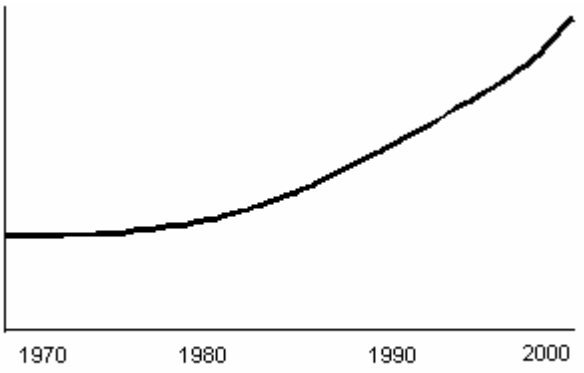


Figura 2. Evolución en la producción de software (tomada de Pressman, 1993).

En España, el informe realizado por el Ministerio de Industria y Energía sobre las Tecnologías de la Información y la Asociación Española de Empresas de Tecnologías de la Información (MINER-SEDISI, 1999) calcula que en 1998 el mantenimiento de software supuso 41.793 millones de pesetas (un 24,07% más que en el año anterior), a los que debería añadirse una parte importante del capítulo de *outsourcing*, cuyo montante total asciende a 53.708 millones. Esta tendencia se confirma también en el resto de países europeos, como por ejemplo en Francia (Bardou, 1997).

1.3. CAUSAS DEL PROBLEMA.

Sin embargo, y a pesar de esta realidad, la mayoría de las organizaciones, aunque utilizan alguna metodología para ejecutar sus nuevos desarrollos, afirman no seguir ninguna para gestionar el proceso de mantenimiento (Piattini et al., 1998a): los departamentos informáticos están orientados al control del proceso de los nuevos desarrollos, pero el mayor porcentaje del trabajo que realizan es de mantenimiento (Singer, 1998, afirma que los programadores dedican el 61% de su vida profesional a trabajos de mantenimiento, y sólo un 39% a nuevos desarrollos).

Esta falta de utilización de metodologías se debe a que, exceptuando tal vez la macroestructura de las *intervenciones* de mantenimiento (recepción de la solicitud del cambio, imple-

mentación y pruebas), no existe una definición más vasta y precisa de este proceso. Basili et al. (1996) advierten que "para gestionar de manera efectiva el mantenimiento de software [...] necesitamos proporcionar y validar metodologías que tomen en cuenta las características específicas de las organizaciones de mantenimiento de software".

Por otro lado, para Fuggetta (1999), la mayoría de los métodos y técnicas desarrollados para dar soporte a procesos software se han construido tomando como base aspectos únicamente "ingenieriles". Para este autor, deberían considerarse características importantes de las organizaciones, como su estructura o cultura. Pigoski (1997) manifiesta que "se ha escrito muy poco acerca del modelado de organizaciones de mantenimiento", y nos remite a Pressman (1993) y a Gamalel-Din y Osterweil (1988) para conocer algunas propuestas. En esta tesis hemos pretendido dar un paso adelante en este sentido y, así, identificamos las diferentes organizaciones que intervienen en el proceso de mantenimiento, sus perfiles y las responsabilidades correspondientes a éstos.

Podemos pensar, sin embargo, que es la propia naturaleza del mantenimiento lo que hace caro este proceso, y que no habrá metodología capaz de hacerlos decrecer. Schneidewind (1987), no obstante, enumera una serie de causas que dificultan el proceso de mantenimiento y que sustentan aún más la necesidad de una base metodológica para gestionar este proceso:

- Prácticamente todas las metodologías se han centrado en el desarrollo de nuevos sistemas y no han tenido en cuenta la importancia del mantenimiento. A menudo, el mantenimiento es realizado de una manera ad hoc, en un estilo libre que establece el propio programador (esta opinión también es compartida por Pressman, 1993, quien afirma que "raramente existen organizaciones formales, de modo que el mantenimiento se lleva a cabo *como se pueda*"). Esta situación no siempre se debe a la falta de tiempo para producir una modificación que podría ser cuidadosamente diseñada. Por esta razón, no existen o son poco conocidos los métodos, técnicas y herramientas que proporcionan una solución global al problema del mantenimiento.

- Cambio tras cambio, los programas tienden a ser menos estructurados. Esto se manifiesta en una documentación desfasada (como afirman Baxter y Pidgeon, 1997, al indicar que "la documentación incompleta o inexistente del sistema es uno de los cuatro problemas más importantes en mantenimiento del software"), código que no cumple los estándares, incremento en el tiempo que los programadores necesitan para entender y comprender los programas o en el incremento en los efectos secundarios producidos por los cambios. Todas estas situaciones implican casi siempre unos costes de mantenimiento del software muy altos.

- Es muy habitual que los sistemas que están siendo sometidos a mantenimiento sean cada vez más difíciles de cambiar (lo cual, como confirman Griswold y Notkin, 1993, provoca que el mantenimiento sea cada vez más costoso). Esto se debe al hecho de que los cambios en un programa por actividades de mantenimiento dificultan la posterior comprensión de la funcionalidad del programa. Sommerville (1992) también apunta que "cualquier cambio con-

lleva la corrupción de la estructura del software y que, a mayor corrupción, la estructura del programa se torna menos comprensible y más difícil de modificar". Por ejemplo, el programa original puede basarse en decisiones de programación no documentadas a las que no puede acceder el personal de mantenimiento. En estas situaciones, es normal que el software no pueda ser cambiado sin correr el riesgo de introducir efectos colaterales no deseados, debidos a interdependencias entre variables y procedimientos que el mantenedor no ha detectado.

- La falta de una metodología adecuada suele conducir a que los usuarios participen poco durante el desarrollo del sistema software. Esto tiene como consecuencia que, cuando el producto se entrega a los usuarios, no satisface sus necesidades y se tienen que producir esfuerzos de mantenimiento mayores en el futuro.

- Además de los problemas de carácter técnico anteriores, también pueden existir problemas humanos que influyen negativamente en el mantenimiento: muchos programadores consideran el trabajo de mantenimiento como una actividad inferior -menos creativa- que les distrae del desarrollo de software, que resulta mucho más interesante. Esta visión puede verse reforzada por las condiciones laborales y salariales y crea una baja moral entre las personas dedicadas al mantenimiento. Como resultado de lo anterior, cuando se hace necesario modificar el software, en vez de emplear una estrategia sistemática, las intervenciones tienden a ser realizadas con precipitación, sin ser pensadas de forma suficiente, no documentadas adecuadamente y pobremente integradas con el código existente.

1.4. SOLUCIONES PROPUESTAS.

Parece claro, por tanto, que resulta necesario dotar a las organizaciones y empresas de métodos que faciliten el proceso de mantenimiento del software, de manera que disminuyan sus costes y dificultades. Existen, de hecho, diferentes tipos de soluciones parciales -algunas de ellas automatizadas- que facilitan las labores de modificación del software. Dependiendo de su naturaleza, se pueden clasificar en:

- Soluciones técnicas, destinadas por lo general a asistir en momentos puntuales del proceso de modificación del software, y entre las que destacan la *Ingeniería inversa, Reingeniería* y *Reestructuración*.
- Soluciones de gestión, basadas fundamentalmente en:
  - *Aseguramiento de la calidad del proceso* mediante, por ejemplo, el empleo de estándares.
  - *Gestión estructurada del proceso.*
  - *Utilización de recursos humanos especializados en mantenimiento:* típicamente se tiende a utilizar en el mantenimiento a personal nuevo, habitualmente más inexperto y con mayor facilidad para desestabilizar el software (Sommerville, 1992); según Jones (1996), las intervenciones realizadas por personal especializado en mante-

nimiento resultan más productivas y de mayor calidad que las efectuadas por desarrolladores. Es conveniente, por tanto, definir las responsabilidades del personal relacionado con un producto software a lo largo de su ciclo de vida.

- ❖ *Documentación de los cambios*, almacenando los datos más relevantes de cada modificación.

En el Capítulo 3, dedicado al estado del arte, analizamos algunas de estas soluciones.

1.5. MARCO DE LA TESIS

Los trabajos que han permitido la realización de esta tesis surgieron a partir de la puesta en marcha del proyecto *MANTEMA: una metodología para el mantenimiento de sistemas de información*, en el cual participaban el Grupo Alarcos de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha y Atos ODS, S.A.:

- El Grupo Alarcos está formado por Personal Docente e Investigador del Departamento de Informática de la Escuela Superior de Informática (campus de Ciudad Real), así como por becarios de investigación y colaboradores de otras universidades. Las áreas de mayor interés investigador para el grupo son el mantenimiento de software y las métricas para bases de datos.
- Atos ODS, S.A. es una compañía multinacional que, entre otros servicios, ofrece los de externalización o subcontratación de mantenimiento del software de otras grandes empresas (industriales, bancarias, de telecomunicaciones, etc.). Por volumen de negocio, Atos ODS es la sexta compañía europea de servicios informáticos.

El proyecto MANTEMA se englobó en la iniciativa ATYCA del Ministerio de Industria y Energía y contó para su realización con un presupuesto de 59.600.000 pesetas (358.203,21 euros). A lo largo de su ejecución, el proyecto fue mostrando una serie de líneas de investigación que permanecían abiertas, como el mantenimiento de bases de datos. Para continuar las investigaciones en esta vía, se presentó a la Comisión Interministerial de Ciencia y Tecnología (CICYT-FEDER) el proyecto *MÁNTICA: Definición de un conjunto de métricas para la mantenibilidad de bases de datos objeto-relacional*, que fue aprobado en diciembre de 1998 y que cuenta con una financiación de 17.000.000 de pesetas (102.172,06 euros) para su ejecución.

Esta tesis es fruto, por tanto, de los dos proyectos arriba mencionados, si bien se ha apoyado, principalmente, en el proyecto MANTEMA. Además, algunos de los resultados obtenidos en la tesis sirven como punto de partida para el reciente proyecto *MANTIS: Entorno para el Mantenimiento Integral del Software* (CICYT 1FD97-1608), en el que se estudiarán otros aspectos del proceso de mantenimiento, como los aspectos psicológicos, la mejora del proceso y la implementación de un entorno integral para el mantenimiento. La relación de la tesis con los proyectos mencionados la ilustramos en la Figura 3.

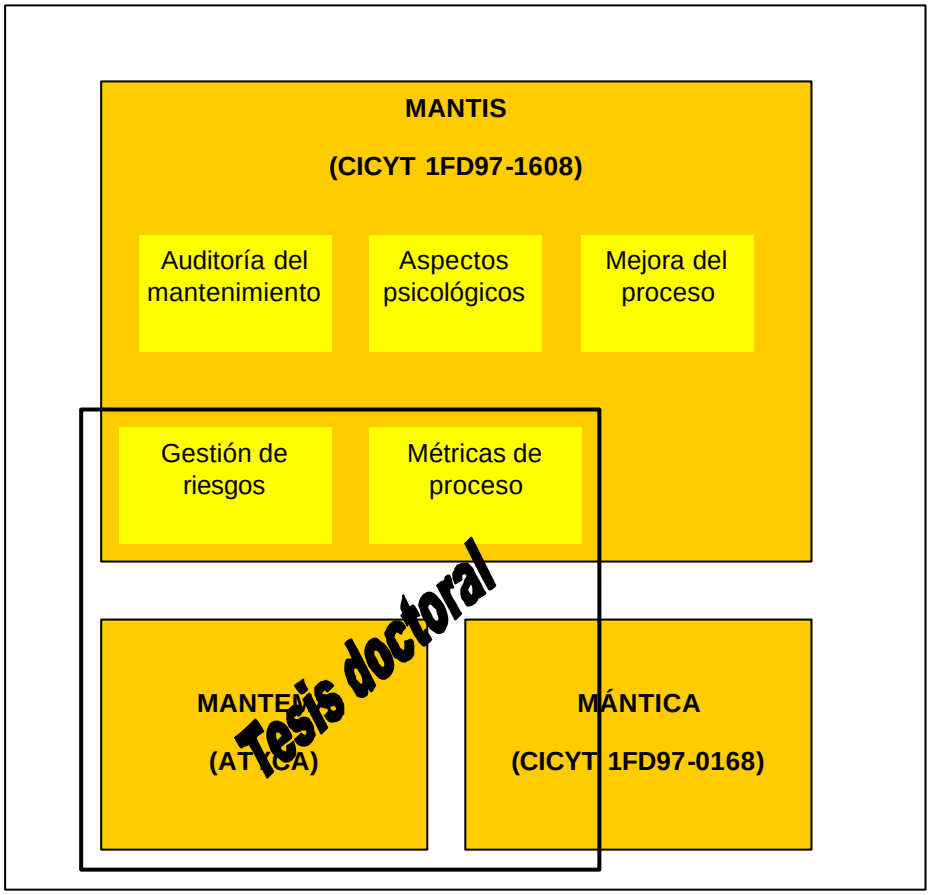


Figura 3. Proyectos que conforman el marco de la tesis.

1.6. CONCEPTO DE METODOLOGÍA.

Como veremos en la siguiente sección, tanto en los objetivos que nos marcamos en esta tesis como en la hipótesis que pretendemos demostrar, se hace uso de la palabra "metodología", término cuyo significado tal vez resulte demasiado amplio y demasiado vago. Graham, Henderson-Sellers y Younessi (1997) definen el término en su libro "The OPEN Process Specification" en el cual, aunque en el contexto de una metodología para desarrollo de software orientado a objetos, proponen una definición válida para el nuestro: estos autores afirman que una metodología debe disponer del soporte necesario para cubrir el ciclo de vida completo desde los puntos de vista técnico y de gestión (en nuestro caso, el modelo de procesos del ciclo de vida se limitaría a la provisión de un modelo para el proceso de mantenimiento).

Entre los elementos con que debe contar una metodología, los autores citan:

1. Un modelo de procesos de ciclo de vida.
2. Un conjunto de técnicas.
3. Un conjunto de entregables.
4. Un conjunto de métricas.
5. Guías para la gestión del proyecto, incluyendo roles y estructura del equipo.

6. Herramientas.

Con esta definición, el concepto de metodología continúa siendo amplio pero encierra un significado mucho más preciso. Para guiar en la lectura y ubicación de las diferentes vertientes que exponemos en la metodología descrita en esta tesis, utilizaremos repetidamente la Figura 4, especialmente a lo largo de los capítulos tercero y cuarto. Como se observa, en ella hemos hecho la siguiente distinción dentro de las categorías de técnicas y métricas:

- Técnicas de estimación de recursos, de estimación de esfuerzo y de identificación y estimación de riesgos.
- Métricas de producto y métricas de proceso. Además, dentro de las primeras, distinguiremos entre métricas para bases de datos y para sistemas orientados al objeto.

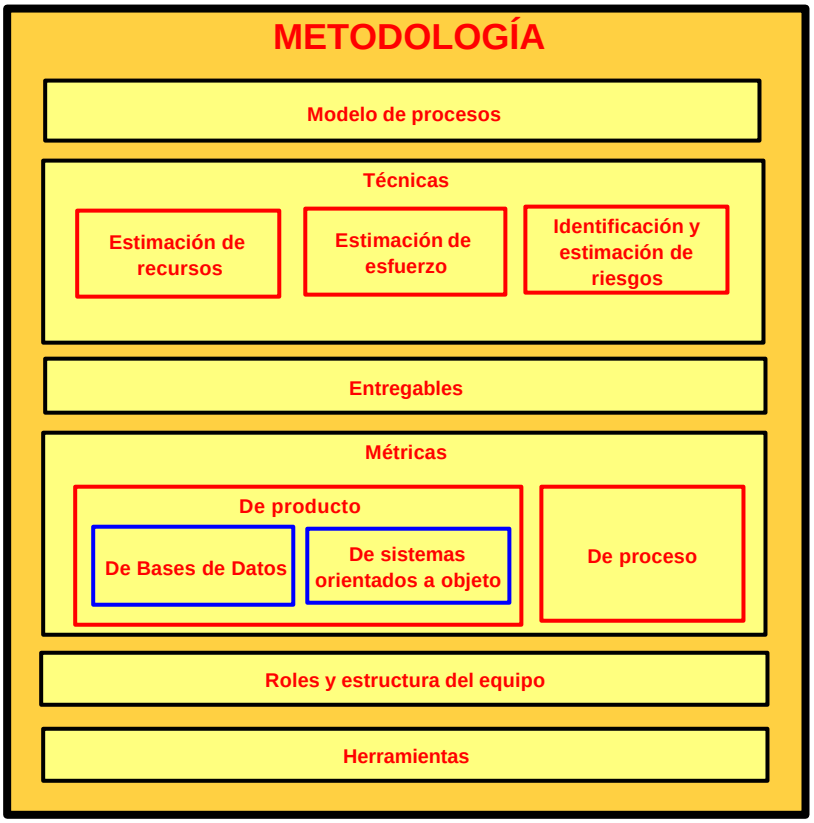


Figura 4. Componentes de una metodología (adaptada de Graham et al., 1996).

1.7. HIPÓTESIS Y OBJETIVOS.

Los objetivos que nos marcamos al comenzar los trabajos que han conducido a esta tesis fueron los siguientes:

1. Analizar las distintas metodologías aplicables al mantenimiento del software, determinando sus aportaciones y limitaciones.

2. Definir una metodología específica para mantenimiento, en el caso de no encontrar una adecuada.
3. Analizar técnicas susceptibles de ser utilizadas a lo largo del proceso de mantenimiento del software.
4. Definir, cuando no existan o no sean apropiadas, técnicas específicas para mantenimiento, susceptibles de ser integradas en la metodología propuesta.
5. Definir un conjunto de métricas para controlar el proceso de mantenimiento.
6. Validar la metodología de mantenimiento producida, así como las diferentes técnicas definidas.
7. Desarrollar un soporte automático que facilite la aplicación de la metodología.

Estos siete objetivos *parciales* podrían resumirse en un único *objetivo principal*:

OBJETIVO PRINCIPAL

«Definir una metodología de mantenimiento que permita gestionar con métodos de ingeniería el proceso de mantenimiento del software».

El cumplimiento de este objetivo principal parte, evidentemente, de la premisa encerrada en la siguiente hipótesis:

HIPÓTESIS

«Es posible gestionar el proceso de mantenimiento del software utilizando métodos de ingeniería».

1.8. ORGANIZACIÓN DE LA TESIS.

La organización de los restantes capítulos de esta tesis es la siguiente:

Capítulo segundo.

En este capítulo se presentan los métodos de trabajo e investigación seguidos durante la realización de la tesis.

Capítulo tercero.

Se resume el estado actual del mantenimiento del software, realizando una introducción a la problemática que crea y presentando algunas de las soluciones utilizadas en la actualidad para resolverla.

Capítulo cuarto.

Presentamos las diferentes vertientes que, de acuerdo a la Figura 4, juntas conforman la metodología MANTEMA.

Capítulo quinto.

Mostramos los diferentes casos en los que hemos validado el modelo del proceso de mantenimiento descrito en MANTEMA.

Capítulo sexto.

Se presenta MANTOOL, una herramienta automática para la gestión del proceso de mantenimiento que incorpora gran parte de las características de la metodología.

Capítulo séptimo.

En este capítulo se presenta el análisis de los resultados, las principales aportaciones del trabajo y las líneas de investigación que quedan abiertas.

Apéndices, acrónimos y bibliografía.

Se incluyen al final de la tesis dos apéndices, la lista de siglas utilizadas y una extensa bibliografía que contiene todas las referencias mencionadas en esta tesis.

2 MÉTODO DE TRABAJO

Tal y como hemos visto en el apartado 1.6, en el desarrollo de una metodología nos encontramos con problemas de distinta naturaleza que deben ser solucionados de diferente forma. En la elaboración y validación de esta tesis debemos enfrentarnos principalmente a dos tipos de problemas:

- Por un lado, los derivados de lo que, de acuerdo con la notación de Graham et al. (1996), se corresponde con el *modelo de procesos*, los *roles y estructura del equipo* y los *entregables.*
- Por otro, los resultantes de la definición del conjunto de técnicas y métricas.

En este capítulo presentamos los dos métodos de investigación que se han utilizado para realizar los trabajos que han conducido a esta tesis doctoral:

- El primero de ellos (Investigación en acción) es un método de investigación cualitativo y se ha aplicado para validar los trabajos mencionados en el primero de los dos puntos anteriores, mediante su aplicación en proyectos reales.
- El segundo es el método hipotético-deductivo tradicional, y se ha utilizado para probar la validez de las distintas métricas y técnicas presentadas.

2.1. INVESTIGACIÓN EN ACCIÓN *(ACTION RESEARCH)*

2.1.1. PRESENTACIÓN

En la edición de 1998 de la Conferencia sobre Procesamiento de Información se celebró la aceptación de métodos de investigación cualitativos como métodos de investigación apropiados para el campo de los sistemas de información (Avison et al., 1999), y han merecido la atención de la revista *IEEE Transactions on Software Engineering* en su número especial sobre Ingeniería del Software Empírica (Seaman, 1999).

Investigación en acción *(Action research)* es un método de investigación cualitativo que McTaggart (1991) define como “la forma que tienen los grupos de personas para preparar las condiciones necesarias para aprender de sus propias experiencias, y hacer estas experiencias accesibles a otros”. French y Bell (1996) matizan un poco más el concepto y lo definen como "el proceso de recopilar de forma sistemática datos de la investigación acerca de un sistema actual en relación con algún objetivo, meta o necesidad de ese sistema; de alimentar de nuevo con esos datos al sistema; de emprender acciones por medio de variables alternativas seleccionadas dentro del sistema, basándose tanto en los datos como en las hipótesis; y de evaluar los resultados de las acciones, recopilando datos adicionales". Según Wadsworth (1998), en Investigación en acción participan "todas las partes involucradas en la investigación, examinando la situación existente (que sienten como problemática), con los objetivos de cambiarla y mejorarla”. Esta autora identifica los siguientes cuatro tipos de participantes en este método, que pueden coincidir en algunas ocasiones:

- El investigador, que es aquel que impulsa como sujeto el proceso investigador.

- El objeto investigado.
- Aquél para quien se investiga, en el sentido de que tiene un problema que necesita ser resuelto y que participa en el proceso investigador. En este contexto se le denomina *grupo crítico de referencia*, y en él hay tanto personas que saben que están participando en la investigación, como otras que participan sin saberlo (como los pacientes sometidos a un tratamiento con placebo).
- Aquél para quien se investiga, en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participe directamente en el proceso. Puede ser el receptor de documentos, informes, etc. En este grupo, por ejemplo, caben tanto los enfermos que se benefician de un tratamiento novedoso creado lejos de ellos, como el médico que aplica dicho tratamiento.

Chein, Cook y Harding (citado en French y Bell, 1996) distinguen cuatro tipos de Investigación en acción:

- *De diagnóstico*, en la que el investigador se adentra en una situación problemática, la diagnostica y realiza recomendaciones al grupo crítico de referencia, pero sin que haya habido una evaluación de tales recomendaciones y sin un control posterior de su efecto.
- *Participante,* en la que el grupo de crítico de referencia pone en práctica las recomendaciones realizadas por el investigador, compartiendo con él sus efectos y resultados.
- *Empírica,* en la que el grupo crítico de referencia lleva un registro amplio y sistemático de sus acciones y sus efectos, característica que hace que esta variante sea difícilmente aplicable.
- *Experimental,* que consiste en evaluar las diferentes formas que existen para conseguir un objetivo. El principal inconveniente de esta variante reside, precisamente, en la dificultad de poder medir objetivamente las diferentes formas, ya que por lo general serán o bien aplicadas en distintas organizaciones con distintas características que enturbian los resultados de la investigación, o bien en una sola organización pero en distintos momentos, con lo que el entorno experimental habrá variado.

Para Padak y Padak (1994), el objeto de la investigación en Investigación en acción es habitualmente un problema que necesita ser resuelto. Estos autores identifican los siguientes pasos, que deben seguirse en las investigaciones que utilicen este método:

- *Identificar las cuestiones que guiarán la investigación*, que deben ser relevantes, estar directamente relacionadas con el objeto que se está investigando y ser susceptibles de encontrarles respuesta.
- *Recoger información relacionada con tales cuestiones*, la cual puede proceder de prácticamente cualquier sitio (bibliografía, medidas, resultados de pruebas, observaciones, entrevistas, documentos, etc.).

- *Analizar la información recogida*, que debe realizarse cuando se pasa por un periodo durante el que no se obtiene información relevante.
- *Compartir los resultados con el resto de interesados*, de tal manera que se invite al planteamiento de nuevas cuestiones relevantes y, como añade Wadsworth (1998), "a profundizar en la materia que se está investigando para proporcionar conocimientos nuevos que puedan mejorar las prácticas, modificando éstas como parte del propio proceso investigador, para luego volver a investigar sobre estas prácticas una vez modificadas".

Con estas características, el proceso de investigación definido por Investigación en acción no es un proceso lineal, sino que va avanzando mediante la compleción de ciclos, en cada uno de los cuales se ponen en marcha nuevas ideas, que son puestas en práctica y comprobadas hasta el siguiente ciclo (Wadsworth, 1998), como se muestra en la Figura 5.

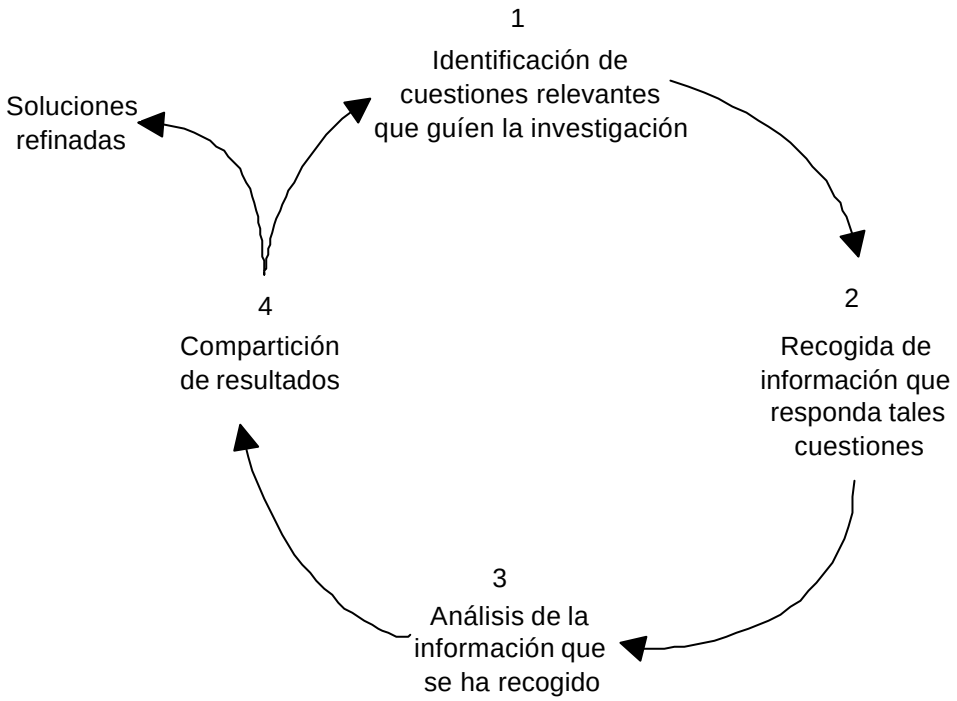


Figura 5. Proceso cíclico de Investigación en Acción.

2.1.2. APLICACIÓN DE INVESTIGACIÓN EN ACCIÓN DURANTE LA REALIZACIÓN DE ESTA TESIS

Partiendo de la base de que ha sido MANTEMA el proyecto en el que se enmarcan los trabajos relacionados con esta tesis, explicamos a continuación cómo se ha aplicado a su desarrollo la modalidad "participante" de Investigación en acción.

2.1.2.1. Participantes

En el sentido de Investigación en acción, el grupo de investigación de la Universidad ha actuado como *investigador*, siendo el proceso de mantenimiento el *objeto investigado*. El *grupo crítico de referencia* (aquel para el que se investiga y que además participa en el proceso) lo ha constituido la empresa Atos ODS, que deseaba mejorar el proceso que aplicaba al mantenimiento de los sistemas de información de sus clientes. El cuarto participante, que es aquel otro para quien se investiga sin que participe directamente en el proceso, está constituido, al menos, por todas las organizaciones dedicadas al mantenimiento de software.

En la Figura 6 ilustramos las relaciones entre los diferentes actores:

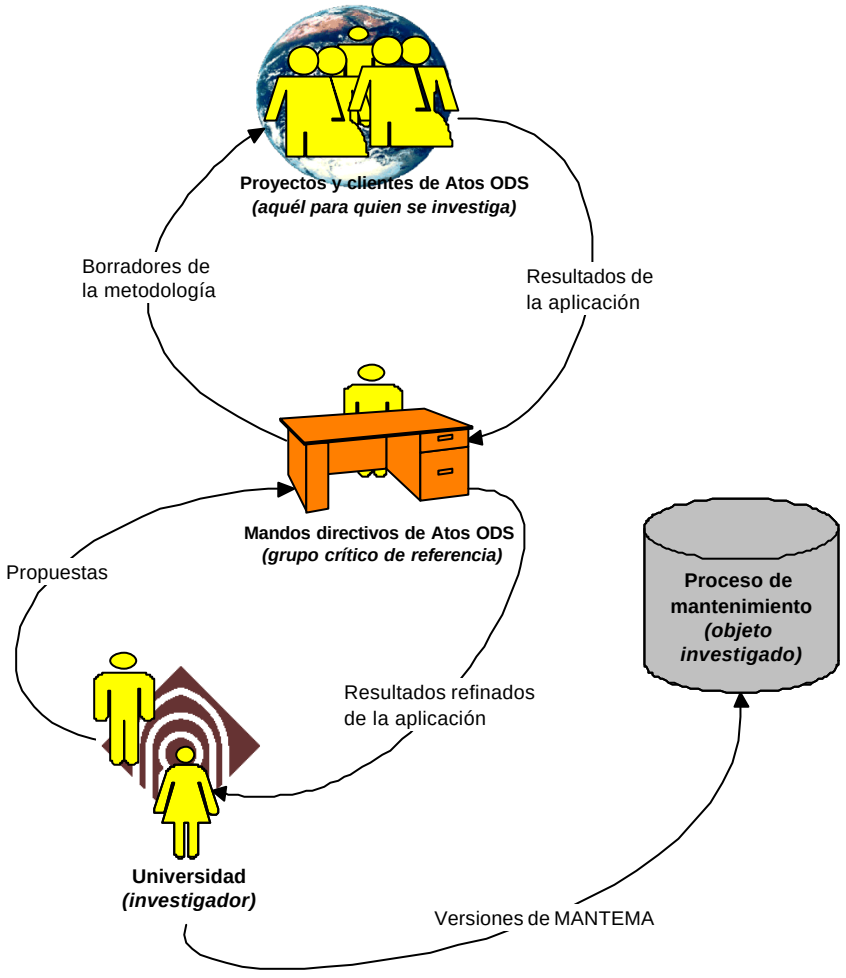


Figura 6. Aplicación del método Investigación en acción.

2.1.2.2. Aplicación de los pasos de Investigación en acción

La implementación real del método de trabajo seleccionado ha implicado, durante todo el proceso investigador, una continua retroalimentación, que se generaba en el grupo crítico de referencia y que iba dirigida al investigador. En efecto, las primeras reuniones entre ambos actores sirvieron para definir la problemática del mantenimiento, explicar el contexto en el que

se movía el grupo crítico de referencia e intentar dibujar las líneas que se podrían seguir para avanzar hacia la solución.

A partir del momento en que los dos primeros aspectos quedaron claros y el tercero empezaba a sustentarse con trazos firmes, el investigador comenzó por estudiar posibles soluciones ya existentes acerca del objeto investigado. Existían diferentes propuestas, algunas incompletas, otras obsoletas, que el grupo crítico de referencia no podía asumir como solución directa.

El investigador, por tanto, mejoraba y refinaba las propuestas, construyendo y ampliando progresivamente la metodología de mantenimiento deseada. En algunas ocasiones, el acercamiento hacia la solución se lograba por la partición del problema en otros más menudos y manejables. Así, por ejemplo, en una de las reuniones mantenidas entre el grupo de investigación y Atos ODS, se decidió enfocar los esfuerzos del investigador hacia la definición precisa de diferentes tipos de mantenimiento.

Hasta ese momento, en que se intentaba definir un único proceso de mantenimiento en el que cupieran todos los tipos, no se veía cercana la convergencia hacia la solución. Esta distinción, sin embargo, consiguió:

- Facilitar y acelerar el trabajo del investigador.
- Favorecer la incorporación de la empresa en la investigación cada vez más como parte activa: es decir, Atos ODS pudo comenzar a aplicar las diferentes guías técnicas de mantenimiento que se iban construyendo para los diferentes tipos.

Estos dos aspectos permitieron la generación de una valiosa información, que era utilizable por el investigador tanto para refinar el tipo de mantenimiento que en esos momentos se estaba aplicando, como posteriormente, al evitar errores ya cometidos durante la posterior definición de otros tipos. Del mismo modo, gracias a la aplicación de estas “guías técnicas de mantenimiento” se logró que el grupo crítico de referencia aceptara o rechazara con más velocidad y seguridad cada una de las propuestas, lo que a su vez aceleraba el trabajo del investigador. En la Tabla 2 resumimos algunos de los hitos más importantes durante la ejecución del proyecto MANTEMA.

Año	**Hecho**	**Participantes**	
		Atos ODS	**UCLM**
1997	Firma del proyecto MANTEMA	X	X
	Exposición del problema	X	X
	Recopilación y estudio de bibliografía		X
	Planteamiento de una estructura inicial de la metodología		X
	Reuniones técnicas para refinar dicha estructura	X	X
	Refinamiento		X
	Nuevas reuniones técnicas	X	X
	Propuesta de tratamiento separado de cada tipo de mantenimiento	X	
1998	Definición de una guía técnica para mantenimiento correctivo		X

		Participantes	
	Aplicación de esta guía técnica a proyectos reales	X	
	Definición de guías para el resto de tipos de mantenimiento		X
	División del correctivo en urgente y no urgente	X	
	Presentación de una guía técnica de perfectivo		X
	Estudio y comentarios a la guía de perfectivo	X	
	Presentación de MANTEMA versión 1: cinco guías técnicas para cinco tipos de mantenimiento: correctivo urgente y no urgente; perfectivo; preventivo; adaptativo		X
	Aplicación de MANTEMA versión 1 a proyectos reales	X	
1999	Estudio de MANTEMA versión 1 para corregir defectos y añadir nuevas características		X
	Búsqueda en la literatura de técnicas de apoyo para la ejecución del proceso		X
	Construcción de plantillas para los documentos de mantenimiento		X
	Informe de fallos descubiertos durante la aplicación de MANTEMA versión 1	X	
	Informe aconsejando agrupar los mantenimientos correctivo no urgente, perfectivo, preventivo y adaptativo en un solo tipo de mantenimiento (el *planificable*), y dejar aislado el correctivo urgente (*no planificable*)	X	
	Borrador de MANTEMA versión 2		X
	Fin del Proyecto MANTEMA		
	Aplicación de este borrador a proyectos reales	X	
	Adición de nuevas métricas de proceso	X	
	Inicio del proyecto MANTIS	X	X
	Refinamiento de MANTEMA versión 2		X
	Planteamiento de guía para identificación y estimación de riesgos		X
2000	Refinamiento de guía para identificar y estimar riesgos	X	X
	Elaboración de MANTEMA versión 2.1		X
	Redacción de la versión presentada en esta tesis		X

Tabla 2. Desarrollo de la "porción metodológica" de esta tesis, según Investigación en acción.

La aplicación del método Investigación en acción, por tanto, ha permitido construir la metodología presentada en esta tesis, y que ha ido refinándose progresivamente. En la Figura 7 mostramos una tabla, modificada tras una reunión entre el grupo crítico de referencia y el investigador, en la que se observa la denominación *evolutivo*, ya inexistente, para algún tipo de mantenimiento.

ACTIVIDADES Y TAREAS EVOLUTIVO								
	Análisis del error			Intervención correctiva				Cierre intervención
	1.1 Investigar y analizar causas	1.2 Documentar diagnóstico y posibles soluciones	1.3 Planificar plazos de ejecución	2.1 Ejecutar correcciones	2.2 Ejecutar pruebas unitarias	2.3 Ejecutar pruebas de integración en la aplicación	2.4 Indicar momento de la puesta en servicio	3.1-3.3
Entradas	Producto software en explotación con Error no bloqueante y no crítico Petición de modificación	Conjunto de elementos software a corregir	Elementos software con errores visibles Datos históricos de proyectos o intervenciones anteriores	Elementos software con errores visibles Causa del error Alternativa de corrección seleccionada	Elementos software corregidos	Elementos software corregidos comprobados individualmente	Elementos software corregidos, sin errores comprobados individuales ni de integración	Muy similares a las de la tabla anterior (correctivo), salvo la redocumentación del manual de usuario
Salidas	Elementos software a corregir	Causa del error Estudio de alternativas de implementación de la corrección	Cobaquario de resolución del problema	Elementos software corregidos Documento con las acciones evolutivas realizadas	Elementos software corregidos comprobados individualmente Documento con las pruebas unitarias realizadas	Elementos software corregidos, sin errores comprobados individuales ni de integración Documento con las pruebas de integración realizadas	Comunicación sobre el momento de la puesta en servicio	
Técnicas		Análisis del código fuente Análisis de la documentación externa						
Métricas								
Responsable								
Interfaces con otros procesos								

Tabla 7. Sinopsis de las actividades y tareas del evolutivo

Figura 7. Ejemplo de modificación de una tabla, tras una reunión entre dos actores del proceso investigador.

2.2. MÉTODO SEGUIDO EN LA PROPUESTA DE MÉTRICAS Y TÉCNICAS

Como observará el lector en el capítulo cuarto, además del modelo del proceso del mantenimiento, en la tesis definimos una serie de métricas y técnicas que pueden ser utilizadas en diversos momentos del mantenimiento.

Hasta hace poco tiempo, la validación de métricas se basaba fundamentalmente en la realización de experimentos con ellas, de manera que se pudiera demostrar o refutar su adecuación al propósito para el que hubieran sido construidas. No obstante, la aplicación de la Teoría de la Medida a la medición del software ha ido cobrando cada vez más importancia, habiéndose propuesto diferentes marcos formales para la caracterización teórica de las métricas de software.

En esta tesis se propone la utilización de algunas novedosas métricas para bases de datos y sistemas orientados a objeto que han sido caracterizadas, desde el punto de vista teórico, utilizando ciertos conjuntos de propiedades propuestos recientemente en la literatura científica. Esta "validación teórica" ha sido complementada en algunos casos con una validación empírica, en el sentido de que se han realizado experimentos para comprobar si las métricas resultan adecuadas para medir las propiedades del software que se pretenden. Este método de doble verificación se resume en la Figura 8: como se observa, los resultados de cualquiera de ambas validaciones pueden utilizarse para redefinir la métrica objeto del estudio.

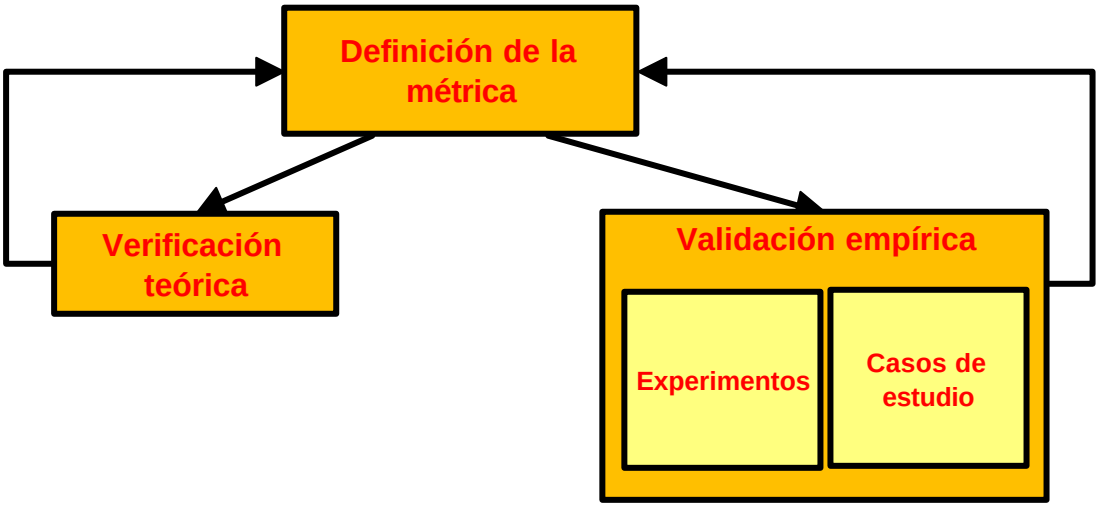


Figura 8. Verificación y validación de métricas.

3 ESTADO DEL ARTE.

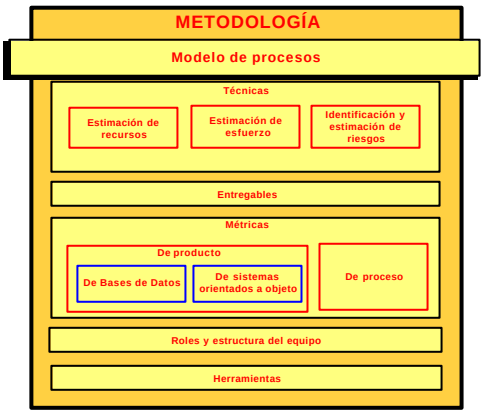
Tras la introducción al mantenimiento, la exposición de su carestía con respecto al resto de procesos del ciclo de vida software y la mención de algunas de las causas del problema, todo ello desarrollado en el capítulo primero, en este capítulo analizamos algunas de las ideas propuestas para solucionar el problema del mantenimiento del software.

Hemos estructurado este capítulo de forma parecida a la del capítulo siguiente, en el que expondremos las diferentes vertientes de la tesis, de la siguiente manera:

- Estudio de modelos de procesos para mantenimiento de software, sección en la que hemos incluido los estándares más relevantes en este campo, algunas propuestas específicas extraídas de la literatura reciente y la externalización del mantenimiento.
- Estudio de las principales propuestas en cuanto a métricas utilizables a lo largo del proceso de mantenimiento.
- Identificación de los tipos de herramientas CASE utilizables durante el mantenimiento.
- Identificación de las soluciones que, en la sección 1.4, denominamos *técnicas*.
- Análisis de algunos métodos de estimación del esfuerzo de mantenimiento.
- Estudio de algunos métodos de gestión de riesgos del mantenimiento.

3.1. MODELOS DE PROCESOS PARA MANTENIMIENTO

En esta sección analizamos algunos modelos de procesos propuestos para el mantenimiento, empezando por un estudio de estándares (de ciclo de vida y específicos para el mantenimiento) y algunas propuestas específicas para el proceso de mantenimiento. También estudiamos aquí la externalización del mantenimiento ya que, si bien podría haberse tratado fuera de esta sección, la cesión de la gestión del proceso a un proveedor externo puede considerarse el singular método que una empresa sigue para gestionar el mantenimiento de su software.



Al final de la sección (página 71) hemos incluido una tabla que utilizamos para comparar algunas de las características de los modelos del proceso que aquí analizamos.

3.1.1. ESTÁNDARES DE CICLO DE VIDA

En este epígrafe analizamos los estándares ISO/IEC 12207 e IEEE 1074.

3.1.1.1. El Estándar Internacional ISO/IEC 12207

La norma ISO/IEC 12207 (ISO/IEC, 1995) es un marco de referencia estándar que cubre todos los aspectos del proceso del ciclo de vida del software. La norma es aplicable a la adquisición de sistemas, productos y servicios software.

Este estándar internacional supone la existencia de al menos dos partes: el adquiriente (*acquirer*) y el suministrador (*supplier*). Aquél es el que tiene la necesidad de algún tipo de producto o servicio software, y éste es el encargado de proporcionársela. Sin embargo, habrá situaciones en las que ambas partes sean la misma.

La norma describe la arquitectura de los procesos del ciclo de vida software, pero no entra en los detalles de cómo implementar o realizar las actividades y tareas incluidas en tales procesos.

ISO/IEC 12207 es una norma abierta, adaptable a diferentes modelos del ciclo de vida, y es responsabilidad de las partes su adecuación al ciclo de vida utilizado.

3.1.1.1.1. Los procesos del ciclo de vida

En ISO/IEC 12207, las actividades que pueden realizarse durante el ciclo de vida software son separadas en tres grupos, que ilustramos en la Figura 9. El primero contiene cinco *procesos principales*; el segundo está formado por ocho *procesos de soporte*; en el tercero, ISO/IEC 12207 sitúa cuatro *procesos organizacionales*. Existe, además, otro proceso que no está incluido en ninguno de los grupos anteriores y que es el *proceso de adaptación* (*tailoring*).

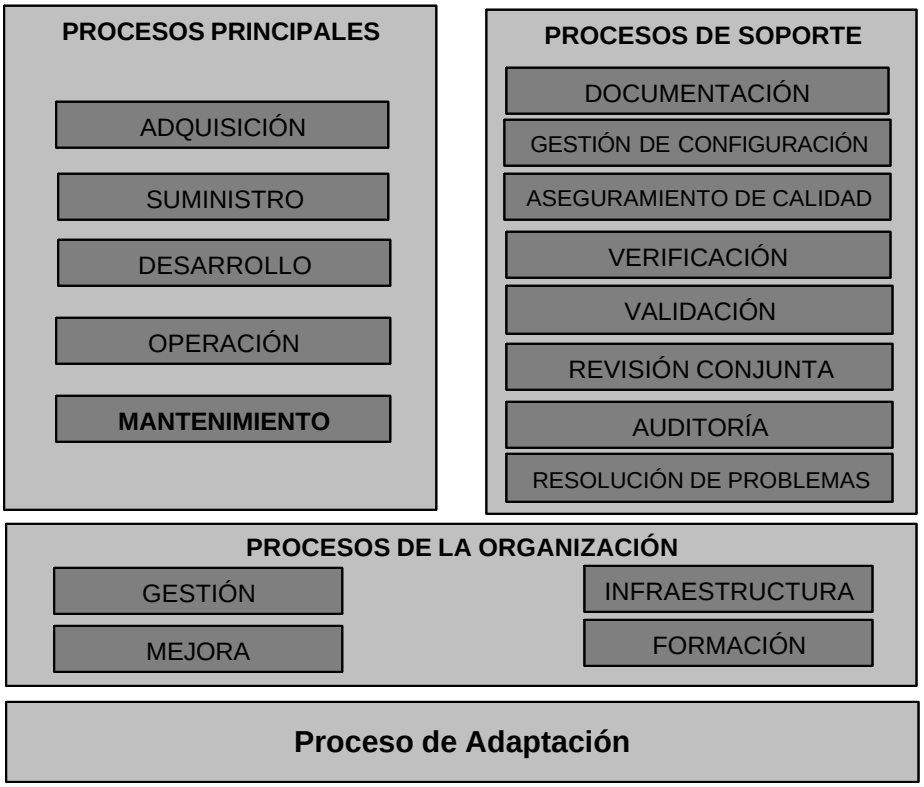


Figura 9. Procesos del ciclo de vida en ISO-12207.

Cada proceso consta de una serie de actividades, y cada actividad de un conjunto de tareas. El mantenimiento propiamente dicho es realizado en el proceso de mantenimiento, perteneciente al grupo de procesos principales, aunque utiliza procesos tanto de soporte como de organización.

3.1.1.1.2. Procesos principales

Aquí se encuentran los cinco procesos que sirven a las *partes principales* durante el ciclo de vida software. Una parte es *principal* cuando comienza o realiza las tareas de desarrollo, operación o mantenimiento de productos software. Las partes principales son, por tanto, el *adquiriente*, el *desarrollador*, el *proveedor*, el *operador* y el *personal de mantenimiento* de software.

Los procesos principales son los siguientes:

1) *Adquisición*: define las actividades del adquiriente, que es la organización que adquiere un producto, servicio o sistema software.
2) *Suministro*: define las actividades del proveedor, que es la organización que proporciona el producto, servicio o sistema software al comprador.
3) *Desarrollo*: define las actividades del desarrollador, que es la organización que define y desarrolla el producto software.
4) *Operación*: define las actividades del operador, que es la organización que provee el servicio de operación o explotación de un sistema informático en su entorno de trabajo real.
5) *Mantenimiento*: define las actividades de la organización de mantenimiento, que da el servicio de mantenimiento del producto software; es decir, la gestión de las modificaciones en el producto software para conservarlo actualizado y operativo. Este proceso incluye la migración y la retirada del producto.

3.1.1.1.3. Procesos de soporte

Estos procesos sirven de apoyo al resto y se aplican en cualquier punto del ciclo de vida. Los procesos de soporte son los siguientes:

1) *Documentación*: define las actividades necesarias para almacenar la información generada durante el ciclo de vida.
2) *Gestión de la configuración*: define las actividades de gestión de la configuración.
3) *Aseguramiento de la calidad*: define las actividades para garantizar que los productos y procesos software son conformes a los requisitos especificados y a los planes establecidos.
4) *Verificación*: define las actividades necesarias para verificar los productos software. Estas actividades pueden ser realizadas por el comprador, el proveedor o una parte independiente.

5) *Validación*: define las actividades necesarias para validar los productos software. Estas actividades pueden ser realizadas por el comprador, el proveedor o por una parte independiente.
6) *Revisión conjunta*: define las actividades para evaluar el estado y los productos de una actividad. Este proceso puede ser utilizado por dos partes cualesquiera: una de ellas es la parte *revisora*, y se encarga de revisar a la otra, la parte *revisada*.
7) *Auditoría*: define las actividades que deben llevarse a cabo para determinar el cumplimiento de los requisitos, planes y contratos. Este proceso puede ser utilizado por dos partes cualesquiera: una de las partes (el *auditor*) audita los productos o actividades de la otra parte (la *auditada*).
8) *Resolución de problemas*: define el proceso que debe llevarse a cabo para analizar y eliminar los problemas descubiertos durante el desarrollo, operación, mantenimiento u otros procesos.

3.1.1.1.4. Procesos organizacionales

Estos procesos los emplea una organización para llevar a cabo funciones de gestión, formación del personal o mejora del proceso. Ayudan a establecer, implementar y mejorar la estructura y los procesos, consiguiendo una organización más eficiente. Se llevan a cabo normalmente a nivel organizacional (corporativo), fuera del ámbito de proyectos y contratos específicos.

Los procesos organizacionales son los siguientes:

1) *Gestión*: define las actividades básicas de gestión, incluyendo la gestión del proyecto, durante el proceso del ciclo de vida.
2) *Infraestructura*: establece la infraestructura necesaria para cualquier otro proceso. Puede incluir hardware, software, herramientas, técnicas, normas e instalaciones para desarrollo, operación o mantenimiento.
3) *Mejora*: es un proceso para establecer, valorar, medir, controlar y mejorar los procesos del ciclo de vida del software.
4) *Formación*: sirve para formar y mantener formado al personal.

3.1.1.1.5. Relaciones entre los diferentes procesos

Como ya hemos indicado, el mantenimiento propiamente dicho es uno de los procesos principales, estando los procesos de los otros tipos (soporte y organizacionales) a disposición de aquéllos.

Estas relaciones entre los procesos, así como las partes que intervienen en cada uno son mostradas en la Figura 10.

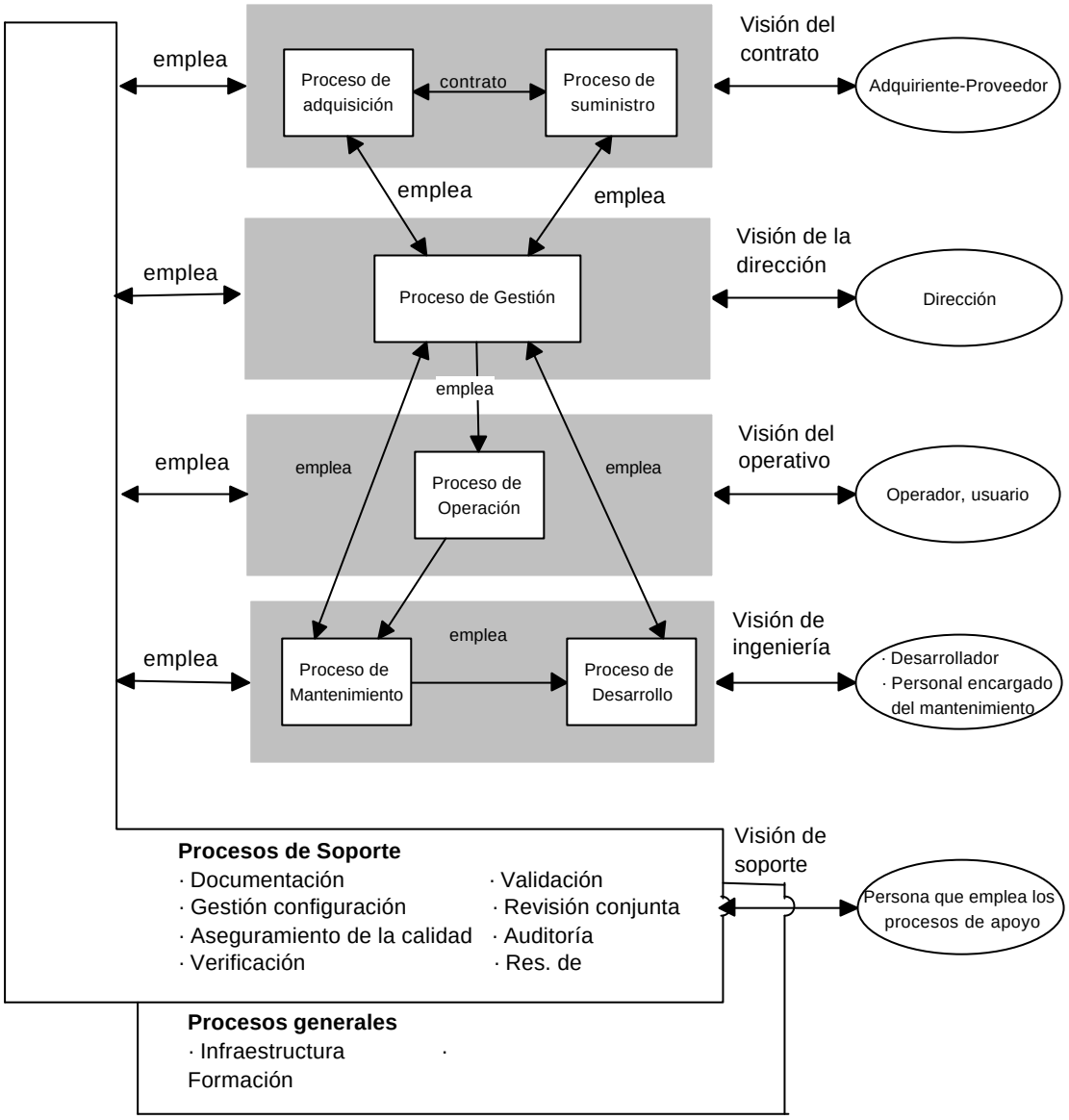


Figura 10. Relaciones entre los procesos del ciclo de vida según ISO/IEC 12207.

3.1.1.1.6. El Proceso de Desarrollo en ISO/IEC-12207

El Proceso de Desarrollo es el que más actividades y tareas contiene de los procesos propuestos por ISO en la norma que aquí analizamos (véase Tabla 3). Incluye las actividades que debe realizar el desarrollador. Entre estas actividades se encuentran el análisis de requisitos, diseño, codificación, integración, pruebas y aceptación del software relacionado. También puede contener actividades relacionadas con el sistema si así se estipula en el contrato. El desarrollador realiza sus actividades de acuerdo a lo pactado en el contrato (que se prepara en el proceso de adquisición y es aceptado por las partes en el proceso de suministro).

PROCESO DE DESARROLLO	1) Implementación del proceso
	2) Análisis de requisitos del sistema
	3) Diseño de la arquitectura del sistema
	4) Análisis de requisitos software
	5) Diseño de la arquitectura software
	6) Diseño detallado del software
	7) Codificación y pruebas del software
	8) Integración del software
	9) Pruebas de calidad del software
	10) Integración del sistema
	11) Pruebas de calidad del sistema
	12) Instalación del software
	13) Aceptación del software

Tabla 3. Actividades del proceso de desarrollo.

3.1.1.1.7. El Proceso de Mantenimiento en ISO-12207

Como hemos visto en el epígrafe 3.1.1.1.2, la norma ISO considera el mantenimiento como uno de los procesos principales del ciclo de vida del software, ya que "define las actividades de la organización (*maintainer*) que proporciona el servicio de mantener del producto software; es decir, la gestión de las modificaciones del producto software con el fin de mantenerlo actualizado y adecuado a su uso". En la definición de ISO, el mantenimiento incluye la migración y la retirada del software y consta de las siguientes actividades:

Implementación del proceso

En esta actividad se desarrollan los planes correspondientes para llevar a cabo las actividades y tareas del mantenimiento. También se deben definir los procedimientos necesarios para la gestión de problemas y petición de modificaciones (empleando el proceso de resolución de problemas), e implementar el proceso de gestión de la configuración para gestionar las modificaciones del sistema existente.

Análisis de problemas y modificaciones

Esta actividad consiste en analizar los problemas o peticiones de modificación con el fin de evaluar su impacto en el sistema y la organización existentes, determinando el tipo de modificación (preventiva, correctiva, etc.), su alcance (tamaño, coste, tiempo, etc.) y su gravedad (rendimiento, seguridad, etc.).

La organización encargada del mantenimiento debe también verificar el problema, elaborar distintas opciones para implementar las modificaciones, y documentar el problema o la petición

de modificación, así como los resultados del análisis y las opciones de implementación. Por último, deberá obtener la aprobación para la opción seleccionada.

Implementación de las modificaciones

En esta actividad se incluyen todas las tareas relativas a determinar qué documentación, unidades de software y versiones deben modificarse, y se utiliza el proceso de desarrollo para implementar las modificaciones.

Los requisitos del proceso de desarrollo deberán complementarse, según el estándar, de la siguiente manera:

a) Se deberán definir y documentar los criterios de evaluación y prueba para probar y evaluar las partes del sistema (unidades, componentes y elementos de la configuración) modificadas y no modificadas.

b) Se deberá asegurar la completa y correcta implementación de los requisitos nuevos y modificados, así como que no se vean afectados los requisitos originales. También se deberán documentar los resultados de las pruebas.

