Guía para la Gestión de la Fiabilidad y Mantenibilidad del Software

Traducción al español de la ARMP-9 Edición 1. Propuesta de PEFM-9

Grupo 1 y Grupo 2

Comité de Industrias y Servicios para la Defensa

Edición 1, Mayo 2013

Página intencionadamente en blanco

COPYRIGHT

PRIMERA EDICIÓN: 21 de mayo de 2013

Este documento ha sido elaborado por el G-1 de SW y G-2 de Fiabilidad y Mantenibilidad en Defensa. Este grupo está formado por voluntarios pertenecientes a diversas empresas participantes en el Comité de Industrias para la Defensa de la Asociación Española de la Calidad (Q-AEC).

ASOCIACIÓN ESPAÑOLA PARA LA CALIDAD
Claudio Coello 92
28006 Madrid
www.aec.es
publi@aec.es

Este documento se publica bajo licencia *Creative Commons* del tipo:

Reconocimiento – NoComercial – SinObraDerivada (by-nc-nd). Se permite su copia y distribución por cualquier medio siempre que mantenga el reconocimiento de sus autores. No se permite un uso comercial de la obra original ni la generación de obras derivadas.
La licencia completa puede consultarse en:
http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es

Página intencionadamente en blanco

AUTORES Y APORTES AL DOCUMENTO

Se agradece cualquier comentario o contribución que pueda mejorar la presente versión. Para ello, por favor dirigir las propuestas a:

D. Juan Bautista Pérez Mínguez
Área de Calidad de la SDGTIC
Ministerio de Defensa
Calle Arturo Soria, 289. 28033 Madrid

Han participado en la elaboración de la traducción del la guía:

Cor. D. Juan Bautista Pérez Mínguez	MINISTERIO DE DEFENSA
D. José Gómez-Randulfe	TECNOBIT
Fernando R. Armada García	NAVANTIA
Isidoro Atienza Manso	INDRA
José Luis Báez García	FCC INDUSTRIAL
Francisco Javier Casquero García	GTD
Estela Fernández Menéndez	SENER
María de Frutos Gómez	INFORMÁTICA EL CORTE INGLÉS
Carlos José García Gallardo	UNITRONICS
Carmen García Zafra	
Cte. Estrella Garrido Garrido	MINISTERIO De DEFENSA
Dolores Guerra Pérez	SIMAVE
Belén Lasanta Herrero	SENER
Manuel Arturo Lea Pereira	GMV AEROSPACE & DEFENCE S.A
Francisco Javier Magaña Tristante	EADS
Raquel Martín Gómez	MINISTERIO DE DEFENSA
Jesús Megía Vega	TECNOBIT
Concepción Torrijos Rodriguez	INSA
Cte. Javier Velasco Prieto	MINISTERIO De DEFENSA
Sergio Vicens	TELEFÓNICA

Página intencionadamente en blanco

LISTA DE PÁGINAS EFECTIVAS

El número total de páginas de esta publicación es de 53, consisten en lo siguiente: 6 páginas de control del documento y 41 páginas de capítulos y anexos.

Este documento ha sido:

	Firma	Fecha
Preparado por:	G1 y G2	Mayo 2013
Revisado por:	Interna y Externa	Mayo 2013
Aprobado por:	Comité Industrias y Servicios Defensa	Mayo 2013

Página intencionadamente en blanco

HOJA DE REVISIONES

Edición	Fecha	Propuesta de alteración	Páginas revisadas
0	Mayo 2013	-	Documento Original

Página intencionadamente en blanco

ÍNDICE DE CONTENIDOS

Guía para la Gestión de la Fiabilidad y Mantenibilidad del Software

RESUMEN EJECUTIVO

Muchos Sistemas de Defensa se basan en el uso de ordenadores para funcionar adecuadamente; por consiguiente es esencial que el Software que los soporta se gestione como parte integrante del sistema completo. Aunque la naturaleza del Hardware y del Software es diferente, existen muchas similitudes que deberían explotarse para manejar de forma efectiva el Software.

Es comúnmente conocido que, pese al esfuerzo en el diseño y las pruebas, no es posible producir software sin defectos. Por tanto, la especificación de la Fiabilidad del Software debiera incluir objetivos como evitar, detectar y corregir los defectos, así como la tolerancia a los mismos, adicionalmente a la totalidad de los requisitos cuantitativos. Los mecanismos fundamentales para la gestión basada en el funcionamiento de un programa de Fiabilidad de Software se contienen en el "Plan de Fiabilidad y Soportabilidad (FyS) del Software"[1] y el "Repositorio de Fiabilidad y Soportabilidad (FyS) del Software". El Plan y el Repositorio son herramientas de gestión de propósito general que son adaptables para su empleo en distintos campos de la ingeniería de sistemas.

El mayor o menor éxito de cualquier proyecto reside en gran parte en la calidad de los contratos de adquisición y apoyo. Si éstos son exigentes, están bien escritos y de forma exhaustiva, habrá una elevada probabilidad de que el producto resultante cumpla los requisitos y proporcione la capacidad deseada durante el ciclo de vida completo del equipo a un coste óptimo.

[1] SAE JA –1002 par 4.3 "Management Framework".

Página intencionadamente en blanco

1 INTRODUCCIÓN

(1) General

Con la creciente sofistificación de los equipos modernos, la miniaturización de la electrónica y el cambio del procesamiento analógico al digital, cada vez más un mayor número de sistemas incorpora algún tipo de tecnología de computación para monitorizar o controlar su funcionamiento. Si bien esto es lógico en plataformas complejas como barcos o aviones, también se da en dispositivos domésticos como teléfonos móviles, lavadoras… En consecuencia, los nuevos sistemas se están haciendo cada vez más dependientes del Software para mejorar sus prestaciones.

Para los equipos de Defensa, el Software forma una parte integral del sistema y debe considerarse en conjunto con el resto de características, a lo largo del ciclo de vida completo del equipo. Las características individuales de cualquier Software contribuirán a los niveles totales de Fiabilidad y Mantenibilidad alcanzados por el sistema como un todo.

Aunque el Software comparte multitud de atributos con los sistemas mecánicos y electrónicos, también posee algunas propiedades especiales: de hecho no se desgasta o degrada con el tiempo[2]. Muchos de los problemas asociados con los sistemas híbridos surgen de la interfaz entre el Software y los elementos físicos del sistema. En cualquier caso, como parte de un programa completo para el sistema, el Software debería gestionarse de forma similar al hardware. Este proceso debe iniciarse lo más pronto posible si se han de minimizar los riesgos del proyecto.

Mientras que la Fiabilidad es tan importante para el Software, como lo es para los sistemas mecánicos o electrónicos, el concepto de Mantenibilidad no es tan fácil de aplicar. En un contexto puramente de Software, la Mantenibilidad se ha sustituido por el término Soportabilidad, que es un concepto más amplio que comprende el diseño, herramientas, métodos y entorno de soporte. El concepto de Fiabilidad y Soportabilidad (FyS) se ha desarrollado en la comunidad del Software durante años para reflejar las actividades requeridas para desarrollar y mantener el Software durante el ciclo de vida de un producto (ver SAE JA1002, AAP-48).

A pesar de la expansión de los ordenadores en todos los aspectos de la vida y los millones de líneas de código escritas cada año, solamente una pequeña parte de los sistemas de software entran en explotación en plazo y coste. Esto representa una enorme pérdida de inversión y pone en riesgo el funcionamiento operativo. Por ello es esencial que los requisitos del Software se determinen cuidadosamente y se especifiquen con precisión, se demuestren adecuadamente y que la adquisición se gestione apropiadamente.

La gestión del Software en general y de la FyS del Software en particular puede suponer un reto. La aplicación de buenas técnicas y herramientas de Ingeniería del Software, en el marco completo de los sistemas es esencial. No obstante, las metodologías deben ser adaptadas para cumplir las necesidades de los proyectos particulares. Para minimizar los riesgos del proyecto, debiera disponerse de consejo y asistencia especializados durante la Fase de Concepto.

Frecuentemente se emplea en los sistemas de defensa software comercial (COTS -*Commercial Off The Shelf*-). Sin embargo, hay algunas consideraciones especiales, que solo se aplican a COTS y que pueden incluir: pérdida de control de la configuración, licenciamiento, obsolescencia y

[2] El Software no se desgasta en un sentido mecánico; no obstante el cambio de entorno operacional, produce la obsolescencia del software y puede dar lugar a similares condiciones.

Soportabilidad. Estos elementos deberían considerarse activamente en un Plan de Gestión del Ciclo de Vida, ya que pueden tener un impacto significativo en los costes de soporte a lo largo de la vida del sistema.

A la vista de la diversidad y complejidad del Software, no existe una solución universal para la entrega de programas eficientes, y menos aún fiables y soportables.

1.1 Objetivo

El Objetivo de la PEFM-9 es proporcionar una guía para los Patrocinadores[3], Gestores, personal de Proyectos y Suministradores para la adquisición y el mantenimiento de sistemas Software Fiables y Soportables, durante el ciclo de vida del equipo.

1.2 Propósito

El propósito de esta guía es:

a) Describir la naturaleza y la importancia del Software;

b) Describir la especificación de los requisitos de FyS del Software;

c) Describir técnicas y herramientas que puedan usarse para evaluar y demostrar la FyS del Software;

d) Describir los principios y el marco requeridos para la adquisición y el sostenimiento de Software fiable y soportable;

e) Describir las funciones y responsabilidades asociadas con la adquisición y el sostenimiento de Software fiable y soportable;

f) Describir las disposiciones recomendadas a incluir en los contratos para adquirir y sostener Software fiable y soportable.

