BSSC (96)2 Nro1
May 1996

Guía para la aplicación de Estándares de Ingeniería de Software ESA
Para proyectos de software pequeños.

Preparado por:
ESA Comité de Estandarización y Control de Software
(BSSC)

Traducido por: Manuel Ibarra Cabrera
Julio 2010

Contenido

PREFACIO

Los estándares de ingeniería de software ESA PSS-05-0, definen las prácticas de software que deben aplicarse en todos los proyectos de la agencia. Mientras que su aplicación es bastante directa en grandes proyectos, la experiencia ha demostrado que un enfoque simplificado es apropiado para proyectos de software pequeños. Este documento proporciona las pautas sobre cómo aplicar los estándares de ingeniería de software (ESA) a proyectos de software pequeños. Por consiguiente a la guía se le ha dado el seudónimo de "PSS-05 lite".

Los siguientes miembros de BSSC, tanto los pasados como los actuales, han contribuido a la elaboración de esta guía: Michael Jones (co-presidente), Uffe Mortensen (co-presidente), Gianfranco Alvisi, Bryan Melton, Daniel de Pablo, Adriaan Scheffer y Jacques Marcoux. La BSSC desea agradecer a Jon Fairclough por redactar y editar la guía. Los autores desean agradecer a todas las personas que aportaron con ideas sobre la aplicación de ESA PSS-05-0 a proyectos pequeños.

Las solicitudes para aclaraciones, propuestas de cambios u otros comentarios relacionados con esta guía deben dirigirse a:

Secretaria de BSSC/ESOC
Atención Señor M. Jones
ESOC
Robert Bosch Strasse 5
D-64293 Darmstadt
Alemania

Secretaría BSS/ESTEC
Atención Señor U. Mortensen
ESTEC
Postbus 299
NL-2200 AG Noordwijk
Holanda.

CAPÍTULO 1: INTRODUCCIÓN

1.1 PROPÓSITO

ESA PSS-05-0 describe los estándares de ingeniería de software que se aplicarán a todos los artefactos software implementados por la Agencia Espacial Europea (ESA) [Referencia 1].

Este documento se ha producido para proporcionar directrices a las organizaciones y a los administradores de los proyectos de software, como una guía para la aplicación de los estándares a proyectos de software pequeños. En una serie de documentos descritos en ESA PSS-05-0, se entregan directrices adicionales para la aplicación de los estándares, Guía de los Estándares de Ingeniería de Software [Referencias 2 a 12].

Existen muchos criterios para decidir si un proyecto de software es pequeño, y por lo tanto si se puede aplicar esta guía. Esto se tratará en el Capítulo 2. Cualquiera sea el resultado de la evaluación, esta guía no debería aplicarse cuando el software es crítico (por ejemplo, un fallo de un software podría provocar la pérdida de vidas, propiedades, o mayores inconvenientes a los usuarios) o cuando se aplica aseguramiento de los requisitos de producto de software para ESA Space System ESA PSS-01-21

1.2 ASPECTOS GENERALES

En el capítulo 2 se define qué es lo que se entiende por un proyecto de software pequeño, y se analizan algunas estrategias simples para la aplicación de ESA PSS-05-0 en los mismos.

El capítulo 3 explica como deberían aplicarse las prácticas obligatorias de ESA PSS-05-0 en proyectos de software pequeños. El apéndice A contiene un glosario de siglas y abreviaturas. El apéndice B contiene una lista de referencias. El apéndice C contiene plantillas de documentos simplificados.

CAPÍTULO 2: PROYECTOS DE SOFTWARE PEQUEÑOS

2.1 INTRODUCCIÓN

En ESA PSS-05-0 aparece un listado con varios factores a considerar cuando se aplican a un proyecto de software (vea la Introducción, Sección 4.1). Los factores que están relacionados con el tamaño del desarrollo de un proyecto de software son:

- Costo del desarrollo del proyecto.
- Cantidad de personas que se necesitan para desarrollar el software.
- Cantidad que se va a producir de software.

Puede considerarse que un proyecto de software es pequeño si se aplican uno o más de los siguientes criterios:

- Si se necesitan menos de dos años hombre de esfuerzo para el desarrollo.
- Si se requiere un equipo único de desarrollo de cinco o menos personas.
- Si la cantidad de código fuente es inferior a 10000 líneas, excluyendo los comentarios.

Con frecuencia una o más de las siguientes estrategias son apropiadas a los proyectos pequeños que generan software no crítico:

- Combinación de los requisitos de software y las fases del diseño arquitectónico.
- Documentación simplificada.
- Simplificación de los planes.
- Reducción de la formalidad de los requisitos.
- Utilización de las especificaciones del plan de pruebas para las pruebas de aceptación.

En las siguientes secciones se revisarán estas estrategias.

2.2 COMBINACIÓN DE LAS FASES RS Y DA

ESA PSS-05-0 requiere que la definición de los requisitos de software y el diseño arquitectónico se desarrollen en fases separadas. Estas fases terminan con una revisión formal del Documento de Requisitos de Software y el Documento del Diseño Arquitectónico. Por lo general, cada una de estas revisiones involucra al usuario, y puede durar entre dos semanas y un mes. Para un proyecto de software pequeño, las revisiones separadas de DRS y DDA pueden prolongar de manera significativa el proyecto. Por lo tanto, una manera eficiente de organizar el proyecto es:

- Combinar las fases RS y DA en una fase única RS/DA.
- Combinar las fases RS/R y DA/R en una única revisión formal al final de la fase RS/DA.

2.3 SIMPLIFICACIÓN DE LA DOCUMENTACIÓN

Las plantillas de documentos proporcionados en ESA PSS-05-0 se basan en los estándares ANSI/IEEE, y están diseñados para satisfacer los requisitos de documentación de todos los proyectos. Los desarrolladores deberán utilizar las plantillas de documentos simplificadas que se entregan en el Apéndice C.

Los desarrolladores deberán combinar el DRS y DDA en un único Documento de Especificación de Software (DES) cuando la fase SR y AD se combinan.

Los desarrolladores deberán documentar el diseño detallado poniendo la mayor cantidad de información detallada del diseño en el código fuente y extendiendo el DES para que incluya cualquier información de diseño detallada que no pueda ubicarse en el código fuente (por ejemplo: La información sobre la estructura de software).

La elaboración de un Documento Histórico del Proyecto se considera una buena práctica, pero su entrega es opcional.

2.4 SIMPLIFICACIÓN DE LOS PLANES

En los proyectos de software pequeños es aconsejable planificar todas las fases al comienzo de los mismos. Esto significa que la sección de las fases del Plan de Administración del Proyecto de Software, Plan de Administración de la Configuración de Software y los Planes de Validación y Verificación de Software se combinan. Los planes pueden generarse cuando se escribe la propuesta de producto.

El propósito de la función de aseguramiento de calidad de software en un proyecto consiste en revisar que el proyecto siga los estándares y planes. Normalmente en un proyecto de software pequeño el administrador del proyecto es quien lo realiza de manera informal. La práctica de producir un Plan de Aseguramiento de Calidad puede postergarse.