Revisión y aceptación del mantenimiento

Esta actividad consiste en la revisión de la integridad del sistema modificado. La deben llevar a cabo la organización encargada del mantenimiento junto con la organización que autorizó la modificación.

La organización encargada del mantenimiento deberá obtener también la aprobación de terminación satisfactoria de la modificación.

Migración

ISO/IEC especifica que se deberá asegurar que cualquier software o dato producido o modificado durante la migración se ajuste a las normas establecidas.

El estándar aconseja desarrollar un plan de migración en el que se especifiquen al menos las siguientes cuestiones:

a) Análisis de requisitos y definición de la migración.

b) Desarrollo de herramientas de migración.

c) Conversión del software y de los datos.

d) Ejecución de la migración.

e) Verificación de la migración.

f) Soporte del entorno antiguo en el futuro.

En esta actividad también se incluye una tarea de revisión postoperación con el fin de evaluar el impacto que suponga el cambio al nuevo entorno.

Por último, en el estándar se insiste en que se deberá poder acceder a los datos utilizados o asociados al antiguo entorno de acuerdo con los requisitos organizacionales para la protección y auditoría aplicables a los datos.

Retirada de software

De acuerdo al estándar, es necesario desarrollar y documentar un "plan de retirada" que aborde cuestiones como las siguientes:

- Cese de soporte total o parcial después de un cierto tiempo.
- Archivo del producto software y su documentación asociada.
- Responsabilidad sobre cuestiones de soporte residual futuro.
- Transición al nuevo producto.
- Accesibilidad de copias de datos.

Es importante que se tenga en cuenta a los usuarios a la hora de planificar la retirada del software y que se les notifique el plan. Las notificaciones deberán incluir lo siguiente:

a) Descripción de la sustitución o actualización con su fecha de disponibilidad.

b) Informe de por qué no se soportará más el software.

c) Descripción de otras opciones de soporte disponibles una vez que se haya eliminado el soporte.

También se recomienda llevar a cabo operaciones en paralelo entre el software nuevo y el retirado, así como proporcionar formación a los usuarios.

Se deberá notificar a todos los involucrados el momento previsto para la retirada. También se debe archivar toda la documentación, ficheros y código apropiados.

Por último, al igual que en el caso de la migración, se deberá poder acceder a los datos utilizados por, o asociados con, el software retirado de acuerdo a los requisitos organizacionales de protección y auditoría aplicables a los datos.

Tabla de resumen

En la Tabla 4 recogemos las diferentes actividades y tareas del Proceso de Mantenimiento.

PROCESO DE MANTENIMIENTO		
Actividades	*Tareas*	
IMPLEMENTACIÓN DEL PROCESO	1	Desarrollar planes de mantenimiento
	2	Definir procedimientos de petición de modificación
	3	Implementar el proceso de gestión de configuración
ANÁLISIS DE PROBLEMAS Y MODIFICACIONES	4	Evaluar impacto (tipo de mantenimiento, alcance del problema y gravedad)
	5	Verificar problema
	6	Elaborar alternativas
	7	Documentar el problema
	8	Obtener aprobación
IMPLEMENTACIÓN DE LAS MODIFICACIONES	9	Determinar objetos a modificar (documentación, unidades de software y versiones)
	10	Abrir un proceso de desarrollo para implementar las modificaciones
REVISIÓN Y ACEPTACIÓN DEL MANTENIMIENTO	11	Revisar integridad del sistema, ya modificado
	12	Obtener aprobación
MIGRACIÓN	13	Asegurar ajuste a la norma
	14	Desarrollar plan de migración
	15	Notificar la futura migración a los usuarios
	16	Ejecutar paralelamente las operaciones con los entornos antiguo y nuevo
	17	Notificar migración
	18	Realizar revisión postoperación
	19	Archivar datos entorno antiguo
RETIRADA	20	Desarrollar plan
	21	Notificar futura retirada
	22	Ejecutar paralelo
	23	Notificar retirada
	24	Archivar datos entorno antiguo

Tabla 4. Principales actividades y tareas del mantenimiento según ISO-12207.

En la Tabla 4 observamos cómo ISO/IEC propone la introducción de un nuevo proceso de desarrollo dentro del proceso de mantenimiento, pero considerándolo una tarea más (la décima tarea del mantenimiento) de la tercera actividad (implementación de las modificaciones).

Para facilitar el seguimiento de los razonamientos que expondremos a continuación, fundimos en la siguiente tabla las dos anteriores, y usaremos la notación 10.1, 10.2, etc. para referirnos a las 13 actividades del proceso de desarrollo que deben realizarse como décima tarea del mantenimiento.

1	Desarrollar planes de mantenimiento	
2	Definir procedimientos de petición de modificación	
3	Implementar el proceso de gestión de configuración	
4	Evaluar impacto (tipo de mantenimiento, alcance del problema y gravedad)	
5	Verificar problema	
6	Elaborar alternativas	
7	Documentar el problema	
8	Obtener aprobación	
9	Determinar objetos a modificar (documentación, unidades de software y versiones)	
10	10.1	Implementación del proceso
Abrir un proceso de desarrollo para implementar las modificaciones	10.2	Análisis de requisitos del sistema
	10.3	Diseño de la arquitectura del sistema
	10.4	Análisis de requisitos software
	10.5	Diseño de la arquitectura software
	10.6	Diseño detallado del software
	10.7	Codificación y pruebas del software
	10.8	Integración del software
	10.9	Pruebas de calidad del software
	10.10	Integración del sistema
	10.11	Pruebas de calidad del sistema
	10.12	Instalación del software
	10.13	Aceptación del software
11	Revisar integridad del sistema, ya modificado	
12	Obtener aprobación	
13	Asegurar ajuste a la norma	
14	Desarrollar plan de migración	
15	Notificar la futura migración a los usuarios	
16	Ejecutar paralelamente las operaciones con los entornos antiguo y nuevo	
17	Notificar migración	
18	Realizar revisión postoperación	
19	Archivar datos entorno antiguo	
20	Desarrollar plan de retirada	
21	Notificar futura retirada	
22	Ejecutar paralelo	
23	Notificar retirada	
24	Archivar datos entorno antiguo	

Tabla 5. El proceso de desarrollo dentro del mantenimiento.

3.1.1.1.8. Comentario

El seguimiento estricto del Proceso de Mantenimiento de ISO implica la ejecución secuencial de las actividades y tareas recogidas en la Tabla 5, lo cual puede, en ocasiones, llevar consigo la realización repetida de algunas tareas y actividades (aspecto que ha sido señalado en Polo et al., 1999c) y que discutimos más abajo, en el epígrafe *Problemas de solapamiento de actividades y tareas*.

La secuencialidad, sin embargo, puede verse alterada si la organización que aplica el estándar realiza una modificación de éste según su Proceso de Adaptación. Este proceso permite la eliminación de los procesos, actividades y tareas que se consideren innecesarias (ISO/IEC, 1995), tras haber identificado el entorno del proyecto y haber mantenido entrevistas con las partes involucradas y haber seleccionado del estándar las tareas, actividades y procesos de utilidad.

Por otra parte, tampoco define métricas para controlar el proceso, aunque sí indica algunos factores para considerar: *Estimación del proceso*, *Cuantificación de los riesgos asociados con las tareas* y *Costes asociados a la ejecución del proceso*. De todos modos, ISO/IEC no sugiere ningún método para estimar dichos factores.

Como todo estándar, el 12207 de ISO/IEC especifica *qué* tareas pueden realizarse, pero no *cuáles* de ellas son necesarias ni *cómo* debemos ejecutarlas. Es de desatacar, sin embargo, el gran impacto internacional que este estándar ha tenido, desde su creación, en las organizaciones de software: "ISO/IEC 12207 dirigirá el mercado mundial del software e impactará en el proceso de mantenimiento" (Pigoski, 1997); para Moore (1998), "es importante que los ingenieros de MITRE entiendan qué aporta la norma 12207 y cómo puede relacionarse con otros estándares". La 12207 también es la principal referencia de Métrica Versión 3 (Consejo Superior de Informática, 2000).

Problemas de solapamiento de actividades y tareas.

a) Actividad 10.13 y tarea 12:

La *aceptación del software* (proceso de desarrollo) consta de las siguientes tareas:

1) Revisión y prueba del producto software por parte del adquiriente (revisión conjunta), auditoría y pruebas de cualificación del software y del sistema.
2) El desarrollador deberá completar y entregar el producto software conforme al contrato.
3) El desarrollador deberá proporcionar formación y soporte continuados conforme al contrato.

Por otro lado, la tarea *obtener aprobación* (proceso de mantenimiento) consiste en que el personal de mantenimiento deberá obtener la aprobación de la modificación conforme al contrato. Pensamos que esta tarea puede verse solapada con la actividad descrita en los párrafos anteriores.

b) Actividad 10.10 y tarea 11:

La actividad *integración del sistema* (proceso de desarrollo) consta de las siguientes tareas:

1) Integrar los elementos de configuración del software con los elementos de configuración del hardware, operaciones manuales y otros sistemas (si es necesario) en el propio sistema. Según son desarrollados, deberá comprobarse que los elementos agregados cumplen los requisitos. Deberá documentarse la integración y los resultados de las pruebas.
2) Deberá desarrollarse y documentarse, para cada requisito de cualificación del sistema, un conjunto de pruebas, casos de prueba (entradas, salidas, criterios de pruebas) y procedimientos de prueba para dirigir las Pruebas de Cualificación del Sistema. El desarrollador debe asegurar que el sistema está preparado para las Prueba de Cualificación del Sistema.
3) Deberá evaluarse el sistema integrado considerando los siguientes criterios (los resultados de tales evaluaciones deberán documentarse):

   3.1. Cobertura de prueba de los requisitos del sistema.

   3.2. Adecuación de los métodos de prueba y los estándares utilizados.

   3.3. Conformidad con los resultados esperados.

   3.4. Factibilidad de las pruebas de cualificación del sistema.

   3.5. Factibilidad de la operación y mantenimiento.

Además, en la tarea *revisar integridad del sistema, ya modificado* el personal de mantenimiento debe llevar a cabo revisiones con la organización que autoricen la modificación para determinar la integridad del sistema modificado. Esta tarea pudiera quizás solaparse con el punto 3 mencionado en los párrafos anteriores.

c) Actividad 10.1 y tarea 3ª:

La actividad 10.1 es la *implementación del proceso*, perteneciente al proceso de desarrollo, y consta de las siguientes tareas:

1) Si no está estipulado en el contrato, el desarrollador deberá definir o seleccionar un modelo de ciclo de vida apropiado al alcance, magnitud y complejidad del proyecto. Las actividades y tareas del proceso de desarrollo deberán ser seleccionadas de manera que correspondan al ciclo de vida elegido.
2) El desarrollador deberá:

   2.1. Documentar las salidas de acuerdo al proceso de documentación.

   2.2. Situar las salidas de acuerdo al proceso de gestión de configuración y realizar el control de cambios de acuerdo a él.

   2.3. Documentar y resolver los problemas y disconformidades encontrados en los productos software y en las tareas, de acuerdo al proceso de resolución de problemas.

   2.4. Realizar el proceso de soporte conforme al contrato.

3) El desarrollador deberá seleccionar, adaptar al cliente y utilizar aquellos estándares, métodos, herramientas y lenguajes de programación (si no está estipulado en el contrato) que estén documentados, sean apropiados y se encuentren establecidos por la organización para desarrollar las actividades del proceso de desarrollo y de los procesos de soporte.

4) El desarrollador debe establecer planes para dirigir las actividades del proceso de desarrollo. Estos planes incluirán estándares específicos, métodos, herramientas, acciones y responsabilidades asociadas con el desarrollo y la cualificación de todos los requisitos, incluyendo los de seguridad. Si es necesario, deben desarrollarse planes diferentes. Estos planes serán documentados y ejecutados.

5) Los elementos no entregables pueden ser utilizados en el desarrollo o mantenimiento del producto software. Sin embargo, debe asegurarse que pueden llevarse a cabo la operación y mantenimiento de los productos entregables, una vez realizada esta entrega; en otro caso, estos productos deberían ser considerados entregables.

Por otro lado, la 3ª tarea del mantenimiento (*implementar el proceso de gestión de configuración*) consiste en que el personal de mantenimiento deberá implementar o establecer una interfaz organizacional con el proceso de gestión de la configuración para gestionar las modificaciones del sistema existente.

3.1.1.2. El Estándar IEEE 1074

Este estándar de ciclo de vida (IEEE, 1991) se puede aplicar a proyectos de desarrollo y mantenimiento de software. Consta de diecisiete procesos, cada uno compuesto de una serie de actividades. El estándar clasifica los procesos según la Tabla 6.

El estándar puede utilizarse para cualquier ciclo de vida software; sin embargo, antes de hacer uso del estándar, deben realizarse correspondencias entre las actividades del estándar y las del modelo de ciclo de vida utilizado o desarrollado. Esta adaptación se realiza en el primer proceso (*Modelo del ciclo de vida software*). Tras ella, se ejecutan el proceso de *Inicio del proyecto* para preparar el marco de desarrollo del proyecto. A continuación, se llevan a cabo los procesos de predesarrollo, desarrollo y postdesarrollo (en este grupo encontramos el proceso de mantenimiento).

Los procesos *Control y seguimiento del proyecto* y *Gestión de la calidad software* se ejecutados paralelamente, durante toda la vida del proyecto. Por otra parte, en los Procesos integrales se ejecutan los procesos que son necesarios para asegurar el cumplimiento satisfactorio de un proyecto, pero que no pueden ser considerados procesos de desarrollo:

- Verificación y validación
- Gestión de la configuración del software
- Desarrollo de la documentación
- Formación

Proceso del modelo del ciclo de vida software	• Modelo del ciclo de vida software	
Procesos de gestión del proyecto	• Inicio del proyecto • Control y seguimiento del proyecto • Gestión de la calidad software	*El* Inicio del proyecto *sirve para crear y mantener el marco del proyecto.* *Los otros dos procesos se ejecutan a lo largo de la vida del proyecto para asegurar el nivel apropiado de la gestión del proyecto y el cumplimiento de las actividades obligatorias.*
Procesos de predesarrollo	• Exploración de conceptos • Localización del sistema	*Procesos que deben ejecutarse antes, durante y después de la ejecución del proyecto.*
Procesos del desarrollo	• Requisitos • Diseño • Implementación	
Procesos del postdesarrollo	• Instalación • Operación y soporte • **Mantenimiento** • Retirada	
Procesos integrales	• Verificación y validación • Gestión de la configuración • Desarrollo de la documentación • Formación	*Procesos necesarios para asegurar la terminación satisfactoria del proyecto, pero que no son procesos del desarrollo.*

Tabla 6. Clasificación de los procesos definidos IEEE 1074.

3.1.1.2.1. El Proceso de Mantenimiento en IEEE 1074.

El proceso de mantenimiento se encarga de la resolución de errores (*errors*), defectos (*faults*) y fallos (*failures*) en el software. El proceso de mantenimiento consta de una única actividad, la *Reaplicación del ciclo de vida software* que, sin embargo, desencadena un gran número de trabajos.

La ejecución de esta actividad necesita las entradas indicadas en la Tabla 7:

Información de entrada	Origen	
	Proceso	Actividad
Información planificada de la gestión del proyecto software	Inicio del proyecto	Planificar la gestión del proyecto
Información reportada de la corrección del problema	Control y seguimiento del proyecto	Implementar el mecanismo de informe de problemas

Tabla 7. Entradas al proceso de mantenimiento.

La primera entrada de la actividad (*Información planificada de la gestión del proyecto software*) debió ser generada al principio del proyecto software, y en ella deben estar definidas las etapas que deben ser ejecutadas para culminar la intervención de mantenimiento. Debe contener las siguientes informaciones:

- Método para almacenar, dirigir y gestionar los informes de problemas.
- Tipos de severidad.
- Método para verificar la resolución de problemas.
- Metodologías y herramientas que deben ser utilizadas durante el proyecto, según el ciclo de vida software utilizado.

La segunda entrada (*Información reportada de la corrección del problema*) es un informe del problema, que debe haber sido cumplimentado conforme al Plan de Informes y Resolución de Problemas.

A partir de ambos documentos de entrada se genera un documento de *Recomendaciones de mantenimiento*, que sale de esta actividad y es tomado como entrada por el proceso de *Exploración de Conceptos* (Tabla 8). A continuación, se evalúan y refinan los requisitos del cambio y se introducen en una *Declaración de necesidades*. Entonces, como también ocurre en ISO/IEC 12207, se realiza una "llamada" al proceso de desarrollo.

Información de salida	Destino	
	Proceso	Actividad
Recomendaciones de mantenimiento	Exploración de conceptos	Identificar ideas o necesidades

Tabla 8. Salidas del proceso de mantenimiento.

Gráficamente, una intervención de mantenimiento de IEEE 1074 se puede describir del modo ilustrado en la Figura 11:

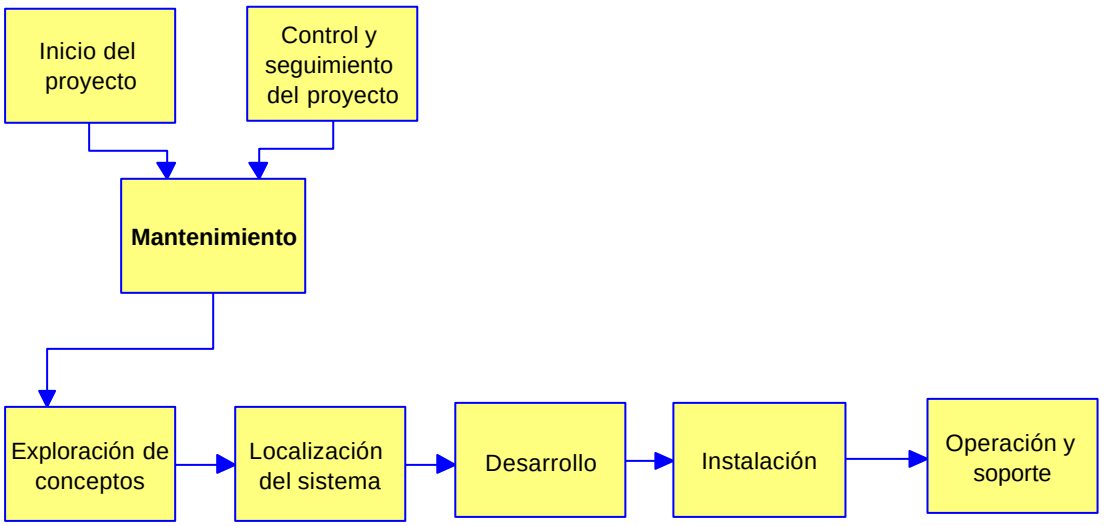


Figura 11. Actividades del mantenimiento en IEEE 1074.

3.1.1.2.2. Comentario

IEEE 1074 adolece de un método claro de mantenimiento, ya que, como hemos visto, para proporcionar la entrada *Información planificada de la gestión del proyecto software* a la única actividad de que consta este proceso, es preciso tener previamente definida la metodología de mantenimiento que se va a utilizar.

Por otro lado, se debe precisar la metodología de mantenimiento en el proceso *Inicio del proyecto*; sin embargo, IEEE 1074 no proporciona guías para definir tal metodología, que se supone debe ir integrada en el ciclo de vida elegido para el proyecto. Por otro lado, tampoco define ninguna métrica para el control del proceso, ni considera la posibilidad de externalizar servicios software.

3.1.2. ESTÁNDARES PARA MANTENIMIENTO

En esta sección se analizan el estándar IEEE 1219 y el borrador del futuro estándar ISO/IEC 14764.

3.1.2.1. El Estándar IEEE 1219 para Mantenimiento

Este estándar IEEE (1992b) describe "un proceso iterativo para gestionar y ejecutar actividades de mantenimiento de software [...]. Es aplicable tanto a la planificación del mantenimiento de software que está todavía en desarrollo como a la planificación y ejecución de actividades de mantenimiento de software para productos software existentes".

Tal proceso iterativo consta de siete fases para actividades de mantenimiento de software:

1. Identificación y clasificación del problema o la modificación.
2. Análisis.
3. Diseño.
4. Implementación.
5. Pruebas del sistema.
6. Pruebas de aceptación.
7. Entrega.

Para cada una de estas fases, el estándar especifica sus entradas y salidas, mecanismos de control y métricas, además de describirse su método de ejecución.

3.1.2.1.1. Fases del Proceso de Mantenimiento

Identificación y clasificación del problema o la modificación

Tras recibir una solicitud de modificación, si ésta es aceptada, es clasificada como de mantenimiento correctivo, perfectivo, adaptativo o de emergencia. A continuación, se realiza un estudio preliminar de su alcance, y se la sitúa en la cola de prioridad de solicitudes. A cada petición de modificación se le asigna un identificador único y se recogen determi-

nadas métricas (Número de omisiones en la solicitud de modificación, Número de solicitudes recibidas, Número de solicitudes duplicadas y Tiempo empleado en la validación del problema).

Análisis

A partir de la solicitud de modificación, de la documentación del sistema y del proyecto, y quizás utilizando alguna información histórica, se realizan dos informes (uno de factibilidad y otro que contenga un análisis detallado), cuyos contenidos mínimos se especifican en el estándar. Del mismo modo, se recogen también algunas métricas: Cambios de requisitos, Tasa de errores en la documentación, Esfuerzo por área funcional, Tiempo transcurrido (planificado) y Tasa de errores generados por prioridad y tipo.

Diseño

En esta fase, se estudia la documentación producida en la fase de Análisis junto a la documentación del proyecto y el sistema, el código fuente y las bases de datos, y se diseña la modificación. Igualmente, se modifica la documentación de los módulos afectados y se genera una lista especificada de documentos para la siguiente fase y se recogen diferentes métricas (Complejidad del software, Cambios de diseño, Esfuerzo por área funcional, Tiempo transcurrido, Cambios en los planes y procedimientos de prueba, Tasa de errores generados por prioridad y tipo, Número de líneas de código añadidas, borradas, modificadas y probadas y Número de aplicaciones).

Implementación

Tomando como entradas las salidas de la fase de diseño, junto al código fuente, bases de datos y la documentación del sistema y el proyecto, se actualizan los productos software (código fuente, bases de datos, documentos de diseño, de prueba, manuales de usuario y de formación). Tras las modificaciones, se somete a pruebas unitarias y de integración a los componentes modificados, incluyendo pruebas de no regresión. Las métricas que se recogen son la Funcionalidad o volumen (en puntos-función o líneas de código) y la Tasa de errores por prioridad y tipo.

Pruebas del sistema

Las pruebas del sistema deben realizarse sobre el sistema completamente integrado, debiendo incluirse la realización de pruebas funcionales, de interfaz y de regresión, así como una revisión de la facilidad de las pruebas para estimar la preparación para las pruebas de aceptación. En cuanto a métricas, únicamente se propone recoger la Tasa de errores por prioridad y tipo (generados y corregidos).

Pruebas de aceptación

Este tipo de pruebas deben ser realizadas por el cliente, el usuario o, en otro caso, por una tercera organización designada por el cliente, debiendo recogerse como única métrica la Tasa de errores por prioridad y tipo (generados y corregidos).

Entrega

En esta fase se realiza la instalación del software modificado. Con anterioridad, se debe notificar a los usuarios la puesta en funcionamiento de la nueva versión del producto. La única métrica propuesta se refiere a los Cambios en la documentación

3.1.2.1.2. Comentario

Este estándar aconseja clasificar cada solicitud de modificación según su tipo de mantenimiento. Sin embargo, deja a continuación el mismo camino para todas las solicitudes, sean del tipo que sean. Incluye, no obstante, un apéndice informativo, no normativo, que "no forma parte del estándar" (IEEE, 1992b, p. 19), en el cual se sugiere la estrategia de "priorizar por conjunto las solicitudes de modificación" cuando el sistema es nuevo y se requiere un esfuerzo de mantenimiento significativo.

Por otra parte, y a pesar de que IEEE 1219 propone controlar ciertos atributos del software mantenido, se echan de menos las métricas que puedan utilizarse para obtener tales medidas en cada una de sus siete fases, así como sugerencias acerca de cómo valorar esos atributos. Tampoco se propone la recolección de métricas para bases de datos.

Por otro lado, el proceso definido no es un proceso global, en el sentido de que no recoge ninguna actividad para preparar o definir el proceso, ni tampoco incluye guías para la integración de la externalización.

3.1.2.2. Borrador del futuro estándar ISO/IEC 14764 para Mantenimiento de software

Este borrador (ISO/IEC, 1999) es parte de la familia de documentos del estándar internacional ISO/IEC 12207 (ISO/IEC, 1995), desarrollando el Proceso de Mantenimiento de este último. El borrador no especifica cómo realizar las actividades y tareas del Proceso de Mantenimiento, sino que únicamente especifica cuáles deben ser.

3.1.2.2.1. Tipos de mantenimiento

En el borrador se definen los siguientes tipos de mantenimiento:

1. Mantenimiento correctivo: modificación realizada sobre un producto software después de su entrega para corregir problemas.
2. Mantenimiento de mejora: cambio realizado en el software que no es una corrección.
3. Mantenimiento adaptativo: modificación de un producto software, después de su entrega, para conservarlo utilizable en un entorno cambiado o cambiante.

4. Mantenimiento preventivo: modificación de un producto software después de su entrega para detectar y corregir errores latentes del software, antes de que lleguen a ser errores efectivos.

3.1.2.2.2. Consideración de la mantenibilidad

Los requisitos de mantenibilidad deberían ser incluidos en el Proceso de Adquisición de ISO/IEC 12207, de manera que sus niveles puedan ser evaluados durante el Desarrollo. Deben considerarse diferentes aspectos de mantenibilidad, relativos a *Lenguaje de programación utilizado, Análisis de requisitos software, Diseño de la arquitectura software, Diseño detallado del software, Codificación* y *Pruebas*.

3.1.2.2.3. Descripción del Proceso de Mantenimiento

Las actividades y tareas del Proceso de Mantenimiento son responsabilidad de la Organización de mantenimiento, definida en ISO/IEC (1995). Las actividades que componen este proceso son las ya descritas en éste (Implementación del proceso, Análisis de problemas y modificaciones, implementación de la modificación, Revisión y aceptación del mantenimiento, Migración y Retirada); sin embargo, el borrador precisa aún más la estructura del proceso, como se muestra en la Figura 12.

Actividad 1. Implementación del proceso

La Organización de mantenimiento establece los planes y procedimientos que serán ejecutados durante el proceso. Consta de tres tareas:

1. Desarrollar, documentar y ejecutar planes y procedimientos para dirigir las actividades y tareas del Proceso de Mantenimiento.
2. Definir procedimientos para recibir, almacenar y controlar los informes de problemas y solicitudes de modificación de los usuarios, y proporcionar a éstos retroalimentación.
3. Implementar o establecer una interfaz con el Proceso de Gestión de Configuración, para gestionar las modificaciones del sistema existente.

Actividad 2: Análisis de problemas y modificaciones

En esta actividad, la Organización de mantenimiento analiza las peticiones de modificación e informes de problemas, replica o verifica el problema, desarrolla alternativas para implementar la modificación, documenta los resultados y opciones de implementación, y obtiene la aprobación para ejecutar la alternativa seleccionada.

En la tarea de análisis de la petición de modificación o informe del problema, la Organización de mantenimiento debe indicar el tipo de mantenimiento de la futura intervención, que será uno de los cuatro indicados en este borrador. Igualmente, se debe indicar la magnitud del cambio (por ejemplo: tamaño de la modificación, coste, tiempo necesario).

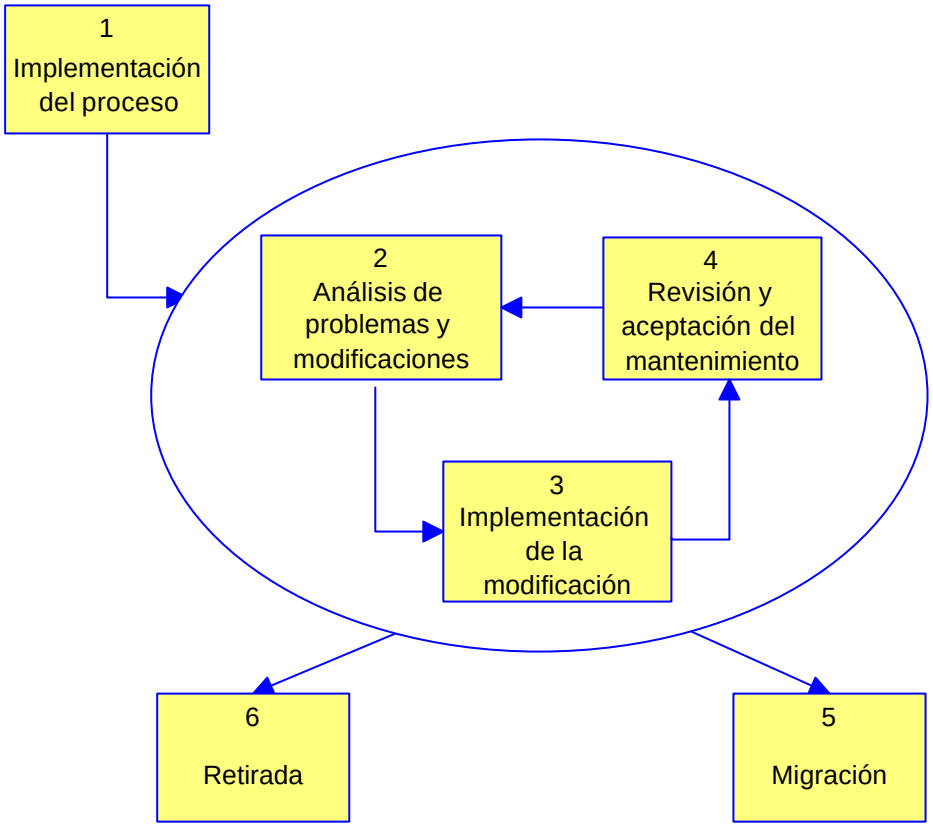


Figura 12. Proceso de mantenimiento de ISO/IEC (1999)

Actividad 3: Implementación de la modificación

La Organización de mantenimiento desarrolla y prueba las modificaciones realizadas sobre el producto software. Las tareas que componen esta actividad son las siguientes:

1. Dirigir y determinar análisis para determinar qué documentación, unidades de software y versiones necesitan ser modificadas.
2. Entrar en el Proceso de Desarrollo de ISO/IEC 12207 (IESO/IEC, 1995) para implementar las modificaciones.
3. Realizar una revisión conjunta de la modificación realizada.
4. Documentar y asegurar la calidad de la modificación.

Actividad 4: Revisión y aceptación del mantenimiento

En esta actividad, se asegura que las modificaciones realizadas sobre el sistema son correctas. Para su ejecución, deben realizarse revisiones con la organización que autorizó la modificación y obtener la aprobación correspondiente.

Actividad 5: Migración

Esta actividad se realiza cuando el sistema tiene que ser modificado para ser ejecutado en un nuevo entorno.

Actividad 6: Retirada

Se realiza cuando el producto software ha alcanzado el final de su vida útil.

3.1.2.2.4. Comentario

En caso de aprobarse por ISO/IEC, este borrador constituirá una buena ayuda para las organizaciones de mantenimiento de software. Sin embargo, y aunque aconseja clasificar las peticiones de modificación e informes de problemas en los diversos tipos de mantenimiento que distingue, no proporciona los conjuntos específicos de actividades y tareas que deben realizarse según dicho tipo. Por otro lado, y aunque indica que el proceso debe evaluarse con la ayuda de métricas, no especifica cuáles de ellas deben ser utilizadas, ni tampoco suministra técnicas "verticales" para ser utilizadas a lo largo del proceso. La externalización está implícitamente considerada, al igual que en ISO/IEC 12207, mediante la aplicación del Proceso de Adaptación de este estándar al Proceso de Mantenimiento definido, pero no explícitamente, puesto que no se contempla ni como proceso ni en forma de conjunto de actividades y tareas.

3.1.3. PROPUESTAS ESPECÍFICAS PARA LA GESTIÓN DEL PROCESO DE MANTENIMIENTO

Existe una llamativa ausencia de modelos de procesos para la gestión y ejecución del mantenimiento de software. Como prueba de ello, mostramos en la siguiente tabla el tema principal de los trabajos publicados en las actas de las dos últimas conferencias (1998 y 1999) Internacional (*International Conference on Software Maintenance*) y Europea (*European Conference on Software Maintenance and Reengineering*, antes *Euromicro Conference on Software Maintenance and Reengineering*) sobre mantenimiento de software, así como durante los años 1998, 1999 y primer bimestre de 2000 en la revista más prestigiosa en este campo, el *Journal of Software Maintenance, Research and Practice*. Como se observa, sólo 4 de las 196 comunicaciones publicadas presentan algún trabajo relacionado con la gestión metodológica del mantenimiento, estando una de ellas relacionada con esta tesis doctoral (Polo et al., 1999a).

El artículo indicado del Journal of Software Maintenance (Schach y Tomer, 2000) presenta un método para desarrollo de productos software con alto grado de mantenibilidad. Este mantenimiento "proactivo" queda fuera de los objetivos de esta tesis, razón por la cual no analizamos esta propuesta, más propia de los nuevos desarrollos. No obstante, existen estudios que indican que, sorprendentemente, el esfuerzo empleado en dar alta mantenibilidad a un producto software no siempre se ve recompensado por una disminución en su posterior esfuerzo de mantenimiento (Swanson, 1999).

Tras la tabla, analizamos las otras dos propuestas publicadas en la 3rd European Conference on Software Maintenance and Reengineering, así como la consideración del proceso de mantenimiento en la Metodología Métrica Versión 3 (Consejo Superior de Informática, 2000).

	ICSM	ECSMR	Journal...	Total
"Keynotes" y estado del arte	4	1	3	8
Identificación de rutinas, objetos o componentes	9	4	6	19
Métricas y medidas	10	8	2	20
Casos de estudio	20	2	4	26
Herramientas	4	6	4	14
Técnicas de prueba	4	2	4	10
Modelos de estimación o previsión	4	2	1	7
Mantenibilidad	3	2	0	5
Reingeniería, Ingeniería inversa y reestructuración	14	19	5	38
Comprensión de programas	11	7	3	21
COTS	4	0	1	5
Metodologías de Mantenimiento	0	3 (una de ellas Polo et al., 1999a)	1	4
Otros	10	6	5	21
Total.....	97	62	39	198

Tabla 9. Tema principal de las comunicaciones presentadas en las últimas publicaciones más importantes sobre mantenimiento.

3.1.3.1. Evolución de requisitos de Lam y Loomes (1998).

Lam y Loomes presentan un método para controlar la evolución de requisitos, a partir del modelo mostrado en la Figura 13. Los autores identifican cuatro tipos básicos de cambios sobre el software (de entorno, de requisitos, de punto de vista y de diseño). En general, los cambios de entorno generan nuevos requisitos, que deben ser controlados conforme a estos puntos:

1. Evolución del modelo, conforme a algún modelo de evolución del software (para lo cual, los autores citan como utilizable el estándar IEEE 1219, comentado en la página 58 y siguientes).
2. Análisis del cambio que, en el momento de la publicación, los autores no habían desarrollado y se encontraba en investigación.

3. Evaluación del impacto, para lo que proponen utilizar algún método de estimación de esfuerzo de mantenimiento.
4. Estimación de riesgos del cambio.
5. Controlar los cambios múltiples.
6. Mantenimiento de las relaciones y la integridad entre los diferentes componentes del sistema.
7. Soporte automático, para lo que proponen la utilización de herramientas de gestión de configuración y de gestión de requisitos.

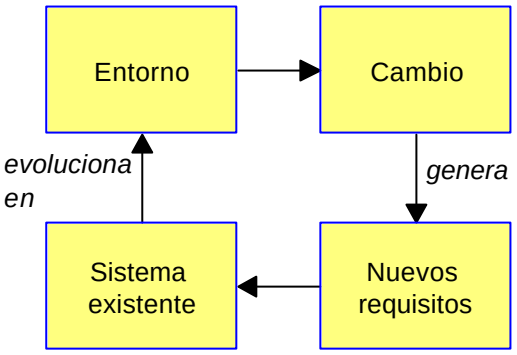


Figura 13. Modelo de evolución de requisitos.

3.1.3.1.1. Comentario

En palabras de los propios autores, el trabajo presentado “discute el problema de la evolución de requisitos y se encuentra en un estado formativo”. No es, por tanto, un modelo de proceso de mantenimiento, aunque hemos querido discutir la propuesta en esta sección debido a que, en cierto modo, presenta un cierto enfoque metodológico, aplicable durante la gestión de este proceso.

3.1.3.2. Modelo del proceso de mantenimiento y reingeniería de Stoecklin, Williams y Stoecklin (1998).

El enfoque de gestión del proceso de mantenimiento presentado en este trabajo puede resumirse en el siguiente algoritmo, propuesto por los autores:

```
Si el mantenimiento es urgente Entonces
    Repetir
        Analizar el problema
        Inspeccionar el código
        Modificar el código
        Realizar pruebas y entregar
    Hasta que el cambio sea satisfactorio
```

```
Si no
    Entrevistar a expertos para determinar los requisitos de la mo-
    dificación
    Definir el impacto del mantenimiento sobre los objetos de con-
    figuración del software
    Adaptar un plan inicial de proceso de mantenimiento basado en
    las tareas del proceso del mantenimiento
    En caso necesario, modificar el plan para planificar recursos
    Implementar la modificación utilizando el plan del proyecto
Fin si
```

3.1.3.2.1. Comentario

El trabajo no presenta realmente una modelo del proceso de mantenimiento, sino, posiblemente, una porción de lo que es la práctica habitual en las organizaciones de mantenimiento.

3.1.3.3. El ciclo de vida del mantenimiento de Kung y Hsu (1998).

Kung y Hsu (1998) proponen un modelo de ciclo de vida del mantenimiento con la intención de "ayudar en la planificación del proceso". El ciclo de vida propuesto consta de cuatro etapas, que dibujan el histórico de solicitudes de modificación de una aplicación. Las etapas son *Introducción*, *Crecimiento*, *Madurez* y *Declive*.

3.1.3.3.1. Etapas del ciclo de vida del mantenimiento.

Introducción.

Durante esta etapa, el uso del sistema es bajo porque el número de usuarios que lo conocen es pequeño, y las demandas de mantenimiento se refieren casi exclusivamente a peticiones de soporte técnico (*help-desk*).

Crecimiento.

Cada vez se va incorporando al uso del sistema un mayor número de usuarios, lo que implica un crecimiento en las solicitudes de mantenimiento para corrección de errores. Esto implica un aumento en el personal necesario para ejecutar las intervenciones.

Madurez.

El número de errores en el sistema se estabiliza, y ahora lo que los usuarios demandan son mejoras en la aplicación, por lo que crece el número de solicitudes de perfectivo. Durante esta etapa, los equipos de mantenimiento se dedican al aumento de las prestaciones del software y a intentar prologar la vida del sistema.

Declive.

En esta etapa, los usuarios demandan el uso de nuevas tecnologías y entornos, por lo que se tiende a disminuir las peticiones de modificación de todo tipo y, o bien a sustituir la aplicación, o bien a integrarla con otros sistemas modernos.

3.1.3.3.2. Representación gráfica.

Los autores representan gráficamente la distribución de peticiones de modificación comentada en las cuatro etapas anteriores con un gráfico similar al mostrado en la Figura 14:

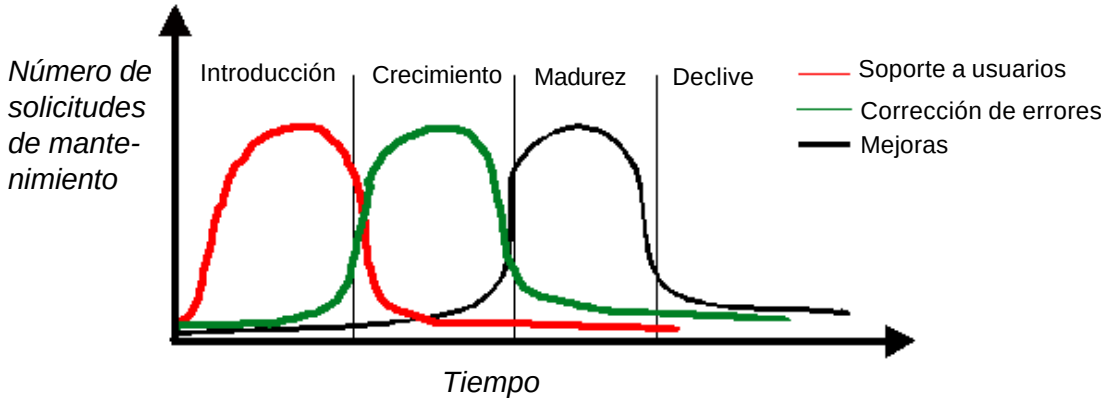


Figura 14. Representación del ciclo de vida del mantenimiento según los autores.

3.1.3.3.3. Resultados mostrados.

Tras la propuesta del modelo, los autores del artículo presentan la distribución de llegadas de solicitudes de mantenimiento en dos proyectos, uno de 67 meses de duración, con 654 solicitudes de mantenimiento, y otro de 18 meses, que sufrió 917 peticiones, observándose curvas de distribución más o menos parecidas a las de la Figura 14.

3.1.3.3.4. Comentario

Más que un modelo de ciclo de vida del mantenimiento, que es la idea deducible del titulo de la comunicación, el trabajo presentado es una muestra de la distribución de solicitudes de mantenimiento a lo largo de la vida de un sistema.

Los autores afirman que las líneas de trabajo futuras irán relacionadas con el siguiente comentario, pronunciado por el director de uno de los dos proyectos: “Hay cierto retroceso cuando instalamos módulos de actualización. Podemos anticipar que ocurrirán problemas durante cierto tiempo: después de instalar una nueva versión con cambios importantes, se experimenta un incremento en el número de solicitudes de asistencia al usuario y correcciones, relacionadas con esos cambios”. Esta observación también ha sido mencionada en Basili et al. (1996) y Schneidewind (1998). Por otra parte, los datos del proyecto de 67 meses han sido también utilizados por Calzolari et al. (1998) para construir el modelo de esfuerzo predictivo que analizamos en la sección 3.5.3 (página 81).

3.1.3.4. Metodología Métrica Versión 3.

En esta reciente versión de la metodología Métrica (Consejo Superior de Informática, 2000) se incluye el Mantenimiento de Sistemas de Información (MSI) como uno de sus proce-

sos principales. Sin embargo, “Métrica V3 sólo reflejará los aspectos del Mantenimiento que tengan relación con el Proceso de Desarrollo”, ya que “esta metodología está dirigida principalmente al proceso de Desarrollo del software”.

En Métrica V3 se distinguen cuatro tipos de mantenimiento (correctivo, evolutivo, adaptativo y perfectivo), si bien se excluyen de su ámbito de aplicación estos dos últimos, para los que “se precisaría el desarrollo de un tipo de metodología específica para resolver su cometido”. Evidentemente, tal «metodología específica» muy bien podría tratarse de MANTEMA.

3.1.4. EXTERNALIZACIÓN DEL MANTENIMIENTO.

La externalización de actividades propias de los procesos del ciclo de vida software constituye en la actualidad una actividad en plena expansión económica (Rao et al., 1996). En efecto, según Hoffman (1997), el cuarenta por ciento de las grandes compañías de EE.UU. han externalizado al menos una de las porciones principales de sus operaciones, e incluso las organizaciones más inesperadas, como el Departamento de Defensa de EE.UU., han decidido externalizar porciones significativas de sus sistemas de información (Brower, 1999).

La externalización de todo o parte del Sistema de Información de una organización permite a ésta obtener una serie de beneficios (Klepper y Jones, 1998; De Looff, 1997), que podemos dividir en dos grandes grupos: estratégicos y financieros.

3.1.4.1. Objetivos estratégicos.

Los primeros se dirigen tanto a la mejora de la posición de la organización en el mercado, como al incremento de la calidad del servicio prestado a sus clientes, lo cual repercute en el valor añadido que se les proporciona. La externalización del S.I. ayuda a la consecución de estos objetivos porque:

- *Los gestores centran su atención en el “core business”*, permitiéndoles concentrarse en los aspectos estratégicos del S.I.
- *Se permite un mayor control del proceso*, ya que las intervenciones de mantenimiento pasan con la externalización a ser intervenciones planificadas (puesto que han sido delegadas en una empresa comprometida con esta tarea) y el mantenimiento deja de formar parte de la vida diaria de la organización. Esto permite dedicar los recursos más cualificados a las actividades más relacionadas con los objetivos del negocio.
- *Libera recursos para desarrollos estratégicos*, lo que también permite mayor flexibilidad en los equipos de trabajo.

3.1.4.2. Objetivos financieros.

Del mismo, la externalización ayuda a conseguir los objetivos financieros porque permite:

- *Disminuir y controlar los costes,* mediante:

- El control riguroso de las actividades de mantenimiento, y el conocimiento de las partidas presupuestarias dedicadas a él.
- El ahorro en formación y gestión de personal, ya que estos costes han sido desplazados al suministrador del servicio.
- La transformación de costes fijos en variables, ya que en el contrato se pacta generalmente un número mínimo de intervenciones de un tipo de mantenimiento.
- El conocimiento de los costes antes de ejecutar intervenciones no pactadas en el contrato, lo que permite realizar análisis previos.
- La disminución del mantenimiento correctivo durante el periodo de externalización, a partir de un compromiso de la organización suministradora de realizar simultáneamente trabajos de preventivo.

- *Incrementar la productividad,* mediante:
  - La implantación de circuitos que, entendiendo las intervenciones como el conjunto de actividades que deben realizarse desde que se detecta la necesidad de la intervención hasta que la modificación es pasada al entorno de producción, definen claramente *quién* hace *qué* cosa, *cuándo* y *cómo*.
  - La eliminación de las interferencias del mantenimiento con los recursos dedicados a nuevos desarrollos.
  - La producción de beneficios, porque ahora el mantenimiento es hecho por un especialista, un *socio tecnológico* especial cuya competencia esencial se centra en esta actividad y que, para mantenerse en el mercado, invierte en Investigación y Desarrollo para nuestro propio provecho.

3.1.4.3. Desventajas.

La externalización, sin embargo, tiene también una serie de desventajas que deben ser tomadas en cuenta por los clientes:

- *Pérdida de control y pérdida de una fuente de aprendizaje,* porque una actividad interna pasa a ser externa.
- *Posibles dependencias del suministrador.*
- *Incremento de costes por cautividad.*
- *Variaciones en la calidad del producto entregado al usuario final.*
- *Problemas entre el personal,* porque éstos pierden parte de las funciones que venían realizando.

3.1.4.4. Situación actual del tema

La externalización del sistema de información puede dar a una organización un alto nivel de flexibilidad, permitiendo a sus miembros la dedicación a los aspectos que forman parte del

core-business de la empresa. La idea de la externalización puede resumirse en la siguiente frase: “Quédate lo mejor, hazlo mejor y externaliza el resto” (Klepper y Jones, 1998).

En general, existen en la literatura gran cantidad de consejos relativos a cómo seleccionar una empresa suministradora, cómo negociar el contrato o cómo gestionar la relación contractual cuando ésta ya está en marcha, incluyendo diferentes casos de estudio ilustrativos. Por el contrario, en ninguna referencia de la numerosa bibliografía manejada se trata la integración de la externalización en el proceso de mantenimiento. Además, prácticamente todos los trabajos publicados estudian la externalización desde el punto de vista de la organización que adquiere el servicio, pero no desde el de la suministradora: además de las desventajas que la externalización encierra para la empresa cliente, la asunción de un contrato de externalización también puede encerrar riesgos para la suministradora (la infravaloración de los recursos necesarios, por ejemplo).

Como se verá en el capítulo cuarto, en esta tesis consideramos detalladamente el establecimiento y finalización de la prestación del servicio de mantenimiento dentro de la metodología definida. Se proponen además varias técnicas, susceptibles de ser utilizadas al inicio de la relación entre ambas organizaciones, así como un conjunto de indicadores de nivel servicio específicos para la externalización del proceso de mantenimiento.

3.1.5. TABLA COMPARATIVA

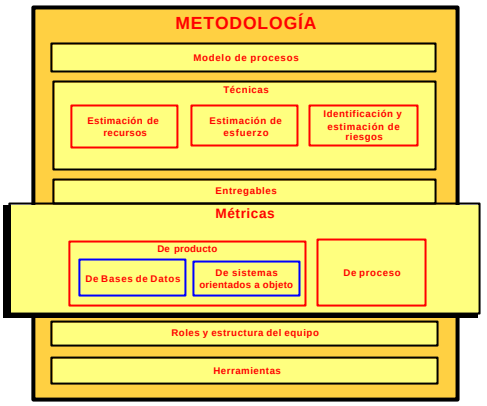
En la Tabla 10 se realiza una comparación, desde el punto de vista metodológico, de los diferentes modelos de procesos analizados en esta sección. En su primera fila se han situado algunas de las características que, de acuerdo a la Figura 4, debe reunir una metodología. No hemos incluido a MANTEMA en la tabla porque, en este punto de la tesis, el lector puede no conocer aún sus características. Sin embargo, conforme se avance en la lectura del capítulo cuarto, podrá ir comprobándose el nivel de cumplimiento de las distintas características.

	Consideración de la externa-lización	Manteni-miento proactivo	Definición de métricas	Consideración de técnicas para estim ación de recursos	Indicación de técnicas para ejecutar las tareas	Consideración de diferentes tipos de mante-nimiento	Consideración de riesgos de externaliz ación	Modelado organi-zacional	Costes de la adaptación
ISO/IEC 12207	No	No	No, aunque recomienda su uso	No	No	Se menciona	No	Se definen cinco partes principales	Altos
IEEE 1074	No	No	No	No	No	Se menciona	No	No	Altos
ISO/IEC 14764	No	No	No, aunque recomienda su uso	No	No	Se menciona	No	Las mismas que en ISO/IEC 12207	Medios
IEEE 1219	No	No	No, aunque menciona propiedades que deben controlarse	No	No	Se menciona	No	No	Bajos, pero poco versátil
Lam y Loomes (1998)	No	No	No	No	No	No	No	No	Altos (la pro-puesta se en-cuentra en un estado muy inicial)
Stoecklin et al. (1998)	No	No	No	No	No	No	No	No	No procede, puesto que no es realmente una metodología de mantenimiento
Kung y Hsu (1998(	No	No	No	No	No	No	No	No	No procede, puesto que no es realmente una metodología de mantenimiento
Schach y Tomer (2000)	No procede	Sí	No	No procede	No procede	No procede	No	No	No procede
Métrica V3	En parte (con-sidera indicado-res de servicio)	No	No, aunque recomienda su uso	No	Sí	Distingue cuatro tipos, pero es aplicable a sólo dos	En parte (habla de pactar indica-dores de servicio)	Sí	Medios-altos (depende del tipo de mantenimien-to)

Tabla 10. Comparación de los diferentes enfoques propuestos para mantenimiento

3.2. METROLOGÍA

De acuerdo con Fenton y Pfleeger (1997), una *medida* es el "número o símbolo asignado para caracterizar un atributo de una entidad, mediante una correspondencia entre el mundo real y el formal", mientras que una *métrica de calidad* es una función "que toma como entrada cierta información del software que se está midiendo, y que devuelve como salida un valor numérico sencillo, que es interpretado como el grado en que el producto software posee un atributo dado que afecta a su calidad" (IEEE, 1992).



Dependiendo del propósito de la medida que realicemos, distinguiremos entre *métricas de producto* y *métricas de proceso*.

- Para Henderson-Sellers (1996), una *medida de producto* es un valor que representa una característica del producto en un instante de tiempo claramente determinado. La medida puede realizarse tanto en la fase de diseño como durante el mantenimiento, pero no contiene sino una descripción instantánea del atributo medido. Una *métrica de producto* será, por tanto, una función que nos devolverá el valor de un determinado atributo del software en un momento dado.
- Una *medida de proceso* sirve para evaluar diferentes aspectos del proceso software, permitiendo a sus responsables efectuar previsiones de costes y esfuerzo, detectar anomalías o mejorar actividades (Henderson-Sellers, 1996). A partir de este concepto, podemos definir una *métrica de proceso* como una función que devuelve un valor relativo al proceso software que se está midiendo.

Suelen utilizarse los valores devueltos por métricas de producto en forma de medidas predictivas, de manera que el análisis (matemático, por analogía, por juicio experto...) que se le aplica a dichos valores constituye una implementación de una métrica de proceso.

3.2.1. MÉTRICAS

Desde el famoso número ciclomático de Thomas McCabe (McCabe, 1976), los investigadores han propuesto multitud de métricas para la medición de diferentes atributos de productos software (sobre todo programas). En un paquete informático divulgativo sobre medición del software, Zuse ha recopilado más de mil métricas de producto entre las que se incluyen métricas de complejidad, tamaño, cohesión, longitud, etc. Sin embargo, existe una ausencia de investigación sobre métricas para bases de datos (Sneed y Foshag, 1998). La mayoría de las métricas propuestas se han centrado en la medida de la complejidad, calidad o mantenibilidad de programas, a pesar de que en los sistemas de información actuales, la base de datos constituye un componente crucial, cuyos niveles de complejidad cobran cada día mayor importancia, sobre todo con la aparición de las bases de datos de tercera generación.

En este sentido, Batini, Ceri y Navathe (1992) proponen un conjunto de atributos de calidad (compleción, corrección, minimalidad, expresividad, legibilidad, autoexplicación, extensibilidad y normalidad), pero no proporcionan para ellos ninguna medida numérica. Del mismo modo, Reingruber y Gregory (1994) dan ciertas normas para construir diseños de bases de datos de alta calidad, pero sin referencias cuantitativas. Kesch (1991) constituye una excepción en este campo, ya que propone una serie de métricas para evaluar la calidad de esquemas conceptuales de bases de datos, pero a través de largos procesos manuales difícilmente automatizables. Gray et al. (1991), por otra parte, proponen un índice de normalización. Muy recientemente, Sneed y Foshag (1998) han propuesto factores cuantitativos, cualitativos y de complejidad para bases de datos heredadas, y Muller (1999) ha propuesto métricas para esquemas conceptuales. En Piattini et al. (2000), analizamos más en profundidad los temas relativos a la calidad de bases de datos y a las diferentes propuestas de métricas.

3.2.1.1. Marcos formales.

En relación sobre todo con las métricas para programas, la enorme cantidad de métricas propuestas ha provocado que los diversos atributos que se pretenden medir se aparezcan más bien como ideas o conceptos más o menos difusos.

Al menos desde 1984, la comunidad científica ha intentado responder con diversos intentos de formalización de conceptos: por lo general, en vez de definiciones rigurosas pero ambiguamente verbales de términos como *complejidad* o *tamaño*, variados autores han propuesto diferentes conjuntos de propiedades que caracterizan a estos conceptos (Prather, 1984; Weyuker, 1988; Zuse, 1991; Kitchenham et al., 1995; Briand et al. 1996). De acuerdo con estos *marcos formales*, las métricas que no se ajusten, por ejemplo, a las propiedades definidas para la complejidad, no deberán ser consideradas de complejidad.

A tenor de lo dicho al principio de este epígrafe, existen pocos atributos del software que aún no se hayan medido de alguna manera. Sin embargo, la aplicación de los marcos formales ha provocado también cierto impacto en los investigadores y, así, aunque Chidamber y Kemerer (1994) definen su conjunto de métricas a partir de diversos conceptos ontológicos, luego comparan cada una de ellas con el marco de Weyuker; recientemente, Poels (1997) ha presentado una métrica para sistemas orientados a objeto que es de acoplamiento conforme a Briand, Morasca y Basili.

3.2.2. MÉTRICAS PARA EXTERNALIZACIÓN DE SERVICIOS SOFTWARE.

De Vogel (1999) es una de las pocas referencias que relacionan métricas y externalización de servicios software. Este autor especifica cinco “dimensiones” para medir y controlar la prestación de servicios software por parte de terceros. Para cada dimensión, el autor propone una serie de posibles medidas, que recogemos en la Tabla 11.

Este autor, además, resalta la importancia de contratar Indicadores del Nivel de Servicio adecuados entre las dos organizaciones.

	Dimensión	Medida
1	Coste	Coste por punto-función de cada aplicación
2	Calidad funcional (cumplimiento de los requisitos funcionales por parte de las aplicaciones)	Disponibilidad del proveedor (días por semana) Accesibilidad del proveedor (horas por día) Tiempo de respuesta
	Calidad técnica (cumplimiento de las especificaciones por parte de las aplicaciones)	Defectos residuales por punto-función Tiempo medio ente fallos Complejidad esencial Complejidad ciclomática Código muerto Código no estructurado Cumplimiento de estándares de codificación
3	Productividad del proveedor (informe de problemas)	Tiempos medio y máximo de respuesta a informes de problemas Tiempo máximo para informar de la corrección de un problema
	Productividad del proveedor (resolución de problemas)	Tiempo medio de reparación Días de calendario para reparación Mantenimiento de un histórico
	Productividad del proveedor (medida general)	Horas por periodo y tipo de actividad Puntos-función soportados Cartera de aplicaciones
4	Satisfacción del cliente (con respecto al proyecto)	Fiabilidad: ¿estaba el proyecto a tiempo? ¿Cumplía los requisitos del negocio? Profesionalidad y disponibilidad del personal Utilidad, claridad y compleción de la documentación Satisfacción general
	Satisfacción del cliente (periódica)	Fiabilidad: ¿tienen los productos y servicios alta calidad? Tiempo de respuesta: ¿responden a tiempo?
	Satisfacción del cliente (retorno de la inversión)	¿Merece la pena la inversión en externalización?
5	Personal	Disponibilidad del personal Horas de formación

Tabla 11. Dimensiones y medidas para controlar la prestación de servicios.