1.3 Aplicabilidad

Esta guía es aplicable a proyectos nacionales y de colaboración para Tierra, Mar y Aire o sistemas conjuntos para el uso de los patrocinadores, gestores y suministradores que tengan responsabilidad en el desarrollo, adquisición o soporte del Software. El documento pretende proporcionar un marco integral y unos principios rectores para una gestión eficiente de la FyS del Software durante el ciclo de vida del equipo.

La guía se aplica principalmente al Software usado en los sistemas de Defensa o en apoyo directo a los mismos, incluyendo:

- Software de aplicación;
- Software a medida de misión crítica;
- Software COTS adaptado para aplicaciones específicas;
- Sistema operativo;
- Software crítico para la seguridad de funcionamiento y de las personas (safety).

[3] Los patrocinadores representan al Usuario y fijan los requisitos completos del sistema y proporcionan financiación al programa.

No obstante, los principios también podrían aplicarse a software más general, incluyendo Aplicaciones Empresariales[4].

1.4 Documentación Relacionada

STANAG 4174	*Allied Reliability and Maintainability Publications*
ARMP-1	*NATO Requirements for Reliability and Maintainability.*
ARMP-4	*Guidance for Writing NATO R&M Requirement Documents.*
ARMP-6	*Guidance for Managing In-Service R&M.*
ARMP-7	*NATO R&M Terminology Applicable to ARMPs.*

[4] Ver definición: Tal como CAN MASIS que proporciona una visibilidad total de activos

Página intencionadamente en blanco

2 LA NATURALEZA DEL SOFTWARE

(2) En términos sencillos, el Software es un conjunto de instrucciones, que permite a un ordenador efectuar una función dada. El conjunto de instrucciones consiste en una serie de instrucciones secuenciales o líneas de código simples, que pueden contener los componentes de algoritmos complejos que requieren ejecución en paralelo. Las instrucciones de Software, pueden escribirse en una variedad de lenguajes de programación, que son traducidos o convertidos a instrucciones máquina. Este proceso ha evolucionado con el tiempo para permitir a los programadores escribir código de ordenador de forma más fácil y conveniente. Los programas Software pueden variar de tamaño desde unas pocas líneas de código para una aplicación muy simple hasta varios millones de líneas para sistemas complejos o sistemas de sistemas. En plataformas sofisticadas tales como un caza, muchos ordenadores se pueden conectar para formar una red distribuida y se pueden ejecutar concurrentemente un número significativo de programas, para operar eficientemente el sistema completo.

(2.1) El Software difiere del hardware en algunos aspectos que se pueden resumir en los siguientes:

a) El Software no es un producto tangible como lo es una radio o un motor, sino un conjunto de instrucciones codificadas que se almacenan en un dispositivo de memoria como es una PROM, EEPROM, CD, lápiz de memoria, disco flexible o disco duro.

b) A diferencia del hardware, la transición entre desarrollo y producción del Software, es relativamente simple, aunque el proceso requiere pruebas extensivas y la documentación apropiada. Una vez que el desarrollo se ha finalizado se pueden hacer cualquier número de copias del Software maestro. Las características inherentes, incluyendo las de FyS de cada copia, serán idénticas a las del programa maestro, suponiendo que los factores externos tales como plataforma de ejecución, otras aplicaciones que se ejecutan en la plataforma y conexiones de red, sean también idénticas si las hay. En consecuencia, las reproducciones múltiples del Software, también incluirán cualquier defecto que pudiera estar presentes en la versión maestra.

c) En principio, los programas Software pueden ser usados cualquier número de veces sin desgaste. Aunque el Software no se desgaste sufre los efectos de la obsolescencia. Este es un hecho importante a tener en cuenta. El ciclo de vida del Software es a menudo similar al del hardware sobre el que funciona, que puede variar desde un ordenador personal a un complejo sistema de armas. Cuando el hardware envejece y la tecnología cambia, se efectúan modificaciones y se introducen nuevas interfaces. Estos cambios de hardware pueden requerir modificaciones al Software, que a menudo hacen que se introduzcan defectos. Cuando se encuentran, estos defectos requieren nuevas correcciones. La documentación de apoyo, si no mantiene el nivel de precisión original, podría producir dificultades cuando se pretende mantener el control de la configuración del Software. Hacia el final de la vida de servicio de un sistema, la disponibilidad de los programadores, conocedores del lenguaje o código original del Software podrían también ser escasos. Finalmente las herramientas de desarrollo y mantenimiento del Software que son paquetes especializados, podrían no funcionar en el nuevo hardware, y por lo tanto introducir riesgos adicionales para el mantenimiento de la aplicación Software objetivo.

d) El Software no se ve afectado por las condiciones físicas, aunque el hardware que lo soporta sí puede verse afectado.

e) El Software puede almacenarse durante varios años sin disposiciones especiales, aunque el medio en que se almacena puede ser objeto de deterioro y de obsolescencia.

f) Se admite frecuentemente que el Software es fiable ya que no se rompe en un sentido mecánico. No obstante, puede fallar dejando de operar correctamente bajo ciertas circunstancias como resultado de errores de diseño o pruebas incompletas.

g) Los defectos no detectados permanecerán en el Software y podrían producir fallos, dependiendo de los datos de entrada o las entradas desde otros sistemas. La acción correctiva debe implicar a la autoridad de diseño o a la organización designada para el soporte y el control de la configuración del Software a mantener.

h) La ingeniería del Software ha desarrollado un vocabulario específico que necesita ser entendido para gestionar adecuadamente el Software. Mientras muchos términos son similares a los de la ingeniería tradicional, otros son específicos del Software (ver ARMP-7 y Anexo A).

i) En general, es más difícil efectuar pruebas integrales al Software que al Hardware. No obstante, la aceptación final debería incluir una prueba completa de sistema (utilizando Hardware y Software integrados), en un entorno representativo del entorno de operación.

(2.2) Como muchos sistemas de Defensa se basan en ordenadores para funcionar adecuadamente, es esencial que el Software que los soporta, se gestione como una parte integrante del sistema completo. Aunque la **naturaleza** del Hardware y del Software es diferente, existen múltiples similitudes que deben explotarse para manejar de forma efectiva el Software.[5]

La mayoría del Software poseerá las siguientes características:

a) **Lenguaje**: Los programas Software se escriben en uno de los muchos lenguajes de programación. Estos abarcan desde lenguajes genéricos de bajo nivel, como Código Máquina, hasta lenguajes de alto nivel especializados como C++ o JAVA. Para aplicaciones de funcionamiento crítico altamente especializadas, sólo debieran utilizarse lenguajes especiales como ADA o Ensamblador. La elección del lenguaje dependerá de la disponibilidad de los programadores, la complejidad de la aplicación y la velocidad de operación requerida al sistema. Si se van a utilizar varios lenguajes en un sistema, deberán gestionarse cuidadosamente las interfaces entre los diferentes módulos.

b) **Tamaño**: El tamaño de un programa Software puede medirse en líneas de código, puntos de función o bytes. Cuanto más grande es un programa, mayor será el equipo de desarrollo y mayor dificultad en su desarrollo.

c) **Modularidad**: Para facilitar el desarrollo del Software, los programas grandes se dividen en módulos. Cada módulo se convierte en un subprograma, que puede ser desarrollado independientemente y luego combinado con otros módulos conformar una aplicación más grande. Si un programa complejo necesita ser modificado para mejorar su funcionamiento o eliminar defectos, es mucho más fácil desarrollar, mantener o corregir un diseño modular. También es posible combinar módulos de diferentes fuentes para crear nuevas aplicaciones con relativa facilidad.

d) **Seguridad**: Muchas aplicaciones incorporan alguna forma de sistema de seguridad. Esto puede incluir la integridad de la información, que incorpora actividades tales como encriptación, restricciones de acceso y cortafuegos (*firewalls*). Estos sistemas se diseñan para reforzar la protección contra virus de ordenador e intervenciones no autorizadas.

[5] Ver SAE JA1003 y SAE JA1005.

2.3 Apoyo a la Operación

Una vez que el equipo entra en la fase de Utilización[6] (Operación), el Software del sistema debiera ser mantenido por una única organización[7] que sería responsable de:

a) Mantener el programa maestro o código fuente de la aplicación. (Esto no siempre es posible si no se dispone de los derechos de propiedad intelectual (IPR –*Intellectual Property Rights*-);

b) El entorno de Ingeniería del Software;

c) Los documentos de ingeniería (Los requeridos por el Plan de Desarrollo Software (SDP -S*oftware Development Plan*-));

d) El análisis de mantenimiento del Software (SAS -*Support Analysis for Software*-);

e) La entrega del Software;

f) La documentación de gestión de configuración y normativas;

g) La gestión de las mejoras y de las actualizaciones;

h) La formación.

2.4 Gestión de Defectos

Si se encuentran defectos en la operación, debería informarse a la organización identificada para el mantenimiento utilizando los informes de incidencias (Ver ARMP-6 FRACAS). Éstas deberían investigarse para confirmar el problema y determinar si el incidente puede atribuirse al Software del sistema, al Hardware, al usuario o a los procedimientos. Si la incidencia es debida al Software, el defecto subyacente necesitará ser identificado ya que podrá presentarse en todos los sistemas en operación.

Los defectos de Software se resuelven normalmente en lotes y se liberan para operación como actualizaciones periódicas, bien como una nueva versión o como un parche. No obstante, los defectos críticos para la seguridad de funcionamiento deberían corregirse con la más alta prioridad y liberarse inmediatamente como actualizaciones de emergencia en la forma de un parche de Software.