ESA PSS-05-0 requiere que la propuesta de prueba sea incluida en el plan de prueba y refinada en el diseño de pruebas. En los proyectos de software pequeños, basta con delinear sólo la propuesta de pruebas. Esto significa que las secciones del Diseño de Pruebas de los SVVP pueden omitirse.

Los desarrolladores pueden considerar que esto es conveniente para combinar los procesos técnicos de la sección del Plan de Administración de Proyecto de Software, el Plan de Administración de Configuración de Software, Plan de Validación y Verificación de software y el Plan de Aseguramiento de Calidad en un único documento (llamado Plan de Calidad en terminología ISO 9000).

2.5 REDUCCIÓN DE LA FORMALIDAD DE LOS REQUISITOS

Las formalidades de los requisitos de un software debieran ajustarse de acuerdo al costo de corrección de los defectos durante el desarrollo frente al costo de:
- Corrección de los defectos durante las operaciones.
- Penalizaciones ocasionadas por los fallos producidos durante la operación.
- Pérdida de reputación sufrida a causa de un producto defectuoso.

La formalidad de un software se logra:
- Diseñando el software para que sea formal.
- Revisando los documentos y código.
- Probando el código.

ESA PSS-05-0 requiere que cada requisito sea probado por lo menos una vez. A veces esto se conoce como el requisito de cobertura de requisito. Hay evidencia de que una cantidad

significativa de defectos permanece en el software cuando el alcance del requisito baja del 70% [Referencia 14], y de que la cantidad de defectos disminuye significativamente cuando la cobertura del requisito sobrepasa el 90%. Sin embargo, lograr una alta protección en las pruebas puede ser costoso en términos de esfuerzo. Los casos de prueba tienen que inventarse para forzar la ejecución de todos los requisitos. El costo de esta prueba exhaustiva puede exceder el costo de reparación de los defectos durante las operaciones.

Los desarrolladores deberían:
- Establecer un objetivo de pruebas de requisitos (por ejemplo: cobertura del 80%).
- Revisar cada requisito no cubierto en las pruebas.

Existen herramientas software disponibles para medir la cobertura de prueba y deberían usarse siempre vez que sea posible.

2.6 UTILIZACIÓN DE LAS ESPECIFICACIONES DEL PLAN DE PRUEBAS PARA LAS PRUEBAS DE ACEPTACIÓN

Cuando el desarrollador es responsable de producir la especificación de la aceptación de pruebas, a menudo las pruebas de aceptación repiten procedimientos y casos de prueba del sistema seleccionado. Un método para la documentación de pruebas de aceptación del sistema consiste simplemente en indicar en la especificación de pruebas de sistema cuales de los casos de prueba y procedimientos debieran ser utilizados en las pruebas de aceptación.

CAPÍTULO 3: PRÁCTICAS PARA PROYECTOS DE SOFTWARE PEQUEÑOS

3.1 INTRODUCCIÓN

La figura 3.1 muestra una aproximación de un ciclo de vida de un proyecto de software pequeño, con las siguientes características:

- Combinación de los requisitos de software y fase del diseño arquitectónico.
- Utilización de las plantillas de documentos simplificadas, descritas en el Apéndice C.
- Información detallada del diseño incluida en el código fuente.
- Combinación de los planes.
- Disminución de los requisitos de formalidad.
- No hay elaboración de documentación de diseño de pruebas.
- Selección de pruebas del sistema que se usan para la aceptación de pruebas.
- No hay aplicación de todas las prácticas de SQA.
- No hay obligación de entregar el Documento Histórico del Proyecto.

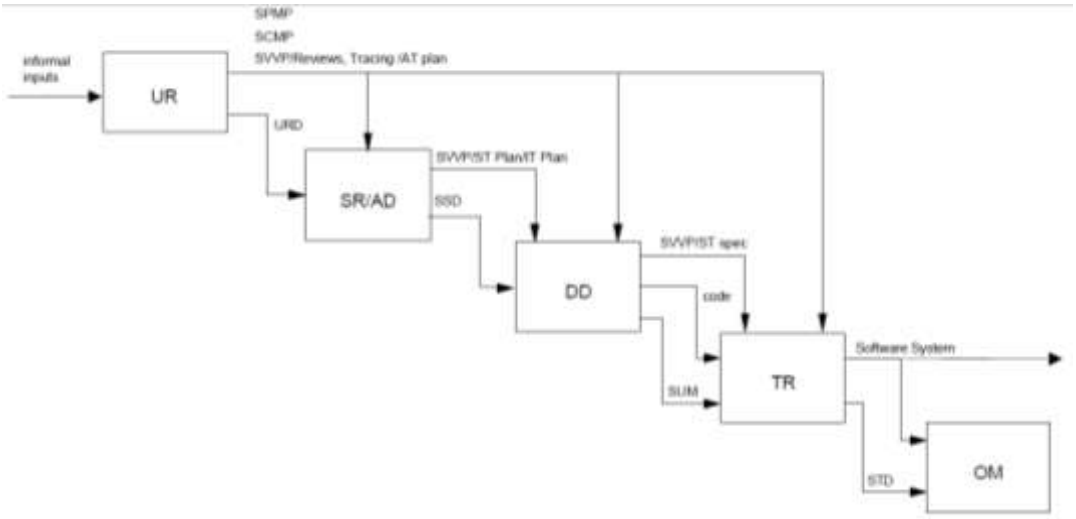

Figura 3.1: Ciclo de vida de un proyecto de software pequeño

Las siguientes secciones describen las prácticas obligatorias en un proyecto de software pequeño. Las prácticas se tabulan con su identificador ESA PSS-05-0.

La adjuntan guías (en *itálica*) en caso de que sea apropiado explicar cómo se aplican estas prácticas en un proyecto pequeño. Estas guías se basan en las estrategias descritas en el Capítulo 2.

3.2 CICLO DE VIDA DE SOFTWARE

- **CVS01** Los productos de desarrollo de un software deberán ser entregados de manera oportuna y deberán cumplir con su propósito.
- **CVS02** Las actividades de desarrollo de un software deberán ser planeadas sistemáticamente y llevadas a cabo.

• **CVS03** Todo proyecto software deberá tener una aproximación de ciclo de vida que incluya las fases básicas, que se muestran en la Parte 1, Figura 1.2 (de PSS-05) (*incluido a continuación*):

- Fase RU - Definición de los requisitos de usuario.
- Fase RS - Definición de los requisitos de software.
- Fase DA - Definición del diseño arquitectónico.
- Fase DD - Diseño detallado y producción del código.
- Fase TR - Transferencia de software a las operaciones.
- Fase OM - Operaciones y mantención.