3.2.3. SITUACIÓN ACTUAL DEL TEMA

En cuanto a las métricas de producto, la tendencia actual parece ir encaminada a la validación de las métricas por partida doble (esto es, teórica y experimental), aunque probablemente todavía queden algunos años para lograr un consenso respecto de las propiedades que caracterizan a las métricas de cada atributo. Algunas de las contestaciones habidas en los debates mencionados han demostrado la falta de compleción de algunos marcos, con lo que seguirían verificándose observaciones como la de Cherniavsky y Smith (1991), mientras que

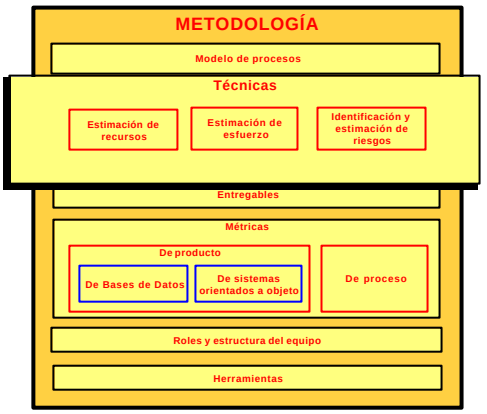
otras (o, incluso, los propios autores al presentar su marco), consideran condición necesaria, pero no suficiente, el cumplimiento de las diferentes propiedades.

Las métricas de proceso, sin embargo, han sido menos criticadas tal vez porque, como hemos mencionado, suelen aplicar alguna clase de análisis matemático a las métricas de producto para ser así utilizadas en forma de medidas predictivas. De este modo, la bondad de sus previsiones será o no aceptable dependiendo de los parámetros y ajustes estadísticos que se hayan utilizado en los experimentos.

Por último, se echan de menos, en las métricas para la externalización, métricas que estén especialmente definidas para la externalización del servicio de mantenimiento de software.

3.3. SOLUCIONES TÉCNICAS

Bajo esta denominación englobamos a las técnicas utilizables en determinados momentos del proceso de mantenimiento. Las técnicas para mantenimiento más habituales son la Ingeniería Inversa, Reingeniería y Reestructuración. Sin embargo, existen algunas otras técnicas utilizables puntualmente, como métodos para gestión de riesgos de las intervenciones de mantenimiento o algoritmos para la detección de clones, que recordamos en esta sección.



3.3.1. INGENIERÍA INVERSA

La ingeniería inversa es "el proceso de construir especificaciones formales abstractas del código fuente de un sistema heredado (*legacy*), de manera que estas especificaciones puedan ser utilizadas para construir una nueva implementación del sistema usando Ingeniería directa o *hacia delante"* (Arnold, 1992). Otros autores no indican que el nivel de partida del proceso de ingeniería inversa tenga que ser siempre el código fuente, sino que puede ser cualquier nivel dado de abstracción (Piattini *et al.,* 1996). Estos mismos autores afirman que la ingeniería inversa "recrea modelos pertenecientes a niveles superiores, ya sean orientados a datos o a procesos", que podemos asociar, respectivamente, a bases de datos y a programas. Parafraseando a Biggerstaff et al. (1994), al final del proceso de ingeniería inversa se debe poder explicar qué hace el programa, cuál es su estructura, qué efectos tiene en su contexto operacional y cuáles son sus relaciones con su dominio de aplicación.

3.3.2. REINGENIERÍA

Arnold (1992) define la reingeniería como "Cualquier actividad que:

1. Mejore la comprensión del software.

2. Prepare o mejore el propio software, normalmente para incrementar su facilidad de mantenimiento, reutilización o evolución".

En general, se entiende por reingeniería "la modificación de un producto software, o de ciertos componentes, usando para el análisis del sistema existente técnicas de ingeniería inversa y, para la etapa de reconstrucción, herramientas de ingeniería directa, de tal manera que se oriente este cambio hacia mayores niveles de facilidad en cuanto a mantenimiento, reutilización, comprensión o evolución" (Piattini et al., 1998a).

3.3.3. REESTRUCTURACIÓN

La reestructuración es la transformación de un producto software a otra forma de representación, pero sin cambiar de nivel de abstracción. En la actualidad, muchas organizaciones se ven obligadas a modificar sus aplicaciones para aprovechar las oportunidades que permiten tecnologías emergentes como Internet, la arquitectura cliente/servidor o la computación distribuida (Jahnke y Wadsack, 1999). Antes de realizar los cambios, las aplicaciones suelen ser sometidas previamente a un proceso de reestructuración, de manera que se genere un producto software equivalente al original, en el mismo nivel de abstracción, pero con mayor comprensibilidad y mantenibilidad, y menos propenso a la introducción de errores. Estas características de la nueva representación del sistema facilitan y abaratan los costes ulteriores de la modificación.

En la actualidad, la mayor parte de las técnicas de reestructuración tienen como objetivo la transformación de aplicaciones construidas hace varias décadas mediante metodologías estructuradas al paradigma orientado a objeto que, como es bien sabido, ofrece niveles muy elevados de portabilidad, reutilización y mantenibilidad (Bucci et al., 1998). Existen, sin embargo, tantas técnicas de reestructuración de sistemas como tipos de sistemas: así, por ejemplo, Sellink et al. (1999) proporcionan una estrategia para la reestructuración de sistemas transaccionales compuestos de código tanto en Cobol como en CICS, para facilitar su migración a otros entornos, como cliente/servidor. Penteado et al. (1999) presentan un método para reestructurar programas en C (generando así mismo código C), que además produce documentación orientada a objetos del sistema. Taschwer et al. (1999) explican un método y una herramienta para convertir código C a C++. Cremer (1998) presenta un método para reestructurar componentes de aplicaciones escritas en Cobol, de manera que sean utilizables en entornos distribuidos.

Además de los métodos mencionados en el párrafo anterior, que trabajan directamente sobre el código fuente, también pueden reestructurarse productos software de un nivel más alto de abstracción: Borne y Romanczuk (1998), por ejemplo, presentan un método para transformar diagramas MERISE a diagramas OMT.

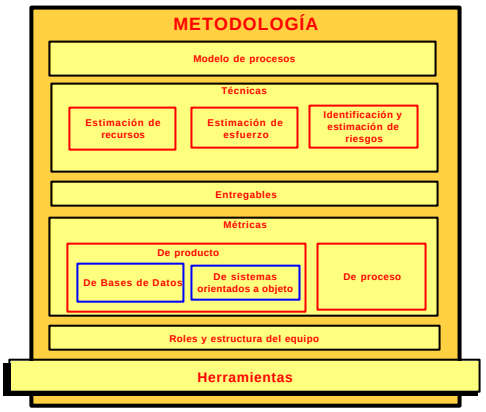
3.3.4. COMENTARIO

Todas las técnicas expuestas en esta sección son utilizables a lo largo del proceso de mantenimiento definido en la metodología MANTEMA, tal vez con algunas modificaciones cuyo

estudio no forma parte de los objetivos de esta tesis. Sin embargo, en ciertos momentos del proceso pueden necesitarse otro tipo de técnicas que no hemos encontrado descritas en la literatura. En este sentido, en MANTEMA se proponen técnicas para identificar y estimar riesgos en proyectos de externalización, desde el punto de vista de la organización suministradora del servicio, y para ayudar a la estimación de los recursos necesarios para mantenimiento no planificable.

3.4. HERRAMIENTAS PARA EL MANTENIMIENTO DE SOFTWARE

Aunque originalmente la mayoría de las herramientas CASE se idearon con otros fines, muchas de ellas pueden utilizarse también para automatizar el proceso de mantenimiento del software (Quang, 1993). Sin embargo, la mayoría de tales herramientas son herramientas *verticales,* en el sentido de que automatizan únicamente determinadas tareas del mantenimiento (estimación de costes, pruebas, reestructuración), y existen pocas herramientas *horizontales*, susceptibles de ser utilizadas a lo largo de todo el proceso. Esta opinión es corroborada por Pigoski (1997), para quien "la ausencia de un proceso de mantenimiento definido dificulta la utilización de herramientas CASE para la gestión del proceso, y las herramientas disponibles sólo son aplicables a actividades concretas del mantenimiento".



Casi todas las herramientas automáticas para mantenimiento presentadas en los últimos años son de reestructuración y reingeniería (en las dos últimas conferencias europeas sobre mantenimiento de 1998 y 1999, 13 de las 15 herramientas presentadas[1] eran de estas categorías o eran mejoradas con funcionalidades de estas clases, mientras que de las otras dos eran herramientas de verificación y estimación). Sin embargo, se comprueba cómo la gran variedad de tipos de productos software existentes (programas, bases de datos, documentos) y el carácter ampliamente heterogéneo de cada uno de ellos (multitud de lenguajes de programación; de modelos de datos, también cada uno con muchos gestores diferentes), hace que la aplicación de las herramientas que automatizan estas técnicas esté muy restringida por las características del software que se va a modificar.

[1] El número de herramientas indicado no entra en conflicto con el número de referencias mencionado en la fila "Herramientas" de la Tabla 9, ya que en ésta se refleja el número de referencias cuyo *tema principal* es "Herramientas para mantenimiento". Puede haber, por tanto, referencias que presentan alguna herramienta como complemento del tema principal que desarrollan (p.ej., una técnica de Ingeniería inversa).

3.4.1. TIPOS DE HERRAMIENTAS CASE

Las herramientas CASE se pueden agrupar en varias categorías según su funcionalidad (Pressman, 1998; Bardou, 1997) que, sin entrar en más detalles, enumeramos a continuación:

- Herramientas de ingeniería de la información.
- Herramientas de modelado y administración de procesos de empresas.
- Herramientas de estimación, planificación y administración de proyectos.
- Herramientas de análisis de riesgos.
- Herramientas de seguimiento de requisitos.
- Herramientas de medición.
- Herramientas de documentación.
- Herramientas de gestión de la configuración del software.
- Herramientas de análisis y diseño.
- Herramientas para *prototipado* y simulación.
- Herramientas para la generación de aplicaciones y componentes.
- Herramientas de programación.
- Herramientas de pruebas.
- Herramientas para la validación.
- Herramientas de reingeniería e ingeniería inversa.
- Herramientas para el mantenimiento.

3.4.2. AUTOMATIZACIÓN DEL MANTENIMIENTO.

Puesto que las tareas de mantenimiento representan la mayor carga de trabajo durante el ciclo de vida del software, la disponibilidad de herramientas para automatizar estas actividades ayudará a reducir notablemente el coste global del proceso. Entre las herramientas para mantenimiento del software tradicionales, se pueden señalar las siguientes:

- Generadores de referencias cruzadas.
- Generadores de organigramas.
- Controladores de código fuente.
- Analizadores automáticos de interfaces.
- Gestores de ficheros.
- Descompiladores.
- Evaluadores del impacto de las modificaciones.
- Detectores de componentes afectados por los cambios.

Estas herramientas, junto con otras nuevas propuestas en los últimos años, se pueden agrupar en tres categorías Mazza et al. (1996):

- Herramientas de navegación
- Herramientas para perfeccionamiento del código
- Herramientas de ingeniería inversa

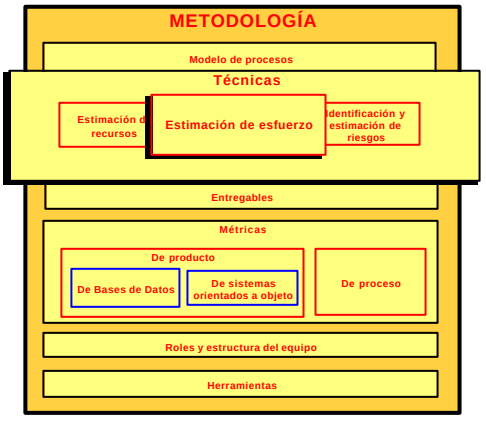
3.4.3. CONCLUSIÓN.

A pesar de la gran cantidad de herramientas CASE existentes, las utilizables para el proceso de mantenimiento son en su mayoría implementaciones de soluciones técnicas, que sin duda resultan de suma utilidad en labores de Ingeniería Inversa, Reingeniería, Reestructuración, Medición de código, Gestión de riesgos, etc., pero no son utilizables como herramientas de gestión del proceso. Además, la gran diversidad de entornos, lenguajes de programación, etc. limita la aplicación de estas herramientas a aquellos productos software para los que fueron diseñados.

Existe, sin embargo, la posibilidad de utilizar algunas herramientas, diseñadas inicialmente con otros fines a lo largo de todo el proceso de mantenimiento, pero su aplicación se limita a la automatización de determinadas franjas del proceso (por ejemplo, herramientas para gestión de configuración).

3.5. MÉTODOS DE ESTIMACIÓN DEL ESFUERZO DE MANTENIMIENTO.

"La estimación del esfuerzo de las intervenciones de mantenimiento puede resultar muy importante para los gestores, cuando planifican actividades de mantenimiento y realizan análisis de costes/beneficios" (Jørgensen, 1995). De acuerdo con este mismo autor, los investigadores han dedicado más tiempo a la elaboración de modelos de estimación de esfuerzo para nuevos desarrollos que para mantenimiento, aunque esta tendencia parece haber comenzado a cambiar desde que el artículo referido fue publicado.



En esta sección presentamos un resumen de algunos métodos de estimación del esfuerzo de mantenimiento propuestos en los últimos años. Puesto que la mayor parte de estos métodos se apoyan en métricas para calcular el esfuerzo de mantenimiento, podríamos haber resaltado, en la figura que venimos utilizando como guía, su carácter tanto *metrológico* como *técnico*.

3.5.1. ESTIMACIÓN POR ANALOGÍA.

La estimación por analogía se basa en el principio de comparar las características del sistema que vamos a modificar con el mismo conjunto de características del mismo u otros sistemas, que previamente hemos modificado. Si se dispone de datos relativos al esfuerzo empleado en estas intervenciones pasadas, se podrá realizar una estimación del esfuerzo necesario para la futura intervención.

El conjunto de características a utilizar debe estar restringido a la información disponible en el momento de hacer la predicción. Una vez seleccionadas las características, se realiza algún tipo de análisis, bien matemático, bien subjetivo (juicio experto), que enfrente los valores de los atributos históricos con los valores del producto que se va a intervenir.

Shepperd et al. (1996) presentan un método y una herramienta para estimación de costes basada en la minimización de la Distancia Euclídea en un espacio n-dimensional, siendo *n* el número de atributos que se comparan. La elección de los atributos debe limitarse a la información disponible en el momento en que se requiere la predicción. El esfuerzo computacional necesario para encontrar las mejores analogías y calcular el esfuerzo aumenta exponencialmente en función del número de atributos seleccionados (20 atributos a comparar de 21 proyectos necesitaron 42 horas de análisis, mientras que 12 atributos utilizaron 10 minutos para el mismo número de proyectos).

3.5.2. MODELO COCOMO PARA MANTENIMIENTO.

Granja y Barranco (1997) realizan una serie de modificaciones al conocido modelo COCOMO de estimación de costes, de tal manera que obtienen una versión aplicable a la previsión de los costes de mantenimiento.

En el modelo COCOMO original se define la *Tasa de Cambios Anual* (TCA) de la siguiente manera:

$$TCA = \frac{NLN + NLM}{NLT}$$

...siendo:
NLN : nº de líneas nuevas.
NLM : nº de líneas modificadas.
NLT: nº total de líneas.

Conociendo la expresión anterior, el *Coste de Mantenimiento* (CM) se calcula con esta expresión:

$$CM = TCA * CD$$

A partir de este punto, los autores introducen un nuevo parámetro, el *Índice de Mantenibilidad* (IM), que mide la facilidad de mantenimiento del producto considerado. Este índice, cuya forma de cálculo mostramos más adelante, influye directamente en el coste de mantenimiento, de manera que la expresión de éste queda de esta forma:

$$CM=TCA*CD*IM$$

Para estos autores, toda acción de mantenimiento puede dividirse en tres tareas:

- Comprensión de los cambios que deben hacerse.
- Realización de las modificaciones necesarias.
- Pruebas de los cambios realizados.

El coste de mantenimiento (CM) puede expresarse como la suma de los costes empleados en cada una de ellas:

$CM = CM_C + CM_R + CM_P$, siendo CM_c los costes de la comprensión, CM_R los costes de la realización de los cambios y CM_P: costes de las pruebas.

Esta expresión se puede desglosar de esta manera:

$CM_C = TCA * CD * I_C$

$CM_R = TCA * CD * I_R$

$CM_P = TCA * CD * I_P$

...donde: I_C : índice de comprensibilidad
I_R : IM de la realización de cambios
I_P : IM de las pruebas

El coste de mantenimiento en personas-mes queda, finalmente, de esta forma:

$CM = CM_C + CM_R + CM_P = TCA * CD * (I_C + I_R + I_P)$

De aquí quedan por conocer los diferentes índices de mantenibilidad para cada una de las tres etapas que Granja y Barranco distinguen en el mantenimiento, y la tasa de cambios anual, que es redefinida en este modelo.

Los índices se calculan en función de las tres métricas siguientes:

· X_C : porcentaje de líneas comentadas por cada cien, para calcular I_C.

· X_R : porcentaje de líneas sin datos constantes por cada cien, para calcular I_R.

· X_P : número de errores comprobados por cada cien líneas de código, para I_P.

Deben utilizarse datos históricos, procedentes de proyectos ya mantenidos, para obtener las funciones que nos den el valor de los índices a partir de las métricas anteriores. Así mismo, de la misma tabla histórica se extraen diferentes matrices que nos permitirán el cálculo de la tasa de cambios anual (TCA).

3.5.3. MODELADO DEL MANTENIMIENTO COMO UN SISTEMA DINÁMICO.

Calzolari et al. (1998) proponen un original método para modelar el esfuerzo de mantenimiento utilizando sistemas dinámicos. Se basan en el modelo predador/presa introducido en 1972 por May.

La idea básica del modelo de May es que, en ausencia de presas, los predadores se extinguen y, en ausencia de predadores, la población de presas alcanza y sobrepasa la capacidad del entorno para alimentarlas. En los casos intermedios se alcanza un equilibrio estable en el sentido de que un incremento en el número de presas permite a los predadores predar y

reproducirse, lo cual causa que decrezca el número de presas; esto, a su vez, provoca que se reduzca el número de predadores.

Los autores realizan la siguiente comparación del modelo dinámico de May con el esfuerzo de mantenimiento correctivo: cuando se corrigen defectos, la actividad de correctivo disminuye. Sin embargo, a diferencia del modelo biológico-dinámico de May, los defectos no tienen capacidad para reproducirse. La única forma de que aparezcan nuevos errores en el software es mediante la entrega de una nueva versión del producto (esto no es realmente una reproducción de defectos -como cabría esperar de una población en sentido estricto-, sino una inserción instantánea de defectos; esto es, de presas).

La similitud del esfuerzo de mantenimiento con el modelo dinámico de May se basa en que:

1) El mantenimiento correctivo es esencialmente predador de defectos software, y el esfuerzo de mantenimiento se alimenta de errores descubiertos por el usuario.
2) Los mantenimientos perfectivo y adaptativo se alimentan de necesidades del usuario, y el esfuerzo de estos dos tipos de mantenimiento se adapta a la cantidad de solicitudes de estos dos tipos de mantenimiento.

En el artículo referido, se expone el método aplicándolo a mantenimiento correctivo. Tras la adaptación del modelo original al mantenimiento, los autores concluyen lo siguiente:

1) Se prevén aumentos de correctivo inmediatamente después de cada nueva versión de la aplicación. Tras este incremento, el esfuerzo de mantenimiento disminuye, debido a que el número de presas ha disminuido suficientemente. Se espera que este comportamiento se repita tras cada entrega. Además, el ajuste en la predicción resulta muy bueno.
2) La duración del esfuerzo de mantenimiento (amplitud de las campanas mostradas en la Figura 14, sección 3.1.3.3, página 67), dependerá del grado en que se haya modificado el producto software.

En esta tesis se presenta una técnica para estimar los recursos necesarios para mantenimiento correctivo urgente, para cuya aplicación se necesita conocer la futura distribución de llegadas de peticiones de este tipo de mantenimiento, por lo que el método expuesto en esta sección resulta muy interesante.

3.5.4. ESTIMACIÓN DEL ESFUERZO DE MANTENIMIENTO CON PUNTOS-FUNCIÓN.

Niessink y van Vliet (1997) realizan ciertos análisis comparativos entre el esfuerzo de mantenimiento de varias intervenciones y el número de puntos-función modificados, contados según métodos diferentes. No obtienen, sin embargo, resultados satisfactorios para ningún modelo predictivo, pero sí que observan que el esfuerzo de mantenimiento es mucho más depen-

diente del tamaño del componente que se va a cambiar, que del tamaño del propio cambio (medidos ambos parámetros en puntos-función). Es decir, que:

Esfuerzo » K ´ Tamaño del componente ´ (1 + e ´ Tamaño del cambio)

...en vez de:

Esfuerzo » K ´ Tamaño del cambio

La falta de adecuación de los puntos-función para realizar estimaciones ha sido también discutida en Dolado y Fernández (1999).

3.5.5. ANÁLISIS DE MÉTODOS DE JÆRGENSEN (1995).

Este autor compara la fiabilidad en la predicción de diferentes métodos de estimación, de los tipos que listamos más abajo. El estudio es realizado aplicando cada método a un conjunto de 100 intervenciones de mantenimiento. De cada intervención de mantenimiento se recogen los siguientes datos:

- *Causa*, que se corresponde con el tipo de mantenimiento (correctivo, adaptativo, perfectivo o preventivo).
- Prioridad de la tarea (alta, media o baja).
- *Confianza* del mantenedor, o grado de conocimiento del mantenedor, en lo relativo a si sabe resolver la petición inmediatamente después de haber leído u oído la especificación. El grado de conocimiento puede ser *Sabe cómo resolver la tarea*, *Puede saber resolverla* o *No sabe cómo resolverla.*
- Años de experiencia como mantenedor.
- Nivel de estudios del mantenedor.
- Horas dedicadas a la intervención.
- Tamaño de la intervención (medida como la suma del nº de LDC añadidas, modificadas y borradas, incluyendo líneas de comentario).
- Lenguaje de programación
- Tipo del cambio (introducción o borrado de módulos, cambio de interfaz, cambio del flujo de control, cambio de declaraciones de datos, cambio de datos o sentencias de asignación).
- Fuentes de información utilizadas (comunicación con los usuarios, documentación del usuario, documentación del lenguaje o la herramienta, comunicación con personal del sistema, comunicación otros mantenedores y uso de la documentación del sistema).
- Edad de la aplicación cambiada.
- Tamaño de la aplicación cambiada.
- Descripción informal de la intervención.

Una vez caracterizadas con estas informaciones, el autor realizó un análisis de correlación entre cada atributo y el esfuerzo dedicado a la intervención, de manera que deja de considerar algunos atributos, y redefine otros. Finalmente, las variables que utilizará son la siguientes:

Atributo	Observaciones
Causa	0→ correctivo; 1→ cualquier otro
Cambio	0→ más del 50% del esfuerzo se empleó en actualización de código, comparado con la inserción y borrado. 1→ otro caso
Modo	0→ más del 50% del esfuerzo se empleó en desarrollas nuevos módulos; 1→ otro caso
Confianza	0→ el mantenedor está seguro de cómo resolver la tarea inmediatamente después de haber leído u oído la especificación por primera vez. 1→ otro caso

Tabla 12 Atributos considerados en el análisis.

El resto de variables dejaron de ser consideradas, pues no había una correlación suficientemente significativa. Los modelos analizados toman como parámetros los valores de los atributos mostrados en la Tabla 12, y son los siguientes:

- Modelos simples: *Esfuerzo = Tamaño / Productividad media*
- Modelos de regresión simple.
- Modelos de regresión múltiple.
- Modelos que utilizan redes neuronales, tomando como muestra la proporcionada por la herramienta PlaNet versión 5.6.
- Modelos que utilizan reconocimiento de patrones.

Para cada modelo, el autor calcula diferentes medidas, basadas en la *Magnitud del Error Relativo*:

- *MMRE*, que es la media de la magnitud del error relativo.
- *MdMRE*, que es la mediana de la magnitud del error relativo.
- *PRED1(0.25)*, que es el porcentaje de intervenciones con un error relativo menor que 0.25.
- *PRED1(0.50)*, que es el porcentaje de intervenciones con un error relativo menor que 0.50.
- *PRED2(0.25, 050)*, que es el porcentaje de intervenciones con un error relativo menor que 0.25, o con |Esfuerzo real - Esfuerzo predicho|≤ 0.5 días

Tras el cálculo de estas medidas, el autor concluye que los modelos más fiables son:

- *Log(Esfuerzo) = K + a ´ Modo + b ´ Modo ´ log(Tamaño) + c ´ Confianza, min-log(Esfuerzo actual - Esfuerzo predicho))*^{2}
- *Log(Esfuerzo) = K + a ´ log(Tamaño) + b ´ Causa + c ´ log(Tamaño) ´ Confianza + d ´ Cambio + e ´ log(Tamaño) ´ Cambio + f ´ Modo + g ´ log(Tamaño) ´ Modo + h ´ Confianza + i ´ log(Tamaño) ´ Confianza, minlog(Esfuerzo actual - Esfuerzo predicho))*^{2}
- *Un método híbrido, que utiliza regresión y reconocimiento de patrones.*

3.5.6. CONCLUSIÓN.

Hemos visto en esta sección diferentes métodos para estimar tanto el esfuerzo de mantenimiento de futuras intervenciones como de distribución de peticiones de modificación en el futuro. Resultaría muy interesante, sin embargo, ser capaces de estimar la cantidad de recursos necesarios para servir las peticiones de modificación durante un periodo de tiempo determinado, de manera que la cantidad de recursos obtenida sea óptima desde el punto de vista económico.

En la metodología descrita en esta tesis se propone aplicar un modelo económico-matemático a los resultados obtenidos de algún método de estimación de la distribución de peticiones de modificación (el descrito en el epígrafe 3.5.3, por ejemplo) para realizar la planificación óptima de recursos necesarios para mantenimiento correctivo urgente. Esta técnica es muy interesante para las empresas suministradoras de mantenimiento, aunque es perfectamente utilizable exista o no externalización del servicio. La exponemos en el epígrafe 4.3.2 (página 133).

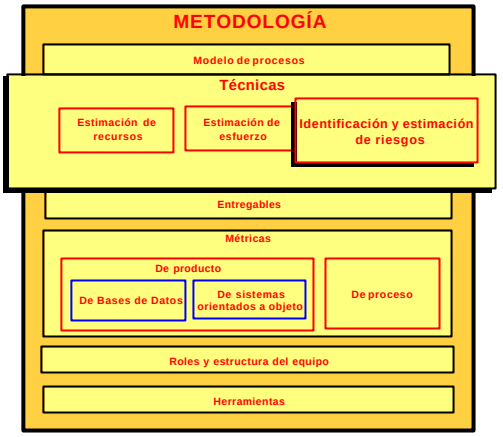
Por otro lado, en MANTEMA también se propone utilizar la familia de métricas *CC* para estimar el esfuerzo de mantenimiento adaptativo de sistemas orientados al objeto.

3.6. RIESGOS EN EL PROCESO DE MANTENIMIENTO.

3.6.1. RIESGOS DE LAS INTERVENCIONES DE MANTENIMIENTO.

Schneidewind (1997a) propone una serie de criterios y ecuaciones para estimar los riesgos del mantenimiento de sistemas críticos (misiones espaciales), basándose fundamentalmente en el tiempo total de prueba del sistema, en el número de errores encontrados durante las pruebas y en el tiempo transcurrido entre cada par de errores. Con estos datos, el autor propone una ecuación para calcular el instante en que ocurrirá el siguiente fallo, de manera que, sin salirnos del contexto, la misión espacial podrá despegar si su regreso tendrá lugar antes de que ocurra el próximo fallo crítico. Además, estos valores pueden utilizarse como parámetros para la medición de la calidad del proceso de mantenimiento, de tal manera que un incremento en el número de errores por unidad de tiempo debe invitar a revisar el proceso de mantenimiento que se está aplicando.

Por otro lado, para Sherer (1997) existen tres tipos de riesgos en el proceso de mantenimiento:



- *Riesgos de proyecto:* el proyecto de mantenimiento no puede completarse debido a la falta de capacidad del personal, el software no puede mantenerse en tiempo y presupuesto o la organización no tiene un proceso de mantenimiento efectivo.
- *Riesgos de utilización:* los sistemas mantenidos producirán problemas al ser utilizados.
- *Riesgos de mantenibilidad:* el sistema será difícil de mantener en el futuro, una vez que los cambios han sido realizados.

La autora propone un método para estimar el riesgo de fallo del software y la magnitud de la pérdida provocada por este fallo, cuyas cuatro etapas resumimos a continuación:

1. *Evaluación del riesgo externo*. Consiste en estudiar el entorno en el que el software se ejecutará, con el objetivo de anticiparse a las posibles situaciones que pueden provocar pérdidas económicas.
2. *Exposición de módulos*. En esta etapa se calcula la magnitud de la pérdida económica debida a errores en un módulo. Existirá una pérdida potencial en un módulo cuando éste esté relacionado con alguna de las posibles situaciones identificadas en la etapa anterior. La magnitud de la pérdida producida por cada módulo se estima sumando la probabilidad uso de cada módulo, ponderada con la consecuencia esperada de todas las situaciones que pueden resultar de ese uso.
3. *Probabilidad de fallo*. Es el número esperado de fallos producidos por un módulo durante un periodo de tiempo debido a errores en dicho módulo. Se estima con un modelo de fiabilidad del software dependiente del tiempo.
4. *Riesgo del módulo*. Se estima como el producto de la exposición del módulo y el número esperado de fallos.

En la referencia se detallan cada una de las etapas anteriores, siendo el método presentado un modelo cíclico, en el sentido de que deben recalcularse los riesgos tras cada intervención de mantenimiento. El método pretende ayudar a mejorar los esfuerzos de mantenimiento mediante la consideración explícita de los riesgos de mantenimiento.

3.6.2. RIESGOS DE LA EXTERNALIZACIÓN.

Además de las desventajas de la externalización, identificables con posibles factores de riesgos, que hemos enumerado en la sección 3.1.4 (página 69), la externalización de todo o

parte del sistema de información ofrece diferentes tipos de riesgos, analizados por Klepper y Jones (1998) a partir de varias referencias, y que resumimos en la Tabla 13.

En esta misma línea, Willcocks y Lacity (1999) exponen un caso de estudio que relata el proceso de externalización seguido por un grupo de compañías aseguradoras del Reino Unido. Para estos autores, los mayores riesgos en las relaciones de externalización se derivan de la falta de seguridad acerca de qué externalizar y por qué hacerlo, así como en la mala selección del suministrador del servicio. Igualmente, insisten en que sólo deben externalizarse las partes del sistema de información que no juegan un papel primordial en el *core business* de la organización. La cesión a un tercero de aspectos cruciales del sistema de información de la empresa eleva el riesgo hasta niveles inadmisibles.

La adecuada negociación del contrato (que debe incluir cláusulas de rescisión), la correcta definición de los indicadores de servicio y los mecanismos de control y seguimiento de los términos contratados se revelan como uno de los mejores modos de mantener los riesgos bajo control.

Riesgo	Mecanismo de control o mitigación
Pérdida de control sobre plazos de entrega y calidad del servicio	Seguimiento de los términos contractuales
Pérdida de flexibilidad (modificaciones a lo contratado necesitan la aceptación del suministrador del servicio)	Contrato abierto Incentivos al suministrador
Incremento de costes por “excesos de servicio”	Realizar buenos análisis de requisitos Control de indicadores de servicio
Acceso a información confidencial por parte del suministrador del servicio	Previsión en el contrato
Propiedad intelectual	Previsiones de copyright en el contrato
Incremento de costes por abuso de los usuarios de los servicios prestados por el suministrador	Filtrado de solicitudes en la organización cliente
Se externaliza demasiado	Realizar un análisis previo, de manera que se externalice sólo aquello que no forma parte del *core business*
El cliente no sabe expresar adecuadamente el tipo de servicio que quiere externalizar	Firma de contratos a corto plazo
El suministrador no presta su servicio con el nivel esperado	Definición de indicadores de servicio y penalizaciones en el contrato
“Anclaje tecnológico” del suministrador	Selección cuidadosa del suministrador Incentivar al suministrador por el avance tecnológico Firmar contratos de corta duración, para considerar en los siguientes contratos la utilización de nuevas tecnologías
Pérdida de una fuente de aprendizaje	Aunque se incrementen los costes, mantener personal trabajando sobre los entornos y sistemas externalizados

Tabla 13. Riesgos asociados a la externalización (Klepper y Jones, 1998).

3.6.3. CONCLUSIÓN.

En esta sección se han repasado algunos métodos de identificación y/o gestión de riesgos para mantenimiento. En el epígrafe 3.6.1 se han mencionado los métodos de Schneidewind (1997a) y Sherer (1997), que pueden utilizarse para estimar los riesgos de futuras intervenciones de mantenimiento y, especialmente el segundo caso, para conocer las partes más arriesgadas del software mantenido, permitiendo así la priorización de recursos y el análisis de diferentes alternativas.

En el epígrafe 3.6.2 se han presentado los riesgos inherentes a la externalización. Como se observa, todos los riesgos mencionados en la Tabla 13 han sido analizados desde el punto de vista de la organización cliente (aquélla que decide externalizar). Sin embargo, la asunción de un contrato de externalización implica también una serie de riesgos para la organización suministradora, que no son estudiados en la bibliografía. Por ello, y puesto que MANTEMA es una metodología de mantenimiento que puede ser utilizada por organizaciones proveedoras de este servicio, es preciso incorporar a la metodología mecanismos que permitan identificar y estimar los riesgos derivados de un proyecto de mantenimiento. Además, tales mecanismos podrán ser utilizados no sólo cuando haya relaciones contractuales de externalización, sino también cuando una organización decide mantener su propio software.

Es importante notar que, aunque hay buenos trabajos que analizan la problemática de la identificación, estimación y gestión de riesgos una vez el proceso de mantenimiento ha comenzado (como los de Schneidewind y Sherer analizados en esta sección), no existen guías que ayuden en la identificación y estimación de riesgos en las etapas iniciales del proceso de mantenimiento (o, dependiendo del concepto de *proceso* que se tenga, anteriores al proceso de mantenimiento propiamente dicho). Las técnicas de estimación de costes constituyen quizás una excepción a este punto; sin embargo, no se consideran métodos de estimación de riesgos ni consideran factores de riesgos en sus estimaciones o, si lo hacen, éstos se confían al juicio de un experto sin proporcionar más detalles (Briand et al., 1998b).

Los métodos generales (no sólo de mantenimiento) de gestión de riesgos comúnmente utilizados suelen dividir el proceso de gestión de riesgos en un conjunto de subprocesos (McConnell, 1998; Karolak, 1996; Project Management Institute, 1996), que puede y debe ser iterativo:

1. Identificación de riesgos
2. Estimación de riesgos
3. Desarrollo de respuesta a riesgos
4. Control de los mecanismos de respuesta

Durante esta tesis se ha desarrollado un cuestionario para identificar los riesgos principales de los proyectos de mantenimiento antes de acometerlos. Este cuestionario permite a la organización de mantenimiento obtener una idea muy aproximada del estado del software que se va a mantener, planificar recursos, realizar presupuestos, etc.

4 MANTEMA: UNA METODOLOGÍA PARA EL MANTENIMIENTO DE SOFTWARE.

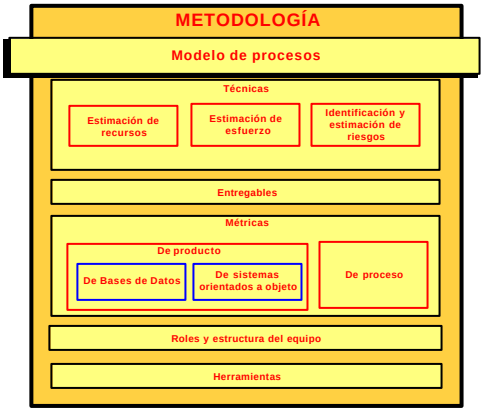
Si bien los programadores entenderán el mantenimiento como el conjunto de tareas que ellos realizan cambiando, eliminando y añadiendo sentencias de código fuente, probablemente los directivos de las empresas en las que trabajan lo perciben como una de las bases más importantes de su negocio y que debe, por tanto, ser absolutamente controlada y continuamente mejorada.

Teniendo presente esta diferencia de conceptos, la metodología MANTEMA muestra la visión del proceso de mantenimiento desde el mayor nivel de abstracción, en el que probablemente no interesa el contenido de las instrucciones ni los campos de los ficheros, aunque sí las mejores técnicas para entenderlos y modificarlos. Desde este punto de vista, el proceso puede verse como el conjunto de todas las operaciones que es necesario realizar sobre el software para implementar las modificaciones solicitadas. Sin embargo, para dotar a este conjunto de operaciones de una base metodológica, es preciso definir con anterioridad el propio proceso de mantenimiento, detallando qué debe realizarse, cuándo, cómo y por quién, de tal manera que cada intervención de mantenimiento que se ejecute conforme una instancia de un proceso de mantenimiento predefinido.

En este capítulo se presenta la propuesta metodológica y metrológica ofrecida por MANTEMA, y está organizado de la siguiente forma: en la sección 4.1 explicamos cómo se obtuvo una estructura básica válida para la definición rigurosa del proceso de mantenimiento a partir del definido en ISO/IEC 12207, presentando gráficamente la estructura general de la metodología y explicando la integración de algunos de los procesos del ciclo de vida en el de mantenimiento, así como los diferentes tipos de mantenimiento que se consideran y la identificación de las organizaciones que intervienen en el proceso. Posteriormente, en la sección 4.2 pasamos a describir muy en detalle las diferentes partes de la metodología, deteniéndonos en cada actividad y tarea y especificando cuidadosamente cada una de ellas. Por último, concluimos con la sección 4.3, en la que explicamos las métricas y técnicas propuestas como apoyo a la ejecución del proceso.

4.1. ESTRUCTURA GENERAL DE LA METODOLOGÍA

El estudio detallado del proceso de Mantenimiento de ISO/IEC 12207 permitió la identificación en él de la *macroestructura* mostrada en la Figura 15, que es también deducible de las actividades reproducidas en la Tabla 4 (Polo et al., 1999b). Tal estructura fue elegida como punto de partida para esta metodología.



En el *Comentario* dedicado al Proceso de Mantenimiento propuesto por ISO/IEC 12207 (epígrafe 3.1.1.1.8, página 53), hemos visto que existe cierto solapamiento de actividades y tareas, además de la propuesta de ciertas tareas cuya realización puede no resultar útil en algunas

ocasiones. Este problema, que surge también en la norma IEEE 1219, es debido al hecho de que todas las actividades aparecen agrupadas y dispuestas en forma de secuencia, sin distinguir cuáles de ellas deben realizarse y cuáles no, dependiendo del tipo de cada caso concreto.

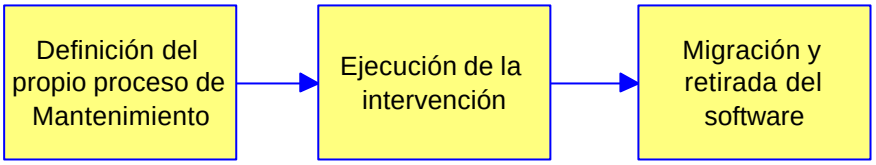


Figura 15. Macroestructura del proceso de mantenimiento de ISO/IEC 12207.

La casuística es desde luego tremendamente variada y sin duda es arriesgado dar unas normas generales de validez universal. No obstante, en las dos normas mencionadas (ISO/IEC 12207 e IEEE 1219) se expone la necesidad de realizar algún tipo de selección de actividades y tareas dependiendo del caso:

- ISO/IEC 12207 propone la utilización de su Proceso de Adaptación, que permite realizar la adaptación básica de la norma ISO a distintos proyectos de software, teniendo en cuenta las variaciones en las políticas y procedimientos organizacionales, los métodos y estrategias de adquisición, el tamaño y complejidad de los proyectos, los requisitos de sistema y métodos de desarrollo, etc. Igualmente, la actividad "Análisis de problemas y modificaciones" del proceso de mantenimiento de ISO contempla la clasificación de la modificación de acuerdo al tipo de mantenimiento, aunque luego no precisa qué actividades deben seguirse según este tipo.
- En la misma línea, la norma de IEEE 1219 aconseja en su apéndice A "clasificar las peticiones de modificación en correctivas, adaptativas, perfectivas y preventivas y agruparlas todas ellas en conjuntos que compartan las mismas áreas de diseño". De este modo, "pueden priorizarse las peticiones de modificación no individualmente, sino de forma conjunta".

Por otro lado, Pressman (1993) también considera la necesidad de establecer diferentes flujos de acciones según el tipo de intervención de mantenimiento que deba aplicarse, como podemos ver en los tres nodos de decisión que hemos rodeado con círculos en la Figura 16.

Es visible, por tanto, la necesidad de plantear diferentes conjuntos de actividades y tareas según el tipo de mantenimiento que deba aplicarse a cada petición de modificación.

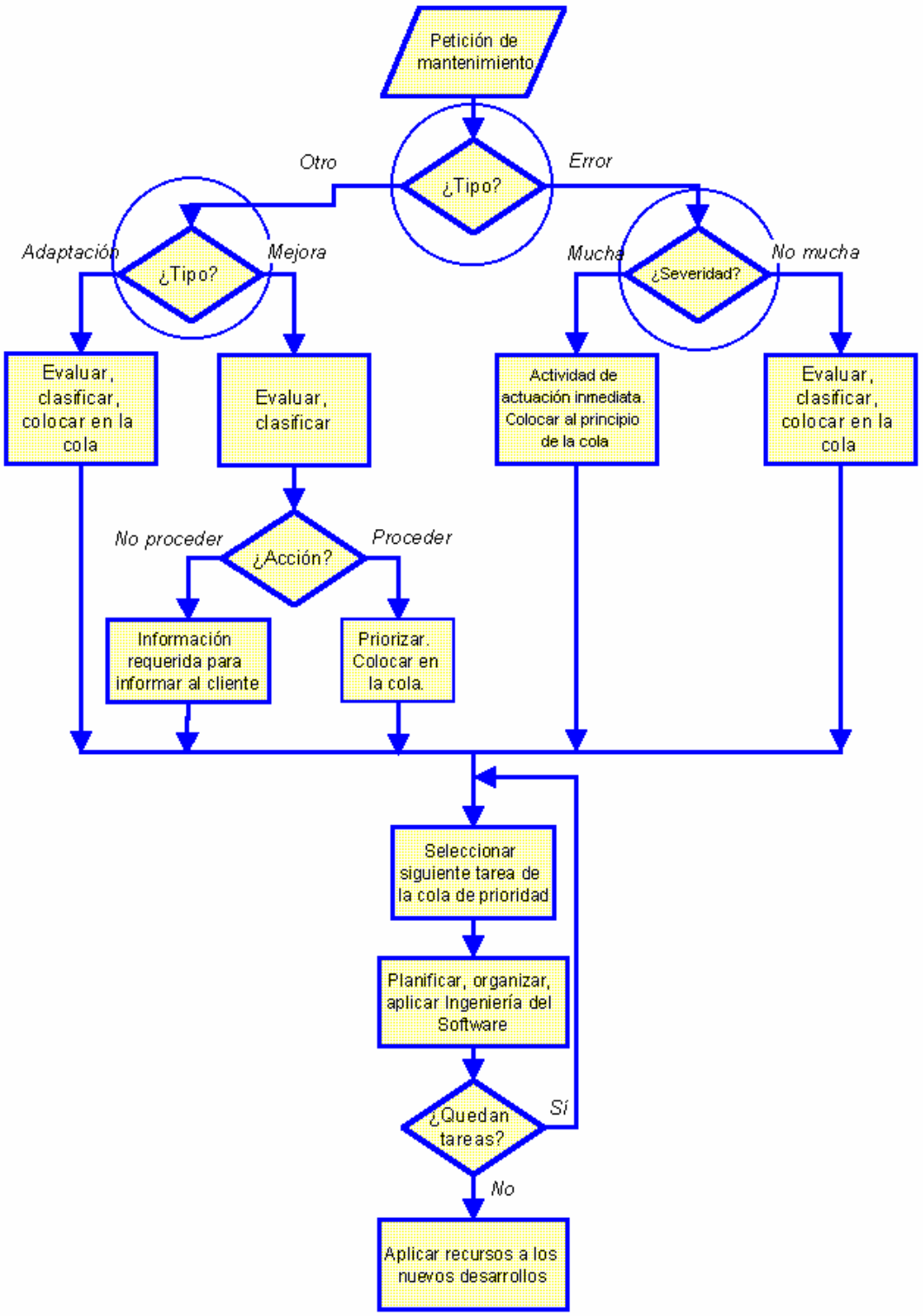


Figura 16. Flujo de tareas en el mantenimiento (tomada de Pressman, 1993).

4.1.1. INTEGRACIÓN DE PROCESOS.

Por otro lado, la metodología pretende la definición de un único proceso de aquéllos que componen el ciclo de vida software: el mantenimiento. Durante éste se realizan trabajos de

análisis y diseño, programación, pruebas, documentación, etc. (la carga de cada trabajo dependerá del tipo de intervención). Se pretende que esta metodología constituya una guía completa de mantenimiento, por lo que integramos y describimos, en el único proceso software descrito, algunas actividades de los procesos habituales del ciclo de vida software.

Aunque los dos estándares del ciclo de vida analizados en la sección 3.1.1 (IEEE 1074 e ISO/IEC 12207) difieren en el número de procesos identificados, coinciden básicamente en lo sustancial. Tomando como referencia los procesos de ISO/IEC 12207, en nuestra metodología integramos actividades de los siguientes procesos del ciclo de vida:

- Adquisición.
- Suministro.
- Desarrollo.
- Mantenimiento.
- Documentación.
- Resolución de problemas.
- Gestión.

Esquemáticamente, el resultado de tal integración puede verse de la forma expresada en la Figura 17, en la que se observan, como procesos satélites al mantenimiento, otros procesos del ciclo de vida cuya integración no se ha llevado a cabo pero que, sin embargo, son utilizados por nuestro proceso de mantenimiento estableciendo interfaces en algunas tareas:

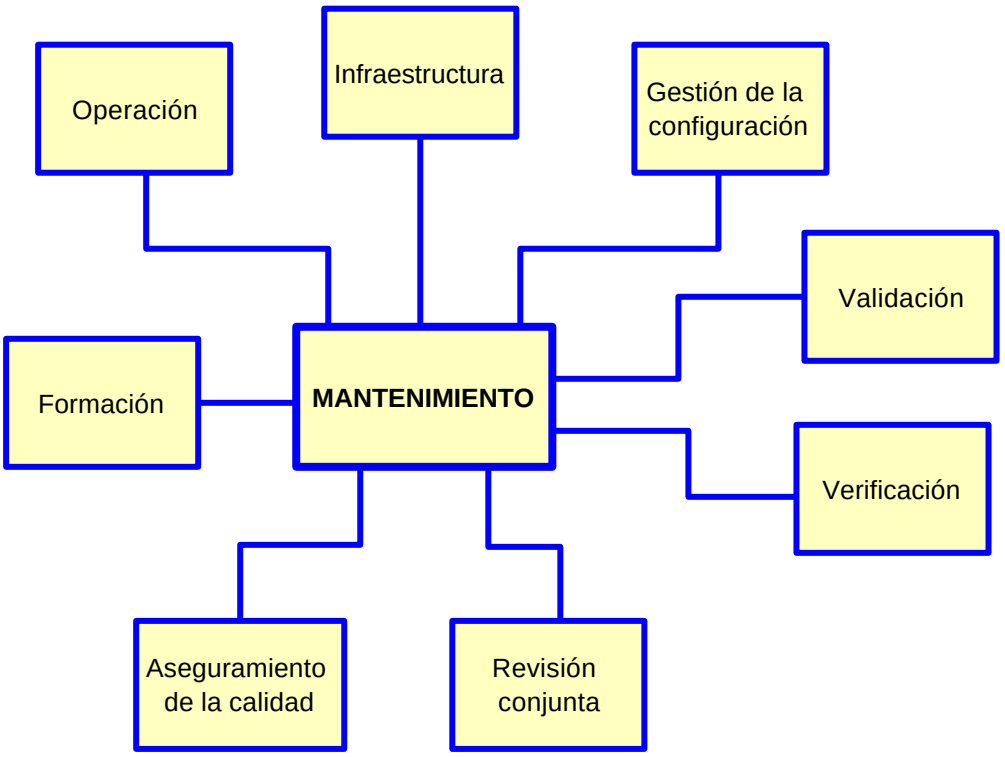


Figura 17. Visión general de los procesos en la metodología.

4.1.2. TIPOS DE MANTENIMIENTO EN LA METODOLOGÍA.

Habitualmente se distinguen diferentes tipos de mantenimiento cuyos grados de importancia, criticidad y formas de actuar son muy diferentes. Por regla general (véase sección 1.1), se distinguen los mantenimientos correctivo, adaptativo, perfectivo y preventivo, aunque puede haber ligeros matices en las definiciones de cada uno según la referencia. Con el fin de unificar criterios e identificar unívocamente cada situación concreta, hablaremos de los siguientes cinco tipos de mantenimiento (Ruiz et al., 1999a):

a) Correctivo *urgente*, que es aquel que se da en situaciones en que existe un error en el producto software que bloquea la aplicación o el proceso de funcionamiento de la empresa, que debe ser resuelto con brevedad (por ejemplo, el día veintiocho falla la aplicación de cálculo de nóminas). Estas peticiones deben ser rápidamente servidas.

b) Correctivo *no urgente*, que se produce cuando existe un error en el producto software que no es crítico, pero que tal vez impida el funcionamiento de la aplicación o el normal funcionamiento de la empresa en un periodo de tiempo relativamente corto (por ejemplo, el fallo en la aplicación de nóminas se produce el día diez).

c) Perfectivo: se ocupa de añadir al software en explotación nuevas características o funcionalidades solicitadas por los usuarios.

d) Adaptativo: se aplica cuando el software en explotación va a cambiarse para que continúe funcionado correctamente en un entorno cambiante.

e) Preventivo: es aplicado cuando se desea mejorar las características internas de un producto software para hacerlo, por ejemplo, más fácilmente mantenible

Sin embargo, a pesar de que las intervenciones de correctivo no urgente, perfectivo, preventivo y adaptativo poseen características propias que las diferencian unas de otras, la aplicación de la metodología a casos reales evidenció que la ejecución de las intervenciones de estos cuatro tipos seguía patrones de comportamiento bastante similares, cosa que ya habíamos advertido durante el diseño. Por ello, se decidió integrar estos cuatro tipos de mantenimiento en uno solo (que denominamos “planificable”), pasando a llamar “mantenimiento no planificable” al que hasta entonces había sido “correctivo urgente”. De este modo, realizamos la clasificación que se muestra en la Figura 18.

En esta metodología de mantenimiento conservamos la terminología ISO al considerar una *Actividad* como una agregación de tareas lógicamente relacionadas, y una *Tarea* como un conjunto de operaciones que deben ejecutarse, pero que, desde el punto de vista del proceso, forman una unidad atómica y transaccional. Utilizaremos la clasificación de tipos de mantenimiento indicada en la Figura 18 para especificar los conjuntos de actividades y tareas que deban ejecutarse según el tipo de intervención, definiendo por tanto una serie diferente para cada uno de los dos tipos. Sin embargo, y puesto que todavía se mantienen diferencias entre los mantenimientos planificables, indicaremos cuando sea necesario las particularidades existentes de los tipos que pertenezcan a este grupo.

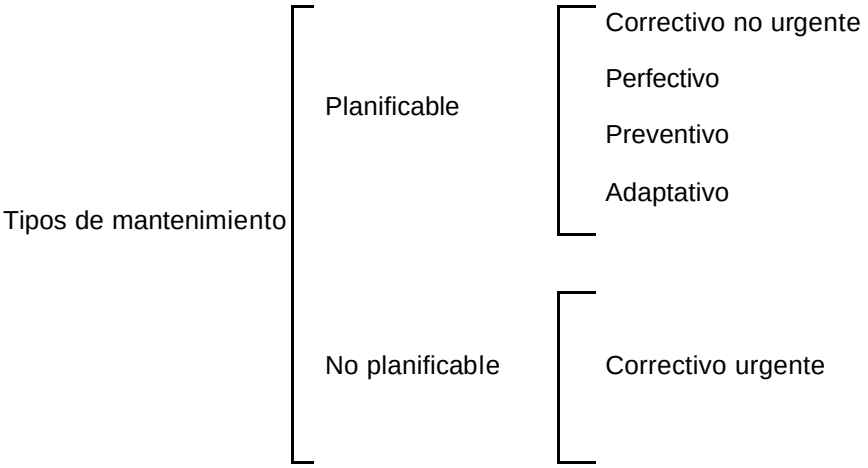


Figura 18. Tipos de mantenimiento en la metodología.

Con esta clasificación y con la definición precisa de sus actividades correspondientes pretendemos que las organizaciones apliquen un proceso de mantenimiento rigurosamente definido, cuantitativamente controlable y que garantice la calidad de las intervenciones. Además, se conseguirá minimizar ciertos costes "burocráticos" del mantenimiento, íntimamente ligados a las labores de selección de actividades y adaptación o creación de metodologías. Igualmente, en las intervenciones de mantenimiento que se adapten a esta metodología se generará la documentación rigurosamente necesaria en documentos cuyos contenidos se encuentran estrictamente definidos (véase Apéndice I). De este modo también se disminuirán los costes relacionados con esta actividad, que en algunos casos se sitúa en el 15% de los costes totales.

La clasificación de actividades y tareas de acuerdo al tipo de mantenimiento permite observar a MANTEMA como un grafo polietápico con cinco nodos en la etapa intermedia, cuatro de ellos obviamente agrupables, como se muestra en la Figura 19. Como observa el lector, se incluyen dos conjuntos de actividades y tareas iniciales y finales, que son comunes a todos los tipos de mantenimiento, y cuya descripción veremos más adelante.

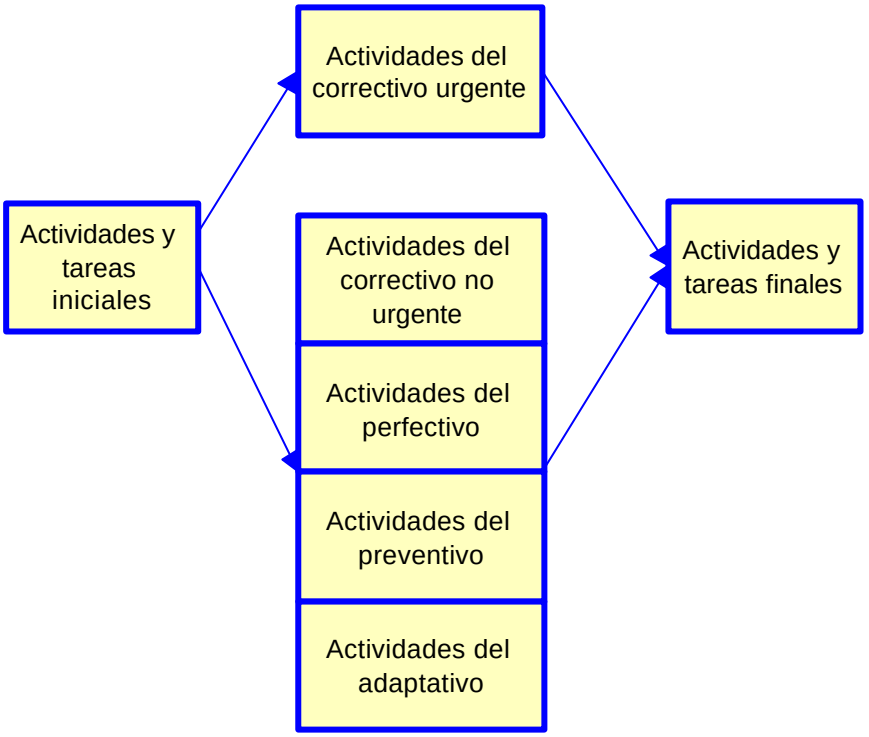


Figura 19. Visión de la metodología como un grafo polietápico.

4.1.3. ESTRUCTURA DE UNA TAREA.

Cada uno de los nodos de la Figura 19 está formado por un conjunto de actividades y, cada actividad, por un conjunto de tareas. Como hemos mencionado en el epígrafe precedente, una *Actividad* es una agrupación de tareas lógicamente relacionadas o, más bien, el nombre de una actividad es una forma de referirnos a un conjunto determinado de tareas. De este modo, una actividad estará finalizada cuando lo estén todas las tareas que la componen.

Antes de comenzar una tarea es necesario disponer de un conjunto de elementos de entrada que permitan su ejecución. Tales entradas podrán proceder del entorno en que se está aplicando la metodología (un programa que contiene un error, por ejemplo), o pueden ser el resultado de salida de una tarea ejecutada con anterioridad. Igualmente, las salidas de una tarea pueden ir destinadas al entorno (un programa que contenía un error y que ha sido corregido, por ejemplo, es puesto en el entorno de producción para permitir su uso), o a la entrada de una tarea posterior. Nótese que, realmente, son las tareas las que toman el protagonismo en la ejecución del proceso de mantenimiento, hecho que en cierto modo se apuntaba al mencionar que las actividades son tareas agrupadas *lógicamente*.

Es conveniente, por otro lado, designar los roles responsables de la ejecución de cada tarea, con el fin de conocer la dedicación de los miembros asignados a estos roles, o para controlar su nivel de eficiencia en la corrección de errores, por ejemplo. Del mismo modo, deben enunciarse ciertos parámetros que deben ser medidos en cada tarea, de manera que se pueda

conocer, entre otras cosas, el nivel de calidad del proceso de mantenimiento (o de la instancia del proceso) que se está aplicando. Por último, para la ejecución de determinadas tareas puede resultar interesante enunciar ciertas técnicas o métodos que faciliten o normalicen los resultados de la tarea.