2.5 Mejoras

La funcionalidad de un sistema de defensa puede ser modificada para mejorar su fiabilidad, para mejorar la capacidad o rendimiento o para extender el uso a nuevos entornos operacionales. Como resultado, la configuración del hardware y los procedimientos de operación pueden verse afectados y esto requerirá la mejora del Software. Para realizarlo de manera correcta, la Especificación de Requisitos Software (SRS -*Software Requirements Specification*-), debería ser revisada previamente para documentar la nueva capacidad y especificar los requisitos de prueba en toda su extensión. [8]Las mejoras de Software deberían seguir el mismo ciclo de desarrollo que la aplicación original (ver ISO 12207).

[6] Ver AAP-48

[7] e.g. Centro de Ingeniería Software, Agencia de Soporte de software o Autoridad de Diseño.

[8] Las pruebas de software suelen ser poco exhaustivas debido a la dificultad de realizar pruebas de estrés (simulando el entorno real de funcionamiento y en condiciones extremas), con las restricciones normales de coste y tiempo

Una vez desarrollada completamente una mejora, debería seguir un proceso de Liberación de Software para asegurar su mantenibilidad y validez para el uso. Para mantener el control de la configuración del sistema desplegado, la distribución e instalación de las mejoras de software deberían ser controladas estrictamente.

Se deberían combinar cuando sea posible, mejoras y actualizaciones para minimizar el número de versiones de Software y consecuentemente los inconvenientes al Usuario. Puede también ser apropiado combinar los cambios de versión del Software con los puntos programados de mantenimiento preventivo del Hardware.

2.6 Soportabilidad

El concepto de Mantenibilidad Hardware no aplica al Software de igual forma ya que está relacionado con la facilidad con que un sistema puede ser mantenido o reparado. Por ello el Software se suele tratar con el término Soportabilidad[9]. De lo anterior se evidencia que el Usuario o Mantenedor del Software no tienen la capacidad para efectuar el mantenimiento del código del Software. Todas las reparaciones las efectúa la organización identificada sobre el programa maestro, donde están disponibles la documentación, el entorno de ingeniería y los condicionantes. Lo que resulta relevante para la organización identificada es la estabilidad del Software en relación con la facilidad con que una aplicación se puede corregir, probar y replicar, con los recursos y personal disponibles a lo largo del ciclo de vida del Software. Además el método y la facilidad con que el Software puede ser cargado en el sistema influyen en la Soportabilidad del sistema.

[9] Ver el concepto de soporte en SAE JA1006.

3 ESPECIFICACIÓN DE REQUISITOS DE FyS

3.1 Fiabilidad

Los requisitos de fiabilidad se establecen habitualmente para un sistema completo, que mayoritariamente está formado por componentes hardware y software. Así los requisitos de fiabilidad del Software han de derivarse de los requisitos de fiabilidad del sistema. Los requisitos de fiabilidad del Software y su correcta especificación son los elementos más críticos para alcanzar una buena Fiabilidad y Soportabilidad para el Software. Esto se puede alcanzar por medio de una correcta asignación de fiabilidad para el sistema o, donde sea apropiado, como un requisito específico. Los elementos que se deberían tener en cuenta cuando se consideran requisitos de fiabilidad software por separado, incluyen:

a) Criticidad del fallo para el sistema completo;

b) Tiempo de recuperación desde un fallo;

c) Impacto de un fallo sobre los elementos de datos o de software;

d) Intervención requerida de un operador tras la ocurrencia de un fallo.

Para que sean significativos, los requisitos de fiabilidad del software deben especificarse cuantitativamente, lo que requiere la aplicación de métricas adecuadas.

Las medidas tradicionales de fiabilidad del hardware, tales como Tiempo Medio Hasta el Fallo (MTTF - *Mean Time to Failure*-) o tasa de fallos (ver ARMP-4), no son necesariamente apropiados para utilizarse con aplicaciones Software. Mientras que la fiabilidad del hardware se basa en una combinación de fallos sistemáticos y aleatorios, la fiabilidad del Software se determina por errores internos y por consiguiente pueden ser considerados completamente sistemáticos. No obstante la impredecibilidad de los errores de los programadores y las condiciones de la ejecución de un programa, a menudo hacen que los errores del software parezcan de naturaleza aleatoria.

(3.2) La fiabilidad del Software depende estrechamente de la reducción del número de defectos en el código y de la efectividad de cualquier actividad de reparación acometida para corregirlos. Los defectos del Software se clasifican en defectos, fallos y errores que se definen en el Anexo A.

Es importante notar la distinción entre defecto y fallo. El término fallo se refiere al comportamiento de un programa y sólo puede ocurrir cuando se ejecuta el software. Un defecto es una propiedad de un programa más que una característica de su comportamiento y comúnmente se le llama "*bug*".

(3.3) La fiabilidad del **sistema** se refiere generalmente al tiempo. Referidos a la fiabilidad del Software, se pueden tomar en consideración tres tipos de tiempos:

a) **Tiempo de Ejecución**: tiempo empleado por un procesador en ejecutar un programa;

b) **Tiempo de Calendario**: tiempo correspondiente al concepto de días naturales;

c) **Tiempo transcurrido**: el tiempo desde el inicio de la ejecución de un software, incluyendo el tiempo de espera y de ejecución de otros programas.

La relación entre el tiempo y los fallos, como uno de los posibles parámetros para la especificación de la fiabilidad del Software, puede expresarse mediante cantidades tales como:

a) Intervalo de tiempo entre fallos;

b) Tiempo hasta el fallo (o periodo de operación libre de fallos);

c) Intensidad o densidad de fallos (El número de fallos por unidad de tiempo).[10]

Cuando el número de fallos forma parte de una especificación, es importante considerar la severidad del fallo y el tiempo de su manifestación. Además, las consecuencias del fallo del sistema deben también ser consideradas, incluyendo:

a) Pérdida de funcionalidades críticas de seguridad de funcionamiento (*safety*);

b) Pérdida de funcionalidades críticas de misión;

c) Cantidad de pérdida de datos tolerable;

d) Información vital que debe protegerse contra fallos.

3.4 Gestión de Defectos

Es bien conocido que, pese al esfuerzo en el diseño y las pruebas, no será posible producir software sin defectos. Por tanto, la especificación de la Fiabilidad del Software debiera incluir objetivos como evitar, detectar y corregir los defectos, así como la tolerancia a los mismos, adicionalmente a la totalidad de los requisitos cuantitativos:

a) **Evitar defectos**: esfuerzo para obtener software correcto al primer intento utilizando técnicas formales tales como las del Instituto de Ingeniería Software en su Modelo Integrado de Madurez de la Capacidad (CMMI – *Capability Maturity Model Integration*-) o el enfoque propuesto por RTCA DO178.

b) **Tolerancia a Defectos**: una propiedad del diseño que permite a un sistema continuar su operación[11] pese a los defectos de software o devolverle a una condición segura conocida. Esto es el concepto de *no punto de fallo único* y comúnmente utiliza la redundancia, con funcionalidades software en paralelo o con intervención humana (HITL -*Human In The Loop*-). La tolerancia a defectos puede requerir hardware adicional o redundante.

c) **Detección de Defectos**: una propiedad del diseño que utiliza programas de vigilancia (*watchdog*) para monitorizar el sistema Software utilizando técnicas tales como pruebas internas (BIT -*Built In Test*-) y equipos de pruebas internas (BITE -*Built In Test Equipment*-). Estos pueden comprobar parámetros críticos y sus límites, realizar pruebas de consistencia independientes o medir la carga del sistema. Típicamente se utilizan para proporcionar avisos (*warnings*) y procesos de limitación o de apagado seguro antes de que se produzca un fallo.

d) **Corrección de defectos**: En principio esto se refiere al software de auto-reparación, que es un concepto muy avanzado. En la actualidad, las correcciones se hacen manualmente.

[10] Esta característica permite el cálculo de la fiabilidad asociada al software, como una función del tiempo, admitiendo procesos estocásticos del tipo Poisson.

[11] Puede ser en un modo degradado especial para funciones de seguridad de funcionamiento críticas.

(3.5) Métricas. Una métrica común utilizada para determinar la calidad de un Software es el número de defectos por cada 1000 líneas de código. Para investigar si se ha cumplido tal requisito, se pueden aplicar técnicas tales como la simulación de Monte Carlo. Un número conocido de defectos (*bugs* inyectados) se insertan deliberadamente en el programa; el software se prueba y la proporción de defectos insertados detectados puede utilizarse para inferir el número total de defectos restantes en el programa. **Ejemplos de métricas de software se dan en el Anexo B**.

(3.6) Definiciones Contractuales. Para evitar disputas comerciales, se deberán definir claramente en el contrato los términos clave incluyendo defecto, fallo, fiabilidad y soportabilidad. **Nota: En el Anexo A se incluyen estas definiciones**.

Página intencionadamente en blanco

4 DESARROLLO DE UN PROGRAMA DE FyS DEL SOFTWARE

(4) El objetivo de este capítulo es describir un marco simple y flexible para la gestión basada en el rendimiento de un programa de fiabilidad del Software. Los mecanismos principales se denominan "plan de Fiabilidad y Soportabilidad del Software"[12] y "Repositorio de Fiabilidad y Soportabilidad del Software". El Plan y el Repositorio son herramientas de gestión de propósito general que se utilizan en varios campos de la Ingeniería de Sistemas.