En proyectos pequeños, las fases RS y DA están combinadas en una fase RS/DA

3.3 FASE RU

• **RU01** Deberá ser responsabilidad del usuario la definición de los requisitos de usuario.
• **RU02** Cada requisito de usuario deberá incluir un identificador.
• **RU03** Los requisitos esenciales de usuario deberán ser marcados como tales.
• **RU04** Para hacer entregas incrementales, cada requisito de usuario deberá incluir un grado de prioridad, de tal manera que el desarrollador pueda decidir la agenda de producción.
• **RU05** Deberá establecerse la fuente de cada requisito de usuario.
• **RU06** Deberá verificarse cada requisito de usuario.
• **RU07** El usuario deberá describir las consecuencias de las pérdidas de disponibilidad, o violaciones de seguridad, de tal manera que los desarrolladores puedan apreciar en su totalidad lo crítico de cada función.
• **RU08** Deberán revisarse formalmente las salidas de la fase RU durante la Revisión de Requisitos de Usuario.
• **RU09** Deberán señalarse claramente los requisitos de usuario no aplicables.
• **RU10** Un producto de la fase RU deberá ser el Documento de Requisitos de Usuario (DRU).
• **RU11** El DRU deberá producirse siempre antes de que el proyecto de software empiece.
• **RU12** El DRU proporcionará una descripción general de lo que el usuario espera que haga el software.
• **RU13** Deberán incluirse todos los requisitos de usuario conocidos en el DRU.
• **RU14** El DRU deberá describir las operaciones que el usuario quiere realizar con el sistema software.
• **RU15** El DRU deberá definir todas las restricciones a las que el usuario desea imponer una solución.
• **RU16** El DRU deberá describir las interfaces externas del sistema software o deberá hacer mención de ellas en los Documentos de Control de Interfaces que existan o que se deben escribir.

3.4 FASE RS/DA

• **RS01** Las actividades de la fase RS deberán ser llevadas a cabo de acuerdo a los planes definidos en la fase RU.
• **RS02** El desarrollador deberá construir un modelo de implementación independiente de lo que necesita el usuario.
• **RS03** Deberá adoptarse un método reconocido para analizar los requisitos de software y aplicarlo adecuadamente en la fase RS.
Esta práctica se aplica a la fase RS/DA
• **RS04** Cada requisito de software deberá incluir un identificador.
• **RS05** Deberán señalarse como tales los requisitos esenciales de un software.
• **RS06** Para entregas incrementales cada requisito de software deberá incluir un grado de prioridad de tal manera que el desarrollador pueda decidir la agenda de producción.
• **RS07** Los requisitos software deberán ir acompañados de las referencias a los RU que trazan, según el etiquetado del DRU.
• **RS08** Deberá ser verificable cada requisito de software.
• **RS09** Las salidas de la fase RS deberán ser formalmente revisadas durante la Revisión de los Requisitos de Software.
Las salidas de las actividades de definición de los requisitos de software se revisaron en la fase de repaso RS/DA.
• **RS10** Una salida de la fase RS será el Documento de Requisitos de Software DRS.
Esta práctica se aplica a la fase RS/AD.
La información de DRS se establece en el Documento de Especificación de Software (DES).
• **RS11** El DRS debe ser completo.
Esta práctica se aplica al DES.
• **RS12** El DES deberá cubrir todos los requisitos establecidos en DRU
Esta práctica se aplica al DES.
• **RS13** Una tabla mostrará como los requisitos de usuario corresponden a los requisitos de software que deberán ser establecidos en el DRS.
Esta práctica se aplica al DES.
• **RS14** El DRS deberá ser consistente.
Esta práctica se aplica al DES.
• **RS15** El DRS no deberá incluir detalles de implementación o terminología, a menos que estos tengan que presentarse como una restricción.
Esta práctica se aplica al DES.
• **RS16** Descripciones de funciones...deberán decir para qué es el software, y evitar decir cómo se elaboró.
• **RS17** El DRS deberá evitar la especificación del hardware o equipamiento, a menos que esto sea una restricción establecida por el usuario.
Esta práctica no se aplica.
• **RS18** El DRS deberá compilarse de acuerdo al índice estipulado en el Apéndice C.
Esta práctica se aplica al DES. El DES deberá compilarse de acuerdo al índice en el Apéndice C de esta guía.
• **DA01** Las actividades de la fase DA deberán realizarse de acuerdo a los planes definidos en la fase RS.
En proyectos pequeños, los planes de DA deberán realizarse en la fase RU.
• **DA02** Deberá adoptarse y aplicarse adecuadamente un método reconocido para el diseño de software en la fase DA.
Esta práctica se aplica a la fase RS/DA.
• **DA03** El desarrollador deberá construir un modelo físico, que describa el diseño de software utilizando terminología de implementación.

• **DA04** El método utilizado para descomponer el software en las partes que lo componen deberá permitir un acercamiento "Top-Down".
• **DA05** Deberá reflejarse en el DDA sólo el método del diseño seleccionado.
Esta práctica se aplica al DES.
La siguiente información para cada componente deberá detallarse en el DDA.
• **DA06** * Datos de entrada;
• **DA07** * Funciones que se van a realizar;
• **DA08** * Datos de Salida.
DA06, 7, 8 se aplican al DES.
• **DA09** Deberá definirse en el DDA la estructura de los datos que componen la interfaz.
Esta práctica se aplica al DES.
Las definiciones de la estructura de datos deberán incluir:
• **DA10** * La descripción de cada elemento (por ejemplo: nombre, tipo, tamaño);
• **DA11** * Las relaciones entre los elementos (por ejemplo: la estructura);
• **DA12** * La variación de los posibles valores de cada elemento;
• **DA13** * Valores iniciales de cada elemento *DA10, 11, 12, 13 se aplican al DES.*
• **DA14** Deberá definirse en el DDA el flujo de control entre los componentes.
Esta práctica se aplica al DES.
• **DA15** Los recursos del computador (por ejemplo: Velocidad de CPU, memoria, almacenamiento, software del sistema) necesarios en el ambiente de desarrollo y en el ambiente operacional deberán evaluarse en la fase DA y definirse en el DDA.
Esta práctica se aplica al DES.
• **DA16** Las salidas de la fase DA deberán revisarse formalmente durante la Revisión del Diseño Arquitectónico.
Los productos de las actividades del diseño arquitectónico de software se revisan en la fase de revisión RS/DA.
• **DA17** El DDA definirá los principales componentes de software y las interfaces entre ellos.
Esta práctica se aplica al DES.
• **DA18** El DDA mencionará todas las interfaces externas.
Esta práctica se aplica al DES.
• **DA19** El DDA será una salida de la fase DA.
Esta práctica se aplica al DES.
• **DA20** El DDA se completará, abarcando todos los requisitos de software descritos en el DRS.
Esta práctica se aplica al DES.
• **DA21** Se establecerá en el DDA una tabla de referencias cruzadas para trazar los requisitos de software del diseño arquitectónico.
Esta práctica se aplica al DES.
• **DA22** El DDA deberá ser consistente.
Esta práctica se aplica al DES.
• **DA23** El DDA deberá ser lo suficientemente detallado para permitir al jefe de proyecto preparar un documento con un plan de implementación detallado y para controlar todo el proyecto durante las fases de desarrollo restantes.
Esta práctica se aplica al DES.
• **DA24** El DDA deberá compilarse de acuerdo al índice estipulado en el Apéndice C.
Esta práctica se aplica al DES. El DES debería compilarse de acuerdo al índice que se encuentra en el Apéndice C de esta guía.