Con estas consideraciones, podemos representar gráficamente una tarea de la siguiente manera (Figura 20):

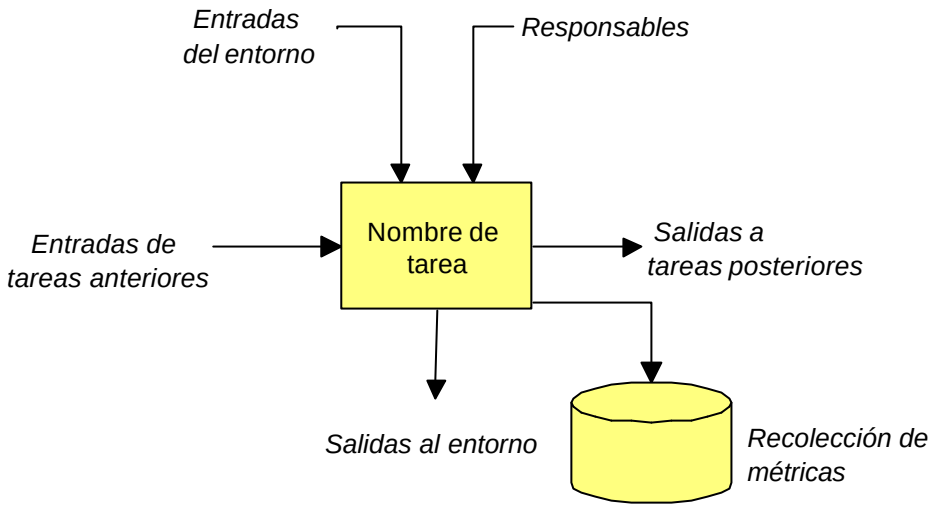


Figura 20. Estructura de una tarea.

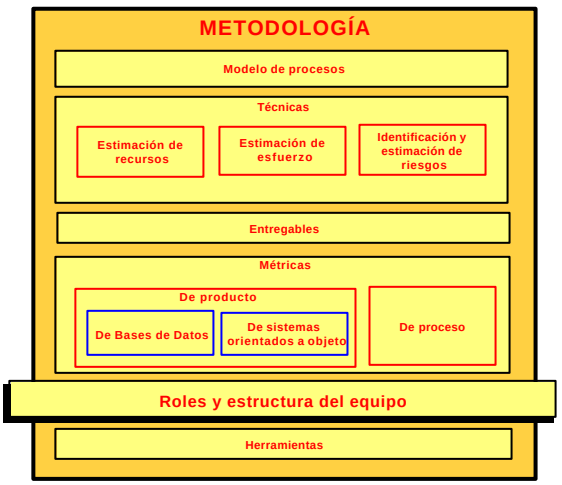
4.1.4. PARTICIPANTES EN EL PROCESO DE MANTENIMIENTO.

Varias organizaciones pueden participar en el proceso de mantenimiento del software (Polo et al., 1999d):

a) Cliente: es la organización propietaria del software y por tanto, la que recibe el servicio de mantenimiento.

b) Organización de mantenimiento: es la organización que realiza el servicio de mantenimiento.

c) Usuario: es la organización que utiliza el software objeto del mantenimiento.



Dependiendo de la situación, cada una de estas organizaciones puede ser una organización diferente, o coincidir varias organizaciones en una sola. Lo mismo puede ocurrir con los siguientes *perfiles*, que definimos para cada una de las organizaciones, de acuerdo a la conveniencia de así hacerlo manifestada en Mazza et al. (1994).

4.1.4.1. Perfiles de cliente.

- *Solicitante*: es quien presenta las solicitudes de modificación. Establece los requerimientos necesarios para su implementación y los entrega a la organización de mantenimiento.
- *Organización del Sistema*: es el departamento que conoce el sistema que será mantenido.
- *Atención a Usuarios*: es el departamento que presta asistencia a los usuarios.

4.1.4.2. Perfiles de la organización de mantenimiento.

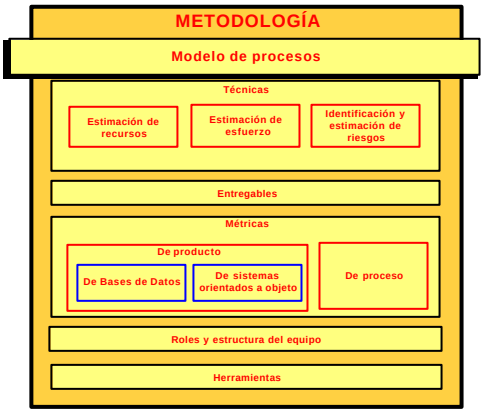
- *Gestor de peticiones*: acepta o rechaza las peticiones modificación y decide el tipo de mantenimiento que debe aplicarse.
- *Planificador*: planifica la cola de peticiones de modificación aceptadas.
- *Equipo de Mantenimiento*: es el grupo de personas que implementa la solicitud de modificación.
- *Responsable de Mantenimiento*: prepara la etapa de mantenimiento, y establece las normas y procedimientos necesarios para llevar a cabo la metodología de mantenimiento usada.

4.1.4.3. Perfiles de usuario.

- *Usuario*: utiliza el software mantenido. Comunica las incidencias a Atención a Usuarios.

4.1.5. CICLO DE VIDA DEL PROCESO DE MANTENIMIENTO.

El proceso de mantenimiento descrito en esta tesis doctoral comienza por la solicitud de prestación del servicio de mantenimiento de una organización a otra. Como hemos apuntado en la sección anterior, dos o tres organizaciones pueden coincidir en una sola, de manera que la solicitud de prestación del servicio podría tener lugar dentro una misma organización.



En cualquier caso, la consideración explícita de tres organizaciones participantes permite considerar, en la metodología, el establecimiento y finalización de relaciones de subcontratación de servicios de mantenimiento.

Una vez que la organización cliente se pone en contacto con la organización suministradora del servicio de mantenimiento (que, insistimos una vez más, puede ser ella misma), ésta realiza un estudio inicial del software cuyo mantenimiento se solicita y del hardware sobre el que éste se ejecuta. Tras este estudio, se pretende que la organización que presta el servicio

conozca las características del sistema de información que se va a mantener, de manera que, si procede, pueda ir preparando presupuestos, contratos de prestación de servicios, etc.

Cuando la organización cliente acepta la propuesta de mantenimiento de la organización suministradora del servicio, se realiza una planificación del proceso a la medida de la organización cliente. Estas acciones se llevan a cabo en el conjunto de *Actividades y tareas iniciales* que presentábamos en la Figura 19 y cuyo nivel de detalle aumentamos en la Figura 21. Tras esta segunda actividad (*Planificación del proceso*), la responsabilidad de la ejecución de las intervenciones de mantenimiento sobre el software convenido en la actividad primera (*Estudio inicial*) recae ya sobre la organización de mantenimiento, que estará preparada para recibir peticiones de modificación, clasificarlas de acuerdo a uno de los tipos de mantenimiento definidos en MANTEMA y servirlas conforme a las actividades (y a sus correspondientes tareas) enunciadas en los nodos centrales de la Figura 21.

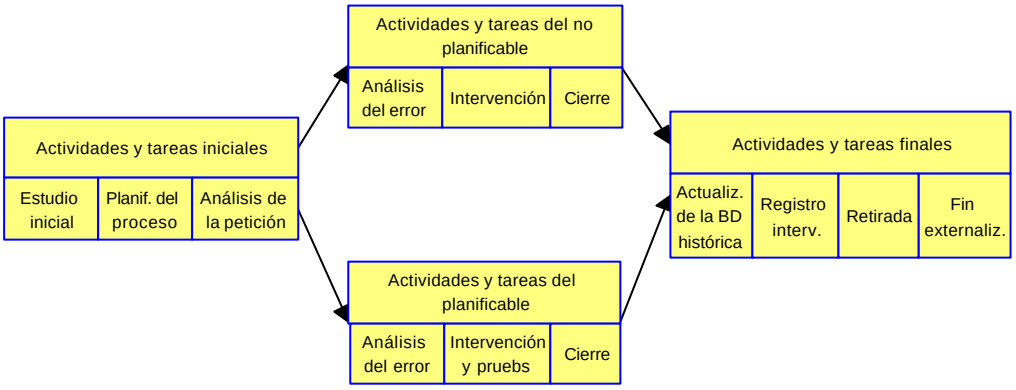


Figura 21. Actividades que componen cada nodo de los que representan la metodología.

Ya servida completamente la petición (lo cual incluye el paso a producción del software modificado), la organización de mantenimiento llega al nodo final, en el que, para cada petición de modificación, debe ejecutar algunas de las *Actividades y tareas finales*, comunes también a todos los tipos de mantenimiento. En particular, debe realizarse la actualización de la base de datos histórica para almacenar toda la información relevante de la petición, registrar la intervención y, en caso necesario, proceder a la retirada del software. La actividad denominada *Fin de la externalización* es realizada cuando ha existido una relación contractual de prestación de servicios de mantenimiento entre organizaciones diferentes que llega a su punto final o, más genéricamente, cuando va a cambiar la organización suministradora del servicio de mantenimiento.

Se observan dos conjuntos de actividades claramente diferentes (Figura 22):

- El primero, formado por las actividades que, por lo general, son realizados solamente una vez, tanto al principio como al final del proceso de mantenimiento (*Estudio inicial* y *Planificación del proceso*, en las Actividades y tareas iniciales, y *Fin de la externalización*, en las Actividades y tareas finales).

- El segundo, compuesto de un conjunto de actividades repetitivo, que se ejecuta una vez para cada petición de modificación recibida, y que son las que se encuentran situadas, en la Figura 21, entre las mencionadas en el punto anterior.

Teniendo en cuenta estas observaciones, podemos precisar un poco más el concepto de "proceso de mantenimiento" indicando que está formado por:

1. Un conjunto de actividades destinadas a preparar y planificar las actividades posteriores del propio proceso.
2. El conjunto de intervenciones que producen modificaciones sobre el software, y cuya ejecución debe seguir las normas que se hayan preparado en el punto anterior.
3. Un conjunto de actividades que deben realizarse con posterioridad a cada modificación sobre el software.
4. Opcionalmente, una actividad que guíe en la finalización de la prestación del servicio de mantenimiento.

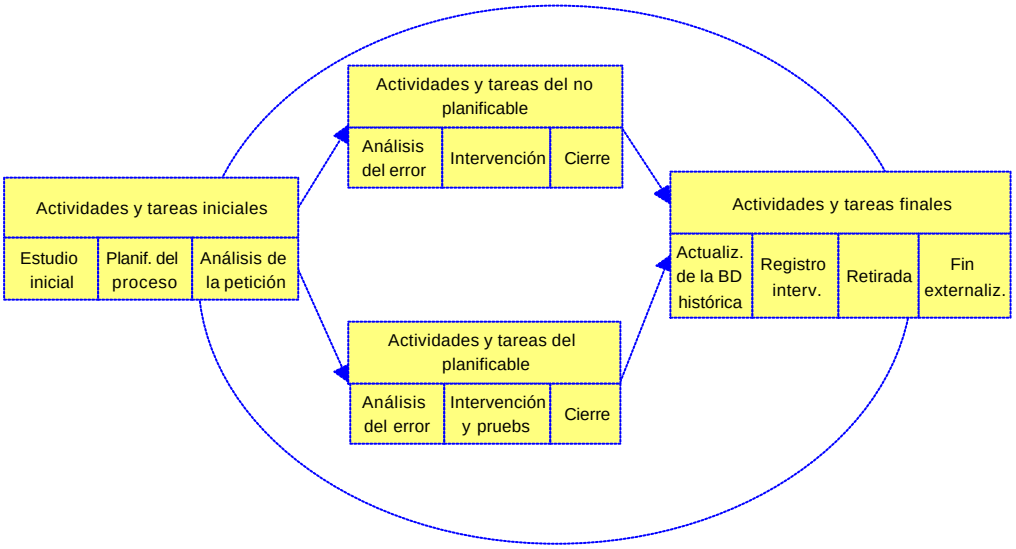


Figura 22. Actividades cíclicas en la metodología.

4.2. ESTRUCTURA DETALLADA DE LA METODOLOGÍA.

En esta sección dedicamos un epígrafe a cada uno de los nodos que componen la Figura 19. Para cada actividad de cada nodo, indicamos las tareas que lo componen. Del mismo modo, detallamos las entradas, salidas, personal responsable, técnicas y métricas de cada tarea, de manera que determinamos los "qué, cómo, dónde, cuándo, cómo y por qué" del proceso (Kellner y Hansen, 1988), como ya apuntamos en la Figura 20.

Muchos de los objetos que las tareas toman como entradas y que producen como salidas son documentos, que en el texto aparecerán etiquetados con el prefijo *DOC* seguido de un número; las plantillas de estos documentos se encuentran recogidas en el Apéndice I. No obs-

tante, al finalizar la sección dedicada a cada nodo se ofrece una tabla con la lista de documentos que se pueden generar.

4.2.1. ACTIVIDADES Y TAREAS INICIALES COMUNES.

Actividad I0. Estudio inicial.

Esta actividad se ejecuta cuando la organización Cliente ha contactado con la Organización de mantenimiento, con el fin de que ésta le proporcione el servicio de mantenimiento de su sistema de información o de parte de él. La Organización de mantenimiento no es todavía la responsable de la ejecución de las intervenciones.

Esta actividad consta de las siguientes tareas:

Tarea I0.1 Iniciar y recoger información

En esta tarea, como respuesta a una *Solicitud de prestación del servicio de mantenimiento*, el Equipo de mantenimiento (perfil de la Organización de mantenimiento) y la Organización del sistema (Cliente) rellena un *Cuestionario inicial* (DOC1, página 219) en el que detalla ciertos aspectos del software que se deberá mantener (sistema operativo, lenguaje o lenguajes de programación, nº de programas y cualesquiera otros datos que puedan resultar de interés para la Organización de mantenimiento), según una muestra tomada del producto software.

Tarea I0.2 Preparar propuesta de mantenimiento

De los datos recogidos en la tarea I0.1, el Responsable de mantenimiento (uno de los perfiles de la Organización de mantenimiento) crea una *Propuesta de mantenimiento* (DOC2) del software. En la preparación de la propuesta (a la que, en su caso, acompañará una propuesta económica), puede utilizarse la técnica de Identificación de Riesgos que describimos en la sección 4.3.1 (página 127), utilizando en este caso el documento DOC4.

Como se indica en la plantilla del documento DOC2 (página 220), en este documento deben recogerse los valores de los Indicadores de servicio (sección 4.4.3.1, página 174) propuestos por la Organización de mantenimiento.

Tarea I0.3 Contrato

Se redacta y se firma el *Contrato de mantenimiento* (DOC3, página 220) entre la Organización del sistema y el Responsable de mantenimiento. Esta tarea puede omitirse si ambas organizaciones son la misma, aunque su ejecución sea obligatoria cuando exista externalización.

Actividad I1. Planificación del proceso.

Esta actividad tiene como fin la cesión de la responsabilidad del mantenimiento a la Organización de mantenimiento. Sin embargo, antes de adquirir tal responsabilidad, deben realizarse las siguientes tareas:

Tarea I1.1 Planificar calendario y responsables

Se genera un calendario de reuniones y se enumeran los interlocutores válidos y los responsables de ambas partes (Organización de mantenimiento y Cliente). Las reuniones planificadas pueden servir como hitos de comprobación de resultados.

Tarea I1.2 Adquirir conocimiento de la aplicación

En esta tarea, el Equipo de mantenimiento debe estudiar la documentación existente del software objeto de la prestación del servicio, el código de los programas, referencias cruzadas, entrevistarse con los usuarios, observar cómo trabaja la organización de mantenimiento actual, etc. Durante esta tarea, el encargado de resolver las peticiones de modificación es el propio Cliente (o la organización que tenga en ese momento contratada); sin embargo, el Equipo de mantenimiento observa su trabajo para conocer mejor el software objeto de la prestación del servicio.

Esta tarea puede durar entre uno y dos meses, durante los cuales el Equipo de mantenimiento no habrá probablemente modificado una sola línea de código. Al finalizar esta tarea, el Equipo de mantenimiento debe entregar al Cliente un informe acerca del estado de su software, de manera que, entre otras cosas, se permite verificar que el Equipo de mantenimiento ha adquirido un conocimiento adecuado del software que deberá mantener.

Por otro lado, las circunstancias pueden hacer que esta tarea no sea ejecutada total o parcialmente (el software no ha sido mantenido con anterioridad, por ejemplo).

Tarea I1.3 Desarrollar planes

El Equipo de mantenimiento desarrolla los planes de mantenimiento y construye el *Resumen Técnico* (DOC5, página 221) del aplicativo, una vez adquirido el conocimiento. Puede utilizarse la técnica descrita en la sección 0, página 129) para estimar la cantidad de recursos que deben dedicarse a atender peticiones de mantenimiento no planificable.

Debe inventariarse el sistema de información que se va a mantener, debiendo recogerse como mínimo, para cada aplicación y base de datos, la siguiente información:

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
PF	Número de puntos-función
NMód	Número de módulos
NLC	Número de líneas de código
CC	Complejidad ciclomática media de cada módulo
AcopEst	Número de variables globales (visibles externamente) de cada módulo
AcopDin	Número de rutinas públicas (visibles externamente) de cada módulo
MétBD	Métricas de cada una de las bases de datos del sistema (véase página 140)

Tarea I1.4 Definir procedimientos de petición de modificación

El Equipo de mantenimiento genera modelos de documentos para la presentación de cada *Petición de modificación* (DOC6, página 222). Así mismo, establece, junto al Cliente, los procedimientos de petición de modificación, en los que indica quién será el receptor de la petición, quién es el encargado de su estudio, etc.

Tarea I1.5 Implementar proceso de G.C.S.

Esta tarea establece una interfaz con el proceso de gestión de configuración existente en la organización para gestionar las modificaciones que se realicen sobre el sistema existente.

Tarea I1.6 Preparar entornos de pruebas

El Equipo de mantenimiento prepara copias del entorno software de producción para implementar sobre ellas las intervenciones de mantenimiento.

Actividad I2. Análisis de la petición de modificación.

Ésta es la primera de las actividades que se ejecutarán para cada petición de modificación recibida. En esta actividad, la Organización de mantenimiento ya es responsable del mantenimiento y debe satisfacer cada una de las peticiones (en el caso de que exista contrato de pres-

tación del servicio, es posible que no se hayan contratado intervenciones de todos los tipos de mantenimiento, lo que deberá ser tenido en cuenta por la organización).

Esta actividad consta de las siguientes tareas:

Tarea I2.1 Recibir petición de modificación

El Solicitante (perfil de Cliente) o, en caso de estar autorizado para ello, el Usuario, entrega una *Petición de modificación* (DOC6), que es recibida y registrada por el Gestor de peticiones (uno de los perfiles de la Organización de mantenimiento), que le asigna un identificador único.

Las siguientes informaciones deben recogerse:

Información	Explicación
PerQPide	Persona que hace la petición
MedioDPet	Medio por el que se realiza la petición

Tarea I2.2 Decidir tipo de mantenimiento

A partir de la *Petición de modificación* (DOC6) recibida y registrada en la tarea I2.1, el Gestor de peticiones decide o bien rechazar la petición, o bien aceptarla y decidir el tipo de mantenimiento que debe aplicarse. En caso de elegir la primera opción, deben justificarse las razones. En caso de aceptación, la petición de modificación es entregada al Planificador (perfil de la Organización de Mantenimiento), que la distribuye y encauza al tipo de mantenimiento que corresponda.

Las siguientes informaciones deben ser recogidas:

Información	Explicación
AcRech	Petición aceptada o rechazada
TipoMantto	Tipo de mantenimiento de la petición
CausaPet	Causa de la petición: ❖ Cambios legales ❖ Evolución del negocio ❖ Mejora del proceso ❖ Cambio en las reglas de negocio ❖ Otras (indicar)
AreaPet	Área o dominio funcional que ha presentado la petición

4.2.1.1. Tablas de resumen: Actividades y tareas iniciales comunes

En las siguientes dos tablas se ofrece resumidamente este conjunto de actividades y tareas.

	Actividades y tareas iniciales comunes (1 de 2)					
	Estudio inicial			Planificación del proceso		
	I0.1 Iniciar y recoger información	I0.2 Preparar propuesta de mantenimiento	I0.3 Contrato	I1.1 Planificar calendario y responsables	I1.2 Adquirir conocimiento de la aplicación	I1.3 Desarrollar planes
Entradas	Solicitud de prestación del servicio de mantenimiento	Cuestionario inicial Entrevistas	Propuesta de mantenimiento	Contrato de mantenimiento (DOC3)	Producto software en operación o desarrollo	Producto software en operación Lista de elementos software Nueva documentación del aplicativo
Salidas	Cuestionario inicial (DOC1)	Propuesta de mantenimiento (DOC2) Documento de riesgos (DOC4)	Contrato de mantenimiento (DOC3)	Listado de responsables, calendario de reuniones, etc..	Lista de elementos software Nueva documentación del aplicativo Contrato definitivo de mantenimiento (DOC3)	Resumen técnico (DOC5) Plan de mantenimiento del periodo
Técnicas	Entrevista	Identificación y estimación de riesgos (sección 4.3.1, página 127)			Estudio de la documentación Observación y entrevistas Instalación de herramientas Ingeniería inversa	Planificación Estimación de recursos para mantenimiento no planificable (sección 0, página 129)
Responsable	Equipo de mantenimiento Organización del sistema	Responsable de mantenimiento	Organización del sistema Responsable de mantenimiento	Organización del sistema Responsable de mantenimiento	Equipo de mantenimiento Organización del sistema Usuario	Equipo de mantenimiento
Interfaces con otros procesos			Aseguramiento de la calidad			
	Pueden omitirse si no se externaliza					

	Actividades y tareas iniciales comunes (2 de 2)				
	Planificación del proceso			Análisis de la petición de modificación	
	I1.4 Definir procedimientos de petición de modificación	I1.5 Implementar proceso de G.C.S.	I1.6 Preparar entornos de pruebas	I2.1 Recibir petición de modificación	I2.2 Decidir tipo de mantenimiento
Entradas	Plan de mantenimiento	Sistema en explotación	Elementos software del sistema en operación	Petición de modificación (DOC6)	Petición de modificación recibida
Salidas	Modelos de documentos Procedimientos	Proceso de gestión de la configuración	Copias de los elementos software en operación	Petición de modificación recibi-da	Informe sobre el tipo de mantenimiento necesario, su criticidad, etc. Decisión sobre el tipo de mantenimien-to a aplicar
Técnicas					Evaluación de impacto
Responsable	Equipo de mantenimiento Organización del s istema Solicitante	Equipo de mantenimiento	Equipo de mantenimiento	Gestor de peticiones Usuario Solicitante	Gestor de peticiones Planificador
Interfaces con otros procesos	Gestión de la Configuración	Gestión de la Configuración	Gestión de la Configuración		

4.2.1.2. Relación de documentos de las Actividades y tareas iniciales comunes.

Los documentos que se han generado durante esta etapa se recogen en la tabla siguiente. Los contenidos de estos documentos se encuentran en el Apéndice I.

Nombre del documento	*Código*
Cuestionario inicial	DOC1
Propuesta de mantenimiento	DOC2
Contrato de mantenimiento (del cual se preparan dos ejemplares: uno provisional y otro definitivo)	DOC3
Tabla de factores de riesgo (véase sección 4.3.1)	DOC4
Resumen técnico	DOC5
Petición de modificación	DOC6

4.2.2. ACTIVIDADES Y TAREAS DEL MANTENIMIENTO NO PLANIFICABLE (CORRECTIVO URGENTE)

Estaremos en este caso cuando el error paralice de manera seria el funcionamiento normal del resto del sistema de información o el de la organización, de forma que la corrección del error deba ser inminente.

Actividad NP1. Análisis del error. Esta actividad consta de las siguientes tareas:

Tarea NP1.1 Investigar y analizar causas

El Equipo de mantenimiento analiza la *Petición de Modificación* (DOC6, página 222), verifica el problema (quizás con la colaboración del Usuario) y lo reproduce, y estudia diferentes alternativas para implementar la modificación. Además, debe construir una lista de los elementos software a corregir (módulos, rutinas, documentos, etc.).

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NPFAfect	Número de puntos-función que previsiblemente se verán afectados por el error

También deben recogerse las siguientes informaciones:

Información	Explicación
Bas-OrigenError	Origen del error (de acuerdo al Apéndice II)
Bas-CausaError	Causa del error (de acuerdo al Apéndice II)

Actividad NP2. Intervención correctiva urgente. Esta actividad consta de las siguientes tareas:

Tarea NP2.1 Realizar acciones correctivas

El equipo de mantenimiento ejecuta las acciones necesarias para corregir el problema detectado.

Deben identificarse las rutinas y bases de datos afectadas por la intervención, debiendo recogerse, para cada una, las siguientes métricas:

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NPFAñad	Número de puntos-función añadidos
NPFModif	Número de puntos-función modificados
NPFBorr	Número de puntos-función borrados
NModAlter	Número de módulos alterados
NLCAñad	Número de líneas de código añadidas
NLCModif	Número de líneas de código modificadas
NLCBorr	Número de líneas de código borradas
CC	Complejidad ciclomática media de cada módulo alterado
AcopEst	Número de variables globales (visibles externamente) de cada módulo
AcopDin	Número de rutinas públicas (visibles externamente) de cada módulo
MétBD	Métricas para bases de datos, en caso de que se haya modificado alguna (véase sección 4.4)

Tarea NP2.2 Cumplimentar documentación

El Equipo de mantenimiento debe documentar los cambios realizados en el *Documento de Acciones Correctivas Realizadas* (DOC7, página 222).

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NPág	Número de páginas añadidas a la documentación, según el estándar de la Organización

Tarea NP2.3 Ejecutar pruebas unitarias

El Equipo de mantenimiento debe comprobar la corrección de todos los cambios realizados. Estas pruebas se documentarán en el *Documento de Pruebas Unitarias Realizadas* (DOC8, página 223).

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NErrDetec	Número de errores detectados

Actividad NP3. Cierre intervención. Esta actividad consta de la siguiente tarea:

Tarea NP3.1 Pasar a producción

El Equipo de mantenimiento pasa al entorno de producción el software corregido para su utilización por parte de los usuarios. Se establece aquí una interfaz con el proceso de Gestión de la Configuración para garantizar la corrección de la tarea.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea

4.2.2.1. Tablas de resumen: Actividades y tareas del mantenimiento no planificable (correctivo urgente)

En la siguiente tabla se ofrece resumidamente este tipo de mantenimiento.

	Actividades y tareas del mantenimiento no planificable (correctivo urgente)				
	Análisis del error	Intervención correctiva urgente			Cierre intervención
	NP1.1 Investigar y analizar causas	NP2.1 Realizar acciones correctivas	NP2.2 Cumplimentar documentación	NP2.3 Ejecutar pruebas unitarias	NP3.1 Pasar a producción
Entradas	Producto software en explotación con error bloqueante o crítico Petición de modificación	Conjunto de elementos software a corregir	Elementos software antiguos (con errores visibles)	Elementos software corregidos Casos de prueba	Elementos software corregidos y probados
Salidas	Conjunto de elementos software a corregir	Conjunto de elementos software corregidos	Elementos software corregidos Documento de acciones correctivas realizadas (DOC7)	Elementos software corregidos y probados Documento con las pruebas unitarias realizadas (DOC8)	Producto software en explotación corregido
Técnicas		Codificación		Técnicas de prueba del software	
Responsable	Equipo de mantenimiento Usuario	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento Usuario
Interfaces con otros procesos		Aseguramiento de la calidad Gestión de la configuración		Aseguramiento de la calidad	Gestión de la configuración

4.2.2.2. Relación de documentos de las Actividades y tareas del correctivo urgente

Los documentos que se han generado durante esta etapa se recogen en la tabla siguiente. Los contenidos de estos documentos se encuentran en el Apéndice I.

Nombre del documento	*Código*
Acciones Correctivas Realizadas	DOC7
Pruebas Unitarias Realizadas	DOC8

4.2.3. ACTIVIDADES Y TAREAS DEL MANTENIMIENTO PLANIFICABLE

Puesto que no todas las actividades del mantenimiento planificable son aplicables a todos los tipos de mantenimiento que hemos englobado bajo esta denominación (correctivo no urgente, perfectivo, preventivo y adaptativo) -ya que conservan algunas diferencias-, indicamos con las siguientes claves los tipos de mantenimiento a los que cada tarea es aplicable.

Claves: CP→ mantenimientos correctivo no urgente y perfectivo.

A→ adaptativo.

P→ preventivo.

Actividad P1. Análisis de la petición. Esta actividad consta de las siguientes tareas:

Tarea P1.1 Valorar petición (CP/P)

La *Petición de modificación* (DOC6, página 222) -que fue recibida en la tarea I2.1 del conjunto de Actividades y tareas iniciales comunes y distribuida en la tarea I2.2- está ya en posesión del Equipo de mantenimiento que va a llevar a cabo la intervención.

En esta tarea, se valoran los costes y esfuerzos necesarios para servir la petición. Igualmente, se valoran la disponibilidad de recursos y de calendario para estimar los plazos de la intervención.

Para realizar la estimación del esfuerzo de la intervención puede utilizarse alguno de los métodos mostrados en la sección 3.5 (página 79).

Para el caso de **mantenimiento preventivo**, y con el fin de conocer cuál es la aplicación más susceptible de modificar, puede utilizarse el Análisis de Cartera de Sneed (1995).

La petición de modificación es puesta en la cola de espera.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea

Informaciones	Explicación
FechIni	Fecha estimada de inicio de la intervención
FechFin	Fecha estimada de finalización de la intervención

Tarea P1.2 Documentar posibles soluciones (CP/P).

A partir del producto software en explotación y de la petición de modificación:

- Si se trata de CP (correctivo no urgente o perfectivo), se documenta la causa del error y se indican las posibles alternativas de implementación en el documento de *Diagnóstico del Error y Posibles Soluciones* (DOC9, página 223), dejando pendiente de rellenar la alternativa de implementación elegida.
- Si se trata de P (preventivo), se produce una *Lista de Elementos Software y Propiedades Mejorables* (DOC12, página 224).

  Se rellena, además, el documento DOC16 (identificado en la tabla-resumen de esta sección como DOC16a, cuya plantilla se detalla en la página 225), en el que se recogen los valores de métricas del producto que correspondan. Los datos de este documento, junto a los del DOC16b –que será generado posteriormente en la tarea P2.6 (Ejecutar paralelamente en software antiguo y nuevo)- son utilizados en las Actividades y tareas finales comunes para incorporarlos a la base de datos histórica.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NPFAfect	Número de puntos-función que previsiblemente se verán afectados por el error

Informaciones	Explicación	Observ.
Bas-OrigenError	Origen del error (de acuerdo al Apéndice II)	Correctivo no urgente
Bas-CausaError	Causa del error (de acuerdo al Apéndice II)	

Tarea P1.3 Elegir alternativa adecuada (CP).

El equipo de mantenimiento completa el documento de *Diagnóstico del Error y Posibles Soluciones* (DOC9, página 223), indicando la alternativa de corrección elegida. Para la elección de la alternativa pueden realizarse estimaciones de datos de proyectos anteriores para optimizar la dedicación de recursos.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea

Actividad P2. Intervención y pruebas. Esta actividad consta de las siguientes tareas:

Tarea P2.1 Planificar calendario (A).

El Equipo de mantenimiento realiza una planificación del calendario de la intervención en el que debe detallar cuándo se acometerán las diferentes fases de la adaptación.

También se rellena el documento DOC16 (identificado en la tabla-resumen de esta sección como DOC16a), en el que se recogen los valores de métricas del producto que correspondan. Los datos de este documento, junto a los del DOC16b (generado más adelante, en la tarea P2.4) son utilizados en las actividades y tareas finales comunes para incorporarlos a la base de datos histórica.

Para los casos en que tanto el sistema origen como el destino son **orientados a objetos**, pueden utilizarse las métricas *CC* de Fioravanti, Nesi y Polo (1999), presentadas en la sección 4.4.2 (página 161).

Nótese que esta tarea sólo se realiza en el mantenimiento adaptativo.

Tarea P2.2 Realizar copia del producto software (A).

Antes de iniciar la intervención adaptativa, el equipo de mantenimiento realiza una copia del producto software y de todas las bases de datos, ficheros, etc. relacionados.

Tarea P2.3 Ejecutar intervención (CP/P/A).

El equipo de mantenimiento debe ejecutar las acciones necesarias para servir la petición de modificación conforme a la alternativa seleccionada -la cual está desarrollada en el DOC9 (*Diagnóstico del Error y Posibles Soluciones*)-. Se deben generar los correspondientes documentos de acciones realizadas (DOC7, DOC11 o DOC13, páginas 222 y 224).

Deben identificarse las rutinas y bases de datos afectadas por la intervención, debiendo recogerse, para cada una, las siguientes métricas:

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NPFAñad	Número de puntos-función añadidos
NPFModif	Número de puntos-función modificados
NPFBorr	Número de puntos-función borrados
NModAlter	Número de módulos alterados
NLCAñad	Número de líneas de código añadidas
NLCModif	Número de líneas de código modificadas
NLCBorr	Número de líneas de código borradas
CC	Complejidad ciclomática media de cada módulo
AcopEst	Número de variables globales (visibles externamente) de cada módulo modificado
AcopDin	Número de rutinas públicas (visibles externamente) de cada módulo modificado
MétBD	Métricas para bases de datos (página 140)

Tarea P2.4 Ejecutar pruebas unitarias (CP/P/A).

El Equipo de mantenimiento realiza las pruebas unitarias sobre el producto software intervenido. Debe comprobarse que la *Petición de Modificación* (DOC6) queda servida. Una vez finalizadas, se genera el *Documento de Pruebas Unitarias Realizadas* (DOC8, página 223).

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NerrDetec	Número de errores detectados

Tarea P2.5 Ejecutar pruebas de integración (CP/P/A).

El Equipo de mantenimiento debe comprobar que los diferentes elementos software funcionan correctamente de forma conjunta.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NErrDetec	Número de errores detectados

Tarea P2.6 Ejecutar paralelamente en software antiguo y nuevo (CP/P/A).

El Equipo de mantenimiento ejecuta acciones reales en el producto software antiguo y en el modificado para detectar y prevenir posibles errores de proceso. Pueden aplicarse pruebas de no regresión, de manera que se repiten casos de prueba anteriores a la intervención. En el caso del mantenimiento adaptativo se reconstruye, si es preciso, la documentación técnica del sistema.

Además, durante esta tarea se rellena un ejemplar del DOC16 (identificado en la tabla como DOC16b), en el que se recogen los valores de las métricas del producto que correspondan al producto ya intervenido.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NerrDetec	Número de errores detectados

Tarea P2.7 Verificar y validar corrección con el cliente (CP/P/A).

El Equipo de mantenimiento y la Organización del sistema (o el perfil en que se haya delegado) se reúnen para comprobar que el producto intervenido funciona correctamente.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea

Tarea P2.8 Redocumentar manual de usuario (CP/P/A).

El Equipo de mantenimiento debe redocumentar el manual de usuario, si es que ha cambiado el modo de operación del software.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea
NPágMan	Número de páginas (conforme al estándar de la organización) añadidas/borradas/modificadas al manual

Tarea P2.9 Pasar a producción (CP/P/A).

El Equipo de mantenimiento instala el producto software modificado en el entorno de trabajo real. En el caso del mantenimiento adaptativo (y tal vez en algún caso aislado de otro tipo de mantenimiento), puede establecerse una interfaz con el Proceso de Infraestructura por si es preciso cambiar o agregar máquinas, etc.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea

Tarea P2.10 Realizar revisión (CP/A).

El Equipo de mantenimiento comprueba que el producto modificado funciona correctamente en el entorno de trabajo real.

Métrica	Explicación
Tiempo	Tiempo dedicado a la tarea

Actividad P3. Cierre de intervención. Esta actividad consta de la siguiente tarea:

Tarea P3.1 Archivar datos del producto software inicial (A).

El Equipo de mantenimiento realiza una copia de seguridad de todo lo relacionado con el producto software que se ha adaptado (programas, datos, documentación, etc.).

4.2.3.1. Tabla-guía.

Para mejorar la legibilidad de la tabla de mantenimiento planificable, marcamos a continuación (Tabla 14) las tareas obligatorias según el tipo de mantenimiento.

	Correctivo no urgente/perfectivo	Preventivo	Adaptativo
P1.1	X	X	
P1.2	X	X	
P1.3	X		
P2.1			X
P2.2			X
P2.3	X	X	X
P2.4	X	X	X
P2.5	X	X	X
P2.6	X	X	X
P2.7	X	X	X
P2.8	X	X	X
P2.9	X	X	X
P2.10	X		X
P3.1			X

Tabla 14. Tareas aplicables en el mantenimiento planificable.

4.2.3.2. Tablas de resumen: Actividades y tareas del mantenimiento planificable (correctivo no urgente, perfectivo, preventivo y adaptativo).

Las tres siguientes tablas muestran resumidamente este tipo de mantenimiento.

	Actividades y tareas del mantenimiento planificable (1 de 3)					
	Análisis de la petición			Intervención y pruebas		
	CP/P	CP/P	CP	A	A	CP/P/A
	P1.1 Valorar petición	P1.2 Documentar posibles soluciones	P1.3 Elegir alternativa adecuada	P2.1 Planificar calendario	P2.2 Realizar copia del producto software	P2.3 Ejecutar intervención
Entradas	Producto software en explotación. Petición de modificación (DOC6).	Producto software en explotación Petición de modificación en espera	Producto software en explotación Alternativas de implementación (DOC10)	Documentación del proyecto	Producto software en explotación	CP: Producto software en explotación Diagnóstico del error (DOC9) Alternativa seleccionada P: Producto software en explotación Lista de elementos software y propiedades mejorables (DOC12) A: Copia del producto software
Salidas	Petición de modificación en espera Calendario de intervención P: Lista de elementos software y propiedades mejorables (DOC12)	CP: Diagnóstico del error y posibles soluciones (DOC9) Alternativas de implementación (DOC10) Medidas del producto (DOC16a) P: Lista de elementos software y propiedades mejorables (DOC12) Medidas del producto software en explotación (DOC16a)	Alternativa seleccionada (DOC9 completo)	Calendario de intervención Estimación de plazos Disponibilidad de recursos	Copia del producto software	CP: Producto software corregido Documento de acciones correctivas/perfectivas P: Producto software perfeccionado Documento de acciones preventivas (DOC13) A: Copia adaptada
Técnicas	Estimación de esfuerzos (sección 3.5, página 79) Gestión de proyectos P: Análisis de cartera (Sneed, 1995)	Ingeniería inversa Análisis de la documentación del proyecto	Consulta a la base de datos histórica	Gestión de proyectos Métricas CC para mantto. adaptativo (sección 4.4.2, página 161)		Reingeniería Reestructuración Codificación Redocumentación
Responsable	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento
Interf. con otros proc		Aseguramiento de la calidad	Aseguramiento de la calidad		Gestión de la configuración	Aseguramiento de la calidad

	Actividades y tareas del mantenimiento planificable (2 de 3)					
	Intervención y pruebas					
	CP/P/A	CP/P/A	CP/P/A	CP/P/A	CP/P/A	CP/P/A
	P2.4 Ejecutar pruebas unitarias	P2.5 Ejecutar pruebas de integración	P2.6 Ejecutar paralelamente en software antiguo y nuevo	P2.7 Verificar y validar corrección con el cliente	P2.8 Redocumentar manual de usuario	P2.9 Pasar a producción
Entradas	Producto intervenido (corregido, perfeccionado o copia adaptada) Documento de intervención (de acciones correctivas, perfectivas, preventivas o adaptativas: DOC7/11/13) Petición de modificación (DOC6)	Producto intervenido comprobado unitariamente Petición de modificación (DOC6)	Producto intervenido comprobado	Documento de intervención (de acciones correctivas, perfectivas, preventivas o adaptativas: DOC7/11/13) Elementos software corregidos y probados Documentación con las pruebas realizadas (DOC8)	Producto software totalmente comprobado Manual del usuario	Producto software totalmente comprobado
Salidas	Producto intervenido comprobado unitariamente Documento con las pruebas unitarias realizadas (DOC8)	Producto intervenido comprobado Documento con las pruebas de integración realizadas	Producto intervenido comprobado y en correcto funcionamiento Medidas del producto (DOC16b) A Documentación actualizada	Documento de aceptación de la corrección validado por el cliente Producto software totalmente comprobado	Manual de usuario redocumentado	Producto software totalmente comprobado en explotación con la Petición de Modificación servida
Técnicas	Técnicas de prueba	Técnicas de prueba	Realización de las operaciones reales en las versiones antigua y nueva del producto software Técnicas de no regresión			
Responsable	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento Usuario	Equipo de mantenimiento Solicitante Usuario	Equipo de mantenimiento	Equipo de mantenimiento
Interf. con otros proc	Aseguramiento de la calidad Validación Verificación	Aseguramiento de la calidad Validación Verificación	Operación	Revisión conjunta		Gestión de la configuración Infraestructura

	Actividades y tareas del mantenimiento planificable (3 de 3)	
	Cierre de intervención	
	CP/A	A (sólo Adaptativo)
	P2.10 Realizar revisión	P3.1 Archivar datos del producto inicial
Entradas	Producto software en explotación con la Petición de Modificación servida	Producto software inicial Datos relacionados con el producto software inicial
Salidas	Producto software en explotación con la Petición de Modificación servida revisado	Producto software inicial archivado Datos del producto software antiguo archivados
Técnicas		
Responsable	Equipo de mantenimiento	Equipo de mantenimiento
Interfaces con otros procesos		Gestión de la configuración

4.2.3.3. Relación de documentos de las Actividades y tareas del mantenimiento planificable

Los documentos que se han generado durante esta etapa se recogen en la tabla siguiente. Los contenidos de estos documentos se encuentran en el Apéndice I.

Nombre del documento	*Código*
Diagnóstico y posibles soluciones	DOC9
Alternativas de implementación	DOC10
Acciones perfectivas realizadas	DOC11
Lista de elementos software y propiedades mejorables	DOC12
Documentación de las acciones preventivas realizadas	DOC13

4.2.4. ACTIVIDADES Y TAREAS FINALES.

Este conjunto de actividades y tareas es común a todos los tipos de mantenimiento. En la primera actividad de este grupo se actualiza la base de datos de datos históricos, que sirve para realizar las estimaciones de recursos y tiempo de cada intervención. A continuación, se incluye la retirada del software (actividad directamente adoptada de la norma ISO/IEC 12207, que será ejecutada cuando el software vaya a retirarse definitivamente del entorno operativo), y el fin de la externalización, que podrá ser omitida cuando no haya existido externalización del mantenimiento.

Actividad F1. Registro de la intervención. Esta actividad consta de la siguiente tarea:

Tarea F1.1 Registrar intervención.

La intervención queda registrada, según los procedimientos de la organización.

Actividad F2. Actualización de la base de datos histórica. Esta actividad consta de las siguiente tareas:

Tarea F2.1 Recoger información de la intervención.

Si las métricas indicadas para cada tarea no se hubieran ido recogiendo y almacenando durante la intervención, tal recolección debería ser hecha en esta tarea.

Tarea F2.2 Actualizar base de datos.

Los datos recogidos en la tarea anterior son incorporados a la base de datos histórica.

Actividad F3. Retirada. Esta actividad consta de las siguientes tareas:

Tarea F3.1 Desarrollar plan de retirada

El Equipo de mantenimiento redacta un documento en el que describe cuándo se llevará a cabo la retirada del software.

Tarea F3.2 Notificar futura retirada

El Equipo de mantenimiento notifica al Cliente el momento en el que se ejecutará la retirada.

Tarea F3.3 Ejecutar paralelo

El Usuario, con el visto bueno del Cliente y bajo la supervisión del Equipo de mantenimiento, realiza operaciones reales sobre el software que se va a retirar y el software nuevo (si es que aquél va a ser sustituido).

Tarea F3.4 Notificar retirada

Se notifica la inminencia de la retirada.

Tarea F3.5 Almacenar datos del entorno antiguo

Los datos del entorno antiguo son almacenados.

Actividad F4. Fin de la externalización.

Tarea F4.1 Entrega del inventario y de la documentación

Dependiendo de lo que se pactara en el contrato, posiblemente la Organización de mantenimiento se encuentre obligada a entregar al cliente todos los productos software generados y modificados durante el periodo en que ha sido responsable del mantenimiento.

Tarea F4.2 Traspaso de experiencia y formación

Esta es la tarea inversa a la I1.2 (Adquirir conocimiento de la aplicación), durante la cual el Equipo de mantenimiento aprendió las características del sistema observando el modo de trabajo de la organización de mantenimiento que el Cliente tuviera a la sazón. En este caso, es la Organización de mantenimiento la que debe formar al nuevo personal de mantenimiento. En el último periodo de esta tarea (variable según el contrato), el mantenimiento se realiza con equipos mixtos.

Tarea F4.3 Cesión definitiva del servicio

La Organización de mantenimiento deja de prestar sus servicios al Cliente con carácter definitivo.

4.2.4.1. Tablas de resumen: Actividades y tareas finales

En la siguiente tabla se ofrecen resumidamente estas actividades

.

	Actividades y tareas comunes finales (1 de 2)							
	Registro de intervención	Actualización de la base de datos histórica		Retirada (procedente de ISO/IEC 122207)				
	F1.1 Registrar intervención	F2.1 Recoger información de la intervención	F2.2 Actualizar base de datos	F3.1 Desarrollar plan de retirada	F3.2 Notificar futura retirada	F3.3 Ejecutar paralelo	F3.4 Notificar retirada	F3.5 Almacenar datos del entorno antiguo
Entradas	Documentación generada en esta etapa	Intervención registrada	Datos históricos de la intervención	Productos software antiguos en explotación Nuevo producto software preparado	Plan de retirada	Plan de retirada Productos software antiguos en explotación Nuevos productos software preparados	Plan de retirada	Datos del entorno antiguo
Salidas	Documentación registrada	Datos históricos de la intervención	Datos históricos de la intervención almacenados	Plan de retirada	Notificación a los usuarios con la retirada	Coexistencia de los productos software antiguos y nuevos	Nuevos productos software en explotación Productos antiguos retirados	Datos del entorno antiguo almacenados
Responsable	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento	Equipo de mantenimiento Cliente	Equipo de mantenimiento Cliente Usuario	Equipo de mantenimiento Cliente	Equipo de mantenimiento
Interfaces con otros procesos	Gestión de la Configuración		Gestión de la Configuración			Formación		

	Actividades y tareas comunes finales (2 de 2)		
	Fin de la externalización		
	F4.1 Entrega del inventario y de la documentación	F4.2 Traspaso de experiencia y formación	F4.3 Cesión definitiva del servicio
Entradas	Documentación y productos generados durante el proceso de externalización Contrato	Experiencia del equipo de mantenimiento de la Organización de mantenimiento sobre el software mantenido objeto del contrato (este producto de entrada es, desde luego, un bien intangible)	La Organización de mantenimiento, que ha estado suministrando el servicio hasta ahora, cesa definitivamente la prestación de éste.
Salidas	Documentación y productos generados durante el proceso de externalización entregados al Cliente	Equipo de mantenimiento del Cliente formado	
Responsable	Equipo de mantenimiento Cliente	Equipo de mantenimiento Cliente	Organización de mantenimiento
Interfaces con otros procesos		Formación	

4.3. TÉCNICAS Y MÉTRICAS PROPUESTAS PARA LA EJECUCIÓN DE LAS TAREAS.

En esta sección se describe la serie de técnicas y métricas que se enumera a continuación, y cuya utilización se recomienda como apoyo al proceso de mantenimiento descrito en esta tesis.

1. Una técnica (epígrafe 4.3.1) para identificar, antes de acometerlos, las características de los proyectos de mantenimiento que más riesgos producen (momento en el que, además, se cuenta con muy poca información cuantitativa acerca de tales proyectos).
2. Un modelo económico del proyecto de mantenimiento (epígrafe 4.3.2), que permite calcular la cantidad de recursos que deben dedicarse en un periodo a servir peticiones de mantenimiento no planificable. Su validación es matemática y se expone a medida que se va explicando la técnica.
3. Una serie de métricas de mantenibilidad de esquemas conceptuales y lógicos de bases de datos (epígrafe 4.4), cuyas validaciones formal y empírica se desarrollan a partir de la página 153.
4. Un conjunto de métricas de tamaño y complejidad para sistemas orientados a objeto. Como indicamos en su exposición (epígrafe 4.4.2), estas métricas se utilizan habitualmente para estimar el esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto, por lo que caerán tanto en la categoría *técnica* como *metrológica*.
5. Un conjunto de métricas para controlar el proceso de mantenimiento (epígrafe 4.4.3).

4.3.1. IDENTIFICACIÓN Y ESTIMACIÓN DE RIESGOS

Utilizada en:	Actividades y tareas iniciales	Actividad I.0. *Estudio inicial*	Tarea I0.2. Preparar propuesta de mantenimiento

Como se mencionó en la sección 3.1.4.4, la asunción de un contrato de externalización encierra una serie de riesgos no sólo para el Cliente, sino también para la Organización de mantenimiento.

A continuación se presenta una técnica para identificar y estimar los riesgos asociados a un proyecto de mantenimiento antes de acometerlo. En tales momentos del proceso (que, en el modelo del proceso descrito en esta tesis corresponde a la tarea I0.2, descrita en la página 101), la información cuantitativa del entorno en el que se ejecutará el mantenimiento es realmente muy poca, limitándose a la recogida en el Cuestionario inicial (DOC1, página 219). Mediante la técnica que describimos, se recoge información subjetiva del proyecto que se va a mantener, la cual puede ser usada para identificar las áreas de mayor riesgo del proyecto.

En esta sección se presenta el cuestionario utilizado para recoger información acerca de los factores circunstanciales, el tratamiento estadístico a que fue sometido para reducir su nú-

mero de preguntas y algunas discusiones sobre estos resultados. Como línea futura de trabajo, relacionada especialmente con el proyecto Mantis (véase sección 1.5, página 26), se encuentra la implementación completa de una técnica de estimación a partir de los datos recogidos en el cuestionario.

4.3.1.1. Cuestionario inicial.

El cuestionario que presentamos a continuación (Tabla 15) utiliza, para identificar los riesgos del proyecto, una lista de 36 factores circunstanciales que ha sido construida a partir de algunos de los enumerados en Eurométodo (Euromethod, 1996), más otros que han sido incorporados a partir de la experiencia en la aplicación de MANTEMA en diversos proyectos de mantenimiento de Atos ODS.

Eurométodo es una metodología de adquisición de productos, servicios y sistemas software que define un factor circunstancial como "una propiedad de la situación que genera riesgo y que debe tenerse en cuenta en el diseño de la estrategia de adquisición" (Euromethod, 1996). Un factor circunstancial, por tanto, puede tener influencia sobre uno o más factores de riesgo. Del mismo modo, y coincidiendo con la idea de Neitzel (1999), cada factor de riesgo puede afectar a una o más áreas de negocio de la organización propietaria del software, que puede a su vez afectar a la Organización de mantenimiento.

La Tabla 15 debe rellenarse para cada aplicación objeto de mantenimiento, deseablemente por varios técnicos de la Organización de mantenimiento. Su primera columna está ocupada por los grupos de factores circunstanciales que se enumeran y explican en la tercera. Cada celda de la cuarta columna de la Tabla 15 debe rellenarse con los valores 1 a 5, según el personal que la rellena se sienta *totalmente de acuerdo*, *de acuerdo*, *indiferente*, *en desacuerdo* o *totalmente en desacuerdo* con la afirmación manifestada en la segunda columna, que está redactada en tono positivo.

A modo de ejemplo, en la Figura 1 reproducimos un fragmento cumplimentado de la Tabla 15, en la que la persona que lo ha rellenado se encuentra indiferente respecto de la afirmación "la calidad de la documentación del software es alta", en desacuerdo respecto de "la calidad del diseño de la arquitectura del sistema es alta", etc.

Factor	Nº	Subfactor	Valor
Especificaciones existentes de la aplicación	F1	La calidad de la documentación del software es alta	3
	F2	La calidad del diseño de la arquitectura del sistema es alta	4
	F3	La calidad del diseño del software es alta	4
	F4	La documentación está actualizada	5
	F5	Los requisitos del SI están disponibles y son claros	3
	F6	Los requisitos del SI son estables	3

Figura 23. Ejemplo de cumplimentación de la Tabla 15.

La tabla completa toma la siguiente forma:

Factor	Nº	Subfactor	Valor
Especificaciones existentes de la aplicación	F1	La calidad de la documentación del software es alta	
	F2	La calidad del diseño de la arquitectura del sistema es alta	
	F3	La calidad del diseño del software es alta	
	F4	La documentación está actualizada	
	F5	Los requisitos del SI están disponibles y son claros	
	F6	Los requisitos del SI son estables	
Equipo actual de desarrollo o mantenimiento del Cliente	F7	...tiene un buen conocimiento del SI	
	F8	...tiene una alta capacidad de trabajo	
	F9	... tiene un alto nivel de conocimiento del dominio del sistema	
	F10	...tiene un alto nivel de conocimiento del entorno técnico del sistema	
	F11	...tiene un muy buen conocimiento funcional del sistema	
Factores externos	F12	La organización no depende mucho de cambios externos	
	F13	La organización no depende mucho de subcontratistas	
Factores del SI	F14	Los procesos de negocio son simples	
	F15	Los procesos de negocio son estables	
	F16	La información es estable	
	F17	El nivel de normalización del sistema es alto	
	F18	Excluyendo tolerancia a fallos, el número de replicaciones del sistema es bajo	
Factores críticos de la aplicación	F19	La aplicación no es crítica para el negocio	
	F20	Hay pocas interfaces con otras aplicaciones del SI	
	F21	Los errores no influyen en la impresión pública de la organización	
	F22	No se producen momentos críticos aleatoriamente	
	F23	No hay momentos periódicos que requieran la incorporación de personal o mayor dedicación de los recursos (a final de mes, p. ej.)	
	F24	El software no es vulnerable a las acciones del usuario (por ejemplo, actualizaciones directas sobre la base de datos)	
	F25	No se depende del usuario para conocer el problema	
Calidad de la aplicación	F26	Se han utilizado estándares de denominación y de programación	
	F27	La calidad de los programas *batch* es alta (depende del número de programas, lenguajes utilizados y otros factores de calidad, como nº de líneas de código, nº de módulos, edad de los programas, nº de intervenciones sufridas, acoplamiento de los módulos, etc.)	
	F28	La calidad de los programas *on-line* es alta (depende del número de programas, lenguajes utilizados y otros factores de calidad, como nº de líneas de código, nº de módulos, edad de los programas, nº de intervenciones sufridas, acoplamiento de los módulos, etc.)	
	F29	Los informes generados por la aplicación están poco influidos por las intervenciones de mantenimiento	
	F30	Las pantallas de la aplicación están poco influidas por las intervenciones de mantenimiento	
Factores de la organización	F31	El entorno es estable	
	F32	Hay poca dependencia de cambios en la organización	
	F33	Hay poca tasa de cambios en los procesos de negocio	
Metodologías	F34	La tecnología utilizada en el proyecto está disponible	
	F35	Se utilizaron metodologías durante el desarrollo del sistema	
	F36	Se utilizaron herramientas durante el desarrollo del sistema	

Tabla 15. Factores circunstanciales considerados.

4.3.1.2. Recolección de datos.

Solicitamos la colaboración de los jefes de varios proyectos de varias organizaciones para cumplimentar el cuestionario; sin embargo, obtuvimos respuestas afirmativas sólo de ocho, siete pertenecientes a Atos ODS, S.A. y uno a la Excma. Diputación Provincial de Ciudad Real. Algunas de las razones por las que los restantes no lo contestaron son: “Pregunta información que no conocemos” y “Pregunta información que no se puede dar”.

En la Tabla 2 recogemos las puntuaciones dadas a los factores circunstanciales de las ocho aplicaciones. Las siete correspondientes a Atos ODS están implementadas en Cobol-CICS-DB2. mientras que la perteneciente a la Excma. Diputación Provincial de Ciudad Real está hecha en 4GL. Todas ellas ocupan entre 125.000 y 275.000 líneas de código.

	Atos ODS							EDP CR
f.c.	P1	P2	P3	P4	P5	P6	P7	Cont. púb.
1	3	2	2	4	2	2	2	3
2	3	3	3	2	3	3	3	4
3	3	4	2	2	3	3	4	3
4	3	4	2	4	2	2	2	2
5	3	1	2	4	4	4	4	3
6	3	4	2	4	2	2	3	3
7	4	5	4	4	3	3	4	5
8	4	5	5	3	3	3	3	4
9	4	5	4	2	3	3	4	4
10	4	5	5	2	3	3	4	5
11	4	5	4	2	2	2	2	5
12	3	3	4	3	2	3	2	1
13	3	3	4	4	1	1	1	1
14	4	2	4	4	2	2	3	1
15	2	4	4	4	2	2	3	5
16	4	3	4	2	2	2	3	2
17	3	3	3	2	2	2	3	2
18	4	3	5	3	2	2	2	5

	Atos ODS							EDP CR
f.c.	P1	P2	P3	P4	P5	P6	P7	Cont. púb.
19	2	1	4	2	5	3	5	5
20	1	1	5	5	3	3	3	3
21	1	1	5	4	5	5	5	1
22	1	1	2	3	4	4	4	3
23	4	3	5	3	2	2	2	3
24	1	4	5	1	2	2	2	1
25	4	3	3	3	2	2	2	2
26	4	3	2	4	3	3	3	3
27	4	4	2	3	3	3	3	3
28	4	3	3	3	2	3	3	3
29	3	3	4	4	3	2	3	2
30	4	3	4	3	3	3	4	3
31	4	2	3	4	2	2	2	3
32	4	2	4	4	3	3	3	3
33	3	3	4	4	3	3	3	3
34	3	3	5	3	3	3	3	5
35	3	1	1	3	3	3	4	4
36	3	1	1	2	4	4	4	5

Tabla 16. Puntuación dada a los diferentes factores circunstanciales.

4.3.1.3. Tratamiento estadístico.

Hemos aplicado un Análisis de Componentes Principales a los datos recogidos en la Tabla 16. El Análisis de Componentes Principales es una técnica de reducción de datos que detecta grupos de variables independientes que representan la misma característica de la población. El objetivo de este análisis es determinar estos componentes, omitiendo los demás, en cuanto se puede prescindir de ellos sin pérdida de mucha información (Sierra, 1994). Los componentes principales detectados representan aspectos ortogonales de la población estudiada, y pueden expresarse como combinaciones lineales de las variables independientes. El primer componente principal explica la mayor varianza en el conjunto de datos, el segundo la siguiente mayor varianza, etc.

La técnica de componentes principales se ha utilizado en multitud de estudios de las más variadas disciplinas, incluyendo la Ingeniería del Software (Briand et al., 1998; Fioravanti et al., 1999).

En nuestro caso, aplicamos componentes principales sin usar matriz de rotación y deteniendo la entrada de variables en cada componente principal cuando los autovalores son mayores que 1. Obtenemos los seis CP que se muestran en la Tabla 17, que representan algo más del 98% de la varianza total.