4.1 Plan de Fiabilidad y Soportabilidad

El Plan de Fiabilidad y soportabilidad (FyS) es un documento que describe las actividades que deben efectuarse durante el desarrollo para asegurar que:

a) Los requisitos de FyS del Software se derivan de los requisitos de nivel de sistema y permanecen consistentes con los mismos;

b) Los requisitos de FyS del Software son entendidos e implementados por el suministrador;

c) Se mantiene un Repositorio de FyS del Software para proporcionar un registro preciso y actualizado del progreso frente a los requisitos.

El plan de FyS debe formar parte integral del plan de desarrollo del Software que a su vez formará parte del plan de proyecto completo.

4.2 Repositorio de FyS del Software

El Repositorio de FyS del Software sirve para proporcionar evidencia que asegure que se ha progresado en el cumplimiento de los requisitos de Fiabilidad y soportabilidad del sistema. Las evidencias han de analizarse periódicamente durante el proyecto para asegurar que el desarrollo de la FyS del software es consistente y cumple los requisitos a nivel de Sistema. El repositorio también ha de asegurar que los requisitos de FyS son entendidos por el suministrador y que se ha resuelto cualquier discrepancia

El Plan de FyS y el Repositorio de FyS en conjunto, proporcionan un medio de seguimiento del progreso, frente a los objetivos de fiabilidad especificados. El Plan y el Repositorio se basan en el concepto de la eliminación temprana de defectos y la prevención continua de los mismos durante el ciclo de vida del Software. El Plan proporciona una visión de los procesos de fiabilidad, las actividades y los requisitos de rendimiento previstos, mientras que el Repositorio proporciona evidencia de los logros alcanzados en la fiabilidad del producto software, documentados mediante mediciones cuantitativas y cualitativas.

Se puede encontrar ayuda para la creación y uso de los Planes y Repositorios de FyS en la Guía de Implantación de la Fiabilidad del Software de SAE (SAE JA1003), que define prácticas para la implantación de un programa de fiabilidad del Software en el marco de una completa ingeniería de sistemas. Se describen prácticas para llevar a cabo un programa de FyS de Software a través de la selección de las actividades del ciclo de vida que se han adaptado a un sistema. Se

[12] SAE JA1002párrafo 4.3 "Management Framework".

proporcionan referencias y se identifican numerosos métodos y técnicas de análisis, diseño y verificación. Las guías para la adaptación de los programas de Fiabilidad del Software incluyen cuestiones de seguridad de funcionamiento y contra intrusión, para la integración de software COTS (*Commercial Off The Shelf*) así como para obtención de los datos apropiados. Se aplican a todos los proyectos que incorporan software, particularmente a los sistemas críticos para la misión o la seguridad, donde resulta esencial una elevada fiabilidad del software. Se proporcionan guías para todas las organizaciones involucradas en el diseño, desarrollo, obtención, integración uso y soporte del software.

5 ENTREGA Y GESTIÓN DE LA FyS DEL SOFTWARE

(5) Dentro del contexto del ciclo de vida del sistema, debe considerarse un ciclo de vida para el Software, que en el caso de los sistemas de armamento de la OTAN se describe en la AAP-48[13] y divide el ciclo de vida en seis fases adaptables:

a) Concepto;

b) Desarrollo;

c) Producción;

d) Utilización;

e) Soporte;

f) Retirada del servicio.

Este capítulo, sobre la entrega y la gestión de la FyS del Software, se alinea con la AAP-48[14]. La mayor parte de las actividades tendrán lugar durante el desarrollo del software, aunque la gestión de configuración, el control de versiones, la estrategia de liberaciones, la gestión de errores y defectos, la formación y la interacción con la comunidad de usuarios serán actividades de mayor importancia durante las fases de utilización y de Soporte.

5.1 Fase de Concepto

La Fase de Concepto es la fase inicial del ciclo de vida de un sistema, que comienza con la identificación de un requisito para una nueva capacidad, la modificación de un sistema existente o la sustitución de un sistema. Debería incluir: estudios de viabilidad y análisis de opciones, el desarrollo de requisitos y la preparación de planes entregables. La fase de Concepto finalizará con un hito de decisión en el que se considere el paso a la fase de Desarrollo.

5.1.1 Planificación para la FyS del Software

Durante la planificación inicial, cuando los requisitos del Software se extraen de los requisitos del sistema, se deberían considerar las capacidades de FyS del Software e incluirlas en los planes de proyecto, de pruebas, de Aseguramiento de la Calidad y Soporte. La especificación de requisitos Software (SRS -*Software Requirements Specification*-) debe establecer claramente la funcionalidad del Software y actualizarse conforme vayan madurando los requisitos de Software a lo largo del proyecto. Durante la última parte de la fase de Concepto, debe crearse un Plan de Desarrollo de Software (SDP – *Software Development Plan*-) que referencie la capacidad de FyS del Software y exprese como se va a obtener la especificación requerida. Debería tenerse en cuenta que casi el 50% de todos los fallos del software se deben a requisitos no definidos claramente.

[13] NATO System Life Cycle Stages and Processes.

[14] La AAP-48organiza los procesos en cuatro grupos:
- Procesos de Acuerdo: Aseguran la conformidad entre empresas y/o proyectos
- Procesos de Empresa: Se enfocan en la estructura del sistema (supersistemas-subsistemas.
- Procesos de proyecto: Se enfocan en el ciclo de vida del sistema de interés.
- Procesos técnicos: Proporcionan apoyo a las funciones del proyecto y facilitan la optimización de los beneficios.

Un Plan de Desarrollo Software (SDP -*Software Development Plan*) exhaustivo constituirá la base para las fases de Desarrollo, Utilización y Soporte del proyecto. Debería estar integrado con el Plan de Desarrollo del Sistema completo y referenciar los siguientes elementos de Ingeniería de Software, adaptados a las necesidades específicas del proyecto:[15]

a) Requisitos de documentación;

b) Métodos de desarrollo Software.[16]Políticas de reutilización de Software (reutilización y modificación del software heredado);

c) Métodos de integración, gestión de problemas y riesgos del Software;

d) Tratamiento de requisitos críticos, incluyendo los relativos a la seguridad de operación (safety), a la misión, al aseguramiento y seguridad (*security*) de la información;

e) Herramientas y técnicas utilizadas en el desarrollo o mantenimiento del software;[17]

f) Control de la configuración: Gestión de cambios, aprobación y seguimiento;

g) Proceso de liberación de versiones;[18]

h) Normas aplicables;

i) Requisitos de hardware necesarios para el desarrollo, utilización y soporte;

j) Estrategia de construcción del Software (incluyendo guías de estilo para el código);

k) Metodología de pruebas, incluyendo pruebas unitarias, de integración, de sistema y la documentación;

l) Verificación y validación de proyecto, así como la Verificación y Validación independientes;

m) Proceso de reporte de datos;

n) Proceso de distribución de mejoras del Software;

o) Auditoría y Verificación del software instalado.

5.1.2 Soportabilidad del Lenguaje del Software

Para el software desarrollado a medida, será necesario seleccionar uno o varios lenguajes de programación apropiados. Esto podrá tener profundas consecuencias, por lo que han de tenerse en cuenta los siguientes elementos antes de elegir un lenguaje dado:

a) La plataforma de destino debe soportar el lenguaje;[19]

b) El compilador deberá ser soportado durante el ciclo de vida completo;

c) El lenguaje debe ser el adecuado para el tipo de aplicación;[20]

[15] Ver ISO 12207para el contenido de un Plan de Desarrollo Software (SDP) y más detalles sobre este asunto.

[16] Algunos ejemplos incluirían los modelos tradicionales en cascada, el modelo en espiral de Boehm Ganar-Ganar, modelos evolutivo e incremental,

[17] Incluyendo: herramientas de modelado y simulación, herramientas de desarrollo y pruebas, lenguajes, compiladores, herramientas de ingeniera software ayudada por ordenador y convenciones sobre notaciones.

[18] Por ejemplo entrega incremental, entrega simultánea, entrega selectiva para pruebas de prototipo, trabajos asociados y documentación.

[19] Es buena práctica habilitar alguna capacidad adicional en la plataforma de destino para el crecimiento del software y la operación del programa.

[20] El software de sistemas de armas no debe desarrollarse en un lenguaje orientado a la empresa.

d) La popularidad del lenguaje;[21]

e) La especificación abierta del lenguaje;[22]

f) La eficiencia del lenguaje (velocidad de ejecución y requisitos de memoria);

g) La flexibilidad, compatibilidad y facilidad de integración; [23]

h) Fiabilidad del compilador y el soporte del suministrador;[24]

i) Los lenguajes y compiladores están certificados conforme a las normas de seguridad de funcionamiento (*safety*) requeridas.

5.1.3 Gestión de Riesgos

El Software se encuentra sometido a riesgos en común con el hardware. La gestión de riesgos debe ejecutarse durante todo el ciclo de vida del Software. Debe prepararse un plan que establezca algún método de gestión de riesgos y que evalúe el nivel de riesgo (ver Capítulo 4).

5.1.4 Software Preexistente COTS (*Commercial Off The Shelf*)

Durante la fase de Concepto, cuando se están considerando soluciones alternativas, puede seleccionarse el uso de software preexistente (COTS), ya sean módulos o aplicaciones completas. No obstante, la selección de un COTS no asegura ni elimina la responsabilidad del cliente de asegurar la compatibilidad en cuanto a la Fiabilidad y la Soportabilidad (FyS)

El software COTS puede incluir aplicaciones que no se van a desarrollar, programas previamente desarrollados o adaptados y comprende: sistemas operativos, compiladores, herramientas software de apoyo, drivers relacionados con el hardware, módulos de comunicaciones, librerías de aplicación o aplicaciones completas. Puede resultar difícil gestionarlo y normalmente su modificación suele ser compleja tanto por razones técnicas como legales (ver ARMP-1 y ARMP-6).