3.5 FASE DD

• **DD01** Las actividades de la fase DD deberán realizarse de acuerdo a los planes definidos en la fase DA.
Los planes de la fase DD están contenidos en los planes desarrollados en la fase RU y actualizados a medida que corresponda durante el proyecto.
El diseño detallado y producción de un software deberá basarse en los tres principios siguientes:
• **DD02** * Descomposición "Top-Down";
• **DD03** * Programación estructurada /Programación OO;
• **DD04** * Documentación y producción concurrente.
• **DD05** Los procesos de integración deberán ser controlados por los procedimientos de gestión de configuración de software, definidos en PGCS.
• **DD06** Antes de que un módulo pueda aceptarse, cada requisito deberá ejecutarse exitosamente por lo menos una vez.
La aceptación debería realizarla el administrador del proyecto después de las pruebas unitarias y por el cliente al final de la fase.
Los proyectos de software deberían:
- Establecer un objetivo de pruebas de requisitos (por ejemplo: cobertura del 80%).
- Revisar cada requisito no cubierto en las pruebas.
Existen herramientas software disponibles para medir la cobertura de las pruebas. Deberán utilizarse siempre que sea posible.
• **DD07** Las pruebas de integración deberán verificar que toda la información intercambiada a través de una interfaz, concuerde con las especificaciones de la estructura de datos en el DDA.
Esta práctica se aplica al DES.
• **DD08** Las pruebas de integración deberán confirmar que el flujo de control definido en el DDA ha sido implementado.
Esta práctica se aplica al DES.
• **DD09** El sistema de pruebas deberá verificar la concordancia con los objetivos del sistema, como está establecido en el DRS.
Esta práctica se aplica al DES.
• **DD10** Cuando el diseño de un componente principal esté terminado, deberá acordarse una revisión crítica del diseño para certificar su aptitud y conveniencia para la implementación.
• **DD11** Después de la producción, la revisión del DD (R/DD) deberá considerar los resultados de las actividades de verificación y decidir cual transferir al software.
• **DD12** Todo el código entregable deberá estar identificado en una lista de elementos de configuración.
• **DD13** El DDD deberá ser una salida de la fase DD.
No aplicable. La información detallada del diseño está establecida en el DES y en el código fuente.
• **DD14** La parte 2 del DDD deberá tener la misma estructura y esquema de identificación que el código en sí mismo, con una correspondencia de 1:1 entre las secciones de la documentación y los componentes de software.
No aplicable.
• **DD15** El DDD deberá completarse, con una descripción de todos los requisitos de software en del DRS.
Esta práctica significa que todos los requisitos en el DES deben implementarse en el código.
• **DD16** Deberá establecerse en el DDD una tabla de referencias cruzadas de los requisitos de software para el diseño detallado de los componentes.

En su lugar se deberá actualizar la matriz de trazabilidad en el DES.
• **DD17** El Manual de Usuario de Software (MUS) deberá ser una salida de la fase DD.

3.6 FASE TR

• **TR01** Los representantes de los usuarios y personal de operaciones deberán participar en las pruebas de aceptación.
• **TR02** El Comité de Revisión de Software (del inglés SRB) deberá revisar el desempeño del software en las pruebas de aceptación y sugerir al encargado del proyecto en la empresa cliente (del término *initiator*), si el software puede aceptarse de manera provisional o no.
• **TR03** Las actividades de la fase TR deberán llevarse a cabo de acuerdo a los planes definidos en la fase DD.
Los planes de la fase TR serán establecidos en la fase RU y actualizados cuando corresponda.
• **TR04** Deberá establecerse la capacidad de construir el sistema comenzando por los componentes que son directamente modificables por el equipo de mantenimiento.
• **TR05** Deberán indicarse en el PVVS las pruebas de aceptación necesarias para la aceptación provisional.
• **TR06** El informe de la aceptación provisional deberá elaborarlo el encargado del proyecto en la empresa cliente, en favor de los usuarios, y enviarlo al desarrollador.
• **TR07** La aceptación provisional del sistema de software deberá consistir en las salidas de todas las fases previas y en las modificaciones que sean necesarias en la fase TR.
• **TR08** Una salida de la fase TR será el DTS.
• **TR09** El DTS será entregado por el desarrollador a la organización de mantenimiento para su aceptación provisional.
• **TR10** El DTS deberá incluir el resumen de los informes de las pruebas de aceptación, y toda la documentación sobre los cambios del software, realizados durante la fase TR.

3.7 FASE OM

• **OM01** Hasta la aceptación final, las actividades de la fase OM que involucren al desarrollador deberán llevarse a cabo de acuerdo a los planes definidos en el PAPS/TR.

• **OM02** Todas las pruebas de aceptación deberán completarse exitosamente antes de que el software sea finalmente aceptado.
• **OM03** Incluso cuando el contratista no esté involucrado, habrá un hito de aceptación final para acordar la entrega formal del desarrollo de software a mantenimiento.
• **OM04** Deberá designarse una organización de mantenimiento para cada producto de software en producción.
• **OM05** Deberán definirse los procedimientos para las modificaciones de software.
• **OM06** Deberá mantenerse la consistencia entre el código y la documentación.
• **OM07** Deberán asignarse recursos al mantenimiento del producto hasta que éste sea retirado.
• **OM08** El **SRB** deberá autorizar todas las modificaciones del software.
• **OM09** La información de la aceptación final deberá producirla el responsable del proyecto en la empresa cliente , a favor de los de los usuarios, y enviarla al desarrollador.

• **OM10** El DHP (Documento Histórico del Proyecto) deberá entregarse al encargado del proyecto en la empresa cliente después de la aceptación final.

No aplicable. Sin embargo, se alienta a los desarrolladores para que produzcan un DHP para uso interno.

3.8 ADMINISTRACIÓN DE PROYECTOS DE SOFTWARE

• **APS01** Todas las actividades de administración de proyectos de software deberán documentarse en el Plan de Administración de Proyectos de Software (PAPS).
• **APS02** Al término de la revisión de RU, deberá producirse la sección de la fase RS de PAPS.
El PAPS no tiene secciones de fase: sólo hay un plan único para todas las fases RS/DA, DD y TR.
• **APS03** El PAPS deberá definir un plan para todo el proyecto.
Esta práctica se aplica al PAPS.
• **APS04** Deberá incluirse en la fase PASP/RS un cálculo exacto del esfuerzo involucrado en la fase RS.
Esta práctica se aplica al PAPS.
• **APS05** Deberá producirse (PAPS/DA) la sección de la fase DA del PAPS durante la fase RS.
No aplicable.
• **APS06** Deberá incluirse en el PAPS/DA un costo estimado del total del proyecto.
• **APS07** Deberá incluirse en la fase PASP/DA un cálculo exacto del esfuerzo involucrado en la fase DA.
Esta práctica se aplica al PAPS.
• **APS08** Deberá producirse (PAPS/DD) la sección de la fase DD del PAPS durante la fase DA.
No aplicable. Sin embargo, el PAPS debería actualizarse cuando el DES se ha producido.
• **APS09** Deberá incluirse en el PAPS/DD un costo estimado del total del proyecto.
Esta práctica se aplica al PAPS.
• **APS10** El PAPS/DD deberá contener un WBS que esté directamente relacionado con la descomposición de software en componentes.
Esta práctica se aplica al PAPS.
• **APS11** El PAPS deberá contener una red de planificación que muestre la relación entre las actividades de codificación, integración y prueba.
Debería utilizarse un diagrama de barras (por ejemplo: Diagrama Gant) para mostrar la relación.
• **APS12** El trabajo de producción de paquetes de software en PAPS/DD no debería durar más de 1 mes-hombre.
Esta práctica se aplica al PAPS.
• **APS13** Durante la fase DD deberá producirse la sección correspondiente a la fase TR del PAPS (PAPS/TR).
Esta práctica se aplica al PAPS.