	Autovalores iniciales		
Componente	Total	% de la varianza	% acumulado
1	11.83	32.86	32.86
2	8.17	22.69	55.50
3	6.24	17.33	72.88
4	4.45	12.38	85.25
5	3.38	9.39	94.63
6	1.46	4.05	98.68

Tabla 17. Componentes principales obtenidos.

De la Tabla 17 se desprende que las preguntas del cuestionario de factores circunstanciales (Tabla 15) puede agruparse en seis componentes. En la Tabla 4 se muestra el peso que cada una de las 36 variables independientes ejerce sobre cada CP (esto es, el coeficiente que tomará la variable en la combinación lineal que permite conocer el valor del CP). En esta tabla hemos señalado únicamente los pesos mayores que 0,7 o menores que -0.7.

	Componente					
	1	2	3	4	5	6
F22	-0,920					
F23	0,879					
F8	0,854					
F13	0,842					
F25	0,822					
F36	-0,810					
F5	-0,805					
F16	0,771					
F35	-0,712					
F17						
F11						
F12						
F7						
F28						
F2		-0,829				
F9		-0,826				
F32		0,766				
F10		-0,765				

	Componente					
	1	2	3	4	5	6
F3		-0,738				
F33		0,731				
F14		0,730				
F20						
F29						
F31						
F26			-0,898			
F27			-0,889			
F6			-0,710			
F4						
F24						
F19						
F21						
F18						
F34						
F1						
F15						
F30					0,754	

Tabla 18. Incidencia de las variables en los CP.

4.3.1.4. Análisis de resultados.

Dependiendo del valor absoluto del umbral de corte que escojamos (en nuestro caso, 0.7), deberemos conservar algunas preguntas para la versión refinada del cuestionario y eliminar otras. En nuestro caso, las preguntas que conservaremos son aquéllas con un peso significativo en la Tabla 18. El nuevo cuestionario se reproduce en la Tabla 19, en el cual el número inicial de preguntas (36) se ha reducido a las 21 preguntas esenciales.

Factor	Subfactor	Valor
Especificaciones existentes de la aplicación	La calidad del diseño de la arquitectura del sistema es alta	
	La calidad del diseño del software es alta	
	Los requisitos del SI están disponibles y son claros	
	Los requisitos del SI son estables	
Equipo actual de desarrollo o mantenimiento del Cliente	...tiene una alta capacidad de trabajo	
	... tiene un alto nivel de conocimiento del dominio del sistema	
	...tiene un alto nivel de conocimiento del entorno técnico del sistema	
Factores externos	La organización no depende mucho de subcontratistas	
Factores del SI	Los procesos de negocio son simples	
	La información es estable	
Factores críticos de la aplicación	No se producen momentos críticos aleatoriamente	
	No hay momentos periódicos que requieran la incorporación de personal o mayor dedicación de los recursos (a final de mes, p. ej.)	
	No se depende del usuario para conocer el problema	
Calidad de la aplicación	Se han utilizado estándares de denominación y de programación	
	La calidad de los programas batch es alta (depende del número de programas, lenguajes utilizados y otros factores de calidad, como nº de líneas de código, nº de módulos, edad de los programas, nº de intervenciones sufridas, acoplamiento de los módulos, etc.)	
	Las pantallas de la aplicación están poco influidas por las intervenciones de mantenimiento	
Factores de la organización	Hay poca dependencia de cambios en la organización	
	Hay poca tasa de cambios en los procesos de negocio	
Metodologías	La tecnología utilizada en el proyecto está disponible	
	Se utilizaron metodologías durante el desarrollo del sistema	
	Se utilizaron herramientas durante el desarrollo del sistema	

Tabla 19. Versión revisada del cuestionario.

4.3.1.4.1. Interpretación de los componentes principales.

Como ya indicamos, cada CP es ortogonal a los demás, por lo que podríamos concluir que el cuestionario inicial mide seis "dimensiones" diferentes de los proyectos de mantenimiento, correspondientes a los seis CP. A continuación realizamos sencillas interpretaciones de cada CP; nótese que, a este solo efecto, los factores con pesos negativos los hemos subrayado y reescrito en tono negativo para facilitar la interpretación.

CP 1. Este CP se ve influido principalmente por los factores circunstanciales F22 (no existencia de momentos críticos aleatorios), F23 (existencia de momentos críticos periódicos), F8 (Alta capacidad de trabajo del equipo actual), F13 (dependencia de subcontratistas), F25 (dependencia del usuario para conocer el problema), F36 (no se usaron herramientas durante el desarrollo del sistema). F5 (no disponibilidad y falta de claridad de requisitos), F16 (estabilidad de requisitos) y F35 (no se usaron metodologías durante la construcción del sistema).

Entre otras cosas, este CP estaría midiendo la calidad con que se ejecutaron las primeras etapas del desarrollo de la aplicación (F35, F36 y F5). Por otro lado, F5 y F25 muestran una correlación de -0.703 (+0.703 si consideramos F5), lo que denota que la no disponibilidad de requisitos implica una mayor dependencia del usuario.

CP 2. En éste influyen los factores F2 (baja calidad de la arquitectura), F9 (bajo nivel de conocimiento del dominio del sistema), F32 (se depende de cambios en la organización), F10 (bajo nivel de conocimiento del entorno técnico del sistema), F3 (baja calidad del diseño del software), F33 (poca tasa de cambios en procesos de negocio) y F14 (simplicidad de los procesos de negocio).

En este CP tiene especial relevancia la cualificación del equipo actual de mantenimiento de la organización, así como la calidad del diseño del sistema (arquitectura y software). El sistema fue diseñado al comienzo de la vida del sistema, por lo que volvemos a observar la clara influencia de las etapas iniciales del ciclo de vida.

CP 3. Influyen en este CP los factores F26 (no se usaron estándares de denominación y programación), F27 (baja calidad de programas *batch*) y F6 (inestabilidad de requisitos).

De nuevo notamos influencias de las etapas anteriores al mantenimiento, mediante los factores 26 y 27.

CP 5. En este CP consideramos únicamente el hecho de que las pantallas estén influidas por las intervenciones (F30).

4.3.1.5. Discusión.

Esperamos que la reducción del número de preguntas en el cuestionario facilite su aceptación por parte de los gestores y jefes de proyecto. En este caso, podremos continuar con la implementación de esta técnica mediante la recolección de mayor cantidad de datos (entre los que se incluirán, además de los relativos a factores circunstanciales, número de peticiones de modificación recibidas, fechas de presentación y tipo de cada de una de ellas, tiempo empleado en su resolución, etc.).

A falta aún de confirmar la influencia del cuestionario en el esfuerzo de mantenimiento, de las discusiones realizadas tras analizar los CP podemos concluir que el nivel de calidad con que hayan sido desarrolladas las etapas iniciales del ciclo de vida de los sistemas son las que más van a influir en el proceso de mantenimiento (componentes 1, 2 y 3). Esto vendría a confirmar resultados como los mostrados por Boehm (1981) o Schach (1992).

4.3.2. ESTIMACIÓN DE RECURSOS PARA EL MANTENIMIENTO NO PLANIFICABLE

La técnica que describimos en este epígrafe utiliza los datos predichos por alguno de los métodos de estimación de esfuerzo de mantenimiento aplicado a correctivo (sección 3.5, página 79) y los parámetros económicos del entorno del proyecto, para determinar la tasa de recur-

sos que es necesario planificar durante un periodo para la corrección de errores sin incurrir en pérdidas económicas. Esta técnica es aplicable en la tarea I1.3 ("Desarrollar planes", descrita en la página 102), aunque puede ser utilizada con cierta regularidad (trimestral o semestralmente, por ejemplo) para estimar los recursos necesarios para no planificable en un determinado periodo.

Al finalizar 1998, se calculaba que existía en Europa un exceso de demanda de informáticos de 500.000 personas con respecto a la oferta, y se estima que este número crecerá hasta 2.400.000 personas para el año 2002 (Bourke, 1999). Por ello, esta técnica puede resultar de gran interés a la hora de asignar los recursos humanos a diferentes proyectos de mantenimiento.

4.3.2.1. Modelo económico

Cuando existe externalización, es habitual que la Organización de mantenimiento sufra penalizaciones económicas si sirve las peticiones de modificación correctivas fuera del tiempo al que se hubiera comprometido (véanse los "Indicadores de servicio" en el epígrafe 4.4.3.1, página 174). Sin embargo, la existencia de tales penalizaciones no exime al Cliente del pago de los recursos a la Organización de mantenimiento. El Cliente también sufre pérdidas económicas por cada hora de parada del sistema, y la Organización de mantenimiento paga por los recursos dedicados al Cliente en función del tiempo que realmente le dedican.

Centrados en el mantenimiento correctivo urgente, identifiquemos cada uno de estos factores:

- m_C es el precio (en euros, por ejemplo) que el Cliente paga a la Organización de mantenimiento por cada hora de recurso contratada.
- m_{MO} es el precio en euros que la Organización de mantenimiento paga a sus recursos. Si no hubiera más parámetros económicos en el proyecto, es evidente que, para que la Organización de mantenimiento tuviera beneficios, m_{MO} debería ser mayor que m_C.
- s es el valor de la penalización que la Organización de mantenimiento debe abonar al Cliente por cada día de retraso en la corrección de errores.

En la Tabla 20 representamos los parámetros de un periodo de n días de duración de un proyecto genérico de mantenimiento. Pretendemos calcular el número p de horas que la Organización de mantenimiento puede dedicar sin incurrir en pérdidas en cada uno de los n días, siendo h_i el número de horas estimado que se necesitarán el día *i-ésimo*. Evidentemente, si $p \geq h_i \ \forall i$, entonces no habrá retraso ninguno de los días. La situación que vamos a modelar, sin embargo, supone que vamos a dedicar un cantidad de recursos menor o igual a los necesarios.

El número de días (o jornadas de trabajo de p horas) requerido para servir las h_i horas de trabajo que se ha estimado llegarán en el día *i-ésimo* se especifican en la cuarta columna, ocupando la quinta el día previsto de finalización de los trabajos recibidos en el día, para cuyo cál-

culo debe considerarse el retraso acumulado, que aparece en la sexta columna. Cada retraso sufrirá la penalización que indicamos en la última columna.

Así, por ejemplo, si h_1=*8 horas* y p=*4 horas*, los errores del primer día requerirán dos jornadas de trabajo de corrección, lo que implicar que comiencen con retraso las peticiones de modificación recibidas el segundo día, que provocarán sucesivos retrasos en las de los días siguientes.

Día	Horas planif.	Horas necesarias	Días necesarios	Día de finalización	Retraso	Penalización
1	p	h_1	$\frac{h_1}{p}$	$\frac{h_1}{p}$	$\frac{h_1}{p}-1=\frac{h_1-p}{p}$	$s\cdot\left(\frac{h_1}{p}-1\right)=s\cdot\frac{h_1-p}{p}$
2	p	h_2	$\frac{h_2}{p}$	$\frac{h_1}{p}+\frac{h_2}{p}$	$\frac{h_1+h_2-2p}{p}$	$s\cdot\left(\frac{h_1+h_2}{p}-2\right)=s\cdot\frac{h_1+h_2-2p}{p}$
3	p	h_3	$\frac{h_3}{p}$	$\frac{h_1+h_2+h_3}{p}$	$\frac{h_1+h_2+h_3-3p}{p}$	$s\cdot\frac{h_1+h_2+h_3-3p}{p}$
.....						
n	p	h_n	$\frac{h_n}{p}$	$\sum_{i=1}^{n}\frac{h_i}{p}$	$\frac{\sum_{i=1}^{n}h_i-np}{p}$	$s\cdot\frac{\sum_{i=1}^{n}h_i-np}{p}$
	pn	$\sum_{i=1}^{n}h_i$			$\sum_{j=1}^{n}\frac{\sum_{i=1}^{j}(h_i-ip)}{p}$	$s\cdot\sum_{j=1}^{n}\frac{\sum_{i=1}^{j}(h_i-ip)}{p}$

Tabla 20. Modelo de un proyecto genérico de mantenimiento.

Por tanto, la Organización de mantenimiento incurrirá en penalizaciones en un periodo si, durante éste, dedica menos recursos de los necesarios; sin embargo, estas penalizaciones sólo supondrán pérdidas cuando su coste más el de los recursos realmente dedicados sea mayor que los honorarios pagados por el Cliente. Éste, por el contrario, percibirá como una fuente de ingresos las cantidades percibidas en concepto de penalizaciones, y le compensará que la Organización de mantenimiento se retrase si los honorarios de ésta más las cantidades perdidas por los errores son menores que el importe de las penalizaciones.

De este modo, el problema puede estudiarse desde estos dos puntos de vista e, incluso, desde uno tercero: el de una organización que mantiene su propio software. A continuación analizamos los tres puntos de vista y los ilustramos con un ejemplo.

4.3.2.1.1. Estudio desde el punto de vista de la Organización de mantenimiento

En el proyecto modelado en la tabla anterior, la Organización de mantenimiento pagará $m_{MO}\,p$ euros diarios a sus recursos, mientras que el Cliente abonará $m_C\,h_i$ a la Organización de mantenimiento. En el periodo considerado de n días, estas cantidades serán, respectivamente,

$C(p) = m_{MO} np$ e $I(p) = \sum_{i=1}^{n} m_C h_i$ (donde con *C* denotamos la función de *Coste* y con *I* la de *Ingresos*, siempre desde el punto de vista de la Organización de mantenimiento). Para simplificar el problema, supongamos que son *h* las horas estimadas necesarias para cada uno de los días del periodo. Con esta consideración, $I(p) = m_C nh$, y los *n* primeros términos de la última columna podemos reescribirlos de esta manera:

$$s \cdot \frac{h-p}{p}; s \cdot \frac{2 \cdot (h-p)}{p}; s \cdot \frac{3 \cdot (h-p)}{p}; ...; s \cdot \frac{n \cdot (h-p)}{p}$$

La última celda de la Tabla 20 especifica la sanción total. Considerando que $h_i = h \ \forall \ i$, podemos operar de la siguiente manera:

$$S(p) = s \cdot \frac{h-p}{p} + s \cdot \frac{2 \cdot (h-p)}{p} + s \cdot \frac{3 \cdot (h-p)}{p} + ... + s \cdot \frac{n \cdot (h-p)}{p} =$$
$$= s \cdot \frac{h-p}{p} \cdot (1+2+3+...+n) = s \cdot \frac{h-p}{p} \cdot \sum_{i=1}^{n} i = s \cdot \frac{h-p}{p} \cdot \frac{n \cdot (n+1)}{2} = \frac{n \cdot (n+1) \cdot s \cdot (h-p)}{2p}$$

Ahora tenemos toda la información necesaria para calcular *p*: la Organización de mantenimiento podrá utilizar sin pérdida económica *p* horas diarias ($p \leq h$) durante el periodo considerado a partir del valor que iguala *I(p)* con *C(p)+S(p)*. Resolvamos, por tanto, la siguiente ecuación:

$$I(p) = C(p) + S(p)$$
$$nm_C h = nm_{MO} p + \frac{n \cdot (n+1) \cdot s \cdot (h-p)}{2p}$$

$$2pnm_C h = 2pnm_{MO} p + n \cdot (n+1) \cdot s \cdot (h-p)$$

$$2pnm_C h = 2pnm_{MO} p + n \cdot (n+1) \cdot s \cdot h - n \cdot (n+1) \cdot sp$$

$$2p^2 nm_{MO} - 2pnm_C h + n \cdot (n+1) \cdot s \cdot h - n \cdot (n+1) \cdot sp = 0$$

$$2p^2 m_{MO} - 2pm_C h + \cdot (n+1) \cdot s \cdot h - \cdot (n+1) \cdot sp = 0$$

$$p^2 2m_{MO} - p[2m_C h + (n+1)s] + \cdot (n+1) \cdot s \cdot h = 0 \qquad \textbf{(1)}$$

Si la Organización de mantenimiento resuelve la ecuación (1) con los parámetros de su proyecto, conocerá la cantidad de recursos que debe dedicar para no incurrir en pérdidas.

4.3.2.1.2. Estudio desde el punto de vista del Cliente

En este caso, el Cliente entiende el pago de penalizaciones (función *S*) por parte de la Organización de mantenimiento como un beneficio, y el pago a ésta (función *I*) como un coste. El Cliente incurre, además, en un coste adicional, que denotamos mediante la función *L*, y que depende del número de horas que tarde en corregirse el error.

$L(p) = d \cdot n \cdot h + d \cdot \frac{n \cdot (n+1) \cdot (h-p)}{2 \cdot p \cdot 24}$, donde d es el coste en euros por hora sufrido por la ocurrencia del error. L depende tanto del tiempo de retraso como del tiempo mínimo de corrección. Hemos dividido por 24 ya que el retraso se mide en días y d en horas.

El Cliente no perderá dinero a partir del punto en que las penalizaciones pagadas por la Organización de mantenimiento igualan el pago a ésta más la pérdida variable (función L):

$$L(p) + I(p) = S(p)$$

$$d \cdot n \cdot h + d \cdot \frac{n \cdot (n+1) \cdot (h-p)}{48p} + m_C \cdot n \cdot h = \frac{n \cdot (n+1) \cdot s \cdot (h-p)}{2p}$$

$$48p \cdot d \cdot h + d \cdot (n+1) \cdot h - d \cdot (n+1) \cdot p + 48 \cdot p \cdot m_C \cdot h = 24 \cdot (n+1) \cdot s \cdot h - 24 \cdot (n+1) \cdot s \cdot p$$

$$p \cdot [48h \cdot (d + m_C) + (n+1) \cdot (24 \cdot s - d)] = (n+1) \cdot (24 \cdot s - d) \cdot h$$

$$p = \frac{h \cdot (n+1) \cdot (24 \cdot s - d)}{48 \cdot h \cdot (d + m_C) + (n+1) \cdot (24 \cdot s - d)} \quad \textbf{(2)}$$

4.3.2.1.3. Estudio desde el punto de vista de una organización que mantiene su propio software.

En este caso, la única empresa que consideramos comparte los roles de Cliente y Organización de mantenimiento. No sufrirá penalizaciones, pero sí pérdidas por los errores del sistema. El coste diario de los recursos se calcula con la expresión $m_{MO}\,p$, ya que la empresa paga por las horas realmente dedicadas. A esta empresa le interesará dedicar menos horas de recurso a partir del punto en que el coste de los recursos más las pérdidas por los errores iguale al coste de dedicar los recursos necesarios para la corrección de errores. Teniendo en cuenta que, debido a la inexistencia de intermediarios, el precio de los recursos es único (es decir, $m_{MO}=m_C$), tenemos:

$$I(p) = L(p) + C(p)$$

$$n \cdot m \cdot h = d \cdot n \cdot h + d \cdot \frac{n \cdot (n+1) \cdot (h-p)}{48p} + n \cdot m \cdot p$$

Si operamos, obtenemos la siguiente ecuación de la que podemos despejar p:

$$48 \cdot m \cdot p^2 + p[48 \cdot h \cdot (d - m) - d \cdot (n+1)] - d \cdot (n+1) \cdot h = 0 \quad \textbf{(3)}$$

4.3.2.1.4. Ejemplo

Supongamos un proyecto de externalización de mantenimiento, y que la Organización de mantenimiento desea conocer cuántos recursos dedicarle durante los próximos 30 días. Se han

estimado 8 horas necesarias al día. La Organización de mantenimiento paga 50 euros/hora a los recursos, y recibe 100 euros/hora por este concepto del Cliente. Existe una penalización de 400 euros por día de retraso. Ambas organizaciones desean conocer los valores de p a partir de los cuales no incurrirían pérdidas.

Si traducimos los parámetros del problema a los términos descritos en los epígrafes anteriores:

n=30 días

m_C=100 euros/hora

m_{MO}=50 euros/hora

s=400 euros/día

h=8 horas

Calculemos primero el valor de p para la Organización de mantenimiento, aplicando la Ecuación (1):

$$p^2 \cdot 2 \cdot 50 - p[2 \cdot 100 \cdot 8 + (30+1) \cdot 400] + (30+1) \cdot 400 \cdot 8 = 0$$

$$100 \cdot p^2 - 14000 \cdot p + 99200 = 0$$

$$p = \frac{14000 \pm \sqrt{14000^2 - 4 \cdot 100 \cdot 99200}}{2 \cdot 100} = \frac{14000 \pm 12502}{200}$$

$$p_1 = 132 \text{ horas}; \; p_2 = 7.49 \text{ horas}$$

Obviamente, utilizar más de 8 horas diarias no tiene sentido para la Organización de mantenimiento, ya que dedicaría más recursos de los necesarios y se produciría una disminución automática de los beneficios. Por tanto, el valor p_1 obtenido no resulta de utilidad. En caso de escasez de recursos, la Organización de mantenimiento podría llegar a dedicar un mínimo de entorno a 7.49 horas diarias sin incurrir en pérdidas económicas. Un valor inferior haría entrar en pérdidas automáticamente, mientras que en este caso los beneficios alcanzan su máximo con 8 horas diarias de dedicación.

El valor de umbral para el Cliente se calcula aplicando la Ecuación (2). Suponiendo que la pérdida por cada hora de caída del sistema (d) es de 120 euros:

$$p = \frac{8 \cdot (30+1) \cdot (24 \cdot 400 - 120)}{48 \cdot 8 \cdot (120+100) + (30+1) \cdot (24 \cdot 400 - 120)} = \frac{2351040}{378360} = 6.21 \text{ horas}$$

Podemos calcular las funciones de beneficio en ambas organizaciones en función del valor de recursos dedicado (p). En el caso de la organización de mantenimiento, su beneficio (B_{OM})

será la diferencia entre lo recibido del Cliente (función *I*) y lo pagado en concepto de los recursos realmente dedicados (función *C*) más las penalizaciones (función *S*). Operando:

$$B_{OM}(p) = I(p) - C(p) - S(p) = 24000 - 1500\,p - 186000 \cdot \frac{8-p}{p}$$

Para el Cliente, las sanciones pagadas por la Organización de Mantenimiento son ingresos, por lo que, para calcular su beneficio, habrá que restar a esta cantidad los costes por caída del sistema (función *L*) más los costes del servicio contratado (función *I*):

$$B_C(p) = S(p) - I(p) - L(p) = 183675 \cdot \frac{8-p}{p} - 52800$$

En la Tabla 21 mostramos las diferentes funciones según el número de horas realmente dedicadas; obsérvese la forma en que varían las funciones de beneficio de ambas organizaciones:

p	I(p)	C(p)	S(p)	L(p)	Bmo(p)	Bc(p)
1	24000	1500	1302000	45075	-1279500	1232925
2	24000	3000	558000	35775	-537000	498225
3	24000	4500	310000	32675	-290500	253325
4	24000	6000	186000	31125	-168000	130875
5	24000	7500	111600	30195	-95100	57405
6	24000	9000	62000	29575	-47000	8425
7	24000	10500	26571	29132	-13071	-26560
8	24000	12000	0	28800	12000	-52800
9	24000	13500	-20666	28541	31166	-73208
10	24000	15000	-37200	28335	46200	-89535

Tabla 21. Valores de las funciones según el parámetro p.

El valor *p=7.49 horas* significa que, desde el punto de vista del Cliente, éste obtendría beneficios gracias a la elevada cantidad de penalizaciones impuestas si la Organización de mantenimiento dedicara menos de 7.49 horas diarias de recurso. La Figura 24 muestra gráficamente la variación de las funciones de beneficio de ambas organizaciones en función del valor real de recursos dedicados.

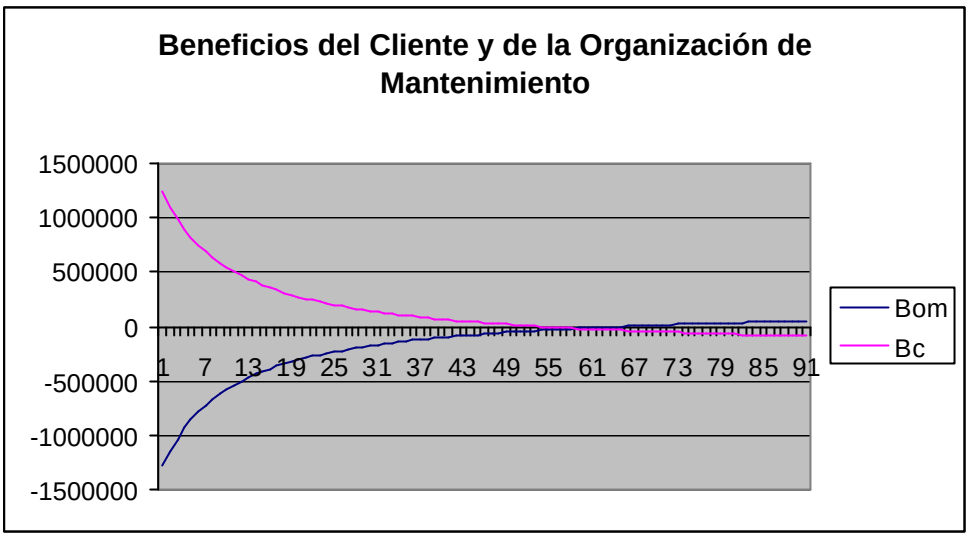


Figura 24. Beneficios de ambas organizaciones según el valor de p.

4.3.2.1.5. Discusión y ampliaciones al modelo

A pesar de que el ejemplo ha mostrado que el Cliente obtendría beneficios si la Organización de mantenimiento dedicara muy pocos recursos, la realidad es que los retrasos muy acusados pueden implicar pérdidas de producción demasiado grandes, con lo que podrían aparecer otros tipos de costes no considerados. Un efecto parecido surge desde el punto de vista de la Organización de mantenimiento, ya que una baja dedicación puede conllevar la aparición de otro tipo de riesgos, como la rescisión o renegociación del contrato de mantenimiento. Este hecho puede considerarse en el modelo haciendo asumir, como un coste adicional, un porcentaje de las pérdidas soportadas por el Cliente, bien de forma lineal o exponencial, como por ejemplo:

$$C(p) = n \cdot m_{MO} \cdot p + k \cdot (d \cdot n \cdot h)^{1 \pm e}, \quad 0 \leq \mathrm{k} < 1, \; e \in \Re$$

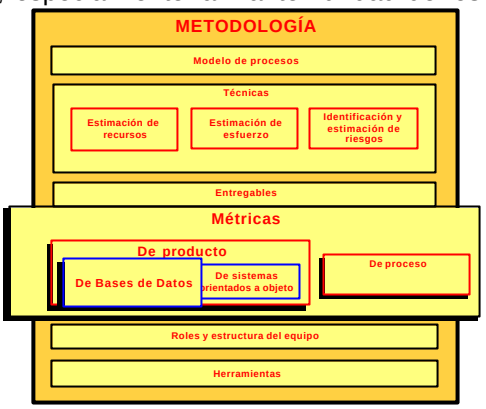
4.4. MÉTRICAS DE PRODUCTO

En esta sección presentamos métricas para bases de datos (esquemas conceptuales y esquemas lógicos) y para sistemas orientados a objeto.

4.4.1. MÉTRICAS PARA BASES DE DATOS

Se presentan aquí un conjunto de métricas para la medición de atributos de calidad de esquemas conceptuales y lógicos de bases de datos. La medida de la calidad de los esquemas conceptuales es interesante tanto en los nuevos desarrollos de bases de datos (ya que permiten decidir entre dos esquemas alternativos), como para estimar el esfuerzo de su modificación.

Las métricas propuestas miden los factores de calidad propuestos por Batini et al. (1992), junto a otros factores que nosotros proponemos, todos ellos enfocados hacia la mantenibilidad de bases de datos. Como es bien sabido la calidad y, especialmente la mantenibilidad de los programas depende en gran medida del diseño de la base de datos que manejan, por lo que resulta muy útil disponer de métodos para estimar la calidad de éstas.



Presentamos en primer lugar las métricas para esquemas conceptuales, y pasamos después a presentar las aplicables a esquemas lógicos. Algunos de estos dos tipos de métricas se han caracterizado formalmente y validado empíricamente.

4.4.1.1. Métricas para el esquema conceptual

El objetivo del diseño conceptual es la obtención de una correcta representación de la organización de la información. Tal representación se realiza mediante un esquema conceptual, usando habitualmente el modelo entidad-interrelación (De Miguel et al., 1999).

4.4.1.1.1. Compleción

Según Batini et al. (1992), "un esquema interrelacional es *completo* cuando refleja fielmente todos los requisitos del universo del discurso". Definimos para la compleción una métrica nominal que puede ser *alta, media, baja* o *muy baja*.

4.4.1.1.2. Corrección

Según Batini et al. (1992), "un esquema conceptual es correcto cuando se aplica correctamente el modelo entidad-interrelación". Esta métrica puede ser también *alta, media, baja* o *muy baja*.

4.4.1.1.3. Minimalidad

La minimalidad la definimos como el número de atributos "sobrantes" (repetidos o calculables mediante operaciones matemáticas o inferencia lógica) presentes en el esquema. Para situar el valor de la expresividad entre 0 y 1, puede utilizarse el siguiente ratio:

$$MR = 1 - \frac{NDA}{NAS}$$

...donde:

NDA es el número de atributos repetidos, deducibles mediante operaciones o derivables por inferencia lógica del esquema

NAS es el número de atributos del esquema

4.4.1.1.4. Expresividad

Definimos la expresividad como el número de atributos, entidades e interrelaciones cuyo nombre tiene pleno contenido semántico. De acuerdo con Reingruber y Gregory (1994), un

atributo, entidad o interrelación tiene pleno contenido semántico cuando “su nombre es descriptivo y cumple las reglas de la organización”.

Para ajustar el valor entre 0 y 1, proponemos los siguientes tres ratios:

$$SFAR = \frac{SFA}{NAS}$$

...donde SFA es el número de atributos del esquema con pleno contenido semántico y NAS es, como ya hemos mencionado, el número de atributos del esquema conceptual.

$$SFER = \frac{SFE}{NES}$$

...donde SFE es el número de entidades con pleno contenido semántico y NES es el número de entidades del esquema.

$$SFRR = \frac{SFR}{NRS}$$

...donde SFR es el número de interrelaciones cuyos nombres tienen pleno contenido semántico y NRS es el número de interrelaciones del esquema.

4.4.1.1.5. Legibilidad y autoexplicación

Para Batini et al. (1992), “un diagrama tiene buena legibilidad cuando respeta ciertos criterios estéticos que hacen el diagrama elegante”; por otro lado, definen la autoexplicación como la característica de un esquema que representa “un gran número de propiedades usando el modelo conceptual por sí mismo, sin otros formalismos (por ejemplo, anotaciones en lenguaje natural”.

En nuestro contexto, esta métrica es, como la compleción y la corrección, una métrica nominal que será *alta,* baja o *muy baja*.

Aunque esta métrica es la más subjetiva de las propuestas, no debemos olvidar a Fenton y Pfleeger (1997), que afirman que “es importante reconocer la utilidad de las métricas subjetivas, desde el momento en que entendemos su imprecisión”.

4.4.1.1.6. Extensibilidad y normalidad

Ya que la tercera forma normal es la más natural para representar el universo del discurso, proponemos usar la siguiente métrica, debida a Gray (1991)

$$N = \frac{A_e \times E_e + R_e \times A_e + R_e \times E_e}{A_3 \times E_3 + R_3 \times A_3 + R_3 \times E_3}$$

...donde el numerador es calculado en función del número de atributos, entidades y relaciones del esquema real, y el denominador lo es en función del mismo esquema en tercera forma normal.

“A” representa el número de atributos, E el de entidades y R el de interrelaciones.

4.4.1.1.7. Cohesión del esquema

Definimos la cohesión del esquema como el número de componentes conexas presente en el esquema entidad-interrelación. El mínimo valor alcanzable por esta métrica es 1, y se encuentra acotado superiormente por NES (Número de Entidades del Esquema).

Para ajustar su valor al intervalo (0,1] (es abierto a la izquierda porque el valor 0 nunca se alcanzará) proponemos el uso del "Ratio de cohesión del esquema":

$$COSR = \frac{\sum_{i=1}^{|US|} NEUS_i^2}{NES^2}$$

|US| es el número de componentes conexas del grafo representado por el esquema conceptual.

$NEUS_i$ es el número de entidades en la componente conexa i-ésima.

NES es el número de entidades del esquema.

4.4.1.1.8. Complejidad intra-entidad

La definimos como el número de interrelaciones de una entidad consigo misma. Ésta es una métrica para cada entidad; no obstante podemos proporcionar una métrica válida para el esquema completo de esta manera:

$IEC_{ESQUEMA}$= (media aritmética de IEC, desviación típica de IEC)

4.4.1.1.9. Complejidad del esquema

La complejidad se define como el número de interrelaciones del esquema entidad-interrelación. Igualmente, podemos utilizar el siguiente ratio:

$$SCR = \frac{SC}{MNR}$$

...donde: SC es el número de interrelaciones en el esquema y MNR es el número máximo teórico de interrelaciones en el esquema. Este "máximo teórico" viene dado por la siguiente expresión, en la que NES es el número de entidades del esquema:

$$MNR = \frac{NES \times (NES - 1)}{2}$$

4.4.1.1.10. Longitud interrelacional

Es la longitud del camino interrelacional más largo que podemos encontrar en el esquema, excluyendo las interrelaciones reflexivas.

4.4.1.1.11. Tamaño

Lo medimos como el siguiente triple, formado por el número de entidades, el número de interrelaciones y el número de atributos del esquema:

S=(NES, NRS, NAS)

4.4.1.2. Métricas para el esquema lógico

El objetivo del diseño lógico es la transformación del esquema conceptual en un esquema lógico, en el que la información se organiza según el modelo lógico soportado por el sistema gestor de bases de datos.

4.4.1.2.1. Compleción

Diremos que un esquema lógico es completo cuando refleja perfectamente los requisitos del universo del discurso. La compleción, igual que ocurría para los esquemas conceptuales, es una métrica nominal que será *alta, media* o *baja.*

4.4.1.2.2. Corrección

Un esquema lógico es correcto cuando representa correctamente el universo del discurso captado en su modelo conceptual correspondiente. La corrección también es *alta, media* o *baja*.

4.4.1.2.3. Minimalidad

Una base de datos tiene minimalidad máxima cuando cada aspecto de los requisitos aparece una sola vez en el esquema relacional. Medimos la minimalidad en función del número de atributos sobrantes en las diferentes tablas de la base de datos, aunque también podemos utilizar el siguiente ratio para ajustar su valor al intervalo [0,1):

MR= 1-NFE/NFS ...donde: NFE es el número de atributos sobrantes

NFS es el número de atributos del esquema

La minimalidad expresada por su ratio es útil para estimar el esfuerzo que debe dedicarse al mantenimiento de la integridad de los datos en la base de datos. En la Figura 25 observamos dos tablas de una base de datos: en *Vehículos* se almacena de forma redundante el nombre de cada conductor, ya que puede ser obtenido de la tabla *Conductores*. El valor óptimo de la minimalidad, sin embargo, no tiene por qué ser siempre cero, ya que, por razones de rendimiento, en ocasiones resulta conveniente almacenar valores derivados de otros atributos.

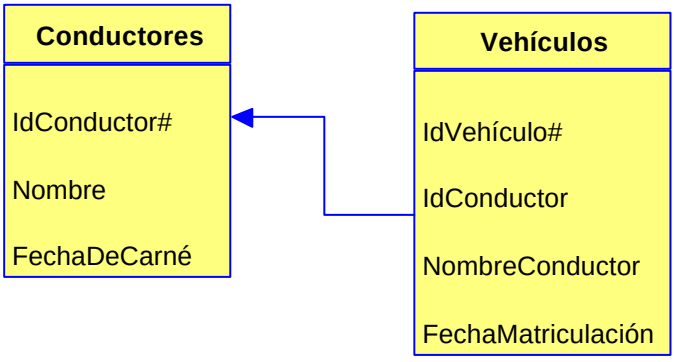


Figura 25. Ejemplo para ilustrar la minimalidad.

4.4.1.2.4. Expresividad

Un esquema lógico es expresivo cuando sus atributos y tablas tienen pleno contenido semántico (en el mismo sentido que en la Expresividad de los esquemas conceptuales) y cuando existe concordancia de nombre entre las claves primarias y sus correspondientes claves ajenas (existe concordancia cuando los nombres de las claves ajenas y primarias han sido mutuamente tenidos en cuenta, de acuerdo a las reglas de la organización).

Para calcular la expresividad pueden utilizarse los siguientes ratios:

$$SFFR = \frac{SFF}{NFS}$$

...donde SFF es el número de atributos con pleno contenido semántico y NFS es el número de atributos del esquema relacional.

$$SFTR = \frac{SFT}{NTS}$$

...donde SFT es el número de tablas del esquema con pleno contenido semántico y NTS es el número de tablas del esquema.

$$FPCR = \frac{FPC}{FKS}$$

...donde FPC es el número de claves ajenas cuyos nombres coinciden con sus correspondientes claves primarias, excepto las claves ajenas cuyas respectivas claves primarias se encuentran en la misma tabla. FKS es el número de claves ajenas del esquema.

4.4.1.2.5. Legibilidad y autoexplicación

Definimos la legibilidad como el siguiente ratio, propuesto por Fenton y Pfleeger (1997) y conocido como “densidad de comentarios”:

$$LR = \frac{CLOC}{LOC + CLOC}$$

...donde:

LR es el ratio de legibilidad

CLOC es el número de líneas de comentario.

LOC es el número de líneas de código.

De acuerdo con Fenton y Pfleeger (1997), la definición más extendida de “Línea de Código” es la propuesta por Hewlett-Packard: cualquier sentencia, salvo las de comentario y las líneas en blanco. Adoptamos esta definición, válida también para bases de datos en general, y para el lenguaje SQL en particular.

4.4.1.2.6. Extensibilidad y normalidad

Definimos la normalidad como el número de tablas en tercera forma normal (3FN) o superior que hay en el esquema lógico. Acotamos este valor en el intervalo [0,1] utilizando el siguiente ratio:

$$NR = \frac{NT3NF}{NTS}$$

...donde:

NT3NF es el número de tablas en 3FN

NTS es el número de tablas del esquema

Recordemos que cada tabla en una forma normal superior a la tercera está también en tercera forma normal, de tal manera que las tablas en BCNF, 4FN y 5FN se encuentran también en esta categoría.

4.4.1.2.7. Cohesión del esquema

La cohesión del esquema se define como el número de componentes conexas que podemos encontrar en el grafo relacional que representa a la base de datos. A partir de este valor, proponemos el uso del ratio de cohesión del esquema:

$$COSR = \frac{\sum_{i=1}^{|US|} NTUS_i^2}{NTS^2}$$

|US| es el número de componentes conexas.

$NTUS_i$ es el número de tablas en la componente conexa i-ésima.

El máximo valor alcanzable por *COSR* es 1, y será alcanzado cuando todas las tablas del esquema se encuentren en la misma componente conexa. En este caso, $|US|=1$ y $NTUS_1 = NTS$, ocurriendo que:

$$COSR = \frac{\sum_{i=1}^{1} NTS^2}{NTS^2} = \frac{NTS^2}{NTS^2} = 1$$

Por otro lado, el valor de *COSR* será cada vez más próximo a cero cuando el número de componentes conexas tienda a infinito y haya una sola tabla en cada componente conexa.

En la Tabla 22 situamos ejemplos de diversos esquemas de bases de datos, todos ellos con cien tablas y dos o tres componentes conexas. En general, valores de *COSR* cercanos a 1 indican que la mayoría de las tablas pertenecen a la misma componente conexa, y que unas pocas permanecen aisladas. Estas tablas "sobrantes" (filas A y J de la Tabla 22) deberían o bien ser eliminadas, o bien incorporadas al esquema (fila J). Los valores medios indican que las tablas se encuentran agrupadas en diversos varias componentes conexas relativamente grandes (filas B, E, F, G, H, I). Probablemente, cada componente conexa constituya realmente una base de datos diferente. Los valores muy bajos significan que las tablas están disgregadas, sin relación unas con otras, lo cual debería invitarnos a rediseñar completamente la base de datos utilizando alguna técnica de Ingeniería Inversa o Reingeniería (por ejemplo, una base de datos formada por 20 componentes conexas con 5 tablas cada una dan un valor *COSR=0.05*).

		Nº de tablas en cada Comp. Conexa.			
		NTUS1	NTUS2	NTUS3	COSR
Base de datos	A	0	10	90	0,82
	B	8	22	70	0,5448
	C	21	4	75	0,6082
	D	23	2	75	0,6158
	E	36	36	28	0,3376
	F	38	50	12	0,4088
	G	40	20	40	0,36
	H	45	23	32	0,3578
	I	85	10	5	0,735
	J	97	2	1	0,9414

Tabla 22. Ejemplos para ilustrar el ratio de cohesión.

4.4.1.2.8. Complejidad intra-tabla

Esta métrica se define, para cada tabla del esquema, como el número de claves ajenas cuya correspondiente clave principal se encuentra en la propia tabla. Para cada tabla se mide utilizando el siguiente ratio:

$$ITCR = \frac{ITC}{NFT}$$ ...donde:

ITC es el número de claves ajenas cuyas respectivas claves primarias se encuentran en la propia tabla

NFT es el número de atributos de la tabla considerada

Para el esquema completo, utilizamos el siguiente par como valor de la métrica:

$ITCR_{SCHEMA}$=(Media de ITCR, Desviación típica de ITCR)

4.4.1.2.9. Complejidad del esquema

La complejidad del esquema es el número de claves ajenas que hay en el esquema. Puede medirse mediante el ratio de complejidad del esquema:

SCR=SC/NFS ...donde: SC (complejidad del esquema) es el número de claves ajenas del esquema y NFS es el número de atributos del esquema.

4.4.1.2.10. Longitud referencial

Es la longitud del camino referencial más largo del esquema.

4.4.1.2.11. Tamaño

Lo medimos como el número de tablas del esquema (NTS).

4.4.1.3. Tabla de resumen

Atributo	Métricas		Posibles valores
	Conceptual	Lógica	
Compleción	Nominal	Nominal	*Alta, Media, Baja*
Corrección	Nominal	Nominal	*Alta, Media, Baja*
Minimalidad	MR=1-(NDA/NAS)	MR=1-(NFE/NFS)	[0, 1)
Expresividad	SFAR=SFA/NAS SFER=SFE/NES SFRR=SFR/NRS	SFFR=SFF/NFS SFTR=SFT/NTS FPCR=FPC/FKS	[0,1) [0,1) [0,1)
Legibilidad	Nominal	LR=CLOC/(LOC+CLOC)	*Alta, Media, Baja* para el esquema conceptual; [0,∞) para el lógico
Normalidad	$N = \frac{A_e \times E_e + R_e \times A_e + R_e \times E_e}{A_3 \times E_3 + R_3 \times A_3 + R_3 \times E_3}$	NR=NT3NF/NTS	[0,1]
Complejidad intra-entidad o intra-tabla	IEC_{SCHEMA}=(media de IEC, desviación típica de IEC)	ITCR=ITC/NFT	[0, 8) para el esquema conceptual, [0,1) para el lógico
Complejidad del esquema	SCR=SC/MNR	SCR=SC/NFS	[0, 8) para el esquema conceptual, [0,1) para el lógico
Longitud interrelacional/referencial	LR = longitud del camino interrelacional más largo	LR = longitud del camino referencial más largo	[0, 8)
Tamaño	S=(NES, NRS, NAS)	S= Número de tablas	[0, 8) para todas las métricas
Cohesión del esquema	$COSR = \frac{\sum_{i=1}^{\lvert US \rvert} NEUSi^2}{NES^2}$	$COSR = \frac{\sum_{i=1}^{\lvert US \rvert} NTUSi^2}{NTS^2}$	(0,1] (cero es un valor límite, inalcanzable en la práctica)

Tabla 23. Resumen de las métricas y ratios.

4.4.1.4. Ejemplo

A continuación, aplicamos las métricas propuestas a un ejemplo, tomado De Miguel y Piattini (1997), relativo a la base de datos de una biblioteca. Mostramos en la Figura 26 el esquema entidad-interrelación y, a continuación, el código SQL correspondiente a su paso al modelo relacional.

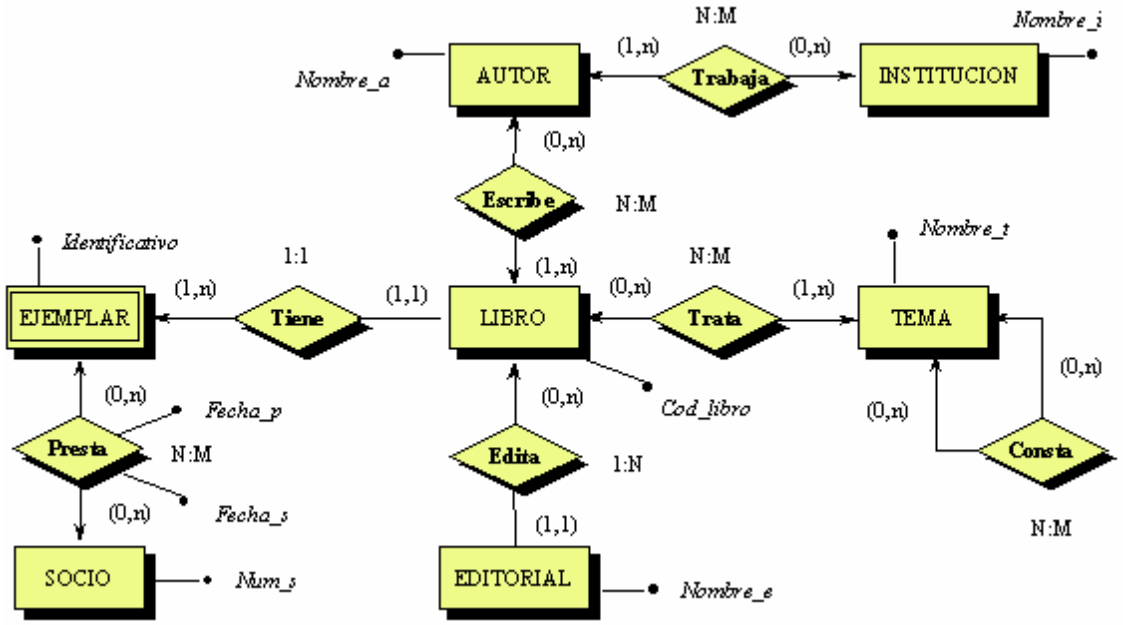


Figura 26. Esquema conceptual para el ejemplo.

```
CREATE SCHEMA BIBLIOTECA
********** DOMINIOS **********
     .....
********** TABLAS **********
CREATE TABLE autor
 (nombre_a nombres,
 nacionalidad        nacionalidades,
 PRIMARY KEY (nombre_a) )

CREATE TABLE trabaja
 (nombre_a nombres,
 nombre_i   instituciones,
 PRIMARY KEY (nombre_a, nombre_i),
 FOREIGN KEY (nombre_a) REFERENCES autor
      ON UPDATE CASCADE)

CREATE TABLE institucion
 (nombre_i  instituciones,
 Dir          lugares,
 Tel          telefonos,
 PRIMARY KEY (nombre_i))

CREATE TABLE libro
(cod_libro   códigos,
 título       título NOT NULL,
 idioma       idiomas NOT NULL,
 num_copias          número_ejemplares NOT NULL,
 nombre_e   nombres NOT NULL,
 año          año,
PRIMARY KEY (cod_libro),
FOREIGN KEY (nombre_e) REFERENCES
editorial
      ON UPDATE CASCADE,

CREATE TABLE escribe
(nombre_a  nombres,
 cod_libro   códigos,
PRIMARY KEY (nombre_a, cod_libro),
FOREIGN KEY (nombre_a) REFERENCES autor
```

```
        ON UPDATE CASCADE,
FOREIGN KEY (cod_libro) REFERENCES libro
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE ejemplar
(cod_libro   códigos,
 identificativo        número_ejemplares,
PRIMARY KEY (cod_libro, identificativo),
FOREIGN KEY (cod_libro) REFERENCES libro
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE socio
(num_s              nums_socio
 dni          dnis NOT NULL,
 nombre       nombres NOT NULL,
 domicilio    lugares NOT NULL,
 tel          teléfonos,
 tipo_s       tipos_socio NOT NULL,
PRIMARY KEY (num_s),
UNIQUE (dni) )

CREATE TABLE presta
(cod_libro   códigos,
 identificativo        número_ejemplar,
 num_s              nums_socio,
 fecha_p      fechas,
 fecha_s      fechas,
 CHECK (fecha_s > = fecha_p),
PRIMARY KEY (cod_libro, identificativo, num_s,
fecha_p),
FOREIGN KEY (cod_libro, identificativo)
REFERENCES ejemplar,
FOREIGN KEY (num_s) REFERENCES socio
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE tema
(nombre_t   temas,
 desc_t      descripciones,
 PRIMARY KEY (nombre_t) )

CREATE TABLE consta
(tema_p      temas,
 tema_s      temas,
 PRIMARY KEY (tema_p, tema_s),
FOREIGN KEY (tema_p) REFERENCES tema
        ON DELETE CASCADE
        ON UPDATE CASCADE,
FOREIGN KEY (tema_s) REFERENCES tema
        ON DELETE CASCADE
        ON UPDATE CASCADE)

CREATE TABLE editorial
(nombre_e  nombre,
 dirección   lugares NOT NULL,
 ciudad              lugares NOT NULL,
 país        lugares NOT NULL,
 PRIMARY KEY (nombre_e) )

CREATE TABLE trata
(cod_libro   códigos,
 nombre_t   temas,
 PRIMARY KEY (cod_libro, nombre_t),
 FOREIGN KEY (nombre_t) REFERENCES tema,
 FOREIGN KEY (cod_libro) REFERENCES libro,
        ON UPDATE CASCADE
        ON DELETE CASCADE)
```

4.4.1.4.1. Medición del esquema conceptual

a) Compleción: *alta*, ya que suponemos que los requisitos se encuentran completamente recogidos en el esquema entidad-interrelación.

b) Corrección: es, evidentemente, *alta*.

c) Minimalidad: si en la entidad *Libro* aparece como atributo el *número de copias*, éste sería un atributo derivado, deducible a partir del *número de ejemplares* que tiene un *libro*, En este caso,

encontraríamos un atributo derivado (repetidos, deducibles o derivables) en el modelo entidad-interrelación, por lo cual:

$$MR = 1 - \frac{NDA}{NAS} = 1 - \frac{0}{9} = 1$$

d) Expresividad: supongamos que, según las reglas de la organización, todos los nombres utilizados en los diseños deben coincidir con los nombres reales utilizando, por ejemplo, notación húngara, excepto en el caso de los atributos clave, que deben ir precedidos del prefijo *Id*. Por tanto:

$$SFAR = \frac{SFA}{NAS} = \frac{1}{9} = 0.11$$

...ningún atributo cumple las reglas de la organización

$$SFER = \frac{SFE}{NES} = \frac{7}{7} = 1$$

$$SFRR = \frac{SFR}{NRS} = \frac{7}{7} = 1$$

...todas las entidades e interrelaciones tienen nombres reales y significativos

e) Legibilidad y autoexplicación: es *alta.*

f) Extensibilidad y normalidad: como el esquema entidad-interrelación está en 3FN, su valor es 1.

g) Cohesión del esquema:

$$COSR = \frac{\sum_{i=1}^{1} NEUS_i^2}{NES^2} = \frac{7^2}{7^2} = 1$$

...porque el grafo ER tiene sólo una componente conexa, formada por 7 entidades.

h) Complejidad intra-entidad: como sólo tenemos una entidad con interrelaciones reflexivas, el valor de esta métrica es:

IEC_{SCHEMA}=(0.14, 0.38)

i) Complejidad del esquema:

$$SCR = \frac{SC}{MNR} = \frac{SC}{\frac{NES \times (NES - 1)}{2}} = \frac{7}{\frac{7 \times 6}{2}} = \frac{7}{21} = 0.33$$

...porque hay 7 interrelaciones (SC) y 7 entidades (NES).

j) Longitud interrelacional: su valor es 4, que podemos obtener al recorrer por el esquema las siguientes entidades: *Institución* → *Autor* → *Libro* → *Ejemplar* → *Socio*

k) Tamaño: S=(7, 7, 9), correspondiente al número de entidades, interrelaciones y atributos del esquema.

4.4.1.4.2. Medición del esquema lógico

a) Compleción: hemos supuesto compleción *alta* para el esquema conceptual, pero en el esquema lógico mostrado no se ha reflejado la interrelación existente en el esquema conceptual entre *Institución* y *Trabaja:* a pesar de que el diseñador ha reservado en esta tabla el atributo *nombre_i* para hacerlo clave ajena de *Institución*, ha olvidado la cláusula "FOREIGN KEY (nombre_i) REFERENCES institucion" en *Trabaja*. Por tanto, el valor de esta métrica es *medio*.

b) Corrección: el esquema representa fielmente la realidad que quiere modelarse, por lo que la corrección es *alta*.

c) Minimalidad:

$$MR = 1 - \frac{NFE}{NFS} = 1 - \frac{1}{38} = 0.97$$

...porque *num_copias* de la tabla *Libro* puede ser calculado mediante una consulta a la tabla *Ejemplares*, por lo que hay un atributo sobrante entre los 38 que forman el esquema lógico.

d) Expresividad:

$$SFFR = \frac{SFF}{NFS} = \frac{1+0+0+3+0+1+3+1+0+0+3+0}{38} = \frac{12}{38} = 0.31$$

...si suponemos las mismas reglas mencionadas en el ejemplo del esquema conceptual, en el esquema lógico nos encontramos con 12 atributos cuyos nombres cumplen las reglas de la organización, frente a los 38 atributos del esquema.

$$SFTR = \frac{SFT}{NTS} = \frac{12}{12} = 1$$

...porque los nombres de todas las tablas cumplen las reglas de la organización.

$$FPCR = \frac{FPC}{FKS} = \frac{12}{12} = 1$$

...porque los nombres de todos los atributos que son claves ajenas concuerdan con sus respectivas claves primarias.

e) Legibilidad y autoexplicación:

$$LR = \frac{CLOC}{LOC + CLOC} = \frac{4}{92 + 4} = 0.041$$

...ya que hay 4 líneas de comentario entre las 96 líneas totales.

f) Extensibilidad y normalidad:

N=1 ...porque todas las tablas están en 3FN.

g) Cohesión del esquema:

$$COSR = \frac{\sum_{i=1}^{|US|} NTUSi^2}{NTS^2} = \frac{1^2 + 11^2}{12^2} = \frac{122}{144} = 0.85$$

...porque hay dos componentes conexas: una con una sola tabla (*Institución)*, y otra con las once restantes.

h) Complejidad intra-tabla:

$ITCR_{SCHEMA}$=(0, 0) ...porque no hay claves ajenas cuya correspondiente clave primaria se encuentre en la misma tabla.

i) Complejidad del esquema:

$$SCR = \frac{SC}{NFS} = \frac{11}{38} = 0.29$$

...ya que hay 11 claves ajenas y 38 atributos.

j) Longitud referencial: su valor es 3, que podemos obtener a partir de las relaciones de integridad referencial siguientes: *Presta* → *Ejemplar* → *Libro* → *Editorial*.

k) Tamaño: es 12 porque hay doce tablas.

4.4.1.5. Validación teórica de las métricas

En este epígrafe comprobamos el ajuste de dos de las métricas propuestas (la validación de las restantes se realiza de forma análoga) al marco formal propuesto en Briand, Morasca y Basili (1996). En este marco se especifican las propiedades que deben cumplir cinco tipos de métricas (llamadas *funciones* en el marco formal): tamaño, longitud, complejidad, acoplamiento y cohesión. De este modo, dada una métrica, ésta no podrá ser considerada -por ejemplo- de acoplamiento, si no cumple las propiedades indicadas para las métricas de acoplamiento.

Las propiedades propuestas se basan en la existencia de sistemas. Un sistema se define de la siguiente manera:

S=(E,R)	E es el conjunto de entidades de S; $R \subseteq ExE$; R es el conjunto de relaciones entre los elementos del sistema

Para estudiar el cumplimiento de propiedades de una función, es preciso identificar el conjunto de entidades sobre el que actúa dicha función, así como su conjunto de relaciones.

4.4.1.5.1. Métrica "Complejidad del esquema"

En el caso de los esquemas conceptuales, la hemos definido como el número de interrelaciones del esquema (SC_C), y como el número de claves ajenas en el caso de los esquemas lógicos (SC_L).

Comprobemos que estas métricas son de complejidad, enfrentándolas a las propiedades de las *funciones de complejidad* del marco formal seleccionado.

A los efectos de este marco formal, consideraremos que un sistema *S* es un par *(E,R)*, donde *E* representa el conjunto de entidades del esquema conceptual o el conjunto de tablas en el caso del esquema lógico, y que *R* es el conjunto de interrelaciones en el esquema conceptual o el conjunto de restricciones referenciales en el esquema lógico. Según el marco formal, $R \subseteq E \times E$, por lo que, en los esquemas conceptuales, *R* estará formado por pares de entidades (e_1, e_2) tales que existe una interrelación en el esquema conceptual que interrelaciona las entidades e_1 y e_2; en los esquemas lógicos, *R* estará formado por pares de tablas (t_1, t_2) tales que t_1 posee una clave ajena cuya respectiva clave primaria se encuentra en t_2.

En el enunciado de las propiedades, nos referiremos con *C(S)* indistintamente tanto a SC_C *(S)* como a SC_L *(S)*.

Propiedad 1. No negatividad.

La complejidad de un sistema *S=(E,R)* es no negativa: $C(S) \geq 0$. ♦

En efecto: es imposible encontrar un esquema conceptual o lógico con un número negativo de interrelaciones o claves ajenas, respectivamente.

Propiedad 2. Valor nulo.

Según esta propiedad, si el conjunto de relaciones del sistema es el conjunto vacío, entonces su complejidad es cero.

$R = \emptyset \Rightarrow C(S) = 0$ ♦

En efecto:

Si el conjunto de interrelaciones del esquema conceptual es el vacío, entonces su cardinal es cero y es cero también el valor de SC_C. Del mismo modo, si el conjunto de claves ajenas de un esquema lógico es el conjunto vacío, su cardinal es cero y es también cero el valor de SC_L.

Propiedad 3. Simetría.