Las ventajas de los productos COTS son su madurez, disponibilidad, elevado número de usuarios[25], fiabilidad[26] y precio reducido. Aun así hay numerosas cuestiones que deben considerarse, incluyendo:

a) Requisitos sobre derechos y licencias;

b) Pérdida de control o capacidad de actualización y mantenimiento del producto;

c) Obsolescencia;

d) Apoyo del Suministrador;

e) Capacidad para interactuar con otros productos;

f) Pruebas (pueden quedar limitadas a pruebas funcionales);

---

[21] Si el lenguaje es poco conocido y/o poco claro, será difícil encontrar programadores de soporte y los costes de soporte serán altos.

[22] Si la especificación del lenguaje es del dominio público, podrán dar soporte varios proveedores, mientras que un lenguaje propietario será dependiente de las circunstancias y supervivencia de un único proveedor.

[23] La facilidad para modificar código, construir módulos independientes, construir interfaces externos, añadir módulos externos escritos con otro lenguaje.

[24] Registro de seguimiento en relación a cuanto se ha extendido, cuantos usuarios hay.

[25] Un mayor número de usuarios proporciona un nivel mayor de pruebas en operación para identificar defectos inherentes.

[26] En su aplicación original de diseño, esto puede cambiar en diferentes aplicaciones

g) Monitorización y mejora de las prestaciones;
h) Mitigación de riesgos;
i) Seguridad (Security);
j) Información y formación;
k) Integración;
l) Plataforma de destino y entorno operacional;
m) Cambios de mercado.

Cuando el software COTS se integra en un sistema militar, debe asegurarse que no afecta a la plataforma de destino que lo acogerá. Por ello ha de ser sometido a un riguroso análisis y régimen de pruebas, de la misma forma que los productos desarrollados a medida.

5.2 Fase de Desarrollo

El Desarrollo es la segunda fase del ciclo de vida del sistema, la cual comienza con la confirmación de los requisitos del proyecto y el arranque del diseño software. Debería incluir: el uso de un plan de construcción de Software, la identificación y establecimiento de líneas de referencia, el control de la configuración y las pruebas de versión. Durante esta fase, el software se desarrollará para cumplir completamente los requisitos establecidos para el sistema, incluyendo la integración con los sistemas hardware representativos y el equipamiento operacional.

La fase de Desarrollo es una fase crítica para la FyS del Software, ya que supone las bases para la entrega de un Software fiable y soportable. La fase de Desarrollo puede abarcar la codificación del software desarrollado a medida, la modificación de software existente o la integración de software COTS en el sistema.

La fase de Desarrollo debería finalizar cuando el software se comprueba en un entorno representativo del hardware y se verifica frente a los requisitos software establecidos. La fiabilidad del sistema completo ha de ser la principal consideración y la fase de Producción sólo debería comenzar cuando se determine y verifique la capacidad del sistema completo.

5.2.1 Aplicación de los Principios de FyS del Software *(Proceso de Análisis de Requisitos, Proceso de Diseño de la Arquitectura, Proceso de Implantación)*

El Capítulo 4 de este documento proporciona guías sobre los requisitos de FyS, el desarrollo y uso del Plan y el Repositorio de FyS. Estos deben implementarse desde la definición de los requisitos hasta el desarrollo o la adquisición del software. Mientras la aplicabilidad de prácticas específicas de FyS estará en función de la importancia de la fiabilidad del software, los requisitos de FyS deben ser trazables a todas las actividades de la fase de desarrollo. El Documento de Diseño Software, que define la arquitectura del producto, debería incorporar los requisitos de FyS, tales como la modularidad, baja complejidad, lenguaje soportable, facilidad de validación y un bajo nivel de dependencia de módulos externos.

5.2.2 Modularidad

Los grandes programas software deberían desarrollarse en forma modular, que puede ayudar a la programación y mejorar la fiabilidad, la soportabilidad y la facilidad de probar. Debe tenerse cuidado para asegurar que los módulos son razonablemente independientes en cuanto a funcionalidad y que las interfaces son simples y bien definidas (Ver Capítulo 2).

5.2.3 Prototipado

Para programas Software grandes y complejos, es altamente deseable comenzar el proceso de desarrollo construyendo una maqueta o modelo de funcionamiento simplificado. Esto ayudará a clarificar las especificaciones del usuario y mejorar el entendimiento de los requisitos por parte del suministrador. El proceso de construir una versión parcial o resumida, utilizando un lenguaje de modelado[27] para simular las funciones de un programa se conoce como prototipado. Este proceso ayudará a definir, simplificar y probar interfaces y algoritmos antes de implantarlos en entornos complejos de codificación. Las aplicaciones de Software en prototipo, previas al desarrollo completo pueden reducir significativamente el tamaño final del programa y el tiempo de desarrollo a la vez que mejoran la FyS.

5.2.4 Modelado y Simulación

Una vez completado el proceso de Prototipado y comenzado el trabajo sobre el programa Software, es deseable utilizar técnicas de Modelado y Simulación para ayudar al desarrollo. Este proceso utiliza sistemas Hardware y Software junto con comunicaciones y recursos humanos para emular el sistema operativo. La simulación es un proceso mediante el cual se emplea una aplicación Software para efectuar una serie de funciones en un modelo del sistema operativo, para verificar la funcionalidad requerida. El uso de la simulación permite: la compresión y expansión del tiempo, un modo de operación de parada y arranque, experimentación con varios estados del sistema y capacita que se puedan medir los resultados, sin la necesidad de construir interfaces ni la infraestructura de acogimiento del sistema. La simulación efectiva de las aplicaciones Software durante el desarrollo puede utilizarse para mejorar la FyS.

5.2.5 Interfaces

En los sistemas complejos es muy importante asegurar la compatibilidad entre módulos Software, aplicaciones y la plataforma de destino. Las interfaces también existen entre el Software y el Hardware y entre el Hardware, los operadores y el entorno. Las interfaces son canales de comunicación entre subsistemas auto contenidos, que pueden ser unidireccionales o bidireccionales. La comunicación a través de los límites de las interfaces debe seguir ciertos protocolos, que sean entendidos a cada lado de las interfaces.

Durante el Desarrollo, todos las interfaces deberían ser mapeados para establecer dependencias entre los subsistemas, las aplicaciones y los módulos.[28] Debería definirse una descripción completa para cada interfaz, listando todas y cada una de las necesidades para el programador para conocer su función dentro del sistema.[29]

[27] Tal como el Lenguaje de Modelado Universal (UML – Universal Modeling Language), que es una herramienta versátil que permite la visualización de las arquitecturas y la estructura del software.

[28] Esto podría soportarse mediante herramientas de mapeo comerciales tales como UML.

[29] La norma IEEE Standard 1016proporciona detalles relativos a las descripciones del diseño software incluyendo las interfaces.

5.2.6 Pruebas

Las pruebas son un elemento importante en el desarrollo Software y un elemento clave para la fiabilidad del Software. Una concepción común, errónea y grave es que las pruebas se efectúen al final del ciclo de desarrollo. Las pruebas deben formar parte integral de cada fase dentro del ciclo de desarrollo del software, que debe estar soportado por un plan de pruebas, con la flexibilidad suficiente para tratar eventos inesperados. El régimen de pruebas se puede enfocar de distintas maneras, desde la prueba funcional a un examen detallado línea a línea del software.

El régimen de pruebas de Software debería descomponerse en un número de etapas progresivas que podrían incluir:

a) Pruebas unitarias: la prueba de módulos de software;

b) Pruebas de integración: la prueba de dos o más módulos combinados;

c) Pruebas de sistema: la prueba de todos los módulos de software de un sistema;

d) Pruebas de regresión: pruebas tras las modificaciones o cambios;

e) Pruebas de estrés: Pruebas del software más allá de sus pretendidos límites;

f) Pruebas de integración Software-Hardware: Las pruebas del software junto con el hardware representativo;

g) Pruebas de aceptación: pruebas formales de un sistema frente a requisitos contractuales;

h) Pruebas operacionales: prueba de las funciones *end-to-end* frente a los escenarios operacionales.

Todas las pruebas de Software deben estar soportadas por documentación:

- **Plan de Pruebas**. Se debería exigir al suministrador un Plan de Pruebas Maestro que describa el régimen de pruebas del Software. Este también incluirá una serie de Planes de Pruebas para las distintas fases del ciclo de desarrollo Software. Los elementos de cada plan de pruebas deberían incluir los criterios de aceptación y rechazo, los riesgos y las implicaciones del fallo[30] (ver ARMP-6 Anexo K)
- **Procedimientos de pruebas**. Estos procedimientos describen las actividades que deberían ser acometidas para probar un módulo o aplicación Software, las condiciones bajo las que deben efectuarse las pruebas y los resultados esperados. Cada procedimiento de pruebas debería atender al menos a un requisito de los documentados en el documento de especificación de requisitos Software (SRS -*Software Requirements Specification*-). Asimismo, cada requisito debería ser trazable al respectivo Procedimiento de Pruebas por medio de una matriz de verificación. Los procedimientos de pruebas deberían incluir la siguiente información:
  a) **Objetivos de pruebas**: definen el objetivo pretendido en la fase de pruebas correspondiente;
  b) **Criterios de finalización**: las condiciones que determinan la finalización con éxito;
  c) **Responsabilidades**: describen los roles del suministrador y el cliente;
  d) **Normas** y **documentos aplicables**;

[30] Para guía en la realización de Planes de Prueba y otra documentación de prueba ver la norma IEEE 829, Documentación de pruebas software.

e) **Herramientas**: incluyendo los productos Hardware y Software, listas de comprobación e instrucciones;

f) **Recursos**: sistemas, instalaciones, recursos humanos, etc;

g) **Procedimientos de informe**: registro y seguimiento de errores (ver FRACAS en la ARMP-6 Anexo F);

h) **Pruebas de Regresión**.