3.9 GESTIÓN DE LA CONFIGURACIÓN DE SOFTWARE

- **ACM01** Todos los artefactos, por ejemplo la documentación, códigos fuente, códigos objeto, códigos ejecutables, archivos, herramientas, información y pruebas de software estarán sujetos a los procedimientos de la gestión de la configuración.
- **ACM02** Los procedimientos de gestión de la configuración establecerán métodos para la identificación, almacenamiento y cambios en los ítems de software, por medio del desarrollo, integración y transferencia.
- **ACM03** Deberá utilizarse un conjunto común de procedimientos de gestión de configuración.

Cada ítem de configuración deberá tener un identificador que lo distinga de otros ítems con diferente(s):

- **ACM04** * requisitos, especialmente funcionalidades e interfaces.
- **ACM05** * implementación.
- **ACM06** Cada componente definido en el proceso de diseño deberá designarse como un Elemento de Configuración (EI) (del término *Configuration Item: CI*) y poseer un identificador.
- **ACM07** El identificador deberá incluir un número o un nombre relacionado con el propósito del EI.
- **ACM08** El identificador deberá indicar el tipo de procesamiento del EI, su finalidad (por ejemplo: Archivo de información).
- **ACM09** El identificador de un CI deberá incluir un número de versión.
- **ACM10** El identificador de documentos deberá incluir un número de publicación y un número de revisión.
- **ACM11** El método de identificación de la configuración deberá ser capaz de acomodar los nuevos EIs, sin requerir la modificación de los identificadores de cualquier EIs existente.
- **ACM12** En la fase TR, deberá incluirse en el DTS una lista de los elementos de configuración en la primera versión.
- **ACM13** En la fase OM, deberá incluirse en cada Nota de Entrega de Software (NES), una lista de los elementos de configuración que se cambiaron.
- **ACM14** Una NES deberá acompañar cada entrega realizada en la fase OM. Como parte del método de identificación de configuración, un módulo de software tendrá una cabecera que incluya:
- **ACM15** * identificador de elemento de configuración (nombre,tipo, versión);
- **ACM16** * Autor original;
- **ACM17** * Fecha de creación;
- **ACM18** * Cambios históricos (versión/fecha/autor/descripción).

Toda la documentación y almacenamiento deberá estar claramente etiquetada en un formato estándar, incluyendo al menos la siguiente información:

- **ACM19** * nombre del proyecto;
- **ACM20** * Identificador del elemento de configuración (nombre, tipo, versión);
- **ACM21** * fecha;
- **ACM22** * descripción del contenido.

Para asegurar la seguridad y control del software, como mínimo, deberán implementarse las siguientes librerías de software para almacenar todos los componentes disponibles (por ejemplo: documentación, códigos fuente y ejecutable, archivos de prueba, procedimientos de instrucciones):

- **ACM23** * librería de Desarrollo (o Dinámica);
- **ACM24** * librería Maestra (o Controlada);
- **ACM25** * librería Estática (o Archivo).

- **ACM26** No deberán modificarse las librerías estáticas.
- **ACM27** Las copias de seguridad actualizadas de las librerías Estáticas y Maestra deberán estar siempre disponibles.
- **ACM28** Deberán establecerse los procedimientos para realizar copias periódicas de respaldo de las librerías desarrolladas.
- **ACM29** El cambio del procedimiento descrito (en Parte 2, Sección 3.2.3.2.1) deberá observarse cuando sean necesarios los cambios para entregar el documento.
- **ACM30** Los problemas de software y las propuestas de cambio deberán manejarse con el procedimiento descrito (en Parte 2, Sección 3.2.3.2.2)
- **ACM31** Deberá grabarse el estado de todos los elementos de configuración.

Para realizar el estado de contabilidad del software, cada proyecto de software deberá registrar:

- **ACM32** * la fecha y versión/número de cada línea base;
- **ACM33** * la fecha y el estado de cada RID y DCR.
- **ACM34** * la fecha y estado de cada SPR, SCR y SMR;
- **ACM35** * una descripción resumida de cada Elemento de Configuración.
- **ACM36** Como mínimo, la NES deberá almacenar los defectos que se han reparado y los nuevos requisitos que se han incorporado.
- **ACM37** Para cada entrega, la documentación y código deberán ser consistentes.
- **ACM38** Las entregas antiguas se guardarán como referencia.
- **ACM39** El software modificado deberá probarse de nuevo antes de la entrega.
- **ACM40** Todas las actividades de gestión de configuración de software deberán documentarse en el Plan de Gestión de Configuración de Software (PGCS).
- **ACM41** Los procedimientos de Gestión de Configuración deberán establecerse antes de que la producción del software comience (código y documentación).
- **ACM42** Al finalizarse la revisión de RU, deberá elaborarse la sección correspondiente a la fase RS del PGCS (PGCS/RS).

Esta práctica se aplica al PGCS.

- **ACM43** El PGCS/RS deberá cubrir los procedimientos de gestión de configuración para la documentación, y cualquier salida de herramienta "CASE" o código prototipo, producidos en la fase RS.

Esta práctica se aplica al PGCS.

- **ACM44** Durante la fase RS, deberá producirse la sección correspondiente a la fase DA del PGCS (PGCS/DA).

No aplicable.

- **ACM45** El PGCS/DA deberá cubrir los procedimientos de gestión de configuración para la documentación, y cualquier salida de herramienta "CASE" o código prototipo, producidos en la fase DA.

Esta práctica se aplica al PGCS.

- **ACM46** Durante la fase DA, se elaborará la sección correspondiente a la fase DD del PGCS (PGCS/DD).

No aplicable.

- **ACM47** El PGCS/DD deberá cubrir los procedimientos de gestión de configuración para la documentación, código disponible, y cualquier salida de herramienta "CASE" o código prototipo, producidos en la fase DD.

Esta práctica se aplica al PGCS.

- **ACM48** Durante la fase DD, se elaborará la sección de la fase TR del PGCS (PGCS/TR).

No aplicable.

- **ACM49** El PGCS/TR deberá cubrir los procedimientos para la gestión de la configuración de los entregables en el entorno operacional.