Esta propiedad afirma que la complejidad del sistema no depende de la convención elegida para representar las relaciones entre sus elementos. Es decir:

$(S=(E,R) \wedge S^{-1}=(E, R^{-1})) \Rightarrow C(S)=C(S^{-1})$ ♦

En efecto:

a) En el caso de los esquemas conceptuales:

$R = \{ (e_i, e_j) / \text{existe una interrelación entre } e_i \text{ y } e_j)\}$

La métrica SC_C se define como el número de interrelaciones que hay en el esquema: esto es, $SC_C = |R|$.

Por otro lado: $R^{-1}=\{ (e_j, e_i) / \text{existe una interrelación entre } e_j \text{ y } e_i)\}$

Evidentemente, $|R|=|R^{-1}|$, con lo que $SC_C=SC_C^{-1}$.

b) En el caso de los esquemas lógicos:

$R = \{ (t_i, t_j) / t_i \text{ tiene una clave ajena cuya respectiva clave primaria se encuentra en } t_j)\}$

La métrica SC_L se define como el número de claves ajenas que hay en el esquema: esto es, $SC_L = |R|$.

Por otro lado:

$R^{-1}= \{ (t_j, t_i) / t_j \text{ tiene un clave primaria a la cual corresponde una clave ajena presente en } t_i)\}$

Evidentemente, $|R|=|R^{-1}|$, con lo que $SC_L=SC_L^{-1}$.

Propiedad 4. Monotonía de módulos.

Esta propiedad afirma que la complejidad de un sistema no es menor (es mayor o igual) que la suma de las complejidades de dos cualesquiera de sus módulos sin relaciones en común. De otro modo:

$S=(E,R) \wedge m_1=(Em_1, Rm_1) \wedge m_2=(Em_2, Rm_2) \wedge m_1 \cup m_2 \subset S \wedge Rm_1 \cap Rm_2 = \varnothing \Rightarrow$

$\Rightarrow C(S) \geq C(m_1) + C(m_2)$ ♦

a) Esquemas conceptuales.

En este caso, un módulo m está formado por un conjunto Em de entidades y las interrelaciones Rm que conectan las entidades pertenecientes a Em.

Supongamos lo contrario a lo propuesto por la propiedad: es decir, que:

$\exists\, m_1=(Em_1, Rm_1), \exists\, m_2=(Em_2, Rm_2), m_1 \cup m_2 \subset S, Rm_1 \cap Rm_2 = \varnothing \;/$

$/\; SC_C(S) < SC_C(m_1) + SC_C(m_2)$ (Ecuación 1)

Puesto que la complejidad del esquema se define como el número de interrelaciones del esquema o, lo que es lo mismo, el cardinal del conjunto de interrelaciones, $SC_C(S)=|R_S|$ y $SC_C(m_1)=|Rm_1|$ y $SC_C(m_2)=|Rm_2|$.

Para que la Ecuación 1 fuera cierta, debería ocurrir que:

$|R_S| < |Rm_1| + |Rm_2|$

Es decir, que el número de interrelaciones del esquema conceptual fuera menor que la suma del número de interrelaciones en los dos módulos, que son dos subconjuntos del esquema. Esto, sin embargo, es imposible, ya que una de las precondiciones de esta propiedad establece que los conjuntos de interrelaciones de m_1 y m_2 son disjuntos:

$Rm_1 \cap Rm_2 = \emptyset$

Con lo cual nunca podrá ocurrir que:

$|Rm_1| + |Rm_2| > |R_S|$, y siempre ocurrirá que:

$|Rm_1| + |Rm_2| \leq |R_S|$

Por otro lado, la desigualdad será estricta ($|Rm_1| + |Rm_2| < |R_S|$) cuando la unión de los dos módulos no sea el sistema completo ($m_1 \cup m_2 \neq S$), causa ésta que puede deberse a los dos casos siguientes, que ilustramos además en la Figura 27:

(1) $E \notin Em_1 \cup Em_2$

(2) $E = Em_1 \cup Em_2$ pero $R \notin Rm_1 \cup Rm_2$

b) Esquemas lógicos:

La demostración realizada para los esquemas conceptuales es igualmente válida para los esquemas lógicos, sustituyendo SC_C por SC_L, y el término “interrelación” por “integridad referencial”.

Propiedad 5. Aditividad de módulos disjuntos.

Esta propiedad afirma que la complejidad de un sistema compuesto de dos módulos disjuntos es igual a la suma de las complejidades de los dos módulos. En otras palabras:

$S = (E, R) \wedge S = m_1 \cup m_2 \wedge m_1 \cap m_2 = \emptyset \Rightarrow C(S) = C(m_1) + C(m_2)$ ♦

En el marco formal utilizado, dos módulos se consideran disjuntos si no tienen ni entidades ni relaciones en común.

En las dos métricas que nos ocupan, resulta evidente que la complejidad del esquema (conceptual o lógico) es la suma de las complejidades de los dos esquemas disjuntos cuya unión compone el esquema completo.

Conclusión.

Las métricas de complejidad del esquema conceptual y lógico son funciones de complejidad conforme al marco de Briand, Morasca y Basili (1996).

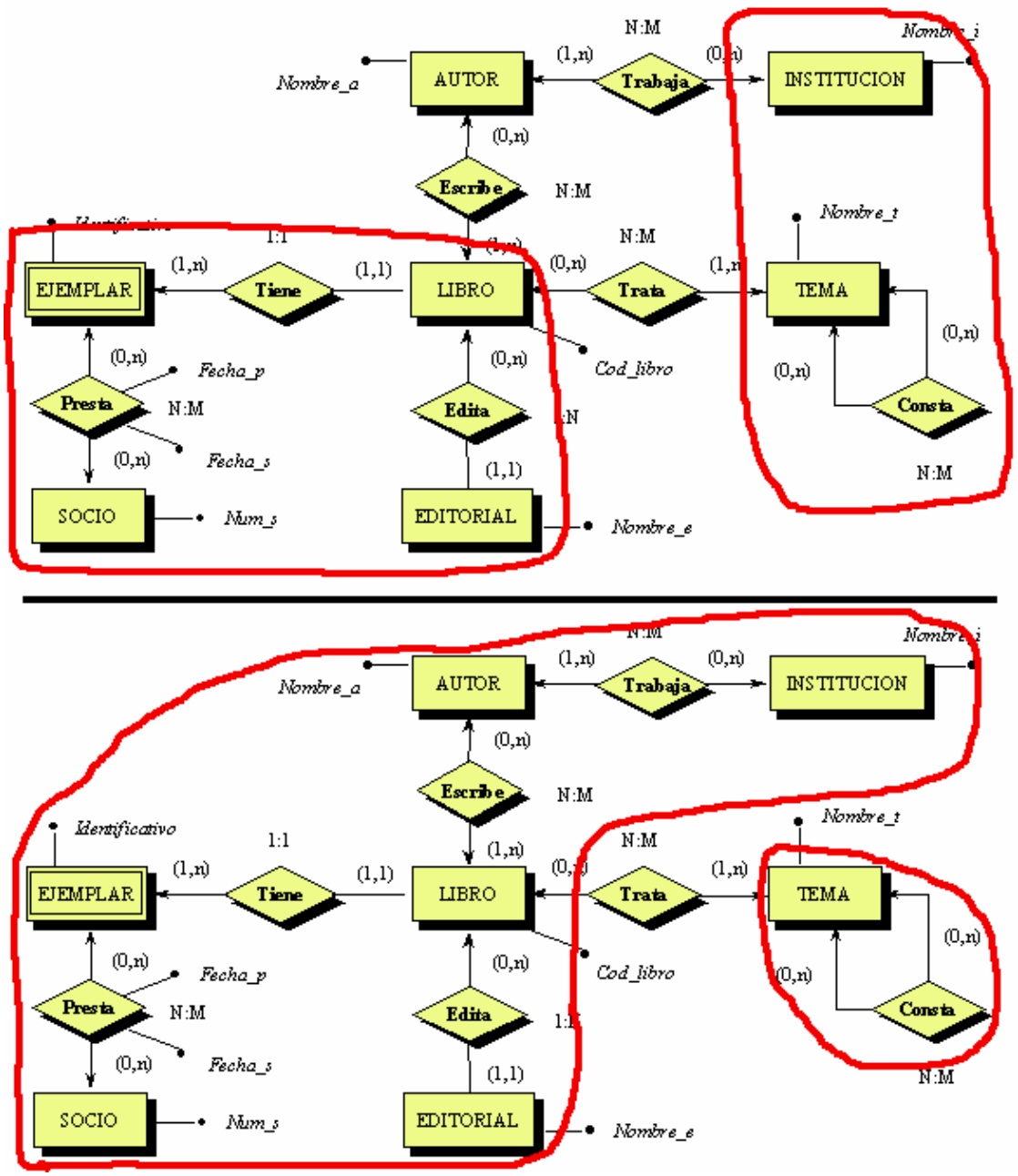


Figura 27. Razones para la desigualdad estricta en la Propiedad de complejidad 4.

4.4.1.5.2. Métrica "Tamaño del esquema"

En el caso de los esquemas conceptuales, el tamaño del esquema se define como un triple formado por el número de entidades, interrelaciones y atributos. Si bien la consideración del triple no tiene cabida dentro del marco seleccionado, sí que lo tiene el hecho de considerar que el tamaño se puede medir como tres métricas distintas, una por cada elemento del triple. El tamaño de los esquemas lógicos lo hemos medido como el número de tablas del esquema.

Comprobemos que las métricas Número de Entidades, Número de Interrelaciones, Número de Atributos y Número de Tablas son *funciones de tamaño* conforme al marco formal de Briand et al. (1996).

A los efectos de este marco formal, consideraremos que un sistema *S* es un par *(E,R)*, donde *E* representa el conjunto de entidades, interrelaciones o atributos del esquema conceptual (según la métrica que estemos considerando), o el conjunto de tablas en el caso del esquema lógico, y que *R* es el conjunto de interrelaciones en el esquema conceptual o el conjunto de claves ajenas en el esquema lógico. Sin embargo, para las funciones de tamaño no interviene el conjunto de relaciones entre los elementos de *E*.

Las propiedades cuyo cumplimiento debemos comprobar son:

Propiedad 1. No negatividad.

El tamaño de un sistema es mayor o igual a cero:

T(S) ³ 0 ♦

En nuestro caso, es imposible encontrar un esquema conceptual o lógico con un número negativo de entidades, interrelaciones, atributos o tablas, por lo que es obvio el cumplimiento de esta propiedad.

Propiedad 2. Valor nulo.

El tamaño de un sistema es cero si no tiene elementos:

S = (E,R) ^ |E| = 0 Þ T(S) = 0 ♦

En efecto: si no hay entidades en el esquema conceptual, entonces el Número de Entidades es cero, y puede decirse lo mismo del Número de Interrelaciones y Número de Atributos, así como del número de tablas del esquema lógico.

Propiedad 3. Aditividad.

Esta propiedad afirma que el tamaño de un sistema es igual a la suma del tamaño de dos de sus módulos sin entidades en común:

$m_1=(Em_1, Rm_1) \wedge m_2=(Em_2, Rm_2) \wedge S = m_1 \cup m_2 \wedge Em_1 \cap Em_2=\emptyset \Rightarrow$

$\Rightarrow T(S)=T(m_1) + T(m_2)$ ♦

En efecto, utilicemos la métrica que utilicemos, el tamaño del esquema (conceptual o lógico) será la suma de dos cualesquiera de sus módulos, puesto que éstos serán subconjuntos disjuntos de entidades, interrelaciones, atributos o tablas para, respectivamente, las métricas Número de Entidades, de Interrelaciones, de Atributos o de Tablas.

4.4.1.6. Validación empírica de las métricas “Complejidad del esquema” y “Longitud del esquema” para esquemas lógicos de bases de datos

Para comprobar la validez no sólo teórica, sino también empírica, de las métricas, se han realizado algunos experimentos con estudiantes de Bases de Datos (correspondientes a tercer curso de Ingeniería Técnica en Informática) en la Escuela Superior de Informática de la Univer-

sidad de Castilla-La Mancha (campus de Ciudad Real). A continuación presentamos uno de los experimentos realizados, con el que pretendemos comprobar si las diferentes métricas propuestas influyen en la comprensibilidad de los esquemas de bases de datos. Como es bien sabido, la mantenibilidad está muy influida por la comprensibilidad; ésta, a su vez, se ve sobre todo afectada por la complejidad (Li, 1987). Por tanto, si del experimento podemos demostrar que las métricas de Complejidad y Longitud influyen en la comprensibilidad, podremos deducir que influyen también en su mantenibilidad: esto es, en su facilidad de mantenimiento.

4.4.1.6.1. Descripción del experimento

Se presentaron a 59 alumnos cuatro esquemas lógicos correspondientes a cuatro diferentes bases de datos, con distintos valores de Complejidad (número de claves ajenas) y Longitud referencial. Se pretendía comprobar los efectos de estas dos propiedades sobre el nivel de comprensión del esquema de base de datos.

Las hipótesis del experimento son las siguientes:

- Hipótesis nula: diferentes valores de las métricas Longitud y Complejidad del esquema no afectan a la comprensión del esquema de base de datos.

- Hipótesis alternativa 1: el valor de la Longitud del esquema afecta a la comprensión del esquema de base de datos.

- Hipótesis alternativa 2: el valor de la Complejidad del esquema afecta a la comprensión del esquema de base de datos.

- Hipótesis alternativa 3: la combinación de la Longitud y Complejidad del esquema afecta a la comprensión del esquema de base de datos.

4.4.1.6.2. Materiales del experimento

La Complejidad de los esquemas presentados a los alumnos podía tomar valor 8 o 5, y la Longitud referencial o bien 2 o bien 5. Combinando estos valores, las métricas de los cuatro esquemas tomaban valores que podían describirse según los pares (8, 2), (8, 5), (5, 2), (5, 5), como se muestra en la Tabla 24:

		Factor B (Complejidad)	
		Bajo	Alto
Factor A (Longitud referencial)	Bajo	2,5	2,8
	Alto	5,5	5,8

Tabla 24. Diseño cruzado para el experimento.

Los alumnos debían realizar las mismas tres operaciones (inserciones, borrados y actualizaciones) sobre cada uno de los cuatro esquemas. La Tabla 25 muestra las plantillas utilizadas por los alumnos para cada uno de los esquemas:

1. ¿Qué tablas y cuántas filas se verán afectadas en cada taba si borramos en la Tabla 5 la fila con cod1=210?					
Tabla 1	Tabla 2	Tabla 3	Tabla 4	Tabla 5	Tabla 6
2. ¿Qué tablas y cuántas filas se verán afectadas en cada taba si actualizamos la columna X de la fila con cod2=11 en la Tabla 3?					
Tabla 1	Tabla 2	Tabla 3	Tabla 4	Tabla 5	Tabla 6
3. ¿Qué tablas y cuántas filas y columnas es necesario añadir si queremos añadir una nueva fila en la tabla 4 (suponga que no existe en la base de datos ninguno de los datos necesarios)?					
Tabla 1	Tabla 2	Tabla 3	Tabla 4	Tabla 5	Tabla 6

Tabla 25. Plantilla utilizada en el experimento.

4.4.1.6.3. Resultados y análisis

Hay tres elementos principales que deben ser tenidos en cuenta cuando se eligen técnicas de análisis: la naturaleza de los datos recogidos, la razón por la cual se realiza el experimento y el tipo de diseño experimental utilizado (Pfleeger, 1995). Debido al tipo de experimento, el estadístico F de Snedecor es el más apropiado para la obtención de resultados (Rohatgi, 1976).

La Tabla 26 muestra los resultados del estadístico F:

Fuente de variación	Qi	Grados de libertad	s_i^2	Ratio F
Longitud	18.457	1	18.5	1.67
Complejidad	531.000	1	531	48.1
Interacción	31.339	1	31.3	2.84
Error	2560.604	1	11.0	
Total	3141.102	232		

Tabla 26. Resultados del estadístico F.

Con un valor α=0.1, comparando estos valores con $F_{1,232}$=2.73 podemos asegurar que:

- Hipótesis alternativa 1: el valor de la Longitud del esquema afecta a la comprensión del esquema de base de datos.

Puesto que 1.67<2.73, **la Longitud referencial no afecta a la comprensión del esquema de base de datos.**

- Hipótesis alternativa 2: el valor de la Complejidad del esquema afecta a la comprensión del esquema de base de datos.

Puesto que 48.1>2.73, **el número de claves ajenas afecta a la comprensión del esquema de base de datos.** Por tanto, **la hipótesis alternativa 2 es válida** porque el valor de la Complejidad del esquema afecta a los resultados obtenidos.

- Hipótesis alternativa 3: la combinación de la Longitud y Complejidad del esquema afecta a la comprensión del esquema de base de datos.

Puesto que 2.84>2.73, **la combinación de las dos métricas afecta a la comprensión del esquema de la base de datos.**

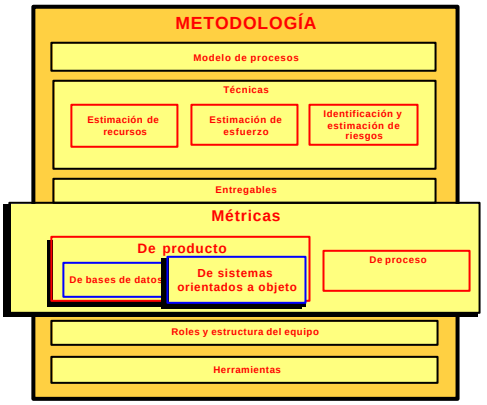
4.4.1.6.4. Conclusión

El número de claves ajenas de una base de datos relacional es un indicador sólido de su comprensibilidad, y la longitud del árbol referencial no es relevante por sí mismo, pero sirve para modular los efectos del número de claves ajenas.

De acuerdo con Schneidewind (1997b), la Complejidad del esquema es una métrica "dominante".

4.4.2. ESTIMACIÓN DEL ESFUERZO DE MANTENIMIENTO ADAPTATIVO EN SISTEMAS ORIENTADOS A OBJETO

La utilización de las métricas que se presentan a continuación constituyen una valiosa técnica de estimación del esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto, cuyo uso se propone en la tarea P2.1 (Análisis de la petición, referida a mantenimiento adaptativo, y que se detalla en la página 114).



Nesi y Querci (1998) presentaron la familia de métricas *CC* (Class Complexity/Size) para estimar el esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto, y cuya validez fue comprobada por esos mismos autores en esa misma referencia.

Algunas de las ecuaciones que definen dichas métricas han sido, sin embargo, redefinidas para facilitar su caracterización formal, por lo que fue necesario realizar nuevos experimentos para seguir constatando su validez. Además, basándonos en dichas métricas, se han definido dos valiosos operadores de composición para sistemas orientados a objetos.

En Fioravanti, Nesi y Polo (1999) se ha publicado esta nueva versión redefinida de las métricas, los mencionados operadores de composición, los resultados de la nueva validación

empírica y la caracterización formal de las métricas. A la definición de los operadores de composición y a la caracterización formal se dedicó especialmente este doctorando durante una estancia de investigación con el Profesor Dr. D. Paolo Nesi en la Universidad de Florencia. A continuación presentamos la familia de métricas, los operadores de composición y su caracterización formal. La validación empírica se encuentra en la citada referencia, en la que se expone su utilización para estimar el esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto.

Las métricas *CC* no han sido utilizadas en los casos prácticos expuestos en el capítulo quinto, ya que estos sistemas son no orientados a objeto. Sin embargo, mantenemos su inclusión en la metodología debido a su utilidad para los casos en que deba estimarse el esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto.

4.4.2.1. Introducción

En las últimos años se han propuesto multitud de métricas para medir las características internas y externas de sistemas orientados al objeto. La complejidad y el tamaño han sido quizás los más estudiados, debido a su relación con el esfuerzo de mantenimiento correctivo, desarrollo, reutilización o mantenimiento adaptativo (Chidamber y Kemerer, 1994; Lorenz y Kidd, 1994; Li y Henry, 1993; Nesi y Querci, 1998; Chidamber, Darci y Kemerer, 1998; Fioravanti, Nesi y Stortoni, 1998; Henderson-Sellers, 1996). En la mayoría de los casos, las métricas han sido validadas empíricamente utilizando datos reales y herramientas estadísticas (Nesi y Querci, 1998; Chidamber et al., 1998; Fioravanti, Nesi y Perlini, 1998). La importancia de la validación empírica está ampliamente reconocida como una forma de demostrar la presencia de relaciones entre las características internas y externas del software (Schneidewind, 1992). Sin embargo, la validación empírica debería aparecer acompañada de una validación teórica que ayude a comprender mejor las métricas y sus propiedades.

En algunos casos, la validación teórica ha sido realizada utilizando marcos formales consistentes en conjuntos de propiedades (Chidamber y Kemerer, 1994, enfrentan sus métricas al marco de Weyuker, 1988). Sin embargo, no hay unanimidad en cuanto a la aceptación de estos marcos y sus propiedades. Por otro lado, la mayoría de los marcos propuestos se centran únicamente en sistemas no orientados a objeto y, en muchos casos, las propiedades propuestas no pueden ser directamente aplicadas a sistemas orientados a objeto.

En esta sección se presenta la validación teórica de la familia de métricas *CC* (*Class Complexity/Size*), cuya validación empírica fue comprobada por Fioravanti et al. (1999a) durante la adaptación de LIOO (*Lectern Interactive Object Oriented*, un sistema que sustituye a los atriles y las partituras sobre papel en las orquestas) desde MS-DOS a UNIX, y al adaptar la versión inicial del editor de música a una versión distribuida colaborativa. En la referencia citada puede consultarse el comportamiento predictivo de las métricas.

El conjunto de propiedades que utilizamos ha sido construido ampliando e integrando la mayoría de los marcos formales más conocidos. El conjunto de propiedades utilizado no repre-

senta un conjunto exhaustivo de propiedades para caracterizar métricas para sistemas orientados a objeto, ni tampoco pretende ser el conjunto mínimo de propiedades que deben ser satisfechas por métricas orientadas a objeto, sino que puede ser visto como un modo de entender mejor el comportamiento de las métricas, así como un paso más para definir formas y herramientas que ayuden a la clasificación de las métricas.

4.4.2.2. Métricas *CC* y *CC1*

La métrica *CC* se define para una clase *c* de la siguiente manera:

$$CC(c) = ECD(c) + ICI(c)$$

...donde *ECD(c)* representa la complejidad o tamaño debida a la descripción externa de la clase, e *ICI(c)* es la debida a la implementación interna. Los componentes *ECD(c)* e *ICI(c)* se pueden descomponer en complejidades o tamaños debidos a miembros de clase locales o heredados:

$$ECD(c) = ECDL(c) + ECDI(c)$$

$$ICI(c) = w_{CL}\, CL(c) + w_{CI}\, CI(c)$$

Además:

$$ECDL(c) = w_{CACL}\, CACL(c) + w_{CMICL}\, CMICL(c)$$

$$ECDI(c) = w_{CACI}\, CACI(c) + w_{CMICI}\, CMICI(c)$$

Aquí, *CACL* es la complejidad o tamaño debidos a atributos definidos localmente; *CACI* es la complejidad o tamaño debidos a atributos heredados; *CMICL* es la complejidad o tamaño de los parámetros de los métodos definidos localmente; *CMICI* es la complejidad o tamaño de los parámetros de los métodos heredados; *CL* es la complejidad o tamaño de los métodos locales, y *CI* es la complejidad o tamaño de los métodos heredados.

Para una clase, cada una de estas funciones se evalúa de la siguiente manera:

$$CACL(c) = \sum_{i=1}^{NAL(c)} CC(a_i)$$

$$CACI(c) = \sum_{i=1}^{NAI(c)} CC(a_i)$$

$$CMICL(c) = \sum_{i=1}^{NML(c)} \sum_{i=1}^{|p(m_j)|} CC(p_i)$$

$$CMICI(c) = \sum_{i=1}^{NMI(c)} \sum_{i=1}^{|p(m_j)|} CC(p_i)$$

$$CL(C) = \sum_{i=1}^{NML(c)} FM(m_i)$$

$$CI(C) = \sum_{i=1}^{NMI(c)} FM(m_i)$$

En estas expresiones: *NAL(c)* y *NAI(c)* representan, respectivamente, los conjuntos de atributos definidos localmente y heredados; *NML(c)* y *NMI(c)* son los conjuntos de métodos definidos localmente y heredados, respectivamente; $p(m_j)$ es el conjunto de parámetros del método *j-ésimo*; por último, $FM(m_i)$ representa una métrica funcional: o bien $Vg'(m_i)$, o bien $LOC(m_i)$. Esta última métrica representa el número de líneas de código del método m_i; la primera se calcula según la siguiente ecuación:

$$Vg'(m) = |Ejes(m)| - |Nodos(m)| + 1$$

...siendo *Ejes(m)* y *Nodos(m)* el número de ejes y nodos del grafo de flujo que representa al método *m*.

Cuando la métrica se aplica a un tipo básico (como un *int* o un *char* de C++), la métrica proporciona los mismos valores para tales atributos: 0 si se utiliza *Vg'*, y 1 cuando se usa *LOC* (suponiendo que cada atributo está escrito en una línea de código diferente).

Cuando los pesos son iguales a 1, se obtiene la versión *CC1* de *CC*:

$$CC1(c) = CACL(c) + CACI(c) + CMICL(c) + CMICI(c) + CL(c) + CI(c)$$

Para un sistema orientado a objetos completo *S*, la métrica *CC* se define de la siguiente manera:

$$CC(S) = \sum_{i=1}^{NCL(S)} CC(c_i)$$

...donde *NCL(S)* representa el número de clases del sistema.

4.4.2.3. Composición mediante el operador IPO ("is part of", es parte de)

Las clases de un sistema orientado a objetos pueden estar relacionadas mediante relaciones de agregación, en la que una clase *es parte de* otra. De este modo, si queremos calcular el valor de *CC* para una clase c_1 que tenga un atributo de clase c_2, debemos añadir a la medida de c_2 el valor de la medida de c_1 de acuerdo a un factor de peso:

$$CC(c_2 \text{ IPO } c_1) = w_{CACL}\, CACL(c_2 \text{ IPO } c_1) + w_{CACI}\, CACI(c_1) + w_{CMICL}\, CMICL(c_1) + w_{CMICI}\, CMICI(c_1) + w_{CL}\, CL(c_1) + w_{CI}\, CI(c_1)$$

Si desarrollamos:

$$CC(c_2 \text{ IPO } c_1) = [w_{CACL}\, CC1(c_2) + w_{CACL}\, CACL(c_1)] + w_{CACI}\, CACI(c_1) + w_{CMICL}\, CMICL(c_1) + w_{CMICI}\, CMICI(c_1) + w_{CL}\, CL(c_1) + w_{CI}\, CI(c_1)$$

Con lo cual:

$$CC(c_2\ IPO\ c_1) = CC(c_1) + w_{CACL}\ CC1(c_2)$$

Esto significa que la métrica *CC* es aditiva por medio de un peso con respecto a la composición mediante el operador IPO. La medida de la clase compuesta mediante IPO es menor que la suma de las medidas de las dos clases; es decir:

$$CC(c_2\ IPO\ c_1)=CC(c_1)+w_{CACL}CC1(c_2) \pounds CC(c_1) + CC(c_2)$$

Si $w_{CACL}\ CC1(c_2) \ ^3\ CC(c_2)$, entonces esta propiedad no es satisfecha y la composición mediante IPO produce medidas mayores que la suma de las medidas de ambas clases. En algunos casos esto puede tener justificación: por ejemplo, si una instancia de una pequeña clase c_2 es usada en c_1, la implementación de los métodos relacionados con la manipulación de la instancia pueden ser incluso más costosa que la construcción de c_2. Esto, obviamente, depende de la métrica funcional utilizada como base de *CC*.

4.4.2.4. Composición mediante el operador de especialización ISA ("is a", es un)

Para las relaciones de especialización (mediante el operador ISA), la métrica *CC* con pesos no es aditiva, tal y como ocurre con muchas otras métricas para sistemas orientados a objeto (Chidamber y Kemerer, 1994). Desde el punto de vista de este operador, la métrica *CC* puede ser vista como si estuviera compuesta de dos partes: *CCL* (complejidad local) y *CCI* (complejidad heredada):

$$CC(c)=CCL(c)+CCI(c)$$

...donde:

$$CCL(c\ )=w_{CACL}\ CACL(c)+w_{CMICL}\ CMICL(c) + w_{CL}\ CL(c)$$

$$CCI(c\ )=w_{CACI}\ CACI(c)+w_{CMICI}\ CMICI(c) + w_{CI}\ CI(c)$$

La especialización de una clase c_1 en una clase c_2 significa que la clase c_2 se implementa heredando los miembros de c_1. En c_2, todos los miembros de la superclase c_1 (que pueden ser localmente definidos en la propia c_1 o heredados de otras clases) son utilizables por las instancias de c_2; de este modo, la superclase contribuye a la complejidad y tamaño de la clase especializada.

De acuerdo a la definición de *CC*, el operador ISA presenta el siguiente comportamiento:

$$CC(c_2\ ISA\ c_1) = w_{CACL}\ CACL(c_2)+w_{CACI}\ CACI(c_2\ ISA\ c_1)+w_{CMICL}\ CMICL(c_2)+$$

$$+ w_{CMICI}\ CMICI(c_2\ ISA\ c_1)+w_{CL}\ CL(c_2)+w_{CI}\ CI(c_2\ ISA\ c_1)$$

Si desarrollamos los términos que contienen el operador que estamos tratando, obtenemos:

$$CC(c_2\ ISA\ c_1) = w_{CACL}\ CACL(c_2)+ w_{CACI}\ CACI(c_2)+ w_{CACI}\ CACI(c_1)+$$

$$+ w_{CACI}\ CACL(c_1) + w_{CMICL}\ CMICL(c_2)+ w_{CMICI}\ CMICI(c_2)+$$

$$+ w_{CMICI}\ CMICI(c_1) + w_{CMICI}\ CMICL(c_1) +$$

$$+ w_{CL}\ CL(c_2) + w_{CI}\ CI(c_2) + w_{CI}\ CI(c_1) + w_{CI}\ CL(c_1)$$

Con lo cual:

$$CC(c_2\ ISA\ c_1) = CC(c_2) + w_{CACI}\ [CACL(c_1) + CACI(c_1)] +$$

$$+ w_{CMICI}\ [CMICL(c_1) + CMICI(c_1)] + w_{CI}\ [CL(c_1) + CI(c_1)]$$

En su parte de herencia, *CC* considera todos los miembros de las superclases hasta que se alcanza la raíz (o raíces) del árbol de herencia. En la ecuación de arriba, $CC(c_2)$ incluye también partes heredadas de otras clases y c_1 puede incluir miembros tanto locales como heredados.

Simplifiquemos la ecuación anterior sumando y restando $CCL(c_1)$ en su lado derecho:

$$CC(c_2\ ISA\ c_1) = CC(c_2) + w_{CACI}\ CACL(c_1) + w_{CACI}\ CACI(c_1) + w_{CMICI}\ CMICL(c_1) +$$

$$+ w_{CMICI}\ CMICI(c_1) + w_{CI}\ CL(c_1) + w_{CI}\ CI(c_1) + w_{CACL}\ CACL(c_1) -$$

$$- w_{CACL}\ CACL(c_1) + w_{CMICL}\ CMICL(c_1) - w_{CMICL}\ CMICL(c_1) +$$

$$+ w_{CL}\ CL(c_1) - w_{CL}\ CL(c_1)$$

Queda:

$$CC(c_2\ ISA\ c_1) = CC(c_2) + CC(c_1) + w_{CACI}\ CACL(c_1) + w_{CMICI}\ CMICL(c_1) +$$

$$+ w_{CI}\ CL(c_1) - w_{CACL}\ CACL(c_1) - w_{CMICL}\ CMICL(c_1) - w_{CL}\ CL(c_1)$$

$$CC(c_2\ ISA\ c_1) = CC(c_2) + CC(c_1) - DCCL(c_1) \qquad (*)$$

En esta última ecuación:

$$DCCL(c_1) = [w_{CACL} - w_{CACI}]\ CACL(c) + [w_{CMICL} - w_{CMICI}]\ CMICL(c) + [w_{CL} - w_{CI}]\ CL(c)$$

Obviamente, el operador ISA no es en general aditivo para *CC*. Sí lo es, no obstante, cuando los pesos relacionados con los términos locales y heredados son idénticos, o bien cuando todos los pesos son iguales a 1; por tanto, ISA es aditivo para *CC1*:

$$CC(c_1\ ISA\ c_2) = CC1(c_1) + CC1(c_2)$$

El tercer término del segundo miembro de la ecuación indicada con (*) denota que la especialización reduce la medida conforme se heredan miembros. Otros trabajos han mostrado también resultados similares (por ejemplo, en Zuse, 1998). Sin embargo, sólo existe realmente disminución de la complejidad y el tamaño de la clase cuando los miembros son heredados y reutilizados sin más; no obstante, la sobrecarga y redefinición de métodos y atributos puede significar un aumento de estas propiedades.

4.4.2.5. Caracterización formal de la familia de métricas *CC*

En este epígrafe se presenta una caracterización formal de la familia de métricas *CC*. Su propósito es presentar y estudiar formalmente las propiedades de las métricas propuestas, así

como favorecer la aparición de un marco que permita analizar sus ventajas y desventajas con respecto a otras métricas para sistemas orientados a objeto.

4.4.2.5.1. Definiciones: sistemas, módulos, clases, miembros y elementos

De acuerdo al paradigma orientado a objeto, una clase es una colección de atributos de cierta clase, junto a los métodos que operan sobre ella. El interfaz de clase define el conjunto de servicios que pueden ser solicitados a las instancias de la clase y éstos constituyen la única manera de modificar el estado de las instancias de clases. Así, una clase *c* incluye atributos A_c y métodos M_c.

En general, los atributos de la clase no son las únicas instancias de clase utilizadas por los métodos de la clase, ya que en éstos pueden así mismo definirse objetos de otra clase para ser usados temporalmente en la realización de algún cálculo (por ejemplo, la clase *Persona* contiene sólo tres atributos de tipo *carácter*, pero uno de sus métodos utiliza una variable de clase *entero* como contador). En este contexto, llamaremos "atributos temporales" a este tipo de objetos, y nos referiremos a ellos mediante TA_c.

Con las consideraciones realizadas hasta este momento, una clase *c* puede entenderse como la siguiente tupla:

$c = (AL_c , AI_c , TA_c , ML_c , MI_c) = (A_c , TA_c , M_c)$

...en donde:

$A_c = (ALc , AI_c)$ y $M_c = (MLc , MI_c)$

En este contexto, podemos representar un sistema orientado a objetos *S* como una tripleta *S=(E, R, C)*, en la cual:

- *E* representa el conjunto de elementos de *S*, en el que incluimos atributos, métodos y atributos temporales: *E=(AL, AI, TA, ML, MI)*
- $R \acute{I} E \acute{} E$ es una relación binaria entre los elementos de *E*.
- $C= \{ c_1, c_2, ..., c_{|C|}\}$ es el conjunto de clases del sistema, siendo $c_i=(Ec_i, Rc_i)$.

Con esta definición, si *c* es una clase del sistema *S*, se verifica que:

- $c \hat{I} C$
- $E_c \acute{I} E$
- $R_c \acute{I} R$.

El conjunto *R* de relaciones está formado por relaciones ente atributos, métodos y atributos temporales. Más concretamente:

$$R = \bigcup_{i=1}^{16} R_i$$

Cada uno de los subconjuntos cuya unión compone *R* se define de la siguiente manera:

$R_1 = \{ (ml(c), al(c)) / ml(c) \text{ usa } al(c) \}$

R_2 = { (ml (c), ta (c)) / ml (c) usa ta (c) }

R_3 = { (ml(c), ai(c)) / ml(c) usa ai(c) }

R_4 = { (ml_1(c), ml_2(c)) / ml_1(c) usa ml_2(c) }

R_5 = { (ml(c), mi(c)) / ml(c) usa mi(c) }

R_6 = { (ml(c_1), al(c_2)) / ml(c_1) usa al(c_2) mediante un parámetro}

R_7 = { (ml(c_1), ta(c_1)) / ml(c_1) usa ta(c_2) mediante un parámetro}

R_8 = { (ml(c_1), ai(c_2)) / ml(c_1) usa ai(c_2) mediante un parámetro}

R_9 = { (ml(c_1), ml(c_2)) / ml(c_1) usa ml(c_2) mediante un parámetro}

R_{10} = { (ml(c_1), mi(c_2)) / ml(c_1) usa mi(c_2) mediante un parámetro}

R_{11} = { (al(c_1), al(c_2)) / al(c_1) hace referencia a al(c_2) }

R_{12} = { (al(c_1), ta(c_2)) / al(c_1) hace referencia a ta(c_2) }

R_{13} = { (al(c_1), ai(c_2)) / al(c_1) hace referencia a ai(c_2) }

R_{14} = { (ta(c_1), al(c_2)) / ta(c_1) hace referencia a al(c_2) }

R_{15} = { (ta(c_1), ta(c_2)) / ta(c_1) hace referencia a ta(c_2) }

R_{16} = { (ta(c_1), ai(c_2)) / tal(c_1) hace referencia a ai(c_2) }

...donde:

- *c* es una clase
- c_1 y c_2 pueden identificar la misma o diferentes clases.
- *a(c)* representa un atributo local o heredado de la clase *c*
- *m(c)* representa un atributo local o heredado de la clase *c*
- *al (c)* representa un atributo definido localmente de la clase *c*
- *ai(c)* representa un atributo de *c*, heredado
- *ta(c)* representa un atributo usado temporalmente en un método (por ejemplo, una variable auxiliar que se declara para realizar algún cálculo)
- *ml(c)* representa un método de la clase *c* definido localmente
- *mi(c)* representa un método de *c*, heredado

Entre las relaciones *R* algunas son interclase (R_1, R_2, R_4, R_6, R_7, R_9, R_{11}, R_{12}, R_{14} y R_{15}) y otras intraclase (R_3, R_5, R_8, R_{10}, R_{13} y R_{16}). Las relaciones mediante TA (R_2, R_7, R_{12}, R_{14}, R_{15} y R_{16}) modelan relaciones de cohesión “ocultas”, que raramente son consideras en los modelos orientados a objeto.

4.4.2.5.2. Validación teórica

Con la adopción del paradigma orientado a objeto, se ha dedicado un gran esfuerzo a hacer corresponder las propiedades de las métricas no orientadas a objeto y a la definición de

nuevos marcos (por ejemplo: Henderson-Sellers, 1996; Whitmire, 1997; Zuse, 1998). Sin embargo, no existe acuerdo sobre las propiedades necesarias ni sobre el modelo más susceptible de ser adoptado.

A continuación comprobamos la validez de la familia de métricas *CC* con respecto a diversas propiedades de diversos marcos. Sin embargo, las propiedades consideradas no han sido utilizadas con el ánimo de proponer un marco unificado de validación teórica de métricas para sistemas orientados a objeto, sino con el fin de caracterizar y discutir las métricas con respecto a propiedades bien conocidas y para estudiar su comportamiento.

Propiedades 1a y 1b: No negatividad.

La medida de un sistema *S=(E, R, C)* o de una clase es no negativa:

" S, m(S)³0, " c Î S, m(c)³0 ♦

De acuerdo con las discusiones realizadas en los epígrafes anteriores, esta propiedad es satisfecha por *CC1* tanto a nivel de sistema como de clase. En este caso, los operadores IPO e ISA son totalmente aditivos.

Con respecto a la métrica *CC*, la propiedad podría no ser satisfecha a nivel de clase, ya que podrían obtenerse valores negativos, dependiendo del signo de los factores de peso. No obstante, estudios empíricos demuestran que esta posibilidad es muy improbable, a pesar de que los valores negativos pueden significar, dependiendo del proyecto al que se aplique la métrica, un ahorro de esfuerzo debido, por ejemplo, a la reutilización. Además, este efecto podría ser evitado mediante la adición de una constante positiva que, por otro lado, tendría tal vez algunos efectos colaterales.

A nivel de sistema, la presencia de valores negativos es muy improbable para un sistema en desarrollo, aunque podrían aparecer en las primeras etapas.

Las discusiones realizadas respecto de esta propiedad son independientes del uso de las métricas funcionales *Vg'* o *LOC*.

Propiedades 2a, 2b, 2c: Valor nulo.

Según esta propiedad, la medida de un sistema *S=(E, R, C)* es cero si algún elemento de *S* es el conjunto vacío:

"S, (E=Æ Ú R=Æ Ú C=Æ) Þ *m(S)=0* ♦

Si un sistema no tiene clases (*C=Æ*), entonces *E=Æ*, lo cual implica que no existen relaciones entre los elementos del sistema (*R=Æ)*. Cuando *R=Æ*, cada subconjunto de *R* es vacío.

Para *CC* y *CC1*, cuando $R_6=R_7=R_8=R_9=R_{10}=\varnothing$, entonces los métodos de la clase no presentan parámetros, con lo que *CMICL* y *CMICI* son nulos. Los términos *CL* y *CI* de *CC* están influidos por varios subconjuntos de *R* que consideran la parte funcional del sistema. Si *R=Æ*, entonces los métodos de las clases del sistema no pueden utilizar ni objetos de ninguna clase ni otros métodos en su código. Si se cumplen estas condiciones, entonces es imposible escribir

un método *m* no vacío en cualquier clase *c* tal que *CL(c)>0* o *CI(c)>0*. En efecto, supongamos un método en una clase con una sentencia nula, como la siguiente:

```
void ClassA :: do_nothing(void)
begin
    ALibraryCall();
end
```

En este caso, su valor correspondiente de *Vg'* es cero, con lo cual esta clase de métodos no incrementa los valores de *CL* o *CI*. Incluso los métodos de secuencias nulas tienen *Vg'=0*. Observemos, por otro lado, el siguiente ejemplo:

```
void ClassA :: do_something(void)
begin
    if (true) then
          AnotherLibraryCall("")
    else
          ALibraryCall();
end
```

En este caso, *Vg'=1*, pero este método no satisface la condición de que *R=Æ*, ya que presenta dos relaciones: la primera con un objeto de la clase booleano y la segunda con una instancia de la clase cadena (aunque sea vacía). Ambas relaciones están incluidas en R_2.

Bajo estas condiciones, por tanto, las construcciones secuenciales producen *Vg'=0*, mientras que los constructores de selección e iteración no pueden ser escritos, con lo que $CL_{Vg'}$= $CI_{Vg'}$*=0.*

Si se usa *Vg'*, los términos *CACI* y *CACL* son cero si *R=Æ*, ya que para las demostraciones de arriba los otros términos de *CC* son nulos, de manera que las clases no presentan complejidad funcional. Esto significa que esta propiedad es satisfecha por $CC_{Vg'}$ y por $CC1_{Vg'}$.

Cuando se utiliza *LOC* como métrica funcional, entonces *CL* y *CI* es diferente de cero incluso aunque todas las relaciones sean vacías. Esto significa que ni CC_{LOC} ni $CC1_{LOC'}$ cumplen esta propiedad.

Cuando *E=Æ*, *R=Æ* ya que no se pueden definir relaciones puesto que no hay elementos en el sistema, con lo que las dos versiones de *CC* y *CC1* cumplirían la propiedad.

Propiedad 3: Antisimetría.

La medida de un sistema *S=(E, R, C)* depende de la dirección de las relaciones entre sus elementos:

$ S=(E, R, C) / S^{-1}=(E, R^{-1}, C) Þ m(S)¹ m(S^{-1}) ♦

Esta propiedad debe ser satisfecha por nuestras métricas de complejidad a nivel de sistema, ya que los operadores IPO e ISA no son conmutativos. A nivel de clase, la propiedad es discutida en la Propiedad 11 (conmutatividad).

Propiedad 4: Adtividad a nivel de sistema.

La medida de un sistema *S=(E, R, C)* no es menor que la suma de las medidas de dos cualesquiera de sus clases:

" c_1, c_2 Î S, m(S) ³ m(c_1) + m(c_2) ♦

En nuestro caso, es evidente que:

$$CC(S)=\sum_{i=1}^{|C|}CC(c_i)=CC(c_1)+CC(c_2)+...+CC(c_n)\geq CC(c_1)+CC(c_2)\geq CC(c_1)+CC(c_2)$$

Esta propiedad la satisfacen las dos versiones (con *LOC* y *Vg'*) de *CC* y *CC1*.

Propiedad 5a: a nivel de clase, la medida no debe ser muy holgada.

Existen clases con diferentes valores de medida:

$ c_1, c_2 / m(c_1)¹ m(c_2) ♦

Esta propiedad es significativa tanto para complejidad como para tamaño. Su origen se encuentra en la teoría de la medida, y su cumplimiento evita la definición de métricas triviales.

Como se discute en Weyuker (1998), *Vg'* es demasiado holgada, ya que puede obtenerse el mismo valor de la métrica para diferentes grafos que tal vez presenten un número muy diferente de líneas de código. Desde este punto de vista, $CC_{Vg'}$ y $CC1_{Vg'}$ pueden considerarse más holgadas que CC_{LOC} y $CC1_{LOC}$.

A nivel de sistema, esta propiedad ofrece los mismos resultados:

Propiedad 5b: a nivel de sistema, la medida no debe ser muy holgada.

Existen sistemas con diferentes valores de medida:

$ S_1, S_2 / m(S_1)¹ m(S_2) ♦

Propiedad 6a: a nivel de clase, la medida no debe ser muy fina.

Una medida de clase debe permitir la existencia de clases diferentes con el mismo valor de la métrica:

$ c_1, c_2 / c_1 ¹ c_2 ^ m(c_1) = m(c_2) ♦

Cualquier versión de *CC* o *CC1* cumple esta propiedad, pues existe una probabilidad no nula de que *m(c_1) = m(c_2)*, siendo c_1 y c_2 clases diferentes. A nivel de sistema, la propiedad también puede ser enunciada y produce los mismos resultados:

Propiedad 6b: a nivel de sistema, la medida no debe ser muy fina.

Una medida de sistema debe permitir la existencia de sistemas diferentes con el mismo valor de la métrica:

$ S_1, S_2 / S_1 ¹ S_2 ^ m(S_1) = m(S_2) ♦

Propiedad 7a: funcionalidad.

Existen clases con la misma funcionalidad y con diferentes valores de medida:

$\$ c_1, c_2 / c_1 \neq c_2 \wedge m(c_1) \neq m(c_2)$ ♦

Si dos clases presentan la misma funcionalidad, significa que han sido implementadas para interactuar con otras clases y elementos de manera equivalente. En general, esto no significa que tengan idénticos interfaz e implementación de sus métodos, con lo que existe una probabilidad no nula de que $m(c_1) \neq m(c_2)$.

Tanto *CC* como *CC1* satisfacen esta propiedad en cualquiera de sus versiones.

Propiedad 7b: estructura de la información.

Existen clases con la misma estructura de información y con diferentes valores de medida:

$\$ c_1, c_2 / c_1 \gg c_2 \wedge m(c_1) \neq m(c_2)$ ♦

Si dos clases presentan la misma estructura de información, significa que han sido diseñadas para modelar entidades similares, pero esto no implica que tengan idénticos interfaz e implementación de sus métodos, con lo que existe una probabilidad no nula de que $m(c_1) \neq m(c_2)$.

Tanto *CC* como *CC1* satisfacen esta propiedad en cualquiera de sus versiones.

Propiedad 8a: positividad débil del operador IPO.

$\forall c_1, c_2, m(c_1) \pounds m(c_2 \text{ IPO } c_1) \wedge m(c_2) \pounds m(c_1 \text{ IPO } c_2)$ ♦

En los casos de *CC* y *CC1*, los términos w_{CACL}, $CC(c_1)$ y $CC1(c_2)$ son positivos, con lo que esta propiedad es cumplida por cualquier versión de *CC* o *CC1*.

Propiedad 8b: positividad débil del operador ISA.

$\forall c_1, c_2, m(c_1) \pounds m(c_2 \text{ ISA } c_1) \wedge m(c_2) \pounds m(c_1 \text{ ISA } c_2)$ ♦

En la ecuación que marcamos con (*) (página 166), los términos $CC(c_2)$ y $CC(c_1)$-$DCCL(c_1)$ son positivos, por lo que ambas versiones de *CC* y *CC1* cumplen esta propiedad.

Propiedad 9a: monotonía débil del operador IPO.

La contribución mediante composición por el operador IPO influye igualmente en entidades que tienen la misma medida:

$\forall p_1, p_2, p_3, m(p_1)=m(p_2) \wedge m(p_1 \text{ IPO } p_3) = m(p_2 \text{ IPO } p_3)$

y

$\forall p_1, p_2, p_3, m(p_1)=m(p_2) \wedge m(p_3 \text{ IPO } p_1) = m(p_3 \text{ IPO } p_2)$ ♦

Tanto *CC* como *CC1* cumplen esta propiedad. En efecto:

$CC(c_3 \text{ IPO } c_1) = CC(c_1) + w_{CACL}\ CC1(c_3)$

$CC(c_3\ IPO\ c_2) = CC(c_2) + w_{CACL}\ CC1(c_3)$

$CC(c_1\ IPO\ c_3) = CC(c_3) + w_{CACL}\ CC1(c_1)$

$CC(c_2\ IPO\ c_3) = CC(c_3) + w_{CACL}\ CC1(c_2)$

Con lo cual:

$\forall\ c_1, c_2, c_3, CC(c_1)=CC(c_2) \wedge CC(c_1\ IPO\ c_3) = CC(c_2\ IPO\ c_3)$

$\forall\ c_1, c_2, c_3, CC(c_1)=CC(c_2) \wedge CC(c_3\ IPO\ c_1) = CC(c_3\ IPO\ c_2)$

Propiedad 9b: monotonía débil del operador ISA.

La contribución mediante composición por el operador ISA influye igualmente en entidades que tienen la misma medida:

$\forall\ p_1, p_2, p_3, m(p_1)=m(p_2) \wedge m(p_1\ ISA\ p_3) = m(p_2\ ISA\ p_3)$

y

$\forall\ p_1, p_2, p_3, m(p_1)=m(p_2) \wedge m(p_3\ ISA\ p_1) = m(p_3\ ISA\ p_2)$ ♦

Consideremos la ecuación (*) (página 166):

$CC(c_3\ ISA\ c_1)= CC(c_3) + CC(c_1) - D\ CCL(c_1)$

$CC(c_3\ ISA\ c_2)= CC(c_3) + CC(c_2) - D\ CCL(c_2)$

$CC(c_2\ ISA\ c_3)= CC(c_2) + CC(c_3) - D\ CCL(c_3)$

$CC(c_1\ ISA\ c_3)= CC(c_1) + CC(c_3) - D\ CCL(c_3)$

Por tanto, ambas consecuencias de esta propiedad son cumplidas:

$\forall\ c_1, c_2, c_3, CC(c_1)=CC(c_2) \wedge CC(c_3\ ISA\ c_1) = CC(c_3\ ISA\ c_1)$

$\forall\ c_1, c_2, c_3, CC(c_1)=CC(c_2) \wedge CC(c_1\ ISA\ c_3) = CC(c_2\ ISA\ c_3)$

Propiedad 10a: permutación de sentencias.

Una medida debe ser insensible a la permutación de sentencias. ♦

Cuando la métrica utiliza *Vg'*, el orden de las sentencias no influye en su valor, ya que esta métrica no es sensible al orden de las sentencias (McCabe, 1976; Weyuker, 1988); del mismo modo, tampoco *LOC* se ve influido por él. Por tanto, *CC* y *CC1* satisfacen en ambas versiones esta propiedad.

Propiedad 10b: permutación de declaraciones.

Una medida debe ser insensible a la permutación de declaraciones. ♦

Esta propiedad es satisfecha por ambas versiones de *CC* y *CC1*.

Propiedad 11: no conmutatividad.

La medida del sistema depende del orden en el cual se realiza la composición:

$\$ c_1, c_2 / m(c_1 ; c_2) \neq m(c_2 ; c_1)$ ♦

El símbolo ";" denota a los operadores IPO e ISA.

Esta propiedad es satisfecha por *CC* ya que los operadores IPO e ISA son no conmutativos cuando los pesos son diferentes de 1:

$CC(c_1 \ IPO \ c_2) \neq CC(c_2 \ IPO \ c_1)$, ya que:

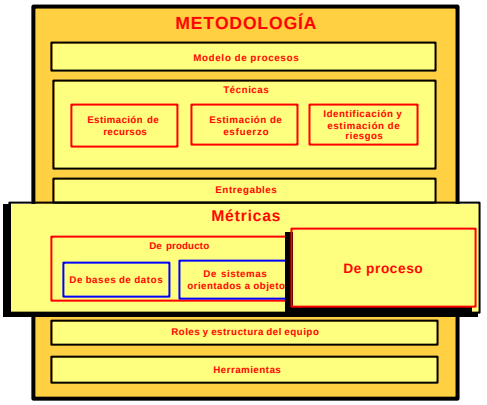
$$CC(c_1 \ IPO \ c_2) = CC(c_2) + w_{CACL} \ CC1(c_1)$$

y: $CC(c_2 \ IPO \ c_1) = CC(_1) + w_{CACL} \ CC1(c_2)$

4.4.3. MÉTRICAS DE PROCESO PARA EL MANTENIMIENTO.

Por lo general, las métricas de proceso pretenden cubrir los siguientes objetivos:

- Controlar los procesos, recursos, actividades, tareas, etc. ya realizados
- Controlar los procesos, actividades, tareas, etc. que se están ejecutando actualmente
- Estimar los costes de futuros procesos, intervenciones, etc.



En esta sección presentamos métricas para los tres puntos mencionados (aunque agrupamos en uno solo los dos primeros), y añadimos otro conjunto de métricas (cuyos valores llamamos *indicadores de servicio*), que permitirán controlar si el mantenimiento se está llevando a cabo según los términos inicialmente planificados.

4.4.3.1. Métricas para la planificación del proceso: indicadores de servicio

Utilizable en:	Mantenimiento planificable (correctivo no urgente y perfectivo)	Actividad I.0. *Estudio inicial*	Tarea I0.2 *Preparar propuesta de mantenimiento*

Los valores de estas métricas (*indicadores de servicio*) se pactan al principio del proceso de mantenimiento entre la Organización de mantenimiento y el Cliente. Sirven para controlar si el proceso se está ejecutando conforme a lo planificado.

- Número máximo de intervenciones atendibles mensualmente (o de otro periodo pactado) de correctivos urgente y no urgente (*NMaxiCU* y *NMaxiCNU*).
- Tiempo de resolución de intervenciones de correctivo urgente (*TriCU*) y no urgente (*TriCNU*). Es el tiempo máximo que la Organización de mantenimiento adquiere como compromiso para resolver hasta *NMaxiCU* intervenciones de correctivo urgente o *NMa-*

xiCNU de no urgente. Dependiendo de lo pactado en el contrato, la Organización de mantenimiento puede incurrir en penalizaciones en caso de superar este tiempo.

- Desviación semanal (u otro periodo, inferior al mencionado en los indicadores *NMaxiCU* y *NMaxiCNU*) máxima para correctivos urgente y no urgente (*DesvCU* y *DesvCNU*). Si se pacta por ejemplo, un valor para *NMaxiCU* mensual de 15 intervenciones, el valor de *DesvCU* representará el número máximo de intervenciones de correctivo urgente que pueden concentrarse en una semana, con compromiso, por parte de la Organización de mantenimiento, de ser atendidas en el tiempo *TriCU*. En otras palabras: la Organización de mantenimiento se compromete a servir en tiempos razonables un cierto número de intervenciones urgentes al mes, pero puede quedar liberada de este compromiso si todas las peticiones (o un número muy elevado de ellas) se concentran en un espacio de tiempo muy pequeño.

De este modo, *DesvCU* y *DesvCNU* representan el número máximo semanal de peticiones de correctivos urgente y no urgente atendibles en los tiempo *TriCU* y *TriCNU*.

4.4.3.2. Métricas para el control del proceso en ejecución y de procesos terminados

Las siguientes métricas ayudan a controlar tanto los procesos ya terminados como los que se encuentran en ejecución.

- Respeto a la planificación del periodo (*ResPlanPer*), que es el porcentaje de tiempo dedicado a ejecutar intervenciones, con respecto al tiempo planificado, para un periodo determinado:

$$\text{ResPlanPer} = \frac{\text{Horas dedicadas durante el periodo}}{\text{Horas planificadas para el periodo}}$$

Cuando vale más de 1, este ratio está denotando que la Organización de mantenimiento ha realizado horas “extras” durante el periodo considerado; valores menores que 1 significan que ha sido necesario dedicar menos tiempo a mantenimiento que el inicialmente planificado.

En los casos de mantenimiento correctivo (urgente y no urgente), los tiempos planificados por periodo vienen dados, respectivamente, por las expresiones *TriCU x NMaxiCU* y *TriCNU x NMaxiCNU*. Para estos dos tipos de mantenimiento, es deseable que tanto el número de horas dedicadas a servir intervenciones como el de horas planificadas sean decrecientes (es decir, que disminuya el número necesario de horas para ambos correctivos). Por lo general, la Organización de mantenimiento recoge su compromiso de alcanzar esta tendencia en el *Plan de mantenimiento del periodo* (DOC17), que puede igualmente ser recogido en el contrato de prestación del servicio, cuando hay externalización.

Además, el compromiso de disminución de correctivo permite abaratar los costes previstos por el Cliente para este tipo de mantenimiento. Así, por ejemplo, en la Figura 28 mostramos el número de horas dedicadas semanalmente a resolver peticiones de mantenimiento de correcti-

vo urgente en un determinado proyecto. Inicialmente, la Organización de mantenimiento planificó 10 horas diarias (70 semanales) para este mantenimiento. El compromiso de realizar preventivo mientras hace correctivo invita a revisar, cada cierto tiempo, el tiempo planificado. En la figura, parece claro que a partir de las semanas 11-12, el número de horas planificadas podría haber sido revisado a la baja, del mismo modo que debería haberse planificado un número de horas superior para las primeras semanas.