- **Resultado de las pruebas**. Todos los resultados de las pruebas deben registrarse, compararse con los criterios de finalización y presentarse al cliente en un formato acordado.

5.2.7 Herramientas Automáticas

El desarrollo Software puede soportarse mediante una variedad de herramientas automáticas asistidas por ordenador. Estas alcanzan desde la definición de requisitos y la traza de los mismos hasta herramientas de diseño Software, herramientas de ingeniería Software asistida por ordenador (CASE -*Computer Aided Software Engineering*-)[31] y herramientas de prueba automatizadas. El uso apropiado de las herramientas seleccionadas durante el Desarrollo podría mejorar la FyS del Software; no obstante, cualquier herramienta utilizada debería ser aprobada por la autoridad apropiada.

5.2.8 Métricas Relacionadas con el Desarrollo del Software

Para monitorizar el progreso durante la Fase de Desarrollo será necesario definir un número de métricas relacionadas con la gestión y con el producto y obtener los elementos de datos que las soporten (ver Capítulo 3). Para proyectos grandes, pueden emplearse herramientas de medición del rendimiento. Algunas métricas comunes incluyen:

a) Líneas de Código Fuente (SLOC -*Source Lines Of Code*-);

b) Defectos por cada mil líneas de código fuente (kSLOC o KSLOC);

c) Número de módulos;

d) Tamaño del programa (Kbytes);

e) Tráfico de cambios anual (ACT -*Annual Change Traffic*-);

f) Números de complejidad del código.

Nota de traducción: ACT significa la porción de código fuente respecto al total que cambia en un año.

5.3 Fase de Producción

La fase de Producción es la tercera en el ciclo de vida del sistema y comienza con la línea de referencia de la primera versión de una aplicación software cuando el Desarrollo ha finalizado. Debería incluir pruebas de comportamiento y de integración final del software con el hardware

[31] CASE es el empleo de apoyo basado en ordenador el proceso de desarrollo software (Software Engineering Institute)

operacional. La fase de Producción finalizará con la instalación del software[32] en todas las plataformas objetivo y la distribución de la documentación de apoyo.

Varias de las actividades de la Fase de Desarrollo pueden continuar en la Fase de Producción y más allá, tales como las pruebas, la gestión de riesgos, la monitorización del rendimiento y la gestión de configuración. No es raro que se efectúen por fases de lanzamiento, conforme se aumenta progresivamente la funcionalidad.[33]

Nota de traducción: la fase de Producción en esta guía, se corresponde con la fase de Paso a Producción de un sistema, o de pre-producción.

5.4 Fase de Utilización

La Utilización es la cuarta fase del ciclo de vida del sistema y es la fase operacional por excelencia relacionada con el funcionamiento continuado del sistema. La Fase de Soporte transcurre paralela con esta fase. La fase de Utilización continuará durante la vida operacional del sistema. Las actividades de FyS del Software durante esta fase se enfocan en el apoyo a las aplicaciones y mediciones del rendimiento.

5.5 Fase de Soporte

El Software, como el Hardware, requiere apoyo durante su ciclo de vida, un elemento clave de éste es la gestión de cambios. El Software puede necesitar ser mejorado como resultado de la corrección de un defecto, cambios en los procedimientos de operación, entornos, el Hardware o las interfaces. Para permitir la modificación o mejora del software sin la necesidad de modificación del Hardware, deberían incluirse con los requisitos originales del hardware, suficientes capacidades de procesamiento y memoria adicionales.

5.5.1 Métricas y Recogida de Datos

La recogida de datos y su análisis resulta muy importante durante la Fase de Soporte para aislar defectos y mejorar la fiabilidad. Las métricas requeridas para apoyar a esta fase pueden incluir:

a) POFOD -Probabilidad de Fallo en la Demanda (*Probability of Failure on Demand*);

b) ROCOF –Tasa de ocurrencia del Fallo (*Rate of occurrence of failure*);

c) MTTF-Tiempo Medio hasta el Fallo (*Mean Time to Failure*).

Una herramienta valiosa para gestionar este proceso es disponer de un sistema de informe del fallo y acciones correctivas (FRACAS –*Failure Reporting and Corrective Action System*-), que puede utilizarse para analizar datos de fallo, desarrollar y apoyar correcciones de los defectos del Software (ver ARMP-6 Anexo F).

[32] Incluyendo el software COTS.

[33] Ejemplos de esto incluyen al modelo en espiral de Boehm (ganar-ganar), el modelo evolutivo y el modelo incremental.

5.5.2 Gestión del Mantenimiento del Software

Durante la Fase de Soporte, la FyS del Software debe ser trazada a un Plan de Gestión del Software (SMP –*Software Management Plan*-) que podría formar parte del Plan de Desarrollo del Software (SDP –*Software Development Plan*-). El SMP debería incluir los siguientes elementos:

a) Gestión de Defectos;
b) Monitorización del funcionamiento del Software;
c) Gestión de Riesgos;
d) Gestión de la Obsolescencia;
e) Gestión de la configuración;
f) Aseguramiento de la Calidad;
g) Gestión de la Documentación;
h) Formación;
i) Soporte del programador.

Nota de traducción: El plan SMP podría ser parte del plan de Aseguramiento de Calidad del Software, o complementarlo.

5.5.3 Gestión de la Configuración

La Gestión de la Configuración (CM –*Configuration Management*) es una disciplina para controlar la evolución del hardware, el software, los servicios y la documentación. CM es un aspecto crítico del Soporte Software ya que es importante que todos los sistemas funcionen con una única versión del Software. Un riesgo importante para el Software proviene de un acceso incontrolado al código fuente pues el Software podría ser cambiado sin dejar trazas. Debe ponerse en marcha una disciplina estricta de CM para evitar cambios no autorizados ni documentados.[34] CM supone la dirección técnica y administrativa de las siguientes actividades:

a) Identificación de la configuración del Software;
b) Monitorización y control de la configuración del Software;
c) Auditorías de la configuración del Software (tanto física como funcional).

5.5.4 Gestión de Riesgos

La gestión de riesgos forma parte integral del Soporte al Software ya que cualquier cambio a una aplicación podría tener efectos indeseables colaterales. Por ello cada petición de cambio al Software debería examinarse y los riesgos asociados evaluados.[35] La evaluación de los riesgos debería ser cuantitativa. Los factores de riesgo asociados con la FyS del Software incluyen:

a) **Factores de Rendimiento**: Tales como el uso de recursos adicionales (memoria, CPU, almacenamiento, ancho de banda de red);
b) **Complejidad**: El número de módulos software afectados por el cambio. El evaluación cualitativa de la complejidad comparada con otros cambios;

[34] ACMP-7Guía OTAN para la aplicación de las ACMPs 1 a 6 de Gestión de Configuración

[35] Ver La Guía para la Gestión continua de Riesgos del Software Engineers Institute.

c) **Tamaño de la modificación**: El número de Líneas de Código del Software y el número de módulos afectados;

d) **Criticidad**: El efecto potencial en la misión principal y en la seguridad de funcionamiento (safety);

e) **Factores de interfaz**: La modificación en relación con los cambios del hardware o de la red;

f) **Testabilidad**: Dificultad del proceso de pruebas;

g) **Factores Humanos**;

h) **Documentación**: Documentación inadecuada o insuficiente.

5.5.5 Proceso de Pruebas

Las pruebas del Software constituyen un aspecto importante del Soporte Software. El papel principal será el de verificar las correcciones a la aplicación. Parte de las herramientas y documentos obtenidos durante el Desarrollo deberían mantenerse durante las Fases de Utilización y de Soporte, incluyendo:

a) Planes de pruebas;

b) Descripciones de las pruebas;

c) Procedimientos de pruebas;

d) Resultados e informes de pruebas;

e) Herramientas y programas de pruebas.

Las pruebas deberían también aplicarse al software COTS a nivel de módulo o sistema.

5.5.6 Verificación y Validación Independientes

Debería considerarse la realización de una Verificación y Validación independientes en todas las fases del desarrollo Software con el fin de mitigar riesgos, incluyendo cualquier modificación o mejora significativa.

5.5.7 Mantenimiento de la Integridad del Software

La gestión de la FyS del Software durante las fases de Utilización y Soporte debería prestar atención a otros factores como son:

a) Obsolescencia del Software;

b) Integridad Software entre sus diferentes versiones;

c) Seguridad de funcionamiento (safety);

d) Mantenimiento de la trazabilidad a los requisitos originales;

e) Mantenimiento de las herramientas de apoyo y de desarrollo software.

5.6 Fase de Retirada del Servicio

Las actividades de FyS durante la fase de retirada del Servicio del Software se limitan a:

a) Retirada controlada de los datos clasificados;

b) Archivo controlado de los datos y documentación del programa;

c) Documentación de las Lecciones Aprendidas durante la vida del software para beneficio de sistemas futuros.

6 ASPECTOS DE LA FyS DEL SOFTWARE EN LA CONTRATACIÓN

(6) El éxito o no de cualquier proyecto depende en gran parte de la calidad de sus contratos de adquisición y de soporte. Si éstos son robustos, bien escritos y comprensibles, habrá una elevada probabilidad de que el producto resultante cumpla con sus requisitos y entregue la capacidad prevista en la fase de utilización del ciclo de vida del equipo a un coste óptimo.