No aplicable.

3.10 VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE

• **VVS01** Para realizar la trazabilidad hacia delante se requiere que cada entrada hacia una fase se pueda rastrear hacia una salida de esa fase.
• **VVS02** La trazabilidad hacia atrás requiere que cada salida de una fase sea rastreable hasta una entrada de esa fase.
• **VVS03** Deberán llevarse a cabo las verificaciones físicas y funcionales antes de la entrega del software.
• **VVS04** Todas las actividades de verificación y validación de software deberán documentarse en el Plan de Verificación y Validación de Software (PVVS).
El PVVS deberá asegurar que las actividades de verificación:
• **VVS05** * sean apropiadas para el grado de criticidad del software;
• **VVS06** * satisfagan los requisitos de pruebas de aceptación y verificación (establecidos en el DRS);
• **VVS07** * verifiquen que el producto satisfaga los requisitos de calidad, confiabilidad, mantenimiento y seguridad...;
• **VVS08** * sean suficientes para asegurar la calidad del producto.
• **VVS09** Al finalizar la revisión RU deberá producirse la sección correspondiente a la fase RS del PVVS (PVVS/RS).
Esta práctica se aplica al PVVS.
• **VVS10** El PVVS/RS definirá como trazar los requisitos de usuario con los requisitos de software, de forma que cada requisito de software esté justificado.
Esta práctica se aplica al PVVS.
• **VVS11** El desarrollador deberá construir un plan de pruebas de aceptación en la fase RU en el PVVS.
• **VVS12** Durante la fase RS, deberá producirse la sección correspondiente a la fase DA del PVVS (PVVS/DA).
No aplicable.
• **VVS13** El PVVS/DA definirá como trazar los requisitos de software con los componentes, de forma que cada componente de software esté justificado.
Esta práctica se aplica al PVVS.
• **VVS14** El desarrollador deberá construir un plan de prueba del sistema en la fase RS y documentarlo en el PVVS.
• **VVS15** Durante la fase DA, deberá producirse la sección correspondiente a la fase DD del PVVS (PVVS/DD).
No aplicable
• **VVS16** El PVVS/DA deberá describir cómo el DDD y código serán evaluados, definiendo los procedimientos de trazabilidad y revisión.
Esta práctica se aplica al PVVS.
• **VVS17** El desarrollador deberá construir un plan de pruebas de integración en la fase DA y documentarlo en el PVVS.
Esto se realiza en la fase RS/DA.
• **VVS18** El desarrollador deberá construir un plan de pruebas unitarias en la fase DD y documentarlo en el PVVS.
• **VVS19** Los diseños de las pruebas unitarias, de integración, de sistema y de aceptación deberán estar descritos en el PVVS.
No aplicable.
• **VVS20** Los casos de prueba deberán describirse en el PVVS, para las pruebas unitarias de integración, sistema y aceptación.

• **VVS21** Deberán describirse en el PVVS los procedimientos de prueba de aceptación, unitaria, de integración y de sistema.
• **VVS22** Deberán describirse en el PVVS los informes de prueba de aceptación, unitaria, integración y sistema.

3.11 ASEGURAMIENTO DE CALIDAD DE SOFTWARE

• **ACS01** Cada empresa de desarrollo de software contratada deberá producir un PACS (Plan de Aseguramiento de Calidad de Software)
No aplicable.
• **ACS02** Todas las actividades de aseguramiento de calidad de software deberán documentarse en el Plan de Aseguramiento de Calidad de Software (PACS).
No aplicable.
• **ACS03** Para el término de la revisión RU, deberá producirse la sección correspondiente a la fase RS del PACS (PACS/RS).
No aplicable.
• **ACS04** El PACS deberá describir, en detalle, las actividades de aseguramiento de calidad para llevarlas a cabo en la fase RS.
No aplicable.
• **ACS05** El PACS/RS resumirá el plan de aseguramiento de calidad para el resto del proyecto.
No aplicable.
• **ACS06** Durante la fase RS deberá producirse la sección correspondiente a la fase DA del PACS (PACS/DA).
No aplicable.
• **ACS07** El PACS/DA cubrirá en detalle todas las actividades de aseguramiento de calidad que se llevarán a cabo en la fase DA.
No aplicable.
• **ACS08** Durante la fase DA deberá producirse la sección correspondiente a la fase DD de PACS (PACS/DD).
No aplicable.
• **ACS09** El PACS/DD cubrirá en detalle todas las actividades de aseguramiento de calidad que se llevarán a cabo en la fase DD.
No aplicable.
• **ACS10** Durante la fase DD deberá producirse la sección de la fase TR del PACS (PACS/TR).
No aplicable.
• **ACS11** El PACS/TR cubrirá en detalle todas las actividades de aseguramiento de calidad que se llevarán a cabo desde el comienzo de la fase TR hasta la aceptación final en la fase OM.
No aplicable.

APÉNDICE A: GLOSARIO

A.1 LISTA DE SIGLAS Y ABREVIATURAS

DA Diseño arquitectónico.
DDA Documento de Diseño Arquitectónico.
ANSI "American National Standards Institute".
PA Pruebas de Aceptación.
DD Diseño Detallado y producción.
DDD Diseño del Documento Detallado.
AEE Agencia Espacial Europea.
IEEE "Institute of Electrical and Electronics Engineers".
PI Pruebas de Integración.
DHP Documento Histórico del Proyecto.
PNE Procedimientos, Normas y Especificaciones[1].
PGCS Plan de Gestión y Configuración de Software.
PAPS Plan de Administración del Proyecto de Software.
PACS Plan de Aseguramiento de Calidad de Software.
RS Requisitos de Software.
PS Prueba del Sistema.
DTS Documento de Transferencia de Software.
DRS Documento de Requisitos de Software.
DES Documento de Especificación de Software.
MUS Manual de Usuario de Software.
PVVS Plan de Validación y Verificación de Software.
DRU Documento de Requisitos del Usuario.
UP Unidad de Prueba.
EAT Estructura de Interrupción del Trabajo.

APÉNDICE B: REFERENCIAS

1. ESA Software Engineering Standards , ESA PSS-05-0 Número 2 Febrero de 1991.
2. Guide to the ESA Software Engineering Standards, ESA PSS-05-01 Número 1 Octubre de 1991.
3. Guide to the User Requirements Definition Phase, ESA PSS-05-02 Número 1 Octubre de 1991.
4. Guide to the Software Requirements Definition Phase, ESA PSS-05-03 Número 1 Octubre de 1991.
5. Guide to the Software Architectural Design Phase, ESA PSS-05-04 Número 1 Enero de 1992.
6. Guide to the Software Detailed Design and Production Phase, ESA PSS-05-05 Número 1 Mayo de 1992.
7. Guide to the Software Transfer Phase, ESA PSS -05-06 Número 1 Octubre de 1994.
8. Guide to the Software Operations and Maintenance Phase, ESA PSS-05-07 Número 1 Diciembre de 1994.
9. Guide to Software Proyect Management, ESA PSS-05-08 Número 1 Junio de 1994.
10. Guide to Software Configuration Management, ESA PSS-05-09 Número 1 Noviembre de 1992.
11. Guide to Software Verification and Validation, ESA PSS-05- 10 Número 1 Febrero de 1994.
12. Guide to Software Quality Assurance, ESA PSS-05-11 Número 1 Julio de 1993.
13. Software Engineering Economics, B.Boehm, Prentice-Hall (1981)
14. Achieving Software Quality with Testing Coverage Measures, J. Horgan, S.London y M. Lyu, "Computer", IIEE, Septiembre de 1994.