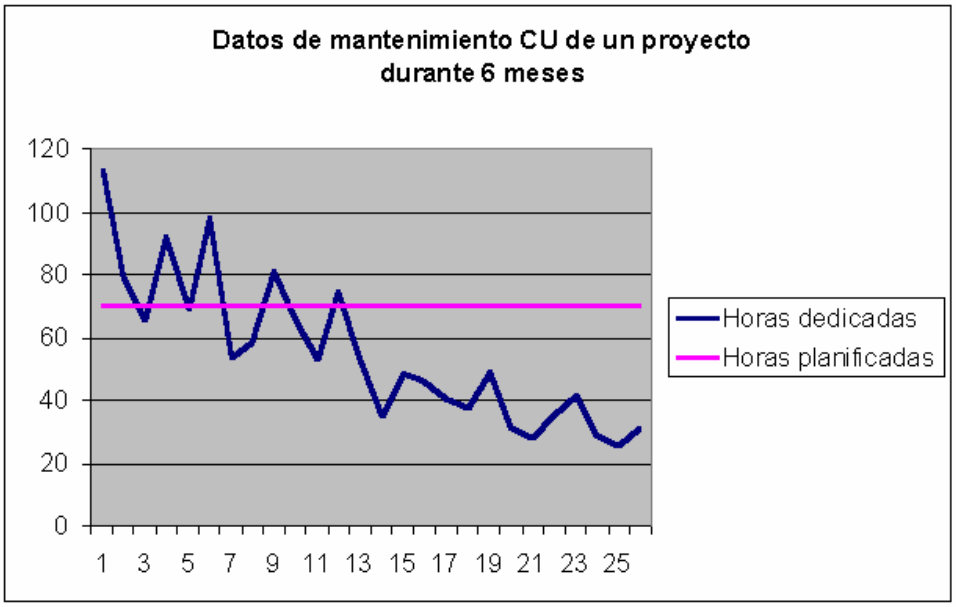


Figura 28. Horas dedicadas y planificadas en un proyecto.

- Número de horas replanificadas en cada periodo, que debiera ser decreciente
- Tiempo medio de respuesta a las peticiones, que debiera ser decreciente.
- Porcentaje de anomalías de cada tipo (urgentes y no urgentes). Nótese que una solicitud de mantenimiento perfectivo, preventivo o adaptativo no se considera una anomalía.
- Evolución de la complejidad y de otras métricas de producto. Es deseable que la complejidad de los programas y bases de datos decrezca a medida que se interviene sobre los productos, lo cual es otro resultado del compromiso de mantenimiento preventivo adquirido por la organización. Puede considerarse la evolución de las métricas de producto de módulos o bases de datos completas, o de rutinas y tablas individualmente. En la Figura 29 mostramos la evolución de la complejidad de la rutina *mnuDVerGraficoDeComponente_click*, perteneciente al código de MANTOOL, que sufrió cuatro modificaciones del 20 al 23 de agosto.

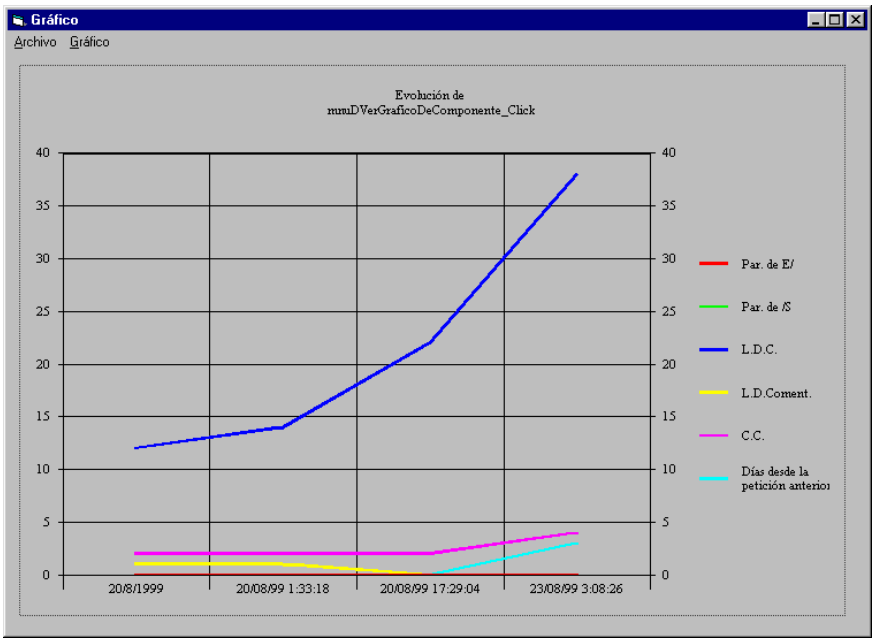


Figura 29. Evolución de diferentes métricas de un componente.

Se observa la tendencia creciente del tamaño del componente en líneas de código, pero un crecimiento mucho más suave de la complejidad. Igualmente, se observa cómo se le quitaron las pocas líneas de comentario que tenía.

- Número de puntos-función (u otras métricas de producto) replanificados en un periodo.
- Número de puntos-función (u otras métricas de producto) servidos por cada tipo de mantenimiento.
- Tiempo de dedicación del personal de mantenimiento.

5 APLICACIÓN EN CASOS PRÁCTICOS

En este capítulo se presenta la validación del proceso metodológico de mantenimiento descrito, mediante la exposición de los casos reales en los que se ha aplicado la metodología y algunas de los beneficios obtenidos de su utilización. Es importante recordar que la validación de las técnicas y métricas ya se ha presentado en el Capítulo cuarto.

5.1. CASOS DE APLICACIÓN

La metodología MANTEMA ha sido aplicada por Atos ODS en los siguientes cuatro proyectos de una importante corporación bancaria española:

- Recaudación de impuestos.
- Domiciliaciones.
- Transferencias.
- Riesgo proactivo.

Además, tras la primera implementación de MANTOOL (la herramienta de soporte a la metodología, que se presenta en el siguiente capítulo), MANTEMA se ha aplicado al propio mantenimiento de MANTOOL.

5.1.1. RECAUDACIÓN DE IMPUESTOS

Esta aplicación entró en producción en el mes de mayo de 1992. Consta de 135 programas con 108.331 líneas de código, implementados en Cobol, Cobol-DB2, Cobol-CICS y Cobol-CICS-DB2.

Las funciones principales de esta aplicación son:

- Gestionar el cobro de impuestos para Hacienda y Comunidades Autónomas.
- Gestionar el cobro de tasas estatales y organismos autónomos.
- Recoger la información de las devoluciones de impuestos para enviársela a Hacienda.
- Realizar embargos sobre cuentas de personas tributarias como consecuencia del impago de un impuesto.

5.1.2. DOMICILIACIONES

Entró en producción en junio de 1992. Consta de 196 programas con 157.281 líneas de código, en los mismos lenguajes que la aplicación anterior.

La aplicación de Domiciliaciones automatiza la gestión de cobro de órdenes de adeudo como consecuencia de domiciliaciones que llegan a la aplicación en ficheros mediante soporte magnético, y del intercambio entre entidades de adeudos domiciliados.

5.1.3. TRANSFERENCIAS

Entró en producción en abril de 1992, constando de 308 programas con 247.156 líneas de código, estando implementado en los mismos lenguajes que las dos aplicaciones anteriores.

La aplicación se encarga de las transferencias entre cuentas bancarias, distinguiéndose dos tipos de transferencias, las periódicas y las unitarias.

5.1.4. RIESGO PROACTIVO

El proyecto se planteó en junio de 1993 con objeto de explotar la información existente de la amplia base de clientela y automatizar la concesión de créditos. Se desarrolla a lo largo de 1994 y 1995, poniéndose en operación en noviembre de 1995 sobre un IBM 9000 modelo 982 en entorno MVS-TSO.

Desde su puesta en operación se ha ido completando con nuevas funcionalidades y tiene relación estrecha con múltiples aplicaciones de las que obtiene datos, como Cuentas Personales, Depósitos, Transferencias, Domiciliaciones, Tarjetas de Crédito o Préstamos. Igualmente, suministra datos a las aplicaciones Concesión Automática de Riesgo y Anticipación del Riesgo.

Consta de 576 programas, de los cuales 235 tienen acceso a DB2 y 94 transacciones “on-line” que ejecutan 227 programas distintos.

El proyecto cubre la siguientes funciones básicas:

- Evaluación automática de toda la clientela una vez al mes.
- Asignación de reglas de clasificación por perfiles de cliente que permitan la concesión y renovación automática de operaciones.
- Gestión diaria de las acciones de recobro a efectuar sobre posiciones vencidas o dudosas.
- Diseño de criterios de selección de clientes para su inclusión en campañas bajo condiciones diferenciadoras.
- Simulación del impacto de la oferta antes de lanzar una campaña.
- Ejecución automatizada de una campaña incluyendo filtros y exclusiones de red.
- Planificación de acciones de márketing periódicas o aperiódicas.
- Base de datos histórica de las campañas a nivel cliente.

5.2. TIPOS DE MANTENIMIENTO EN LA EXPERIENCIA REAL

En los cuatro proyectos presentados se sirvieron, conforme a la metodología, peticiones de mantenimiento pertenecientes a los tipos planificable y no planificable.

5.3. DOCUMENTACIÓN

Durante la aplicación de la metodología en los cuatro casos descritos, se han cumplimentado todos los documentos correspondientes a las diferentes peticiones de modificación servidas. Los modelos de documentos de MANTEMA fueron en general bien aceptados por los usuarios de la metodología, aunque fue preciso realizar modificaciones en algunos de ellos. La *Tabla de factores de riesgo* fue sin duda la más afectada, ya que sufrió una completa reconstrucción: el procedimiento de cumplimentación de este documento era, en un principio, demasiado engorroso, dificultaba su utilización y facilitaba la comisión de errores durante su rellenado. La siguiente figura muestra el diseño de la primera versión de este documento, que constaba además de cuatro páginas del mismo diseño que la reproducida. Como se observa, debían puntuarse en dos columnas diferentes y de distinta manera los factores considerados como positivos y negativos desde el punto de vista del proceso de mantenimiento.

DOC4	FACTORES DE RIESGO	
Aplicación: RIESGO PROACTIVO		
Preparado por : [illegible] Fecha : Junio 1.999	Aprobado por : M. P. V. Fecha : Julio 1.999	

Factores externos	Dependencia de cambios ajenos a la organización	3.1		2
	Dependencia de subcontratistas	3.2		4
Factores del SI	Complejidad de la base de datos	4.1		1
	Complejidad de los procesos de negocio	4.2		4
	Estabilidad de los procesos de negocio	4.3	2	
	Estabilidad de la información	4.4	4	
	Nivel de normalización del sistema	4.5	4	
	Número de replicaciones del sistema informático (excluyendo tolerancia a fallos, etc.)	4.6		3
Factores críticos de la aplicación	Criticidad de la aplicación	5.1		2
	Número de interfaces con otras aplicaciones del SI	5.2		5
	Incidencia de los errores en la imagen de la compañía	5.3		4
	Existencia de momentos críticos (final de mes, p. ej.)	5.4		3
	Existencia de puntas periódicas que necesiten incorporación del personal	5.5		4
	Existencia de puntas no periódicas	5.6		3
	Vulnerabilidad del aplicativo por parte del usuario (actualizaciones directas de la BD, p. ej.)	5.7		1
	Grado de dependencia del usuario para conocer el problema	5.8		3

DOC4 2 de 1

Nota: rellenar unicamente las casillas no sombreadas.

Figura 30. Primera versión de la Tabla de riesgos.

Para facilitar su cumplimentación, se modificó la lista de factores circunstanciales, redactando, para cada uno, una frase en tono positivo (por ejemplo, en la Tabla 15: *La calidad de la documentación del software es alta*), en lugar de mezclar, en la misma tabla, factores positivos y negativos que debían ser puntuados de diferente manera para lograr resultados coherentes.

Otros documentos, sin embargo, han requerido pocas modificaciones y han sido válidos o casi válidos desde el principio. Un ejemplo de estos documentos es el DOC6 (Petición de modi-

ficación), uno de los cuales -el 281, correspondiente a la aplicación “Riesgo proactivo”- reproducimos en la Figura 31.

5.4. LECCIONES APRENDIDAS

Además de otras *lecciones*, filtradas y reabsorbidas por la aplicación del método Investigación en acción, la aplicación en casos reales de la *porción metodológica* de esta tesis doctoral mostró la conveniencia de:

- Distinguir únicamente dos tipos de mantenimiento (planificable y no planificable), en lugar de los cinco que se manejaron en un principio. A lo largo del proyecto MANTEMA, se construyeron diferentes guías técnicas de mantenimiento, en cada una de las cuales se iba refinando la anterior y añadiendo nuevas características (véase Tabla 2).
- Modificar ciertos documentos, eliminando los epígrafes correspondientes a informaciones que nunca eran recogidas y añadiendo epígrafes para informaciones de interés para las que no existía espacio.
- Como ya se ha indicado, fue preciso reformar completamente la tabla de factores de riesgo que, además, se completó con nuevos factores circunstanciales y se dividió en dos tablas distintas pero complementarias (véase “Identificación y estimación de riesgos” en la sección 4.3.1, página 127).
- La aplicación manual de la metodología es engorrosa y es preciso disponer de un entorno automático para aplicarla. Por ello se ha implementado MANTOOL, que se presenta en el capítulo siguiente, y se está adaptando MANUTEN CONDUCT, una herramienta desarrollada y utilizada por Atos ODS para la gestión del mantenimiento.

5.5. VENTAJAS OBTENIDAS DE LA METODOLOGÍA MANTEMA

Una de las ventajas obtenidas por Atos ODS de la utilización de MANTEMA es la facilidad y el bajo costo con que su personal puede ser intercambiado de proyecto, lo cual se logra gracias a la definición de los roles y estructura de la organización (sección 4.1.4, página 97) y a la actualidad de la documentación del software. A su vez, y entre otros factores, en esto influyen la clara estructuración del proceso (secciones 4.1.1, 4.1.5 y 4.2), la definición de las plantillas de documentos (Apéndice I, página 219) y la especificación detallada de los diferentes elementos de entrada y salida de cada tarea (sección 4.2, página 100 y siguientes).

Económicamente, la posesión y utilización de la metodología ha jugado un papel decisivo en la reciente concesión a Atos ODS de un importante contrato de mantenimiento con el Banco de España.

Figura 31. Las dos páginas de una Petición de modificación, rellenada durante la aplicación de la metodología.

6 IMPLEMENTACIÓN DEL PRODUCTO

6.1. INTRODUCCIÓN.

En este capítulo presentamos las características que ofrece MANTOOL para gestionar el proceso de mantenimiento según la metodología descrita en esta tesis. MANTOOL es un prototipo de herramienta *horizontal*, utilizable durante todo el proceso, con algunos complementos *verticales*, susceptibles de ser utilizados en algunos entornos. Como ya mencionamos en la sección 3.4, existen pocas herramientas de gestión para el proceso de mantenimiento (Pigoski, 1997). Ilustraremos las características de MANTOOL con algunos ejemplos, obtenidos mientras se aplicaba MANTOOL a su propio mantenimiento.

En la Figura 32 mostramos la pantalla principal del programa, con el menú de introducción de nueva información totalmente desplegado:

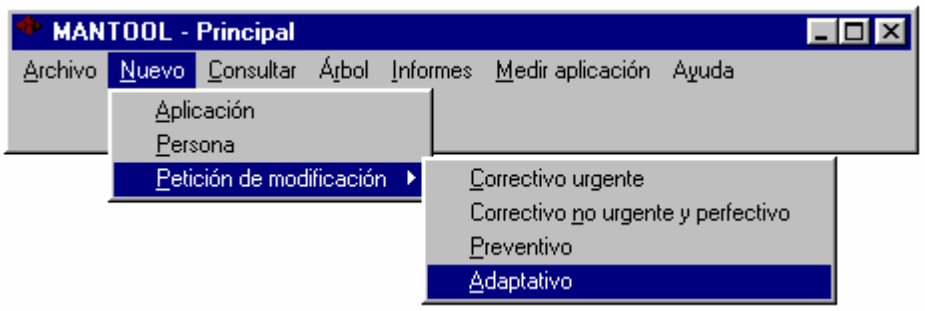


Figura 32. Menú principal de MANTOOL.

La pantalla para la introducción de nuevas peticiones se muestra en la Figura 33. El lugar de presentación de la petición (*Florencia*, en este ejemplo), así como otras informaciones habituales, están almacenados en un fichero de configuración.

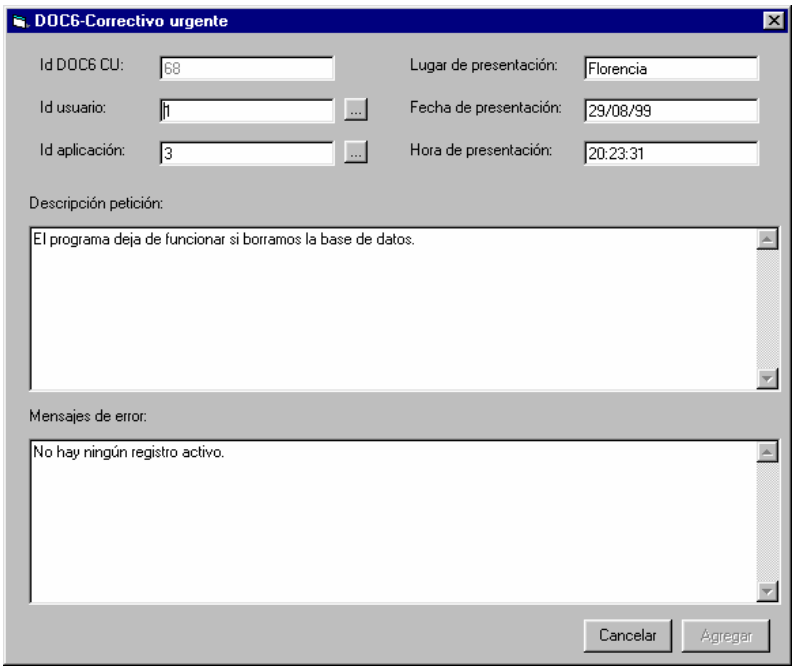


Figura 33. Pantalla para la introducción de nuevas peticiones.

Tenemos tres formas de consultar una petición determinada:

- Introduciendo o seleccionando su número en la ventana mostrada en la Figura 34.

Figura 34. Pantalla habitual para seleccionar una petición de modificación.

- Si sabemos de qué tipo de mantenimiento se trata, podemos seleccionarla en la ventana de la Figura 35.

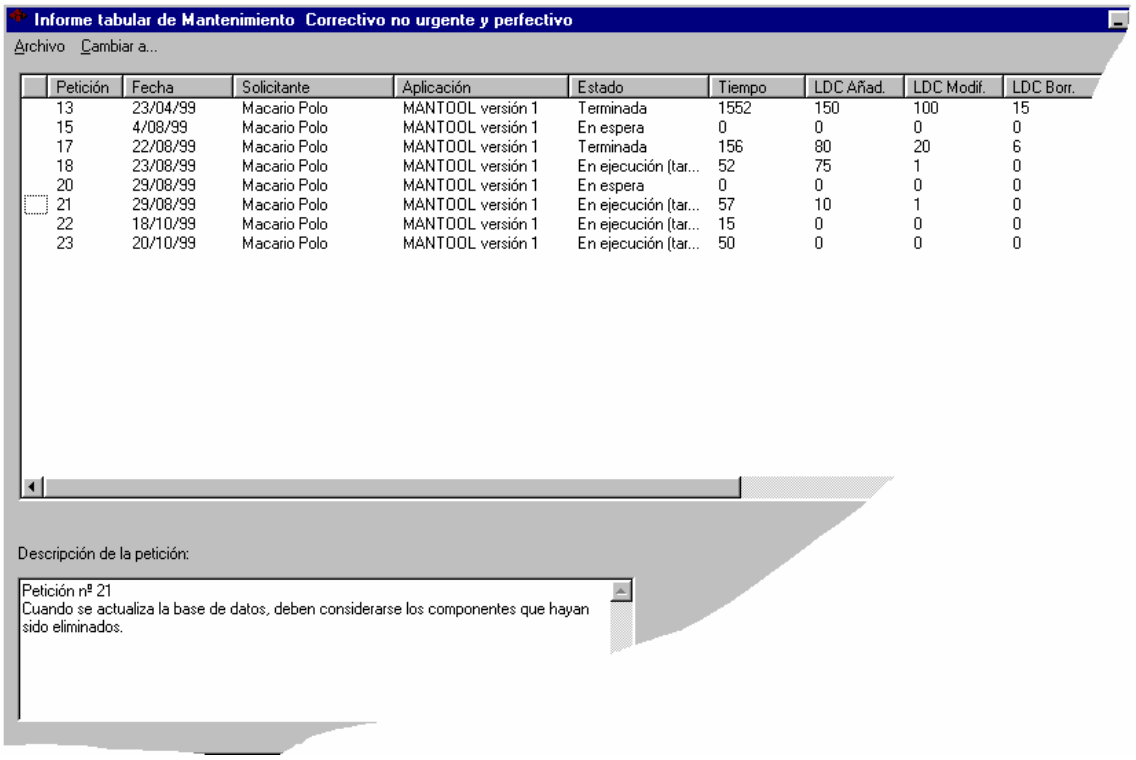


Figura 35. Informe tabular.

- Si preferimos buscarla por la aplicación a la que se refiere, podemos seleccionarla en la Figura 36, que muestra las peticiones de modificación relacionadas con una aplicación. Desde esta pantalla podemos acceder a información relacionada con el nodo en que nos encontremos.

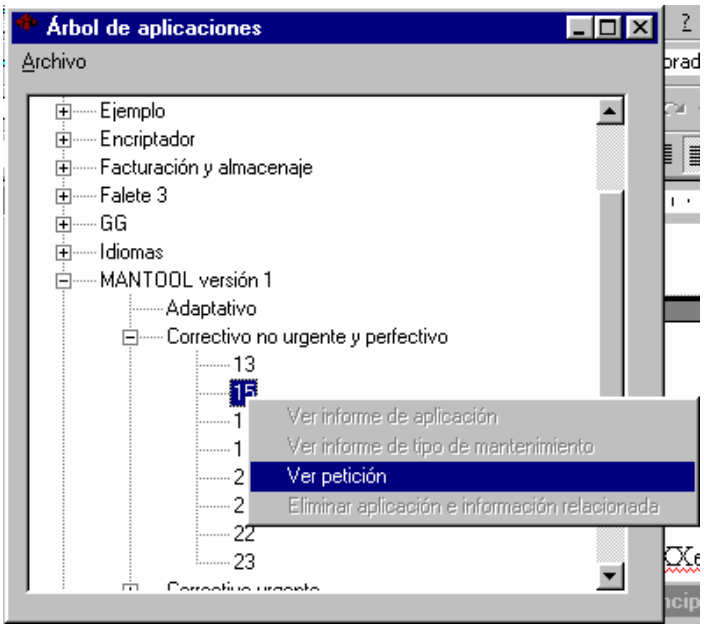


Figura 36. Árbol de aplicaciones.

6.2. ESTADO DE UNA PETICIÓN.

En la Figura 37 mostramos la pantalla utilizada para gestionar las peticiones de modificación. Cada nodo del grafo mostrado en la parte superior representa cada tarea de la petición de mantenimiento, con lo que hay un grafo diferente para cada tipo: correctivo urgente, correctivo no urgente y perfectivo (estos dos comparten el mismo conjunto de tareas), preventivo y adaptativo. El color del nodo asociado a cada tarea pasa de amarillo a rojo, a medida que se van ejecutando las tareas de cada petición.

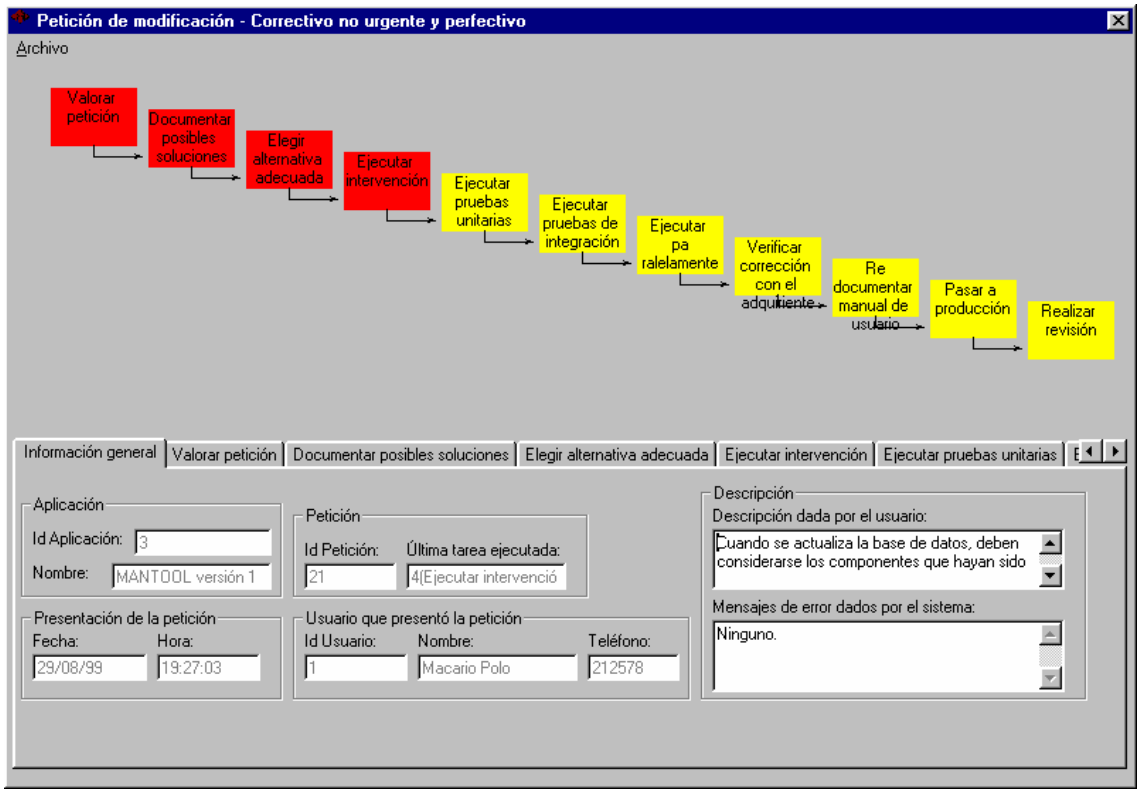


Figura 37. Ventana para la gestión de cada petición.

En la parte inferior hay tantas solapas como tareas tenga el tipo de mantenimiento, más una solapa adicional (la mostrada en la figura anterior), en la que se presenta cierta información general de la petición.

La información contenida en cada solapa es la información relacionada con la tarea a la que representa (documentos, fechas, métricas, etc.). En la siguiente figura mostramos la información de la tarea *Ejecutar intervención*, correspondiente a mantenimiento *Correctivo no urgente y perfectivo*, de la petición cuya información general veíamos en la figura anterior.

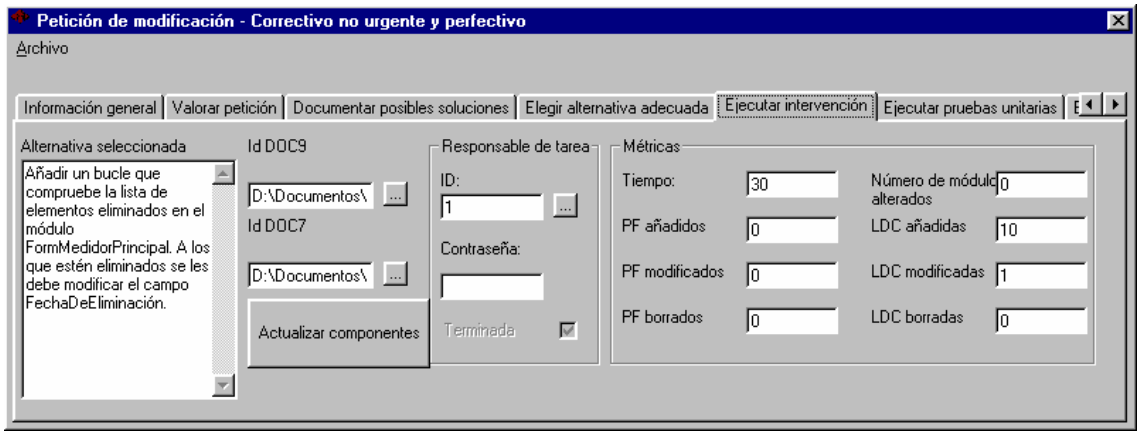


Figura 38. Información de una tarea concreta de una petición de mantenimiento.

Las referencias a los documentos generados en cada tarea corresponden a identificadores únicos de cada documento. En la Figura anterior, por ejemplo, se almacenan las rutas de acceso a los documentos *DOC7* y *DOC9* correspondientes a esta tarea.

6.3. UN COMPONENTE PARA AYUDAR A LA MEDICIÓN DEL IMPACTO DE LOS CAMBIOS.

MANTOOL incorpora un componente *vertical* que puede utilizarse para la medición de atributos de código fuente en Visual Basic 5. Se ha seleccionado este lenguaje porque es el que se ha utilizado para implementar MANTOOL y, como hemos dicho, se ha utilizado MANTOOL durante su propio mantenimiento. Además, el editor de código de Visual Basic va dando formato al código automáticamente a medida que se escribe, lo que facilita enormemente su medición, ya que no hay que realizar prácticamente operaciones de preproceso.

Para cada módulo de un programa, este componente mide las siguientes métricas a nivel de rutina:

- Tamaño de la rutina en líneas de código (LDC).
- Número de líneas de comentario (LDCCom).
- Complejidad ciclomática, calculada como el número de decisiones (*if, case, for, while, do until*) de la rutina.
- Número de parámetros de entrada y de salida de cada rutina.

Todos estos valores pueden totalizarse por módulos y por aplicación.

Este componente de medición se utiliza principalmente en dos ocasiones:

- Cuando se da de alta una aplicación nueva de cuyo mantenimiento nos vamos a encargar. En este caso, el *medidor* se utiliza para insertar en la base de datos la información enumerada arriba de todas las rutinas.
- Cuando se ha modificado el código en las tareas *Realizar acciones correctivas* del mantenimiento correctivo urgente y *Ejecutar intervención*, de los cuatro planificables.

6.3.1. EJEMPLO.

Veamos a continuación un ejemplo de cómo se utiliza este componente tras una modificación del código, aplicándolo a la petición que creamos en la Figura 33. Supongamos que la modificación necesaria para servir tal petición fuera la adición del código recuadrado en la Figura 39 a la rutina *mnuDVerGraficoDeComponente_click* del módulo *FormInformeDeTendencia*.

```
MousePointer = vbDefault
FormGrafico.Show vbModal

' Esto es un trozo que me invento para ilustrar un ejemplo
If 2 > 3 Then
    If 1 < 2 Then
        Dim a As Long
        Select Case a
            Case 1
                For a = 1 To 10
                    a = a + a
                Next
            Case 2
                While a < 10
                    a = a + 1
                Wend
            Case Else
                Dim c1 As Double
                Dim c2 As Double
                Dim h As Double

                Randomize
                c1 = Rnd() * 1000
                c2 = Rnd() * 1000
                h = Sqr(c1 ^ 2 + c1 ^ 2)
                MsgBox "La hipotenusa vale " & h
        End Select
    End If
End If
End Sub
```

Figura 39. Código añadido para, supuestamente, arreglar un error.

Antes de realizar la modificación, el componente de medición nos ofrece la siguiente información de esta rutina: cero parámetros de entrada, cero de salida, 41 líneas de código, cero líneas de comentario y 1 como complejidad ciclomática. La adición del código supone la introducción de 24 líneas nuevas (sin contar la de comentario) y 7 nuevas decisiones (2 *if*, 3 *case* y dos bucles). En la Figura 40 mostramos cómo, efectivamente, el módulo de medida toma nota de estos nuevos valores y los inserta en la base de datos, de tal manera que en ésta se va almacenando la evolución de todas las rutinas que van siendo modificadas.

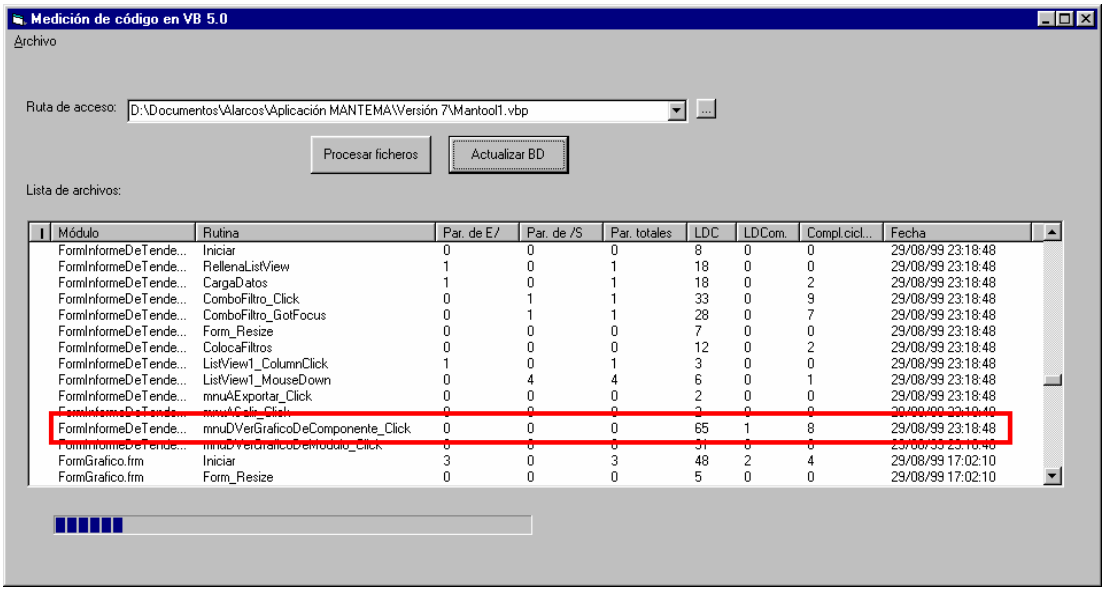


Figura 40. El módulo de medida, durante el proceso de actualización de la base de datos, después de haber medido la aplicación.

Gracias al almacenamiento de los valores instantáneos de las métricas de producto, podemos conocer si los compromisos de mantenimiento preventivo se van cumpliendo (véase Figura 29 en la página 177).

6.4. OTRAS CARACTERÍSTICAS.

6.4.1. RESUMEN DE LOS DATOS DE UNA APLICACIÓN.

La Figura 41 muestra la pantalla en la que aparecen resumidos los datos de mantenimiento de una aplicación, que son calculados a partir de la información histórica.

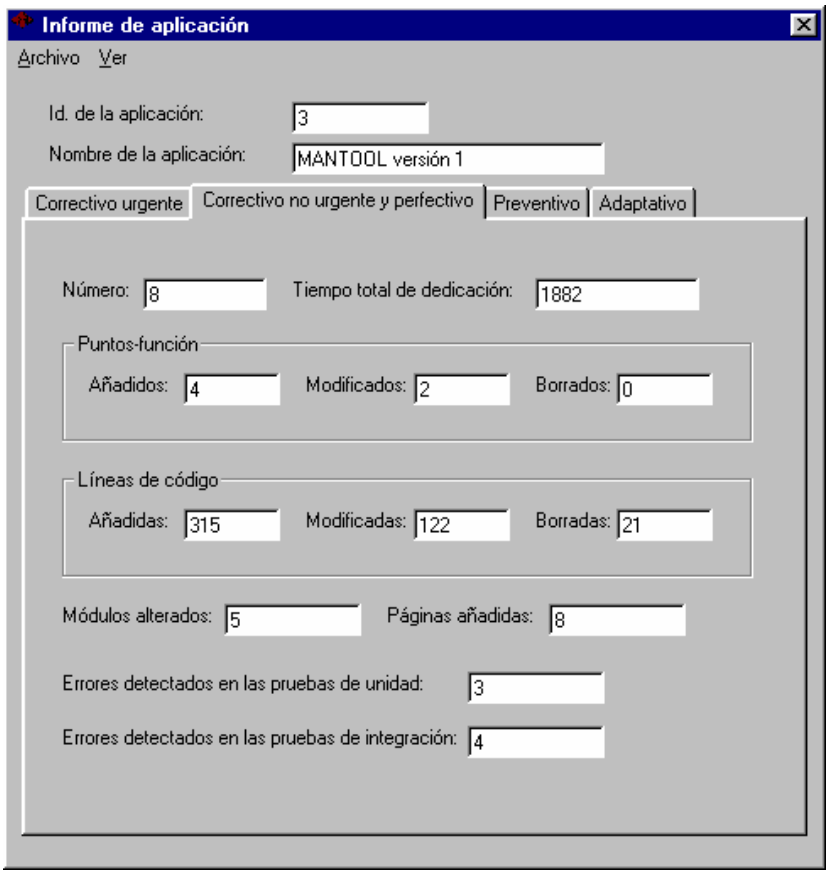


Figura 41. Información del mantenimiento de una aplicación.

6.4.2. ESTADO ACTUAL DE UNA APLICACIÓN.

MANTOOL permite conocer el estado actual de una aplicación desde diferentes puntos de vista:

- Conocer los valores de las diferentes métricas de cada aplicación y el número de cambios sufridos por cada rutina (Figura 42).

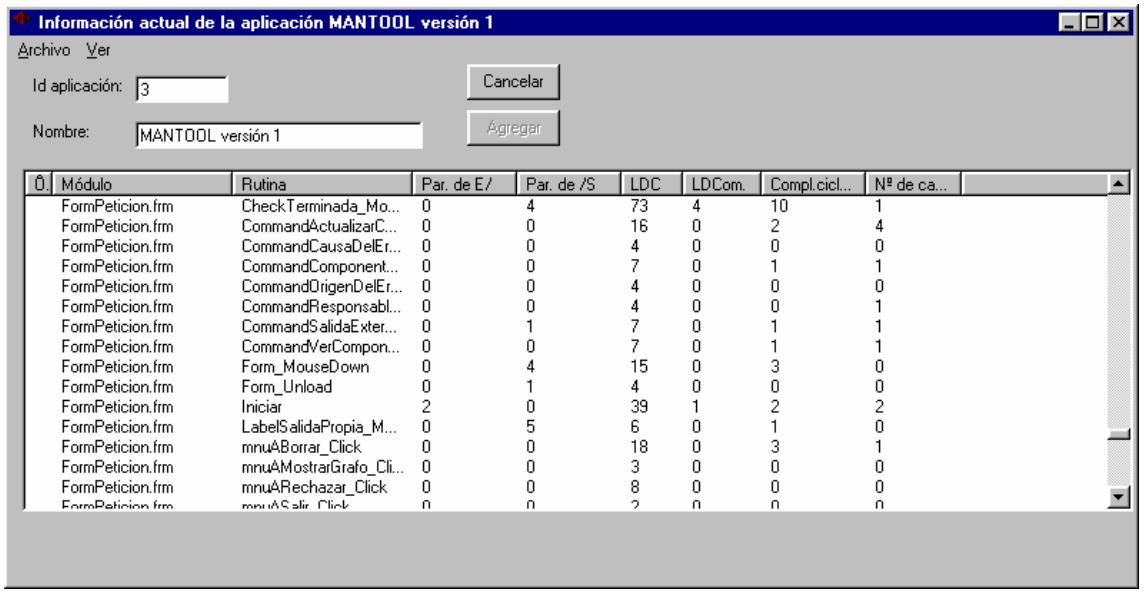


Figura 42. Pantalla con los datos del estado actual de la aplicación MANTOOL.

- Conocer cómo se distribuye el número de cambios según el tamaño, complejidad o número de parámetros de las rutinas. En la Figura 43 sumamos el número de rutinas de MANTOOL con cada complejidad, y mostramos el número de cambios ocurridos para cada valor de ésta. Sorprendentemente, la mayoría de los cambios tienen lugar en las rutinas con menor complejidad ciclomática.

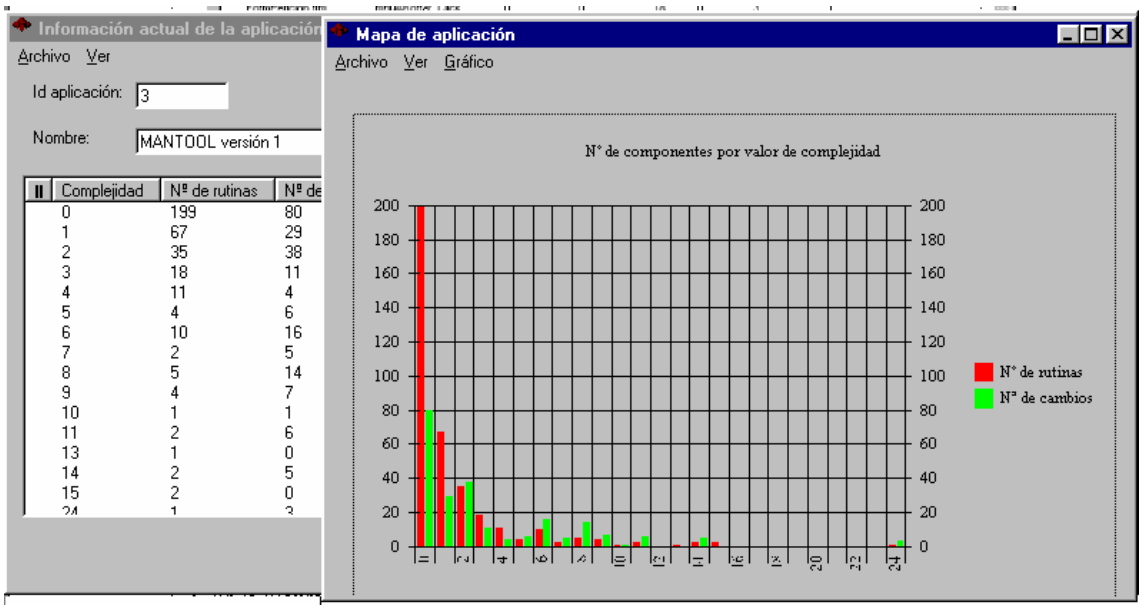


Figura 43. Distribución del número de rutinas según su complejidad, y número de cambios en cada grupo.

6.4.3. CONTROL DE DEDICACIONES.

En MANTOOL puede conocerse el tiempo trabajado por cada miembro del equipo de mantenimiento.

6.4.4. ANÁLISIS DE CARTERA.

En la Figura 44 mostramos las dos pantallas que permite el informe del Análisis de cartera. Los valores de la *calidad técnica* son calculados en función de las métricas de producto de las aplicaciones y de su valor de negocio, el cual es o asignado en el momento de darla de alta o modificado posteriormente. Los valores máximos y mínimos necesarios para calcular la calidad técnica están almacenados en el fichero de configuración y pueden ser modificados en cualquier momento.

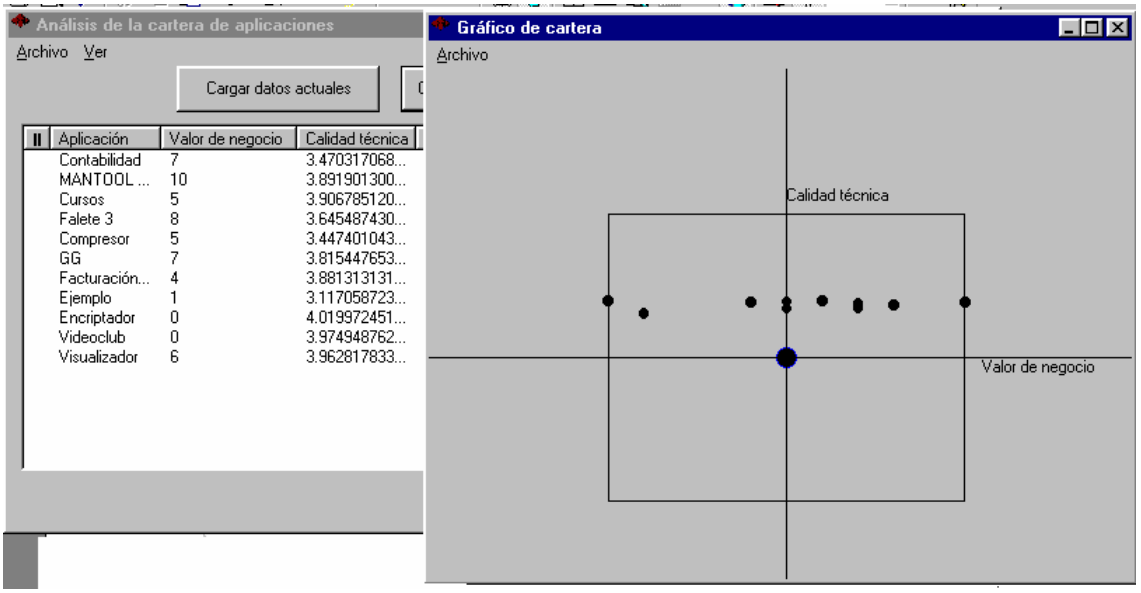


Figura 44. Datos numéricos del análisis de la cartera de aplicaciones y representación gráfica.

6.5. CONCLUSIONES.

Como hemos visto al principio de este capítulo, MANTOOL permite controlar peticiones de modificación conforme a la metodología de mantenimiento presentada en esta tesis. MANTOOL es utilizable a lo largo de todo el proceso de mantenimiento definido, excepto en las dos actividades iniciales (*Estudio inicial* y *Planificación del proceso*) y las dos últimas finales (*Retirada* y *Fin de la externalización*).

Todos los informes de tipo tabular que muestra MANTOOL tienen una opción para exportarlos con formato Ascii, lo que permite procesarlos muy fácilmente con cualquier hoja de cálculo, de cara a realizar otros tipos de análisis.

Por otra parte, es interesante preguntarse a qué tipo de preguntas, referentes al proceso de mantenimiento, es capaz de responder a MANTOOL. A continuación enumeramos algunas de ellas, mostrando el tipo de respuestas que es posible obtener:

- Situación en que se encuentra una petición de modificación, indicando última actividad y tarea ejecutadas (Figura 35 y Figura 37).
- Tiempo consumido por una petición (Figura 35).
- Recursos y tiempo consumidos por cierta tarea de cierta petición (Figura 38).
- Métricas de producto de una aplicación antes y después de cada una de las intervenciones que ha sufrido. Esto se corresponde con los gráficos de tendencia (Figura 29, página 177) y sus respectivos informes, uno de los cuales representamos en la Figura 45.

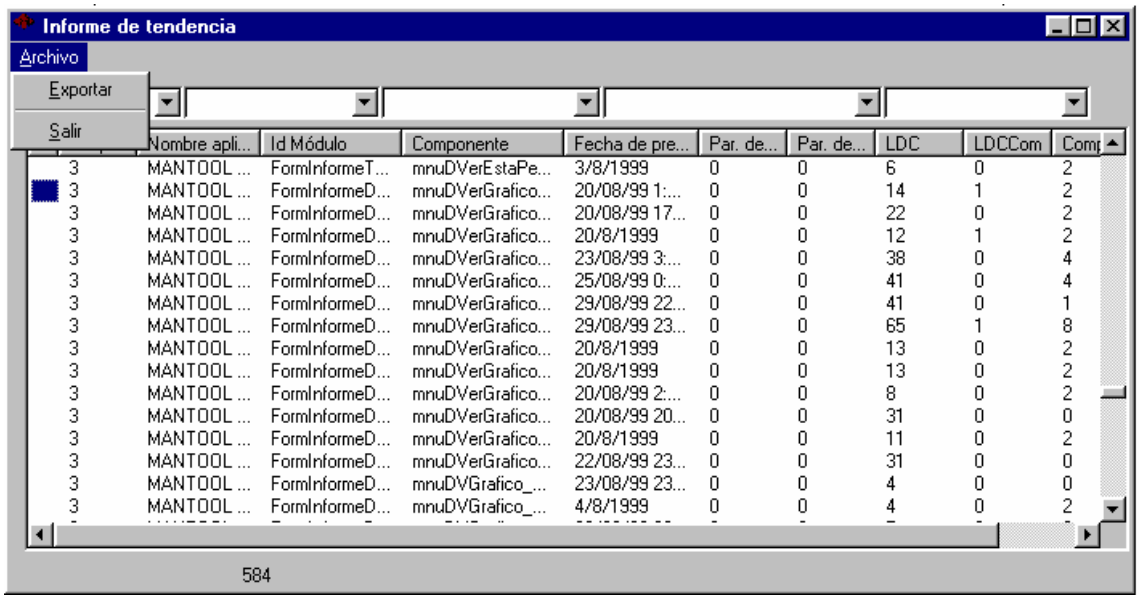


Figura 45. Un informe de tendencia.

- Número de peticiones sufridas por una aplicación. Esta información puede verse directamente en el *Árbol de aplicaciones* (Figura 36), en la ventana de resumen de datos de una aplicación (Figura 41), o mediante el tratamiento, con una hoja de cálculo, de los datos exportados por los diferentes informes de tabulares de cada tipo de mantenimiento.
- Estado en que se encuentra una aplicación de acuerdo al Análisis de cartera (Figura 44).
- Número total de peticiones recibidas: si el número no es muy elevado, puede calcularse en varias de las pantallas de MANTOOL; en caso contrario, pueden tratarse los datos de los informes en una hoja de cálculo.
- Número total de peticiones realizadas por una persona.
- Número de peticiones rechazadas, ya que se contempla el rechazo de peticiones.
- Número de peticiones de cada tipo de mantenimiento.
- Número de peticiones por cambios legales y tiempo total.

- Número, recursos y tiempo de peticiones por evolución del negocio.
- Número, recursos y tiempo de peticiones por mejora del proceso.
- Número, recursos y tiempo de peticiones por cambio en las reglas del negocio.
- Número de peticiones de correctivo (urgente y no urgente) según el origen y causa del error.
- Número de páginas de documentación generadas por cada petición (en los informes tabulares de mantenimiento).
- Número de errores detectados en las pruebas de cada petición (en los informes tabulares de mantenimiento).
- Número de errores detectados en las pruebas de intervenciones realizadas por cierto equipo de mantenimiento.
- Evolución de diferentes métricas de producto de las aplicaciones.
- Valores máximos, mínimos, medios, etc. de todos los puntos anteriores: pueden conocerse fácilmente, ya que todos los informes pueden ser ordenados por cualquier columna.

Por último, es interesante señalar que Atos ODS está añadiendo a su herramienta Conduct (Piattini et al., 1998a) algunas características de MANTEMA.

7 CONCLUSIONES

En este capítulo analizamos cómo se han cumplido los objetivos indicados al principio de esta tesis, mencionamos las principales aportaciones del trabajo de investigación, contrastamos los resultados mediante las publicaciones conseguidas y exponemos las líneas de trabajo que permanecen abiertas.

7.1. ANÁLISIS DE LA CONSECUCIÓN DE OBJETIVOS

En la sección 1.6 de este trabajo se plantearon una serie de objetivos parciales cuyo fin era alcanzar el objetivo principal de esta investigación: la definición de la metodología que hemos presentado a lo largo de esta tesis.

A continuación analizamos el resultado de cada uno de estos objetivos.

Objetivo 1. Analizar las distintas metodologías aplicables al mantenimiento del software, determinando sus aportaciones y limitaciones.

Si bien hemos comprobado la ausencia de metodologías para el proceso de mantenimiento, en el Capítulo tercero hemos realizado un análisis de las propuestas más importantes de los últimos años realizadas en este sentido. Se ha completado cada análisis con un comentario, en el que hemos puesto de manifiesto las aportaciones y limitaciones de cada una.

Objetivo 2. Definir una metodología específica para mantenimiento, en el caso de no encontrar una adecuada.

A la consecución de este objetivo hemos dedicado gran parte del capítulo cuarto, que iniciamos con la presentación de la macroestructura propuesta para la gestión del mantenimiento. Posteriormente, hemos mostrado la integración, en el proceso de mantenimiento, del resto de procesos habituales del ciclo de vida software.

Igualmente, hemos explicado la estructuración del proceso según los diferentes tipos de mantenimiento que, con matices, se definen habitualmente en la literatura. Para cada uno de estos tipos, se han detallado las actividades y tareas que los componen, habiendo indicado en algunas de éstas diferentes técnicas que pueden ser utilizadas para su ejecución, así como las métricas que deben aplicarse para mantener el control cuantitativo del proceso.

Por último, hemos definido también actividades y tareas para la preparación y extinción del proceso de mantenimiento.

Objetivo 3. Analizar técnicas susceptibles de ser utilizadas a lo largo del proceso de mantenimiento del software.

En diferentes secciones del Capítulo tercero hemos repasado algunas de las técnicas más utilizadas durante el mantenimiento del software, habiendo enumerado en la 3.4 los tipos de herramientas automáticas que son capaces de darles soporte.

Objetivo 4. Definir, cuando no existan o no sean apropiadas, técnicas específicas para mantenimiento, susceptibles de ser integradas en la metodología propuesta.

De esto nos hemos ocupado también en el Capítulo cuarto, en donde hemos presentado una técnica para la identificación y estimación de riesgos, utilizable al principio del proceso de mantenimiento. También hemos presentado un conjunto de métricas para controlar diferentes atributos de bases de datos y las métricas *CC*, para estimar el esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto. En el Apéndice I se ofrecen las plantillas de los documentos generables durante el proceso de mantenimiento. También se incluyen la validación empírica y caracterización formal de algunas de las métricas presentadas. Por último, se ha planteado un modelo para planificar óptimamente la cantidad de recursos que deben dedicarse a servir peticiones de mantenimiento no planificable.

Objetivo 5. Definir un conjunto de métricas para controlar el proceso de mantenimiento.

Se han presentado una serie de métricas de proceso para mantenimiento. Igualmente, se han definido diversos Indicadores de Nivel de Servicio, que deben ser considerados cuando exista externalización del mantenimiento.

Objetivo 6. Validar la metodología de mantenimiento producida, así como las diferentes técnicas definidas.

En el Capítulo séptimo hemos explicado los casos reales en los que se ha aplicado MANTEMA, habiendo discutido algunas de las ventajas que reporta su utilización. Sería interesante, sin embargo, conseguir datos de otros proyectos en los que se está aplicando la metodología.

La validación de las diferentes técnicas propuestas en esta tesis se ha ido mostrando, en el capítulo cuarto, tras la presentación de cada una de ellas.

Objetivo 7. Desarrollar un soporte automático que facilite la aplicación de la metodología.

En el Capítulo séptimo hemos presentado MANTOOL, una herramienta para la gestión del proceso de mantenimiento conforme a la metodología MANTEMA. La definición rigurosa de la metodología y su estructuración ha sido uno de los aspectos que más ha facilitado la implementación de MANTOOL.

Objetivo principal. Definir una metodología de mantenimiento que permita gestionar con métodos de ingeniería el proceso de mantenimiento del software.

Puesto que se han alcanzado los diferentes objetivos marcados, y ya que la unión de éstos conformaba este objetivo principal, es claro que éste ha sido alcanzado.

Por otra parte, la demostración de la hipótesis sobre la que hemos trabajado (*Es posible gestionar el proceso de mantenimiento del software utilizando métodos de ingeniería*) se basaba en el cumplimiento del objetivo principal, por lo que podemos concluir que hemos demostrado la veracidad de la hipótesis.

7.2. PRINCIPALES APORTACIONES DE LA INVESTIGACIÓN

El trabajo realizado ha supuesto una serie de aportaciones relativas tanto al tema principal de la investigación (la definición de la propuesta metodológica y metrológica para el mantenimiento), como a otros temas afines, en el contexto del marco del trabajo que mencionábamos en la sección 1.5. Reproducimos a continuación la Figura 4 para mostrar y ubicar cómodamente las principales aportaciones de MANTEMA, que pasamos a desarrollar más abajo:

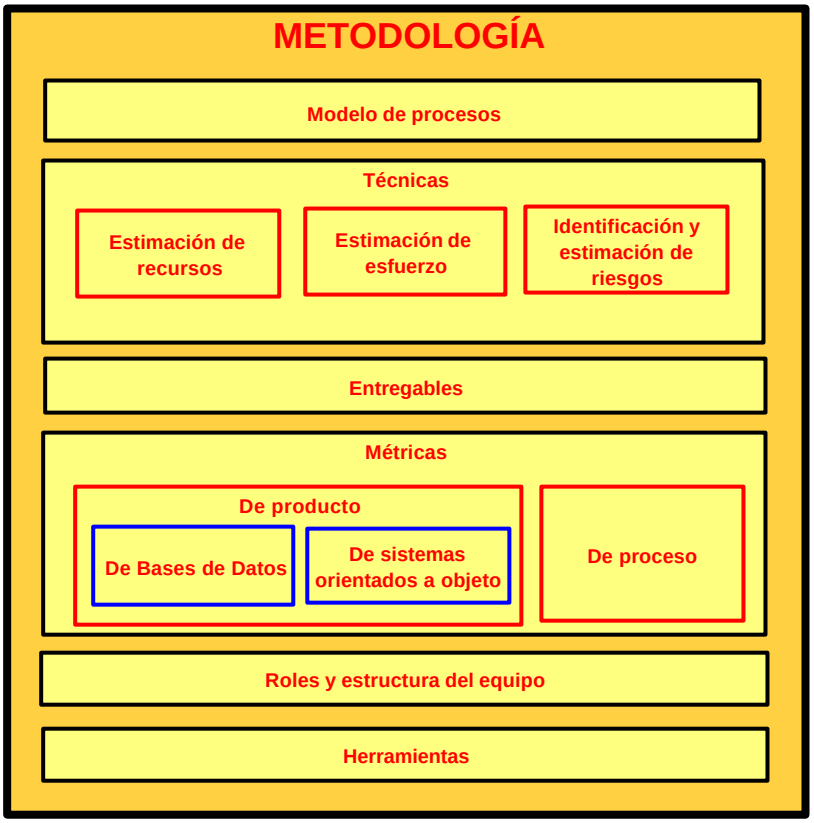


Figura 46. Principales aportaciones de MANTEMA.

7.2.1. MODELO DE PROCESOS

Se ha definido una metodología de mantenimiento de software rigurosa y completa en la que se describen, de manera clara, dos tipos diferentes de mantenimiento (no planificable o correctivo urgente, y planificable, que agrupa al correctivo no urgente, perfectivo, preventivo y adaptativo). La estructuración de cada tipo de mantenimiento en tareas, y la precisa identificación de los elementos que intervienen en cada tarea permite mantener bajo control las intervenciones de mantenimiento que se adapten a MANTEMA.