6.1 El Software como Elemento Único

El sistema Software es un elemento de suficiente importancia para cualquier proyecto como para no ser omitido del proceso de contratación. A la vista de sus características únicas, el Software debería ser considerado como un elemento separado dentro de cualquier contrato de adquisición o de soporte. En la sección de Software del contrato, deben considerarse la Fiabilidad y Soportabilidad mediante requisitos y cláusulas específicos. La falta de cobertura adecuada a esta área puede exponer los proyectos a riesgos significativos, hacer que no queden claramente determinadas las capacidades del software y elevar los costes de soporte.

6.2 El Software como un Componente Integral del Sistema

A pesar de su estado especial, el Software debería considerarse como parte integral de un sistema y también ser referenciado entre los requisitos completos de la capacidad. Así la FyS del Software debería considerarse entre los requisitos globales de la FyM del sistema. La entrega de todos estos requisitos entrelazados debería gestionarse activamente y revisarse su progreso en los hitos del proyecto.

Los requisitos de FyS deben determinarse durante la fase de Concepto junto con las métricas apropiadas y los mecanismos mediante los cuales puedan obtenerse datos y analizarse para medir el progreso (ver Párrafo 3.5). Todos los requisitos deberían cubrirse mediante cláusulas específicas en los contratos de adquisición o de soporte y los planes de trabajo (*statements of work*) que los sustentan.

6.3 Definición de Términos

La definición de cualquier palabra clave asociada con la FyS del Software, su medición y entrega debe ser acordada entre el cliente y el suministrador, y sancionado por el contrato junto con cualquier aclaración que la sustente (ver Anexo A). Debería darse especial consideración a la distinción entre Errores, Defectos y Fallos así como los métodos de contabilizarlos en las incidencias repetidas y sus correcciones asociadas (ver Capítulo 3).

6.4 Evaluación Progresiva

De acuerdo con FyM, el contrato debería requerir que el suministrador proporcione evaluación progresiva de que los requisitos de FyS del Software se van cumpliendo durante las fases de desarrollo, producción y utilización del ciclo de vida del equipo. Se debería requerir al suministrador que proporcione evidencia del progreso frente a los hitos del proyecto (ver ARMP-1) que pueden ser evaluados por el cliente. También deberían preverse disposiciones para los

Comités Conjuntos de Definición de Incidencias (ISC -*Incident Sentencing Committee*-) para acordar la atribución, monitorizar el progreso y gestionar los problemas significativos (éstos temas podrían tratarse en las reuniones de FyM cuando sea apropiado).

El contrato debería especificar cualquier requisito necesario para pruebas formales o demostraciones de F&S durante el desarrollo, la producción o utilización. Se deberían incluir cualesquiera condiciones especiales o implicación del cliente. Cuando se hagan pruebas o demostraciones conjuntas los roles y responsabilidades de todas las partes deben quedar claramente definidas y los límites especificados. El contrato también debería considerar la formación de un grupo conjunto que analice y evalúe los datos resultantes y acuerde los resultados (esto puede ser realizado por el ISC- *Incident Sentencing Committee*-).

6.5 Gestión de los Cambios en los Requisitos

Durante el proceso de negociación del contrato, debe tenerse mucho cuidado para evitar que la FyS del Software a entregar se debilite. Cualquier cambio propuesto a los requisitos de FyS, las definiciones, las métricas, los procesos de medida, la evaluación progresiva, las pruebas, las demostraciones, los hitos o los incentivos deberían discutirse con un especialista de Software antes de que se acuerden con el suministrador.

Para asegurar el compromiso del suministrador con la importancia del Software, la entrega de la FyS del Software debería ligarse a incentivos financieros importantes, en base a mediciones objetivas frente a los objetivos acordados. Los pagos o penalizaciones deberían ligarse a hitos del proyecto y el éxito o no determinado mediante Evaluación Progresiva (ver la ARMP-1).

6.6 Aceptación del Software

Al final del proceso de adquisición, los nuevos sistemas no deberían ser aceptados para el uso, a menos que los requisitos de FyS del Software se hayan conseguido. Si por cualquier razón, se acepta un sistema inmaduro, todos los incumplimientos deben ser formalmente registrados y se deben proporcionar, los recursos apropiados para entregar el programa corregido, de acuerdo a lo establecido en el contrato.

Se debería poner especial atención a las disposiciones para el sostenimiento de la FyS del Software desde la Fase de Concepto, preparado para su implementación en la aceptación del Software, a través de las fases de utilización y retirada prematura. El proceso de evaluación progresiva y la gestión de los defectos deberían revisarse y el contrato de compra corregido para reflejar cualquier afinamiento que pueda requerirse en una fase previa de Utilización. Para evitar confusión, es importante que no se hagan cambios a ninguna de las definiciones o procesos de contabilidad establecidos durante la compra. En el caso de que hubiera un cambio en el nivel de actividad de soporte entre el Desarrollo y la Utilización, el contrato debería permitir que cualquier ahorro fuera compartido entre el Cliente y el Suministrador.

6.7 Apoyo a la Utilización

Cuando finaliza un contrato de adquisición, normalmente será necesario efectuar un contrato de soporte al Software. El trabajo en esta actividad debería comenzar antes que el contrato de adquisición esté próximo a finalizar (se sugieren entre 12 o 24 meses). Para asegurar coherencia y continuidad, el contrato de soporte debería incluir provisiones especiales para FyS y seguir el modelo establecido para la adquisición, utilizando las mismas definiciones y métricas. Deberían

hacerse provisiones específicas para el desarrollo y la entrega de las mejoras del software para asegurar que no socavan las capacidades originales de FyS. También se deberían hacer disposiciones para recoger evidencias de defectos y fallos del Software detectados durante la Utilización y gestionar las acciones correctivas dentro de un marco de trabajo formal. Se debería mantener el uso de incentivos financieros asociados a los hitos del proyecto.

6.8 Contratos de Adquisición

En resumen, deberían tenerse en cuenta los siguientes elementos en el contrato de adquisición de un sistema Software, si se ha de asegurar la FyS:

a) Deben estar especificados los requisitos de FyS;
b) Que están definidos conceptos clave;
c) Que están definidas las métricas a utilizar y los mecanismos de medición;
d) Que están especificadas las pruebas formales de FyS así como las demostraciones;
e) Que está especificada la Evaluación progresiva para la entrega de FyS frente a los hitos fijados;
f) Que se han establecido incentivos financieros para cumplir con los requisitos de FyS;
g) Que se ha establecido el Comité de Actuación frente a Incidentes.

6.9 Contratos de Soporte

En resumen deberían tenerse en cuenta los siguientes elementos en el contrato de soporte de un sistema Software, si se ha de asegurar la FyS:

a) Que los requisitos de FyS se mantienen;
b) Que se mantienen las definiciones de los conceptos clave;
c) Que se mantienen las métricas a utilizar y los mecanismos de medición;
d) Que se mantienen las pruebas formales de FyS así como las demostraciones para las mejoras;
e) Que se ha especificado la evaluación progresiva para mantener FyS frente a los hitos de pago;
f) Que se mantienen incentivos financieros para mantener la FyS frente a los requisitos;
g) Que se mantiene el Comité de Actuación frente a Incidentes;
h) Que se establecen mecanismos para compartir ahorros.

Página intencionadamente en blanco

7 CONCLUSIÓN

(7.1) Todos los aspectos del diseño y la implantación del software en un campo comercial son relevantes y aplicables a los proyectos de sistemas de defensa OTAN. Por ello la consideración de estos elementos en la fase de concepto de dichos proyectos es de gran importancia para asegurar el éxito del proyecto. Este documento ha descrito la naturaleza e importancia del software y ha subrayado los requisitos para especificar la FyS del software. Describe las responsabilidades y las disposiciones contractuales requeridas para obtener y sostener Software fiable y soportable y ha marcado los principios y el marco para cumplir con ello. También se han descrito técnicas y herramientas que pueden utilizarse para evaluar y demostrar la FyS del Software.

(7.2) En el campo de la ingeniería del software en continua evolución es de vital importancia que la diversidad, complejidad y tamaño variable de las posibles soluciones software sea reconocido y entendido. Por ello, la precisión en la especificación de los requisitos, incluyendo la necesidad de mitigar la degradación de los mismos y mantener la trazabilidad de los requisitos fijados, tiene un gran impacto en el éxito del proyecto

(7.3) El objeto de la integración del sistema debería ser tenido en cuenta desde la fase de Concepto del proyecto con los elementos hardware y software considerados como una entidad única. Debería implantarse un proceso de gestión formal para controlar esa integración durante todo el ciclo de vida del proyecto. Esto debería incluir:

a) Gestión de las interfaces;
b) Gestión de la configuración;
c) Análisis de opciones;
d) Proceso de revisión formal;
e) Programa de pruebas;
f) Documentación.

(7.4) Como en todos los proyectos, la contratación eficaz es esencial para asegurar que se obtiene las salidas requeridas. Por ello debería seguirse un proceso de adquisición robusto adaptado específicamente a los proyectos software, tales como CMMI y DO 178. La naturaleza compleja del software también requiere el uso de asesores especializados, bien de la propia empresa o bien como consultoría, para asegurar que no se producen malentendidos y ambigüedades durante el proceso de contratación y adquisición.