APÉNDICE C: PLANTILLAS DE DOCUMENTOS

Todos los documentos deberían contener la siguiente información de servicio:
a) Resumen.
b) Índice.
c) Hoja de Estado de Documento.
d) Información de los cambios del documento realizados desde el último número.

Si no hay información pertinente a esta sección, la sección debería omitirse y numerar de nuevo las secciones.

Las pautas de los contenidos de las secciones de documentos están en itálica. Los títulos de las secciones, asignables por el autor/es del documento, se muestran entre corchetes.

C.1 ÍNDICE DEL DRU

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento.

2 Descripción General
- 2.1 Capacidades Generales.
  - *Describe el proceso soportado por el software.*
  - *Describe las principales funcionalidades requeridas y por qué son necesarias.*
- 2.2 Restricciones generales.
  - *Describe las principales restricciones que se aplican y por qué existen.*
- 2.3 Características del usuario.
  - *Describe quién podrá utilizar el software y cuando.*
- 2.4 Entorno operacional.
  - *Describe qué hacen los sistemas externos y sus interfaces con el producto. Incluye un diagrama de contexto o diagrama de bloque de sistema.*

3 Requisitos específicos

Listado de los requisitos específicos, con atributos.
- 3.1 Requisitos de Capacidad.
- 3.2 Requisitos de Restricción.

C.2 ÍNDICE DES

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento.

2 Descripción del modelo
- *Describe el modelo lógico utilizando un método de análisis reconocido.*

3 Requisitos específicos.
- *Lista de los requisitos específicos, con atributos.*
- *Las subsecciones pueden reagruparse en torno a las funciones de alto nivel.*
- 3.1 Requisitos funcionales.
- 3.2 Requisitos de desempeño.
- 3.3 Requisitos de interfaz.
- 3.4 Requisitos operacionales.
- 3.5 Requisitos de recursos.
- 3.6 Requisitos de verificación.
- 3.7 Requisitos de prueba de aceptación.
- 3.8 Requisitos de documentación.
- 3.9 Requisitos de seguridad.
- 3.10 Requisitos de portabilidad.
- 3.11 Requisitos de calidad.
- 3.12 Requisitos de entrega.
- 3.13 Requisitos de mantenimiento.
- 3.14 Requisitos de seguridad.

4 Diseño del sistema
- 4.1 Método de diseño.
  - *Describe o hace referencia al método de diseño utilizado.*
- 4.2 Descripción de la descomposición
  - *Describe el modelo físico, con diagramas.*
  - *Muestra los componentes y el control y el flujo de datos entre ellos.*

5 Descripción de los componentes
- *Describe cada componente.*
- *Estructure esta sección de acuerdo con el modelo físico.*
- 5.n [Identificador del componente]
- 5.n.1 Tipo
  - *Indica qué es el componente (por ejemplo: módulo, archivo, programa, etc.).*
- 5.n.2 Función
  - *Indica que hace el componente.*
- 5.n.3 Interfaces
  - *Define el control y flujo de datos desde y hacia el componente.*
- 5.n.4 Dependencias
  - *Describe las precondiciones para utilizar estos componentes.*
- 5.n.5 Procesamiento
  - *Describe el control y flujo de datos dentro del componente.*
  - *Resume el procesamiento de los componentes de bajo nivel.*
- 5.n.6 Datos
  - *Define en detalle los datos internos para los componentes, tales como los archivos utilizados para relacionarlos con los componentes principales. Por otra parte entrega una descripción resumida.*

5.n.7 Recursos

Lista de los recursos requeridos, tales como pantallas e impresoras.

6 Viabilidad y estimación de los recursos

Resumen de los recursos del computador requeridos para construir, operar y mantener el software.

7 Matriz de Trazabilidad de Requisitos de Usuario frente a Requisitos de Software

Muestra la correspondencia entre los requisitos de usuario y los requisitos de software que los resuelven.

8 Matriz de Trazabilidad de Requisitos de Software frente a Componentes

Muestra la correspondencia entre los requisitos software y los componentes que los soportan.

C.3 DOCUMENTACIÓN DE LOS REQUISITOS DEL CÓDIGO FUENTE

Cada módulo debería contener una cabecera que defina:
Nombre del módulo
Autor
Fecha de creación
Cambios históricos
    Versión/Fecha/Autor/Descripción
Función
    Indica que hacen los componentes.
Interfaces
    Define las entradas y salidas
Dependencias
    Describe las precondiciones para utilizar este componente.
Procesamiento
    Resume el procesamiento utilizando pseudo-código o un PDL

C.4 ÍNDICE MUS

1 Introducción
- 1.1 Lectores a quienes va dirigido.
  - *Describe quien debería leer el MUS.*
- 1.2 Aplicabilidad de los informes
  - *Establece que versiones de software se aplican al MUS.*
- 1.3 Propósito
  - *Describe el propósito del documento.*
  - *Describe el propósito del software.*
- 1.4 Cómo utilizar este documento
  - *Indica como se debe leer el documento.*
- 1.5 Documentos relacionados
  - *Describe el lugar del MUS en la documentación del proyecto.*
- 1.6 Convenciones
  - *Describe todas las convenciones estilísticas y comandos sintácticos utilizados.*
- 1.7 Instrucciones para informar problemas
  - *Resumen del SPR (Sistema para Informar Problemas).*

2 [Sección de aspectos generales]
- *Describe el proceso que va a soportar el software, y que es lo que hace el software para soportar el proceso, y que es lo que el usuario y/o el operador necesitan proporcionar al software.*

3 [Sección de instalación]
- *Describe los procedimientos necesarios para el software en la máquina de destino.*

4 [Sección de instrucción]
- *Desde el punto de vista del aprendiz, para cada tarea, entrega:*
- (a) Descripción funcional
  - *Qué realizará la tarea.*
- (b) Precauciones y advertencias
  - *Qué hacer y qué no hacer.*
- (c) Procedimientos
  - *Incluye:*
    - *Programa de instalación*
    - *Operaciones de entrada*
    - *Qué resultados esperar*
- (d) Errores probables y sus causas
  - *Qué hacer cuando las cosas van mal.*

5 [Sección de referencia]
- *Desde el punto de vista del experto, para cada operación básica, entrega:*

(a) Descripción funcional
- *Qué hace la operación.*

(b) Precauciones y advertencias
- *Qué hacer y qué no hacer.*

(c) Descripción formal
- *Parámetros requeridos*
- *Parámetros opcionales*
- *Opciones por defecto*
- *Parámetro orden y sintaxis*

(d) Ejemplos
- *Entrega ejemplos probados de la operación.*

(e) Posibles errores de mensajes y sus causas
Lista de los posibles errores y sus causas probables.
(f) Referencias cruzadas con otras operaciones
Se refiere a cualquier operación complementaria, anterior o posterior.