7.2.2. ROLES Y ESTRUCTURA DEL EQUIPO

Se han identificado las tres organizaciones que participan en el proceso de mantenimiento de software, habiéndose listado además una serie de perfiles para cada una. Se mostró la po-

sibilidad de que dos de estas organizaciones *lógicas* estuviesen formadas por una sola organización real, posibilidad ésta que puede también darse con los perfiles de una sola organización. Esta identificación de *roles* resulta muy interesante, ya que, como se puso de manifiesto en los capítulos primero y tercero, la comunidad científica insiste en la necesidad de investigar no sólo aspectos únicamente ingenieriles, sino también organizacionales.

7.2.3. TÉCNICAS

En el capítulo tercero hemos presentado explícitamente dos técnicas novedosas (estimación de recursos para mantenimiento no planificable e identificación de las características con mayor influencia en riesgos), pero también se ha mostrado la integración de la externalización en el proceso de mantenimiento.

7.2.3.1. Estimación de recursos para mantenimiento no planificable

Esta técnica permite calcular la cantidad de recursos que debe dedicarse a mantenimiento no planificable durante un periodo, de manera que la Organización de mantenimiento no sufra pérdidas económicas. Para ello, se plantearon y resolvieron las necesarias ecuaciones para construir un modelo matemático del proyecto de mantenimiento.

Se expuso las diferentes formas en que esta técnica puede ser utilizada: con externalización (pudiendo calcular en este caso la cantidad de recursos desde los puntos de vista de la organizaciones cliente y de mantenimiento) y sin externalización del proceso.

Esta técnica resulta muy útil en situaciones como la actual, en la que la demanda de personal cualificado en Tecnologías de la Información supera ampliamente la oferta.

7.2.3.2. Identificación y estimación de riesgos en proyectos de mantenimiento

Se propuso una técnica para identificar y estimar riesgos -desde el punto de vista de la Organización de mantenimiento- de proyectos de mantenimiento que no se han acometido todavía. Esta técnica se utiliza cuando existe aún muy poca información cuantitativa del software que se va a mantener y resulta útil para estimar la futura distribución de peticiones de modificación en cada tipo de mantenimiento, razón por la que, en la Figura 46, tiene también una flecha de llegada desde el nodo etiquetado “Metrología”.

7.2.3.3. Integración de la externalización en el proceso de mantenimiento

Los nodos inicial y final de la metodología (Figura 21, página 99) incluyen conjuntos de actividades y tareas que permiten preparar y finalizar relaciones de externalización del mantenimiento entre distintas organizaciones.

No obstante, tal integración no se ha llevado a cabo sólo en el apartado del Modelo de Procesos que corresponde a la metodología (como podría desprenderse del párrafo anterior), ya que también las dos técnicas mencionadas en los dos epígrafes inmediatamente anteriores son utilizables en los casos en que existe externalización del proceso. También se han propuesto indicadores de servicio para determinar los niveles de calidad del proceso suministrado por la Organización de mantenimiento.

7.2.4. MÉTRICAS

Se han presentado métricas tanto de producto como de proceso; entre las primeras, se incluyen métricas para bases de datos y para sistemas orientados a objeto. Entre las segundas, métricas para el control de procesos terminados y en ejecución, e indicadores de servicio.

7.2.4.1. Métricas para bases de datos

Las métricas para bases de datos constituyen un aporte de investigación muy novedoso, en un campo de la Informática en el que confluyen las bases de datos y la ingeniería del software. La influencia de estas métricas en la facilidad de comprensión de las bases de datos se ha demostrado mediante algunos experimentos, uno de los cuales se ha detallado en la sección 4.4.1.6. Como se puso de manifiesto, la facilidad de comprensión es uno de los factores que influyen en la mantenibilidad y, por tanto, en la facilidad de mantenimiento. Además, algunas de las métricas se han caracterizado formalmente utilizando un marco axiomático.

7.2.4.2. Métricas para sistemas orientados a objeto

Se han caracterizado formalmente las métricas *CC* para sistemas orientados a objeto, y se ha discutido acerca de la necesidad de nuevos marcos para este tipo de sistemas, ya que poseen características que los diferencian suficientemente de los sistemas tradicionales, no orientados a objeto (véanse secciones 4.4.2.3 y 4.4.2.4, en las que se describen los operadores de composición).

Las métricas *CC* se utilizan para estimar el esfuerzo de mantenimiento adaptativo de sistemas orientados a objeto. Según la definición de "métrica de proceso" vista al inicio de la sección 3.2 (página 72) y de la breve discusión en el párrafo que le sigue, comprobamos cómo, efectivamente, al utilizar los valores devueltos por estas métricas de producto en forma de medidas predictivas, estamos también utilizando métricas de proceso.

7.2.4.3. Métricas para la planificación y el control del proceso

Los indicadores de servicio (sección 4.4.3.1) permiten determinar el nivel de calidad deseado en el proceso de mantenimiento. El conjunto de métricas de proceso propuesto (sección 4.4.3.2, página 175) ayuda a mantener el control del proceso y a evaluar su calidad.

7.2.5. HERRAMIENTA MANTOOL

MANTOOL ofrece soporte automático para realizar el proceso de mantenimiento utilizando MANTEMA. Aunque no es un apoyo completo, porque algunas características de MANTEMA han quedado fuera, el servicio que ofrece MANTOOL a los usuarios de la metodología resulta de gran utilidad.

7.2.6. ENTREGABLES

En el Apéndice I se muestran las plantillas de los documentos generables durante los procesos de mantenimiento adaptados a MANTEMA.

7.2.7. OTRAS APORTACIONES

Resulta importante destacar que MANTEMA es *abierta*, en el sentido de que tanto las técnicas como los diferentes conjuntos de métricas definidos pueden ser utilizados tanto en otras metodologías de mantenimiento como en otros momentos y procesos del ciclo de vida software.

Por otro lado, consideramos interesante la experiencia en la aplicación de un método de investigación cualitativo -como lo es Investigación en acción- en Ingeniería del software, así como el haber trabajado paralela pero integradoramente con el método de investigación hipotético-deductivo. Es decir, que si entendemos que los sucesivos ciclos de Investigación en acción (Figura 5, página 35) van conformando progresivamente una espiral cuya mayor proximidad al centro significa una mejor definición y validación del modelo del proceso, las técnicas y métricas han podido ser propuestas y validadas independientemente del nivel de profundidad en que nos encontráramos.

Señalar por último que, aunque en esta misma sección hemos presentado como aportaciones separadas las técnicas de las métricas, la utilización de éstas para realizar predicciones de esfuerzo, de recursos, etc. constituye realmente la utilización de técnicas.

7.3. CONTRASTE DE RESULTADOS

Diferentes resultados parciales de la presente tesis han sido publicados en diferentes foros. A continuación enumeramos algunos de ellos.

7.3.1. REVISTAS INTERNACIONALES

1. Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999). Roles in the Maintenance Process. ***ACM Software Engineering Notes***, vol. 24, nº 4, páginas 84-86.

   En este artículo se expone la identificación de las organizaciones y los perfiles que intervienen en el proceso de mantenimiento.

2. Calero, C., Piattini, M., Polo, M. y Ruiz, F. (2000). Métricas para la Complejidad de Bases de Datos Relacionales. ***Computación y Sistemas,*** vol. 3, nº 4.

   Se presentan métricas para bases de datos relacionales y se realizan algunas discusiones respecto de sus posibles valores.

3. Fioravanti, F., Nesi, P. y Polo, M. Complexity/Size Metrics for Object-Oriented Systems.

   Este artículo es una revisión del informe técnico mencionado más abajo. Se envió, en noviembre de 1999, a la revista ***IEEE Transactions on Software Engineering***, encontrándose aún en periodo de revisión.

7.3.2. CAPÍTULOS DE LIBRO

4. Polo, M., Piattini, M. y Ruiz, F. (2000). "Managing the Software Maintenance Process" (capítulo 3.1). En van Bon (ed.): **World Class IT Service Management Guide.** ten Hagem & Stam Publishers, La Haya (Holanda).
5. Piattini, M., Genero, M., Calero, C., Polo, M. y Ruiz, F. “Database Quality” (capítulo 14), en Piattini y Díaz (eds.): **Advanced Databases: Technology and Design**. Artech House, Londres (Reino Unido). Aparecerá en junio de 2000.
6. Piattini, M., Genero, M., Calero, C., Polo, M. y Ruiz, F. "Metrics for managing quality information modelling". En Siau y Rossi (eds.): **Information Modelling in the New Millenium.** IDEA Group Publishing, Hershey, Estados Unidos. Se publicará en otoño de 2000.

7.3.3. CONFERENCIAS INTERNACIONALES

7. Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999). *MANTEMA: a Complete Rigorous Methodology for Supporting Maintenance based on the ISO/IEC 12207 Standard*. Proceedings of the **Third IEEE European Conference on Software Maintenance and Reengineering**, páginas 178-181. Celebrado en Amsterdam (Holanda), en marzo de 1999.

   En esta comunicación se presentó la macroestructura de MANTEMA, detallando el mantenimiento correctivo urgente.

8. Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999). *Using the ISO/IEC 12207 Tailoring Process for Defining a Maintenance Process.* Proceedings of the **First IEEE Conference on Standardisation and Innovation in Information Technology**. Celebrado en Aquisgrán (Alemania), en septiembre de 1999.

   En esta comunicación se explicó cómo se había aplicado el Proceso de Adaptación de ISO/IEC 12207 al Proceso de Mantenimiento para obtener la macroestructura de la metodología.

9. Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999). *MANTEMA: a Software Maintenance Methodology based on the ISO/IEC 12207 Standard.* Proceedings of the **Fourth IEEE International Software Engineering Standards Symposium and Forum**. Celebrado en Curitiba (Brasil), en mayo de 1999.

   Se explica en esta comunicación la integración de los procesos del ciclo de vida dentro del Proceso de Mantenimiento y se detallan las actividades y tareas para el inicio y el fin del proceso de mantenimiento.

10. Calero, C., Piattini, M., Polo, M. y Ruiz, F. (1999). *Validating Referential Integrity as a Database Quality Metric.* Proceedings of the **First International Conference on Enterprise Information Systems**, páginas 45-50. Celebrado en Setúbal (Portugal).

    Se realizan las validaciones formal y empírica de las métricas de longitud y complejidad para esquemas lógicos de bases de datos.

11. Polo, M., Piattini, M. y Ruiz, F. (2000). *Planning the non-planneable maintenance.* En "Project Control: The Human Factor", (Proceedings of the combined **11th European Software Control and Metrics conference and the 3rd SCOPE conference on Software Product Quality, ESCOM-SCOPE 2000**), páginas 49-57. Celebrado en Munich (Alemania), en abril de 2000.

    Se presenta la técnica para estimar la cantidad de recursos que deben dedicarse a atender peticiones de mantenimiento no planificable.

12. Piattini, M., Calero, C., Polo, M. y Ruiz, F. (1998). *Maintainability in Object-Relational Databases.* Proceedings of the **First European Software Measurement Conference,** páginas 223-230. Celebrado en Amberes (Bélgica) en mayo de 1998.

    Se presentan métricas para bases de datos objeto-relacional, y la caracterización formal de algunas de ellas conforme al marco de Briand, Morasca y Basili (1996).

13. Polo, M., Piattini, M., Calero, C. y Ruiz, F. (1998). *Measures for control database quality*. **Actas del V Congreso Internacional de Investigación en Ciencias Computacionales.** Celebrado en Aguascalientes (México), en noviembre de 1998.

    Se presentaron métricas para esquemas conceptuales y lógicos de bases de datos.

7.3.4. CONFERENCIAS NACIONALES

14. Polo, M., Calero, C., Ruiz, F. y Piattini, M. (1998). *Métricas de calidad y complejidad para bases de datos.* Actas de las **III Jornadas de Ingeniería de Software**, páginas 79-90. Celebrado en Murcia en noviembre de 1998.

    En esta comunicación se presentan de las métricas para esquemas conceptuales y lógicos de bases de datos.

15. Ruiz, F., Piattini, M., Polo, M. y Calero, C. (1999). *Auditoría del mantenimiento del software: propuesta de objetivos de control.* Actas del **II Congreso Nacional de Auditoría y Control de Sistemas de Información**, páginas 128-143.

    Se proponen diferentes objetivos de control para la auditoría del proceso de mantenimiento.

16. Polo, M. y Piattini, M. (1999). Elaboración de una metodología para el mantenimiento de software. Actas de las **IV Jornadas de Ingeniería de Software**, páginas 219-230. Celebrado en Cáceres en noviembre de 1999.

    En este trabajo se presentó la versión 2.0 de la metodología, así como algunas métricas de proceso y de la herramienta MANTOOL.

7.3.5. REVISTAS NACIONALES

17. Genero, M., Piattini, M., Calero, C., Polo, M. y Ruiz, F. (1999). Calidad de esquemas conceptuales. ***Novática***, nº 140, páginas 23-27.

    En este artículo se realiza un repaso a algunos de los marcos de referencia para la calidad de esquemas conceptuales de bases de datos.

7.3.6. INFORMES TÉCNICOS

18. Fioravanti, F., Nesi, P. y Polo, M. (1999). Complexity/Size Metrics for Object-Oriented Systems. **Informe Técnico 17/1999**, del Departamento de Sistemas e Informática de la Universidad de Florencia.

    En este informe se presenta la familia de métricas *CC*, se definen los operadores de composición, se caracterizan formalmente las métricas y se presenta su validación empírica.

19. Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999). MANTEMA versión 2.0: una Metodología para el Mantenimiento de Software. **Informe Técnico UCLM-DI-99-01**, del Departamento de Informática de la Universidad de Castilla-La Mancha.

    En este informe se presentan algunos aspectos de la versión 2.0 de MANTEMA: se comenta la norma ISO/IEC 12207, se justifica la necesidad de una metodología de mantenimiento y se explica la estructura de actividades y tareas de MANTEMA.

20. Calero, C., Piattini, M., Genero, M., Fernández-Medina, E., Ruiz, F. y Polo, M. (2000). Métricas para bases de datos. **Informe Técnico UCLM-DI-2000-01**, del Departamento de Informática de la Universidad de Castilla-La Mancha.

    Se presenta una amplia visión del estado del arte en métricas para bases de datos: métricas para modelos conceptuales, para bases de datos relacionales, activas, orientadas a objeto y objeto-relacionales. También se explican algunos marcos de verificación

formal, técnicas de validación empírica y se dedica un capítulo a la estimación de esfuerzo en bases de datos.

7.4. LÍNEAS FUTURAS DE TRABAJO E INVESTIGACIÓN.

A pesar de que la tesis ha cumplido los objetivos que nos marcamos al principio (sección 1.6) y de las aportaciones que pensamos que supone (sección 7.2), prácticamente cada uno de los elementos de la metodología (Figura 46) deja abiertos frentes en los que es preciso continuar investigando.

- En el modelo del proceso no han quedado integrados ni se establece interfaz con los procesos de Mejora y Auditoría de ISO/IEC 12207. Entendemos que es importante estudiar más a fondo la integración de estos procesos en la metodología, de manera que la investigación los pueda ir acercando hacia el mantenimiento (Polo et al., 2000; Ruiz et al., 1999b).
- La identificación de organizaciones y perfiles realizada presenta un enorme potencial investigador: en efecto, sería interesante estudiar los aspectos psicológicos de la Ingeniería del software (circunscritos, en nuestro caso, al mantenimiento), aprovechando el método Investigación en acción y las relaciones entre los diferentes actores de este método.
- Será preciso profundizar en la parte *metrológica* de MANTEMA, de manera que se consiga proponer métricas para diferentes entornos. Por su carácter novedoso destacan las bases de datos objeto-relacionales (Piattini et al., 1998b), bases de datos activas (Díaz y Piattini, 1999), los lenguajes de cuarta generación (Martínez y Piattini, 1999) y el modelado de datos con lenguajes de modelado modernos como UML (Genero et al., 1999).
- No obstante lo dicho en el párrafo anterior, además de la propuesta y validación experimental de nuevas métricas, y como Calero y Piattini (1999) han puesto de manifiesto, será preciso validar dichas métricas utilizando otros marcos de formalización (basados en la teoría de la medida, por ejemplo), y no sólo aproximaciones axiomáticas como la mostrada en la sección 4.4.1.5.
- Investigaremos más a fondo el tema de los riesgos aplicados al proceso de mantenimiento, que hemos considerado muy interesante. En esta tesis se ha presentado el primer resultado de la implementación de una técnica de gestión de riesgos de proyectos de mantenimiento, que debe seguir siendo estudiada en profundidad.
- Sería interesante considerar el redesarrollo de MANTOOL utilizando herramientas de Workflow. Además, es importante avanzar en el desarrollo de herramientas verticales que sirvan de apoyo a la ejecución de diversas tareas en MANTOOL (por ejemplo: componentes que permitan realizar estimaciones de futuras intervenciones a partir de los datos almacenados en la base de datos). También será preciso ampliar MANTOOL

horizontalmente, incorporándole los módulos necesarios para gestionar las actividades y tareas iniciales y finales definidas en MANTEMA.

- Como se mostró en la Tabla 10 (Comparación de los diferentes enfoques propuestos para mantenimiento), MANTEMA no contempla el mantenimiento proactivo. Recientemente se han firmado sendos proyectos con las empresas Soluziona International Software Factory y Gedas en los cuales se abordarán, entre otros, aspectos relacionados con este tema.

Con la consideración realizada en el último punto, y retrocediendo hasta la Figura 3 (página 27), podemos observar cómo la investigación en todos estos frentes abiertos se encuentra cubierta por los diferentes proyectos en los que viene trabajando el grupo Alarcos.

APÉNDICES

APÉNDICE I: CONTENIDO DE LOS DIFERENTES DOCUMENTOS

En este apéndice especificamos el índice de contenidos de los diversos documentos durante el proceso de mantenimiento definido en esta tesis.

DOC1 - CUESTIONARIO INICIAL.

A) Identificación de la empresa (datos económicos, personal, etc.) e interlocutores

B) Entorno hardware y software

B.1) Hardware

B.1.1.- Hardware central (marca y modelo, sistema operativo, etc.)

B.1.2.- Hardware periférico

1.2.1.- Relación de terminales

1.2.2.- Relación de estaciones de trabajo (indicando el sistema operativo de cada una)

1.2.3.- Tipo de red

1.2.4.-Tipo de conexión con el hardware central

B.2) Entorno software

B.2.1.- Entorno de desarrollo

B.2.2.- Sistema de ficheros

B.2.3.- Bases de datos

B.2.4.- JCL batch

C) Organización de desarrollo y mantenimiento

C.1.- Metodologías y técnicas de desarrollo

C.2.- Estándares (de denominación, programación, interfaces de usuario, etc.)

C.3.- Normas de gestión de calidad

C.4.- Procedimientos de gestión de proyectos

C5.- Procedimiento de puesta en explotación

C.6.- Procedimientos de auditoría

C.7.- Procedimientos de resolución de problemas

C.8.- Procedimientos de documentación

C.9.- Otros procedimientos o normativas de aplicación

D) Aplicaciones (por cada aplicación)

D.1.- Identificación (nombre, fecha de puesta en marcha, etc.)

D.2.- Unidad organizacional responsable/usuaria

D.3.- Otras aplicaciones relacionadas

D.4.- Programas por lotes (lenguajes, nº de líneas de código/módulos, año de creación, nº de intervenciones sufridas, etc.)

D.5.- Programas en línea (lenguajes, nº de líneas de código/módulos, año de creación, nº de intervenciones sufridas, etc.)

D.6.- Listados (Número, lenguajes, año de creación, nº de intervenciones sufridas, etc.)

D.7.- Pantallas (Número, lenguajes, año de creación, nº de intervenciones sufridas, etc.)

DOC2 - PROPUESTA DE MANTENIMIENTO.

1) Introducción.
2) Resultados de los análisis aplicados a una muestra de programas de distinta complejidad elegidos por la organización proveedora del mantenimiento.
3) Propuesta del servicio formada por:

3.1 Relación técnica que define objetivos, límites, vínculos, responsabilidades, modalidad, actividad y parámetros contractuales. Para cada aplicación que quede dentro de los límites del contrato, se debe especificar el tipo o tipos de mantenimiento que se contratan y los indicadores de servicio.

3.2 Propuesta de contrato.

4) Oferta económica.

DOC3 - CONTRATO DE MANTENIMIENTO.

Nota: los contenidos de este documento son válidos tanto para el contrato provisional de mantenimiento como para el contrato definitivo.

1) Identificación de las partes
2) Objeto del contrato (*el cliente confía al proveedor un conjunto de prestaciones en el campo de los servicios informáticos que tiene como finalidad el mantenimiento del software registrado en el contexto de la propuesta técnica*).
3) Características de la prestación del servicio:

3.1. Inventario de los objetos software a mantener.

3.2. Estado inicial del software.

3.3. Condiciones de organización del trabajo del proveedor y del cliente.

3.4. Condiciones de formalización de la intervención de mantenimiento.

3.5. Adaptación del Plan de Garantía de Calidad al cliente.

3.6. Corrección de las anomalías.

3.7. Mantenimiento de la competencia sobre el software aplicativo y sobre el software estándar por parte del proveedor.

3.8. Redacción de la documentación facilitada a título de prestación anexa. Se determinan, entre otros aspectos, el formato y los plazos de entrega de ésta.

3.9. Modalidad de asistencia a los usuarios finales.

4) Obligaciones del cliente

5) Obligaciones del proveedor

6) Cláusulas de exclusión

7) Cláusulas de organización

8) Garantía de las intervenciones

8.1 Recuperación de datos

8.2 Definición de responsabilidades

9) Otras cláusulas

DOC4 - TABLA DE FACTORES DE RIESGO

Véase sección 4.3.1 (página 127).

DOC5 - RESUMEN TÉCNICO.

1) Introducción

1.1. Alcance y propósito del documento

1.2. Objetivos del proyecto

1.2.1. Objetivos

1.2.2. Funciones principales

1.2.3. Restricciones técnicas y de gestión

2) Estimaciones del proyecto

2.1. Resumen de datos obtenidos de los cuestionarios (nº de programas, tamaño, uso de estándares, etc.)

2.2. Datos históricos (si los hay)

2.3. Técnicas y métricas de estimación

2.4. Valores de las métricas

3) Recursos del proyecto

3.1. Personal

3.2. Hardware y software

3.3. Otros recursos

4) Organización del personal

5) Mecanismos de seguimiento y control

6) Apéndices (si los hay)

DOC6 - PETICIÓN DE MODIFICACIÓN

1) Identificación de la persona que realiza la petición

1.1Nombre

1.2Cargo

1.3Departamento

1.4Teléfono

2) Identificación del producto

2.1 Proyecto afectado

2.2 Componente y versión

3) Para peticiones de modificación por error

3.1. Descripción de la aplicación que falla y de la función que realiza

3.2. Circunstancias en que se produjo el error (fecha y hora, datos de entrada, datos de salida si los hubo, datos de salida esperados, texto libre)

3.3. Mensajes de error dados por el sistema

3.4. Solución recomendada (si es posible)

3.5. Grado de urgencia y fecha en que se necesita la corrección

3.6. Texto libre

4) Para peticiones de modificación por adición de funcionalidades

4.1. Descripción de la funcionalidad que se desea añadir

4.2. Justificación de la adición

4.3. Texto libre

5) Para peticiones de modificación de la calidad del software

5.1. Descripción de las propiedades que se desean mejorar

5.2. Justificación de la adición

5.3. Texto libre

6) Lugar, fecha y hora de presentación de la modificación

DOC7 - ACCIONES CORRECTIVAS REALIZADAS.

1) Identificación de este documento.

2) Identificación del responsable de la intervención (nombre, departamento, teléfono, etc.).

3) Identificación de la Petición de Modificación (DOC6).

4) Lenguaje de programación utilizado.

5) Fecha de instalación del programa.

6) Nº de fallos.

7) Control de dedicaciones (personal y tiempo empleado por cada uno).

8) Tiempo de solución del error.

9) Texto libre.

DOC8 - PRUEBAS UNITARIAS REALIZADAS.

1) Identificación de este documento

2) Identificación del responsable

3) Identificación de la Petición de Modificación

4) Identificación del documento DOC7, DOC11 o DOC13 correspondiente.

5) Propósito general de la prueba

6) Relación de elementos software probados que, para cada elemento, incluya:

   6.1 Identificación del elemento probado

   6.2 Para cada prueba del elemento, indicar:

      6.2.1. Datos de entrada

      6.2.2. Datos de salida esperados y obtenidos

      6.2.3. Evaluación del resultado

      6.2.4. Texto libre

7) Texto libre

DOC9 - DIAGNÓSTICO Y POSIBLES SOLUCIONES.

1) Identificación de este documento.

2) Identificación del responsable de la intervención (nombre, departamento, teléfono, etc.).

3) Identificación de la Petición de Modificación (DOC6).

4) Breve descripción del error

5) Diagnóstico del error

6) Diferentes alternativas de solución

7) Solución adecuada

8) Texto libre

DOC10 - ALTERNATIVAS DE IMPLEMENTACIÓN.

1) Identificación de este documento.

2) Identificación del responsable de la intervención (nombre, departamento, teléfono, etc.).

3) Identificación de la Petición de Modificación (DOC6).

4) Breve descripción de la petición de modificación

5) Planteamiento de las diferentes alternativas de solución

6) Solución adecuada

7) Texto libre

DOC11 - ACCIONES PERFECTIVAS REALIZADAS.

1) Identificación de este documento.

2) Identificación del responsable de la intervención (nombre, departamento, teléfono, etc.).

3) Identificación de la Petición de Modificación (DOC6).

4) Lenguaje de programación utilizado.

5) Descripción de la intervención.

6) Control de dedicaciones (personal y tiempo empleado por cada uno).

7) Tiempo total de la intervención.

8) Otra información de interés (recomendaciones de mantenimiento preventivo, por ejemplo).

9) Texto libre.

DOC12 - LISTA DE ELEMENTOS SOFTWARE Y PROPIEDADES MEJORABLES.

1) Identificación de este documento.

2) Identificación del responsable de la intervención (nombre, departamento, teléfono, etc.).

3) Identificación de la Petición de Modificación (DOC6).

4) Breve descripción de la petición

5) Para cada elemento software, indicar la lista de propiedades valorables (legibilidad, complejidad ciclomática, etc.), el valor actual y el valor deseado. Indicar aquéllos que necesitan mejorar alguna o algunas propiedades.

6) Texto libre.

DOC13 - ACCIONES PREVENTIVAS REALIZADAS.

1) Identificación de este documento.

2) Identificación del responsable de la intervención (nombre, departamento, teléfono, etc.).

3) Identificación de la Petición de Modificación (DOC6).

4) Lenguaje de programación utilizado.

5) Para cada elemento software modificado (que habrá sido identificado en el documento DOC 12), explicar en qué ha consistido la modificación y si se ha cumplido el objetivo de dar a las propiedades el valor deseado.

6) Control de dedicaciones (personal y tiempo empleado por cada uno).

7) Tiempo de la intervención.

8) Texto libre.

DOC16 - MEDIDAS DEL PRODUCTO.

Nota: en cada intervención de mantenimiento se rellenan dos ejemplares de este documento: uno antes de la intervención y otro posterior.

1) Identificación de la petición de modificación.

2) Identificación del producto software.

3) Identificación de los módulos/rutinas/tablas/vistas involucrados en la petición de modificación.

4) Para cada módulo/rutina/tabla/vista identificado en el punto anterior, detallar los valores de métricas de producto que correspondan.

DOC17 - PLAN DE MANTENIMIENTO DEL PERIODO.

1) Identificación de este documento

2) Periodo para el que es válido este documento

3) Valores previstos de:

   3.1 Número de peticiones urgentes

   3.2 Número total de peticiones

   3.3 Horas para peticiones urgentes en el periodo

   3.4 Horas para todas las peticiones en el periodo

   3.5 Tiempos máximos de resolución de anomalías no planificables (correctivo urgente)

   3.6 Tiempos máximos de resolución de anomalías no críticas (correctivo no urgente, perfectivo, preventivo y adaptativo)

4) Texto libre

APÉNDICE II. CAUSAS Y ORÍGENES DE LOS ERRORES

Basili et al. (1998) realizan la siguiente clasificación de los errores humanos en el proceso de mantenimiento, de acuerdo al origen y a la causa del error. Esta clasificación es utilizada en la métrica Bas-OrigenError y Bas-CausaError, utilizada en las tareas NP1.1 y P1.2.

Origen del error	• Cambio en el análisis de requisitos • Cambio en el análisis de localización • Cambio en el análisis de diseño • Codificación
¿Qué provocó el error?	• Desconocimiento del dominio de la aplicación (restricciones operacionales, modelo matemático...) • Desconocimiento del diseño o la implementación del sistema (estructuras de datos, dependencias entre los procesos, restricciones de memoria o rendimiento...) • Requisitos ambiguos o incompletos • Error sintáctico o semántico del lenguaje de programación • Tensión en la planificación • Fallo existente encubierto • Descuido

ACRÓNIMOS

3FN: Tercera Forma Normal.

4FN: Cuarta Forma Normal.

5FN: Quinta Forma Normal.

A: Adaptativo

ANSI: American National Standard Institute.

BCNF: Forma Normal de Boyce-Codd.

CASE: Computer Aided Software Engineering.

CC: Class Complexity/Size

CICS: Customer Information Control System

CICYT: Comisión Interministerial de Ciencia y Tecnología.

COCOMO: Constructive Cost Model

COTS: Commercial-off-the-self

CP: Correctivo no urgente y perfectivo.

ECSMR: European Conference on Software Maintenance and Reengineering (en las ediciones de 1997 y 1998, la *E* significaba *Euromicro* en lugar de *European*)

ICSM: International Conference on Software Maintenance.

IEC: International Electrotechnical Commision.

IEEE: Institute of Electrical and Electronics Engineers.

IPO: Is Part Of.

ISA: Is A.

ISO: International Organization for Standardization.

LDC: Líneas de código.

MINER: Ministerio de Industria y Energía.

NP: No planificable.

OM: Organización de Mantenimiento.

OMT: Object Modelling Technique.

P: Preventivo

SEDISI: Asociación Española de Empresas de Tecnologías de la Información.

SI: Sistema de Información.

SQL: Structured Query Language.

TCA: Tasa de Cambios Anual.

UCLM: Universidad de Castilla-La Mancha.

REFERENCIAS

Albretch, A. J. (1979). Measuring Application Development Productivity. En *Proceedings of the IBM Application Development Symposium* (pp. 83-92). Monterrey, Canadá.

Arnold, R.S. (1986) *An Introduction to Software Restructuring*. Tutorial on Software Restructuring, pp. 1-11. IEEE Computer Society.

Arnold, R. (1992). *Software Reengineering*. IEEE Press.

Avison, D., Lan, F., Myers, M. and Nielsen, A. (1999). Action Research. *Communications of the ACM*, 42(1), 94-97.

Bardou, L. (1997). *Mantenimiento y Soporte Logístico de los Sistemas Informáticos*. Madrid, España: Ra-ma.

Barros S., Bodhuin, T., Escudié, A., Queille, J. P. y Vidrot, J. F. (1995). Supporting Impact Analysis: A Semi-Automated Technique and Associated Tool. En *Proceedings of the International Conference on Software Maintenance* (pp. 42-51). Los Alamitos, California: IEEE Computer Society.

Basili, V., Briand, L., Condon, S., Kim, Y., Melo, W. Y Valett, J.D. (1996). Understanding and Predicting the Process of Software Maintenance Releases. En *Proceedings of the International Conference on Software Engineering* (pp. 464-474). Los Alamitos, California: IEEE Computer Society.

Batini, C. Ceri, S. y Navathe. (1992). *Conceptual Database Design. An Entity-Relationship Approach*. Massachusetts, EE.UU: Addison-Wesley Publishing Company, Inc.

Baxter, I.D. y Pidgeon, W.D. (1997). Software Change Through Design Maintenance. En *Proceedings of the International Conference on Software Engineering* (pp. 250-259). Los Alamitos, California: IEEE Computer Society.

Bennett, K.H.; Martil, R. y Zuylen H.V. (1990). *A Model of Software Reconstruction.* Durham, Reino Unido: Centre of Software Maintenance.

Bennett, K.H. (1991) Automated Support of Software Maintenance, *Information and Software Technology*, 33(1), 74-85.

Berns, G. (1984). Assessing Software Maintainability. *Communications of the ACM*, 27(1), 14-23.

Biggerstaff, Ted. J., Mitbander, Bharat G. y Webster, Dallas E. (1994). Program Understanding and the Conccept Assignment Problem. *Communications of the ACM*, 37(5), 72-83.

Boehm, B.W. (1981). *Software Engineering Economics.* Estados Unidos: Prentice-Hall.

Borne, I. y Romanczuk, A. (1998). Towards a Systematic Object-Oriented Transformation of a Merise Analysis. En Nesi y Lehner (eds.), *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering* (pp. 213-325). Los Alamitos, California: IEEE Computer Society.

Bourke, T.M. (1999). Seven major ICT companies join the European Commission to work towards closing the skills gap in Europe. En *Proceedings of the New Partnerships to Close Europe's Information and Communication Technologies Skills Gap.* Accedido el 10 de enero de 2000 en Internet: http://www.career-space.com/whats_new/press_rel.doc

Brereton, P., Budgen, D. y Hamilton, G. (1999). Hypertext: the Next Maintenance Mountain. *Computer*, 31(12), 49-55.

Briand, L. C., Morasca, S. y Basili, V.R. (1996). Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering*, 22(1), 68-86.

Briand, L., Kim, Y., Melo, W., Seaman, C. y Basili, V.R. (1998a). QMOPP: Qualitative Evaluation of Maintenance Organizations, Processes and Products. *Software maintenance: research and practice*, 10(4). 249-278.

Briand, L.C., El Emam, K. and Bomarius, F. (1998b). COBRA: a Hybrid Method for Software Cost Estimation, Benchmarking and Risk Assessment. En *Proceedings of the 20th International Conference on Software Engineering* (pp. 390-399). Los Alamitos, California: IEEE Computer Society Press.

Briand, L.C., Wüst, J., Ikonomovski, S.V. y Lounis, H. (1998c). Investigating Quality Factors in Object-Oriented Designs: an Industrial Case Study. En *Proceedings of the International Conference on Software Maintenance.*

Brower, J.M. (1999, enero). Outsourcing and Privatizing Information Technology. *Crosstalk, The Journal of Defense Software Engineering*, pp. 28-30.

Bucci, G., Fioravanti, F., Nesi, P. y Perlini, S. (1998). Metrics and Tool for System Assessment. En *Proceedings of the International Conference on Software Engineering* (pp. 36-46). Los Alamitos, California: IEEE Computer Society.

Calero,C. y Piattini, M. (1999). Caracterización formal de métricas para bases de datos relacionales. En Botella, Hernández y Saltor (eds.), *Actas de las IV Jornadas de Ingeniería del Software y Bases de Datos*. Cáceres.

Calzolari, F., Tonella, P. y Antoniol, G. (1998). *Modelling Maintenance Effort by Means of Dynamic Systems.* En Nesi y Lehner (eds.), *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering* (pp. 150-156). Los Alamitos, California: IEEE Computer Society.

Card, D.N. y Glass, R.L. (1990). *Measuring Software Design Quality*. EE.UU.: Englewood Cliffs.

Cherniavsky, J.C. y Smith, C.H. (1991). On Weyuker's Axioms for Software Complexity Measures. *IEEE Transactions on Software Engineering*, 17(6), 636-638.

Chidamber, S.R., Darci, D.P. y Kemerer, F. Managerial use of metrics for object oriented software: an exploration analysis. *IEEE Transactions on Software Engineering*, 24(8), 629-639.

Chidamber, S.R. y Kemerer, F. (1994). A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6), 476-493.

Chikofsky, E.J. y Cross, J.H. (1990). Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software,* 7(1), 13-17.

Consejo Superior de Informática (2000). *Métrica Versión 3*. Disponible en Internet: http://www.map.es/csi/pg5m43.htm (13 de mayo de 2000).

Cremer, K. (1998). A Tool Supporting the Re-Design of Legacy Applications. En Nesi y Lehner (eds.), *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering* (pp. 142-148). Los Alamitos, California: IEEE Computer Society.

De Looff, L. (1997). *Information Systems Outsourcing Decision Making: A Managerial Approach*. Hershey, Palo Alto, EE.UU.: Idea Group Publishing.

De Miguel, A. y Piattini, M. (1997). *Fundamentos y Modelos de Bases de Datos*. Madrid: Ra-Ma.

De Miguel, A., Piattini, M. y Marcos, E. (1999). *Diseño de Bases de Datos Relacionales*. Madrid: Ra-Ma.

De Vogel, M. (1999). Outsourcing and Met*rics.* En Hoof van Huyduynen, Poels, Peeters y Nevalainen (eds.), *Proceedings of the Second European Software Measurement Conference* (pp. 217-225). Federation of European Software Metrics Association y Technologisch Instituut vzw.

Dolado, J. y Fernández, L. (1999). ¿Merece la pena usar los puntos de función? *Novática,* (140), 57-62.

ECMA (1990). *ECMA: Portable Common Tool Environment*. (Technical Report ECMA-149). Ginebra (Suiza): European Computer Manufacturers Association

Euromethod (1996). *Euromethod ver. 1*. Euromethod Project.

Fenton, N. E. y Pfleeger, S. L. (1997). *Software Metrics. A Rigorous & Practical Approach*. International Thomson Computer Press.

Fioravanti, F., Nesi, P. y Stortoni, F. (1999a). Metrics for Controlling Effort During Adaptive Maintenance of Object Oriented Systems. En *Proceedings of the International Conference on Software Maintenance*. Los Alamitos, California: IEEE Computer Society Press.

Fioravanti, F., Nesi, P. y Polo, M. (1999b). *Complexity/Size Metrics for Object-Oriented Systems. Technical Report.* (DSI-RT 17/99). Florencia, Italia: Department of Systems and Informatics, Università di Firenze.

Frazer, A. (1992). *Reverse Engineering-Hype, Hope or Here? Software Reuse and Reverse Engineering in Practice*. Chapman & Hall.

French, W.L. y Bell, C.H. Jr. (1996). *Desarrollo organizacional (quinta edición)*. Naucalpán de Juárez, México: Prentice-Hall.

Fuggetta, A. (1999). Rethinking the models of software engineering research. *The Journal of Systems and Software,* (47), 133-138.

Gamalel-Din, S.A. y Osterweil, L.J. (1988). New Perspectives on Software Maintenance Process. En *Proceedings of the International Conference on Software Maintenance*. Los Alamitos, California: IEEE Computer Society Press.

Genero, M., Piattini, M., Calero, C., Polo, M. y Ruiz, F. (1999). Calidad de esquemas conceptuales. *Novática*, (140), 23-27.

Gillibrand, D. and Kecheng, L. (1998). Quality Metrics for Object-Oriented Design. *JOOP: The Journal of Object-Oriented Programming,* 10(8), 56-59.

Graham, I., Henderson-Sellers, B. y Younessi, H. (1997). *The OPEN Process Specification*. Essex, Reino Unido: ACM Press y Addison-Wesley.

Granja Álvarez, J. C. y Barranco García, M. (1997) A Method for Estimating Maintenance Cost in a Software Project: a Case Study. *Journal of Software Maintenance: research and practice*, 9(3), 161-175.

Gray, L. (1996). ISO/IEC 12207 Software Lifecycle Processes. *Crosstalk, The Journal of Defense Software Engineering*, 9(8).

Gray, R.H.M., Carey, B.N., McGlynn, N.A. y Pengelly, A.D. (1991). Design metrics for database systems. *BT Technology*, 9(4), 69-79.

Griswold, W.G. y Notkin, D. (1993). Automated Assistance for Program Restructuring. *ACM Transactions on Software Engineering and Methodology*, 2(3), 228-269.

Hanna, M. (1993, abril). Maintenance Burden Begging for a Remedy. *Datamation*, pp. 53-63.

Henderson-Sellers, B. (1996). *Object-oriented metrics. Measures of complexity.* Prentice Hall.

Hoffman, T. (1997, marzo). Users Say Move Quickly when Outsourcing your Personnel. *Computer World*, p.77.

IEEE (1990). *ANSI/IEEE Standard 610: IEEE Standard Glossary of Software Engineering Terminology*. Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

IEEE (1987). *ANSI/IEEE, std 1042: Guide to Software Configuration Management.* Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

IEEE (1991). *IEEE Std. 1074-1995, IEEE Standard for Developing Software Life Cycle Processes*. Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

IEEE (1992a). *IEEE Std 1209-1992: IEEE Recommend practice for the evaluation and selection of CASE tools*. Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

IEEE (1992b). *IEEE Std 1219-1992, Standard for Software Maintenance*. Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

IEEE (1992c). *IEEE Standard for a Software Quality Metrics Methodology*. Los Alamitos, California: IEEE Computer Society Press.

IEEE (1995). *IEEE Std 1348-1995, IEEE Recommended practice for the adoption of Computer-Aided Software Engineering (CASE) Tools.* Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

ISO/IEC (1991). *ISO 9126: Information Technology-Software Product Evaluation-Quality Characteristics and Guidelines for Their Use*. Ginebra, Suiza.

ISO/IEC (1995), International Standard Organization /International Electrotechnical Commission. *ISO/IEC 12207: Information Technology-Software Life Cycle Processes*. Ginebra, Suiza.

ISO/IEC (1998), Intemational Standard Organization /International Electrotechnical Commission. *ISO/IEC 9126: Software Quality Characteristics and Metrics*. Canadá.

ISO/IEC (1999), International Standard Organization /International Electrotechnical Commission. *FDIS 14764: Software Engineering - Software Maintenance*. Borrador del futuro estándar ISO/IEC 14764.

Jørgensen, M. (1995). Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. *IEEE Transactions on Software Engineering*, 21(8), 674-681.

Jahnke, J.H. y Wadsack, J. (1999). Integration of Analysis and Redesign Activities in Information System Reengineering. En Nesi y Verhoef (eds.), *Proceedings of the Third European Conference on Software Maintenance and Reengineering* (pp. 160-168)). Los Alamitos, California: IEEE Computer Society.

Jones, C. (1996). *Applied Software Measurement: Assuring Productivity and Quality*. McGraw-Hill.

Karolak, D.W. (1996). *Software Engineering Risk Management*. Los Alamitos, California: IEEE Computer Society Press.

Kellner, M.I. y Hansen, G.A. (1988). *Software Process Modelling*. Pittsburg, EE.UU.: Software Engineering Institute, Carnegie Mellon University.

Kesch, S. (1991). Evaluating the quality of entity relationship models. *Information and Software Technology*, 37(12), 681-689.

Kilpi, T. (1997). New Challenges for Version Control and Configuration Management: a Framework and Evaluation. En *Proceedings of the 1st Euromicro Conference on Software Maintenance and Reengineering*. Los Alamitos, California: IEEE Computer Science.

Kitchenham, B., Pfleeger, S.L. y Fenton, N. (1995). Towards a Framework for Software Measurement Validation. *IEEE Transactions on Software Engineering*, 21(12), 929-944.

Kitchenham, B. y Stell, J.G. (1997, octubre-diciembre). The danger of using axioms in software metrics. IEE Proceedings of Software Engineering, 144(5-6), 279-285.

Klepper, R. y Jones, W.O. (1998). *Outsourcing Information Technology, Systems and Services*. Nueva Jersey, EE.UU.: Prentice-Hall.

Kung, H.-J. y Hsu, Ch. (1998). Software Maintenance Life Cycle Model. En Khoshgoftaar y Bennett (eds.), *Proceedings of the International Conference on Software Maintenance* (pp. 113-121). Los Alamitos, California: IEEE Computer Society.

Lam, W. y Loomes, M. (1998). *Requirements evolution in the midst of environmental change: a managed approach.* En Nesi y Lehner (eds.), *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering* (pp. 121-127). Los Alamitos, California: IEEE Computer Society.

Lear, A.C. (2000). Cobol programmers could be key to new IT. *Computer,* 33(4), 19.

Lehman, M, M. (1980) *Programs, Life Cycles and Laws of Software Evolution.* Proceedings of the IEEE, 68(9), 1060-1076.

Lehman, M.M., Perry, D.E. y Ramil, J.F. (1998). Implications of Evolution Metrics on Software Maintenance. En Khoshgoftaar y Bennet (eds.), *Proceedings of the International Conference on Software Maintenance* (pp. 208-217). Maryland, EE.UU.: IEEE Computer Society. Estados Unidos.

Li, H. F. y Chen, W. K. (1987). An empirical study of software metrics. *IEEE Transactions on Software Engineering,* 12 (6), 679-708.

Li, W. y Henry, S. (1993). Object-oriented metrics that predict maintainability. *The Journal of Systems and Software,* (23), 111-122.

Lientz, B.P. y Swanson, E.F. (1980). *Software Maintenance Management*. Addison-Wesley.

Lorenz, M. y Kidd, J. (1994). *Object-oriented software metrics, a practical guide*. Nueva Jersey: Prentice-Hall.

Mazza, C., Fairclough, J., Melton, B., de Pablo, D., Scheffer, A. and Stevens,R. (1994). *Software Engineering Standards*. Prentice-Hall.

Mazza, C., Fairclough, J., Melton, B., de Pablo, D., Scheffer, A. and Stevens, R. (1996) *Software Engineering Guides*. Reino Unido: Prentice Hall.

McCabe, T. (1976) A Software Complexity Measure. *IEEE Transactions on Software Engineering*, 2(4), 308-320.

McCall, J., Richards, P. and Walters, G. (1977). *Factors in Software Quality.* Vols I, II and III. NTIS AD-a049-014, 015, 055. Rome: US Rome Air Development Center.

McConnell, S. (1998). Desarrollo y gestión de proyectos informáticos. Madrid: McGraw Hill.

McKee, J.R. (1984). Maintenance as a Function of Design. En *Proceedings of the National Computer Conference* (pp. 187-193).

McTaggart, R. (1991). Principles of Participatory Action Research. *Adult Education Quarterly*, 41(3).

Melton, A. (editor) (1995). Software Measurement. International Thompson Computer Press.

MINER-SEDISI (1999), Ministerio de Industria y Energía y Asociación Española de Empresas de Tecnologías de la Información. *Las Tecnologías de la Información en España 1998. Colección Informes y Estudios.* Madrid: Centro de Publicaciones del Ministerio de Industria y Energía.

Moore, J.W. (1998). The Systems Engineering Process Office. ISO/IEC 12207 and Related Software Life-Cycle Process Standards. SEPO TechNote No. 17, R1.

Muller, R.J. (1999). Database Design for Smarties - Using UML for Data Modeling. Morgan Kaufmann Publishers, Inc.

Neitzel, A.C. (1999). Managing Risk Management. *Crosstalk, The Journal of Defense Software Engineering*, 12(7), 17-21.

Nesi, P., Fioravanti, F. y Polo, M. (1999). *Complexity/Size Metric for Object Systems.* Enviado a IEEE Transactions on Software Engineering.

Nesi, P. y Querci, T. (1998). Effort estimation and prediction of object-oriented systems. *The Journal of Systems and Software*, (42), 89-102.

Niessink, F. y van Vliet, H. (1997). Predicting Maintenance Effort with Function Points. En Harrold y Visaggio (eds.), *Proceedings of the International Conference on Software Maintenance* (pp. 32-39). Los Alamitos, California: IEEE Computer Society.

Norman, S. (1992). *Dynamic Testing Tools, a Detailed Product Evaluation*. Ed. Ovum.

O'Neill, D. (1997). Software Maintenance and Global Competitiveness. *Journal on Software Maintenance: Research and Practice*, 9(6), 379-399.

Padak, N. y Padak, G. (1994). *Guidelines for Planning Action Research Projects.* Disponible en Internet: http://archon.educ.kent.edu/Oasis/Pubs/0200-08.html

Pedro de Jesus, L. y Sousa, P. (1998). Selection of Reverse Engineering Methods for Relational Databases. En Nesi y Verhoef (eds.), *Proceedings of the 3rd European Conference on Software Maintenance* (pp. 194-197). Los Alamitos, California: IEEE Computer Society.

Penteado, R., Masiero, P. y Cagnin, M. (1999). An Experiment of Legacy Code Segmentation to Improve Maintainability. En Nesi y Verhoef (eds.), *Proceedings of the Third European Conference on Software Maintenance and Reengineering* (pp. 111-119). Los Alamitos, California: IEEE Computer Society.

Pfleeger, S.L. (1995). Experimental Design and Analysis in Software Engineering. En *Annals of Software Engineering,* (pp. 219-253). J.C. Baltzer AG, Science Publishers.

Piattini, M. G., Calvo-Manzano, J. A., Cervera, J. y Fernández, L. (1996). *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión*. Madrid: Ra-Ma.

Piattini, M., Villalba, J., Ruiz, F., Fernández, I., Polo, M., Bastanchury, T. y Martínez, M.A. (1998a). *Mantenimiento del software: conceptos, métodos, herramientas y outsourcing*. Madrid: Ra-Ma.

Piattini, M., Calero, C., Polo, M. y Ruiz, F. (1998b). Maintainability in Object-Relational Databases. En Coombes, Hooft van Huysduynen y Peeters (eds.), Proceedings of the First European Software Measurement Conference (pp. 223-230). Technological Institute.

Piattini, M., Genero, M., Calero, C., Polo, M. y Ruiz, F. (2000). Database Quality. En Piattini y Díaz (eds.), *Advanced Databases: Technology and Design*. Londres, Reino Unido: Artech House.

Pigoski, T. M. (1997). *Practical Software Maintenance. Best Practices for Managing Your Investment*. Estados Unidos: John Wiley & Sons.

Plaisant, C., Rosse, A., Shneiderman, B. y Vanniamparampil, A.J. (1997). Low-Effort, High-Payoff User Interface Reingeneering. *IEEE Software*, 14(4), 66-72.

Poels, G. (1997). An analytical Evaluation of Static Coupling Measures for Domain Object Classes. En Melo, Morasca y Sahraouis (eds.), *Proceedings of the Object-Oriented Product Metrics for Software Quality Assessment Workshop* (pp. 52-58), dentro de la 12th European Conference on Object Oriented Programming.

Polo, M., Calero, C., Ruiz, F. y Piattini, M. (1998). Métricas de calidad y complejidad para bases de datos. En Toval y Nicolás (eds.), *Actas de las III Jornadas de Ingeniería de Software* (pp. 79-90). Murcia: DM Librero-Editor.

Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999a). MANTEMA: a Complete Rigorous Methodology for Supporting Maintenance based on The ISO/IEC 12207 Standard. En Nesi y Verhoef (eds.), *Proceedings of the Third European Conference on Software Maintenance and Reengineering* (pp. 178-181). Los Alamitos, California: IEEE Computer Society.

Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999b). Using the ISO/IEC 12207 Tailoring Process for Defining a Maintenance Process. En Jakobs y Williams (eds.), *Proceedings of the First IEEE Conference on Standardisation and Innovation in Information Technology* (pp. 205-210). Los Alamitos, California: IEEE Computer Society.

Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999c). MANTEMA: a Software Maintenance Methodology Based on the ISO/IEC 12207 Standard. En *Proceedings of the 4th International Software Engineering Standards Symposium and Forum* (pp. 76-81). Los Alamitos, California: IEEE Computer Society.

Polo, M., Piattini, M., Ruiz, F. y Calero, C. (1999d). Roles in the Maintenance Process. *ACM Software Engineering Notes*, 24(4), 84-86.

Polo, M., Garbajosa, J., Piattini, M. y Ruiz, F. (2000). Métodos y técnicas para la mejora del proceso de mantenimiento. Aceptada para su publicación en las Actas del VII Congreso Internacional de Nuevas Tecnologías y Aplicaciones Informáticas. La Habana (Cuba).

Poston, R. M. y Sexton, M.P. (1992) Evaluating and Selecting Testing Tools. *IEEE Software*, 9(3), 33-42.

Prather, R.E. (1984). An Axiomatic Theory of Complexity Measure. *The Computer Journal*, 27(4), 340-346.

Premerlani, W.J. y Blaha , M.R. (1994). An Approach for Reverse Engineering of Relational Databases. *Communications of the ACM*, 37(5), 42-49.

Pressman, Roger S. (1993). *Ingeniería del Software, un enfoque práctico* (3ª edición). Madrid: McGraw-Hill.

Pressman, R.S. (1998). *Ingeniería del Software. un enfoque práctico* (4ª edición). Madrid: McGraw-Hill.

Project Management Institute (1996). *A Guide to the Project Management Body of Knowledge.*, Upper Darby, PA, EE.UU.: Project Management Institute

Putnam, L. (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering*, 4(4).

Quang, P. T. (1993). Réussir la Mise en Place du Génie Logiciel et de l'Assurance Qualité. París: Eyrolles.

Rao, H.R., Nam, K. and Chaudhury, A. (1996). Information Systems Outsourcing. *Communications of the ACM*, 39(7), 27-28.

Reingruber, M. C. and Gregory, W. W. (1994). *The Data Modeling Handbook*. A Best-Practice Approach to Building Quality Data Models. John Wiley & Sons, Inc.

Rock-Evans, R. y Hales, K. (1990). *Reverse Engineering: Market, Methods and Tools.*

Rohatgi, V.K. (1976). *An introduction to Probability Theory and Mathematical Statistics*. Wiley Series in Probability and Mathematical Statistics.

Ruiz, F., Piattini, M., Polo, M. y Calero, C. (1999a). Maintenance Types in the MANTEMA Methodology. *Proceedings of the First International Conference on Enterprise Information Systems*, (p. 773). Setúbal, Portugal: Instituto Tecnológico de Setúbal.

Ruiz, F., Piattini, M., Polo, M. y Calero, C. (1999b). Auditoría del mantenimiento del software: propuesta de objetivos de control. *Actas del II Congreso Nacional de Auditoría y Control de Sistemas de Información* (pp. 128-143). Valencia: Servicio de Publicaciones de la Universidad Politécnica de Valencia.

Schach, S.R. (1990). *Software Engineering*. EE.UU.: Irwin & Aksen.

Schach, S.R. (1992). *Practical Software Engineering*. EE.UU.: Irwin & Aksen.

Schamp, A. (1995, junio) Configuration Management Tool Evaluation and Selection. *IEEE Software*, pp. 114-118.

Schneidewind, N.F. (1987). The State of Software Maintenance. *IEEE Transactions on Software Engineering*, 13(3), 303-310.

Schneidewind, N.F. (1992). Methodology for validating software metrics. *IEEE Transactions on Software Engineering,* 18(5), 410-421.

Schneidewind, N.F. (1997a). *Measuring and Evaluating Maintenance Process Using Reliability, Risk and Test Metrics.* En *Proceedings of the 19th International Conference on Software Maintenance* (pp. 232-239). Los Alamitos, California: IEEE Computer Society Press.

Schneidewind, N.F. (1997b). *Software metrics for quality control.* En Proceedings of the Fourth International Software Metrics Symposium. IEEE Computer Society Technical Council on Software Engineering.

Schneidewind, N.F. (1998, julio-agosto). How to Evaluate Legacy System Maintenance. *IEEE Software*, pp. 34-42.

Seaman, C.B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering,* 25(4), 557-572.

Sellink, A., Sneed, H. y Verhoef, C. (1999). Reestructuring of COBOL/CICS Legacy Systems. En Nesi y Verhoef (eds.), *Proceedings of the Third European Conference on Software Maintenance and Reengineering* (pp. 72-82). Los Alamitos, California: IEEE Computer Society.

Shepperd, M., Schofield, C. and Kitchenham, B. (1996). Effort Estimation Using Analogy. En *Proceedings of 18th International Conference on Software Engineering*. Los Alamitos, California: IEEE Computer Society Press.

Sherer, S.A. (1997). Using Risk Analysis to Manage Software Maintenance. *Journal on Software Maintenance: Research and Practice*, 9(6), 345-364.

Sierra, R. (1994). *Análisis Estadístico Multivariable. Teoría y Ejercicios.* Madrid: Paraninfo.

Singer, J. (1998). Practices of Software Maintenance. En Khoshgoftaar y Bennet (eds.), *Proceedings of the International Conference on Software Maintenance* (pp. 139-145). Los Alamitos, California: IEEE Computer Society.

Slaughter, S.A. y Banker, R.D. (1996). A Study of the Effects of Software Development Practices on Software Maintenance Effort. En *Proceedings of the International Conference on Software Maintenance* (pp. 197-205). Los Alamitos, California: IEEE Computer Society.

Sneed, H. M. (1995). Planning the Reengineering of Legacy Systems. *IEEE Software*, 12(1), 24-34.

Sneed, H.M. y Foshag, O. (1998). Measuring legacy database structures. En Coombes, Hooft van Huysduynen y Peeters (eds.), Proceedings of the First European Software Measurement Conference. Technological Institute.

Sommerville, I. (1992). *Software Engineering, Fourth Edition.* Reading, Massachussetts: Addison-Wesley.

Stoecklin, S., Williams, D. y Stoecklin, P. (1998). Tailoring the process model for maintenance and reengineering. En Nesi y Lehner (eds.), *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering* (pp. 209-212). Los Alamitos, California: IEEE Computer Society.

Swanson, E.B. (1999). IS "Maintainability": Should it Reduce the Maintenance Effort? *The DATA BASE for Advances in Information Systems*, 30(1), 65-76.

Taschwer, M., Rauner-Reithmayer, D. y Mittermeir, R. (1999). Generating Objects from C Code. Features of the CORET Tool-Set. En Nesi y Verhoef (eds.), *Proceedings of the Third European Conference on Software Maintenance and Reengineering* (pp. 91-100). Los Alamitos, California: IEEE Computer Society.

Wadswroth, Y. (1998). What is Participatory Action Research? *Action Research International*. Accedido el 10 de enero de 2000 en Internet: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html

Weide, B. W., Heym, W. D. y Hollingsworth, J. E. (1995). Reverse Engineering of Legacy Code Exposed. En *Proceedings of the 17th International Conference on Software Engineering* (pp. 327-331). Los Alamitos, California: IEEE Computer Society.

Weyuker, E.J. (1988). Evaluating Software Complexity Measures. *IEEE Transactions on Software Engineering*, 14(9), 1357-1365.

Whitmire, S.A. (1997). *Object-Oriented Design Measurement*. Nueva York: John Wiley & Sons, Inc.

Willcocks, L.P. y Lacity, M. C. (1999). IT outsourcing in insurance services: risk, creative contracting and business advantage. *Information Systems Journal,* 9(3), 163-180.

Zuse, H. (1991). Software Complexity: Measures and Methods. Berlín: DeGruyter Publisher.

Zuse, H. (1998). A framework of Software Measurement. Berlín, Nueva York: Walter de Cruyter.