(7.5) Aunque la historia de la ingeniería software tiene muchos ejemplos de proyectos complejos que han requerido fondos adicionales y fechas de finalización retrasadas, se ha demostrado que una gestión cuidadosa y apropiada puede asegurar que se pueden alcanzar los resultados requeridos en tiempo y presupuesto. La orientación contenida en este documento para Usuarios, Gestores, Personal de proyecto y Suministradores para la adquisición y mantenimiento de sistemas Software fiables y soportables, a lo largo del ciclo de vida del equipo, proporciona las herramientas iniciales para conseguirlo.

Referencias:

1. AAP-48 – NATO *System Life Cycle Stages and Processes*.
2. ACMP-7 NATO *Configuration Management Guidance on the Application of ACMPs 1 to 6*.
3. Def Stan 00-42 Part 3 - *Reliability and Maintainability Assurance Guide*.
4. Def Stan 00-60 Part 0 - *Application of Integrated Logistic Support (ILS).*
5. IEEE Standard 610.12 - *Standard Glossary of Software Engineering Terminology*.
6. IEEE Standard 829 – *Software and System Test Documentation*.
7. IEEE Standard 1016 – *Recommended Practice for Software Design Descriptions*.
8. ISO 12207 – *Software Life Cycle Processes*.
9. RTCA DO178 - *Software Considerations in Airborne Systems and Equipment Certification*.
10. SAE JA1000 - 1998 *Reliability Program Standard*.
11. SAE JA1000-1 - 1999 *Reliability Program Standard Implementation Guide*.
12. SAE JA1002 – 2004 *Software Reliability Program Standard*.
13. SAE JA1003 – 2004 *Software Reliability Program Implementation Guide*.
14. SAE JA1004 – 2004 *Software Supportability Program Standard*.
15. SAE JA1005 – 2004 *Software Supportability Program Implementation Guide*.
16. SAE JA1006 - 2004 *Software Support Concept*.

A. DEFINICIONES DE FyS DEL SOFTWARE

Actualización: Nueva versión de un programa software existente que se ha modificado para eliminar defectos.

Aplicación empresarial: Un producto COTS de software que integra todas las funciones de una organización en un sistema informático único que puede servir todas las necesidades particulares de la organización. Ejemplos de aplicaciones de empresa son SAP R/3, *Mincom* y *PeopleSoft* (CAN MASIS, DAOD 3001-0).

Bug: Ver Defecto (software).

Defecto (software): Condición accidental que hace que una unidad funcional software deje de hacer sus funciones requeridas (SAE JA1003A.2.1).

Error: La diferencia entre una condición computada, observada, o un valor medido y la condición o valor verdaderos, especificados o teóricamente correctos.

Fallo (*software*): La incapacidad de un sistema o componente para efectuar sus funciones requeridas con los requisitos de funcionamiento especificados (Norma IEEE Standard 610.12).

Fiabilidad del Software: La probabilidad de funcionamiento libre de fallos de un programa software, en un momento dado, bajo las condiciones especificadas (SAE JA1002).

Firmware: La combinación de un dispositivo hardware e instrucciones de ordenador o datos de ordenador que residen como un software de sólo lectura en el dispositivo hardware. El software no se puede modificar fácilmente bajo el control de un programa (ISO 12207).

Línea de Referencia (Línea de referencia Software): Una especificación, documento o producto que ha sido revisado formalmente y decidido en un momento dado y que desde entonces sirve como base para posterior desarrollo. Sólo puede cambiarse de forma justificada y aprobada mediante un procedimiento formal de control.

Mantenibilidad del Software: La facilidad con que un sistema o componente software puede modificarse para corregir defectos, mejorar su funcionamiento u otros atributos, o adaptarlo a un entorno cambiante. También un conjunto de atributos que se relacionan con el esfuerzo necesario para efectuar modificaciones específicas (SAE JA1004).

Mejora: Nueva versión de un programa software existente que se ha modificado para proporcionar una nueva capacidad.

Modelo: Una representación física, matemática o lógica de un sistema, una entidad real un fenómeno o un proceso.

Simulación: La implementación de un modelo temporal.

Simulador: Un ser humano en el bucle (HITL –*Human In The Loop*) de simulación.

Software de Sistema: El software de sistema (también conocido como software Operativo) es cualquier software que se necesita para soportar la producción o ejecución de los programas de aplicación, pero que no es específico a una aplicación particular. Los sistemas operativos son ejemplos del software de sistema.

Software: Las instrucciones ejecutadas por un ordenador, en contraposición al dispositivo físico en que se ejecutan (el "*hardware*"). El software puede dividirse en dos tipos fundamentales: software del sistema y software de aplicación o programas de aplicación.

Soportabilidad del Software: Un conjunto de atributos del diseño software, las herramientas y métodos de desarrollo asociados y el entorno de infraestructura de apoyo que permiten que se puedan efectuar las actividades de soporte. (SAE JA1004).

Verificación y Validación (VyV): El proceso de determinar si los requisitos de un componente o sistema son completos y correctos, los productos de cada fase de desarrollo, cumplen los requisitos o condiciones impuestas por la fase previa y el sistema o componente final cumple con los requisitos especificados. (IEEE Standard 610.12).

Verificación y Validación Independientes (VyVI): Verificación y validación efectuada por una organización que es independiente técnicamente, en su gestión, y en su financiación de la organización desarrolladora.

Verificación: Confirmación, mediante examen de una evidencia objetiva de que se han cumplido los requisitos especificados (IEEE Standard 610.12).

Glosario de Términos

Acrónimo principal:

ARMP	*Allied Reliability and Maintainability Publication –NATO–*. (Publicación OTAN de F/M)
BIT	*Built In Test*
BITE	*Built In Test Equipment*
CASE	*Computer Aided Software Engineering*
CM	*Configuration Management*
CMMI	*Capability Maturity Model Integration*
COTS	*Commercial, off-the-shelf* (Elementos comerciales disponibles en el mercado)
EEFPROM	*Electrically Erasable Programmable Read Only Memory*
FRACAS	*Failure Reporting Analysis and Corrective Action System*
FyM	Fiabilidad y Mantenibilidad
FyS	Fiabilidad y Soportabilidad
HITL	*Human In The Loop*
IEC	*International Electrotechnical Commission*
IETM-	*Iterative Electronic Technical Manual*

IPR	*Intellectual Property Rights*
ISC	*Incident Sentencing Committee*
ISO	*International Organization for Standardization*
JA	*Two character code for SAE ground vehicle (J) and aerospace (A) standards and guidelines*
KSLOC	*Thousands (K) of Source Lines of Code*
MTBF	*Mean Time Between Failures*. Este valor refleja el Tiempo Medio entre Fallos a partir de la probabilidad de que éstos se produzcan. Es decir, se calcula como la inversa del valor FR.
NATO	*North Atlantic Treaty Organization*
OTAN	Organización del Tratado del Atlántico Norte (*NATO –North Atlantic Treaty Organization*–)
PEFM	Publicaciones Españolas de Fiabilidad y Mantenibilidad
POFOD	Probabilidad de Fallo en la Demanda (*Probability of Failure on Demand*)
PP F/M	Plan de Programación de Fiabilidad y Mantenibilidad
PROM	Programmable Read Only Memory
ROCOF	Tasa de ocurrencia del Fallo (*Rate of occurrence of failure*)
SAE	*Society of Automotive Engineers*
SDP	*Software Development Plan*
SEI	*Software Engineering Institute*
SLOC	*Source Lines of Code*
SRS	*Software Requirements Specification*
UML	*Unified Modelling Language*
VyV	*Verificación y Validación*

Página intencionadamente en blanco

B. EJEMPLO DE MÉTRICAS DE FIABILIDAD DEL SOFTWARE

El software tiene características estadísticas, que excluyen el uso de métricas de fiabilidad del hardware tales como el MTBF- *Mean Time Between Failures*. Así, la industria del software ha desarrollado algunas métricas específicas para el software que se muestran en la tabla siguiente:

Métrica	Descripción	Enfoque
POFOD ***Probability of failure on demand***	Mide la probabilidad de que un sistema falle cuando se efectúa la petición de un servicio. POFOD = 0.001, significa que 1 de cada 1000 peticiones de servicio puede ser fallida. Se contabilizan el número de fallos y el de peticiones de servicio.	¿Cuál es la probabilidad de poder servir una petición de servicio discreta?
ROCOF ***Rate of occurrence of failure***	Mide la frecuencia de ocurrencia de un comportamiento incorrecto. ROCOF = 0.02 significa que pueden ocurrir 2 fallos en 100 unidades operacionales de tiempo. Se contabiliza el número de fallos durante el tiempo total transcurrido.	¿Cuántos fallos es probable que ocurran en un periodo de tiempo especificado?
MTTF ***Mean time to failure***	Medición del tiempo entre fallos observados en el sistema. MTTF = 1500 significa que puede ocurrir un fallo cada 1500 unidades operacionales de tiempo. Nótese que el MTTF y el ROCOF son inversos uno del otro.	¿Cuánto es el intervalo de tiempo entre sucesos de fallo? No se consideran las consecuencias del fallo.

Página intencionadamente en blanco

C. ESQUEMA DE LA PETICIÓN DE OFERTAS EN LA ADQUISICIÓN DE SOFTWARE

1. Ofertas técnicas:
   a) Guía para las ofertas técnicas;
   b) Criterios de evaluación de las ofertas técnicas.
2. Trabajos y entregables:
   a) Plan de trabajos;
   b) Lista de requisitos contractuales;
   c) Descripciones de los Elementos de datos;
   d) Lista de los Elementos de finalización del contrato.
3. Especificación de requisitos
4. Establecimiento de los requisitos operativos
5. Clases de Manuales Técnicos Electrónicos Interactivos (IETM- *Iteractive Electronic Technical Manual*-)
6. Material suministrado por el Gobierno

Fin del documento