Apéndice A Mensajes de error y procedimientos de recuperación
Lista de todos los mensajes de error.
Apéndice B Glosario
Lista de todos los términos con significados especializados.
Apéndice C Índice (para manuales de 40 páginas o más)
Lista de todos los términos importantes y sus ubicaciones.

C.5 ÍNDICE DTS

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento.

2 Informe de Construcción
- *Describe qué sucedió cuando el software se elaboró a partir de un código fuente*

3 Informe de Instalación
- *Describe qué sucedió cuando el software se instaló en la máquina de destino.*

4 Listado de Elementos de Configuración
- *Listado de todos los elementos de configuración disponibles.*

5 Resumen de los Informes de Pruebas de Aceptación
- *Para cada prueba de aceptación, entrega el:*
  - *Identificador de requisito de usuario y resumen identificador de informe de prueba en el PVVS/PA/Informes de Pruebas resumen de los resultados de las pruebas.*

6 Informe de los Problemas de Software
- *Entrega una lista de los IPS que aparecieron durante la fase TR y su estado en la versión del DTS.*

7 Solicitudes de Cambios de Software
- *Lista de las SCS que aparecieron durante la fase TR y su estado en la versión del DTS.*

8 Informe de las Modificaciones de Software
- *Lista del IMS terminado durante la fase TR.*

C.5 ÍNDICE PAPS

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento

2 Organización del Proyecto
- 2.1 Roles organizativos y responsabilidades
  - *Describe los roles del proyecto, responsabilidades y líneas de reporte.*
- 2.2 Límites organizacionales e interfaces
  - *Describe las interfaces con los clientes y proveedores.*

3 Proceso técnico
- 3.1 Entradas del proyecto
  - *Lista qué documentos serán entradas para el proyecto, cuando y por quién.*
- 3.2 Salidas del proyecto
  - *Lista de qué será entregado, cuándo y dónde.*
- 3.3 Modelo del proceso
  - *Define el informe del ciclo de vida para el proyecto.*
- 3.4 Métodos y herramientas
  - *Describe o hace referencia a los métodos de desarrollo utilizados.*
  - *Define y hace referencia a las herramientas de producción y diseño.*
  - *Describe el formato, estilo y herramientas para la documentación.*
  - *Define y hace referencia a los estándares de codificación.*
- 3.5 Funciones de apoyo al proyecto
  - *Resumen de los procedimientos de ACS y VVS e identifica los procedimientos relacionados y planes.*

4 Paquetes de trabajo, agenda y presupuesto
- 4.1 Paquetes de trabajo
  - *Para cada paquete de trabajo, describe las entradas, tareas, salidas, requisitos de esfuerzo, distribución de recursos y procesos de verificación.*
- 4.2 Agenda
  - *Describe cuando empezará y terminará cada paquete de trabajo (Diagrama Gantt)*
- 4.3 Presupuesto
  - *Describe el costo total del proyecto.*

C.7 ÍNDICE PGCS

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento

2 Actividades GCS
- 2.1 Identificación de la Configuración
  - *Describe las convenciones de identificación de los Elementos de Configuración*
  - *Para cada línea base:*
  - *Describe cuando se creará;*
  - *Describe que contendrá;*
  - *Describe cuanto se reconocerá;*
  - *Describe el nivel de autoridad necesario para cambiarlo.*
- 2.2 Almacenamiento del Elemento de Configuración
  - *Describe los procedimientos para almacenar EIs en las librerías de software.*
  - *Describe los procedimientos para almacenar información contenida en los EIs.*
- 2.3 Control de Cambios de los Elementos de Configuración
  - *Describe los procedimientos de control de cambios.*
- 2.4 Evaluación del Estado de la Configuración
  - *Describe los procedimientos para mantener una revisión de los cambios.*
- 2.5 Procedimientos de construcción
  - *Describe los procedimientos para construir el software desde la fuente.*
- 2.6 Procedimientos de entrega
  - *Describe los procedimientos para la entrega del código y documentación.*

C.8 ÍNDICE PVVS

Sección de Revisiones y Seguimiento

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento

2 Revisiones
- *Describe la inspección, procedimientos de revisión técnica y repaso.*

3 Trazado
- *Describe como rastrear la fase de entradas a salidas.*

Sección de Pruebas Unitarias, de Integración, de Sistema y de Aceptación

1 Introducción
- 1.1 Propósito del documento.
- 1.2 Definiciones, acrónimos y abreviaturas.
- 1.3 Referencias.
- 1.4 Visión general del documento

2 Plan de prueba
- 2.1 Elementos de prueba
  - *Lista de los elementos que se probarán.*
- 2.2 Características que se probarán
  - *Identifica las características que se probarán.*
- 2.3 Entregables
  - *Lista de los elementos que deben entregarse antes de que empiecen las pruebas.*
  - *Lista de los elementos que deben entregarse cuando terminen las pruebas.*
- 2.4 Tareas de prueba
  - *Describe las tareas necesarias para preparar y llevar a cabo las pruebas.*
- 2.5 Necesidades de entorno
  - *Describe las propiedades requeridas en el entorno de pruebas.*
- 2.6 Criterio aceptación/rechazo de un caso de prueba
  - *Define el criterio para aceptar o rechazar un caso de prueba*

3 Especificaciones de los Casos de Prueba
(para cada caso de prueba.....)
- 3.n.1 Identificador de caso de prueba
  - *Asigna un identificador único para cada caso de prueba.*
- 3.n.2 Elementos de prueba
  - *Lista de los elementos que se probarán.*
- 3.n.3 Especificaciones de entrada
  - *Describe la entrada para el caso de prueba.*
- 3.n.4 Especificaciones de salida
  - *Describe la salida requerida del caso de prueba.*
- 3.n.5 Necesidades de entorno
  - *Describe el entorno de prueba*.

4 Procedimientos de prueba
(para cada procedimiento de prueba.....)
- 4.n.1 Identificador del procedimiento de prueba

Asigna un identificador único para el procedimiento de prueba

4.n.2 Propósito

Describe el propósito del procedimiento.

Lista de los casos de prueba que ejecuta este procedimiento.

4.n.3 Pasos del procedimiento

Describe como registrar, ajustar los parámetros, empezar, proceder, medir, cerrar, reiniciar, parar, adaptar la prueba, y como manejar contingencias.

5 Plantilla de Informe de Prueba

5.n.1 Identificador del informe de prueba

Asigna un identificador único para el informe de la prueba.

5.n.2 Descripción

Lista de los elementos que se probarán.

5.n.3 Actividad y eventos de entrada.

Identifica el procedimiento de prueba.

Indica cuándo se hizo la prueba, quién la hizo, y quién la presencio.

Indica si el software aprobó o falló para cada caso de prueba cubierto por el procedimiento.

Describe los problemas.