Revista Española de Innovación, Calidad e Ingeniería del Software

Volumen 1, No. 2, diciembre, 2005

Web de la editorial: www.ati.es
E-mail: reicis@ati.es
ISSN: 1885-4486

Publicado por la Asociación de Técnicos en Informática

Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS)

Contenidos

REICIS

¿Cuál es la madurez que necesitarían los procesos para el desarrollo de sistemas de software crítico?

Patricia Rodríguez Dapena, Josefina Alonso Nocelo, José Carlos Sánchez Domínguez

SoftWcare, S.L

C/. Serafín Avendaño 18 interior, oficina 13

36201 Vigo (Pontevedra)

rodriguezdapena@softwcare.com, alonsonocelo@softwcare.com, sanchezdominguez@softwcare.com

Abstract

Critical systems which function conditions our life need to be verified and certified before its use
The criticism of these systems is founded in the software products related with. The current certification process are, sometimes, insufficient to evaluate all the security and reliability aspects related.
Some international certification organism deals with the development process certification instead of the product or the system. This paper presents how and what must be evaluated respect the development process related with the certification of critics software systems.

Resumen

Los sistemas críticos cuyo funcionamiento condiciona nuestra vida cotidiana necesitan ser verificados, certificados u homologados, antes de su puesta en funcionamiento.
La criticidad de estos sistemas reside cada vez más en los productos software que contienen. Los actuales procesos de certificación resultan en ocasiones insuficientes para evaluar todos los aspectos de seguridad y fiabilidad.
Algunos organismos internacionales de certificación pretenden certificar el proceso de desarrollo en vez del producto o el sistema en sí. Este trabajo presenta cómo y qué se debería evaluar respecto a los procesos de desarrollo para certificar sistemas de software críticos.

Palabras clave: Sistemas, software crítico, evaluación de procesos, seguridad, fiabilidad, proceso, certificación, homologación.

1 Introducción

Las entidades de certificación y homologación (FAA [1], etc.) son conscientes de la necesidad de un nuevo enfoque en la certificación para agilizar la evaluación de la seguridad y fiabilidad de sus

sistemas, para que la incorporación de nuevas tecnologías no retrase la puesta en marcha del sistema (pues resulta compleja y difícil la certificación) y sea una ventaja y no un inconveniente.

Para agilizar este proceso se plantea la certificación del proceso en vez del producto. Podría certificarse que las organizaciones que desarrollan estos sistemas dispongan de procesos internos de diseño, pruebas y aseguramiento de la calidad cuyos resultados sean productos con el nivel de seguridad (*safety*) requerido. Este nuevo enfoque permitiría a las entidades certificadoras centrarse más efectivamente en los aspectos críticos de un sistema.

Para lograr esto, deberían relacionarse los modelos existentes de evaluación y/o certificación de la madurez de los procesos software (tales como SPICE o CMM/CMMI para software) con los requisitos de seguridad (‘*safety*’) del sistema y definir las exigencias según los niveles de criticidad del software —S*afety Integrity Levels* (SILs)—, definiendo así los ‘perfiles de madurez’ de los procesos respecto a los diferentes niveles de criticidad de productos software.

2 Evaluación de procesos

El objetivo buscado en la certificación u homologación de sistemas críticos es el aseguramiento de un mínimo riesgo de fallo del sistema (o al menos, un nivel de riesgo aceptable), una vez puesto en funcionamiento.

A principios de los 80, los militares de EE.UU. y del Reino Unido se propusieron mejorar el mecanismo de selección de proveedores de software con el objetivo de detener el creciente costo de software, reducir riesgos en su desarrollo y mejorar la calidad de los productos de software.

En EE.UU., se creó el *Software Engineering Institute* (SEI), con el objetivo de desarrollar el mecanismo de selección de proveedores. El modelo CMM [7] [8] (cuya primera versión se obtuvo en el año 1991) y el trabajo e impacto de este instituto son bien conocidos (www.cmu.sei.edu)

Por su parte, más adelante, teniendo origen en el Reino Unido se reconoció la necesidad de abordar con mayor rigor el problema de selección de proveedores para los sistemas que dependen en gran medida de software. Se revisaron muchos modelos y métodos existentes en ese momento (Bootstrap, Trillium, etc.), llegándose a un consenso internacional sobre la necesidad y los requisitos para un modelo y método de referencia de evaluación de procesos, constituyendo éste el origen de lo que sería posteriormente la norma ISO 15504.

La Norma SPICE (ISO/IEC 15504)

Hoy día ya existe un estándar ISO, el ISO/IEC TR 15504:1998 [2], que detalla un modelo y un método de referencia para la evaluación de procesos software. Este estándar se encuentra en fase de revisión para su segunda versión [3], y será un modelo de referencia para la evaluación de procesos en general

(no sólo de software) La norma ISO/IEC 15504 [2] [3] (también denominada SPICE —*Software Process Improvement and Capability dEtermination*—, por el proyecto que dio origen a la norma) se caracteriza por:

- Ser aplicable a cualquier organización o empresa.
- Ser independiente de la organización, el modelo del ciclo de vida, la metodología y la tecnología.
- Ser un marco para métodos de evaluación, no un método o un modelo en sí.
- Cubrir diferentes objetivos para la evaluación de procesos:
  - Determinación de la capacidad (niveles de capacidad o de madurez);
  - Mejora de procesos
  - Evaluar el cumplimiento de determinados requisitos del ciclo de vida de desarrollo de software.

La parte 5 de la norma proporciona la guía para un modelo de evaluación de procesos software, de acuerdo con la norma ISO/IEC 12207 [4] [5] para procesos del ciclo de vida del software. La arquitectura SPICE tiene dos dimensiones: procesos y niveles de madurez o capacidad (figura 1).

Dimensión de los Procesos

Contiene los procesos que se han de evaluar y que, en la parte 5 del estándar, se corresponden con los procesos del ciclo de vida del software, definidos en el estándar ISO/IEC 12207 [4] [5].

Los procesos se agrupan en categorías, en función del tipo de actividad al cual se aplican: Clientes y proveedores (CUS), Ingeniería (ENG), Soporte (SUP), Gestión (MAN) y Organización (ORG).

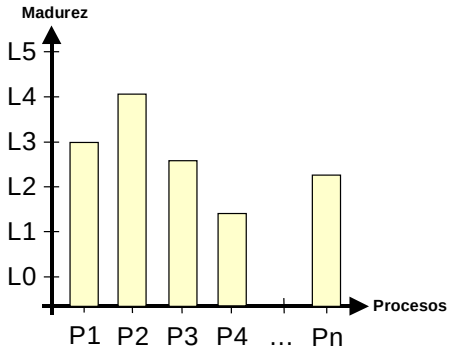

Fig. 1 Dimensiones de SPICE

Dimensión de Niveles de Madurez o Capacidad

Define una escala de medida para determinar la madurez de un proceso. Cuenta con seis niveles de capacidad o madurez —numerados de 0 a 5— asociados a nueve atributos de procesos (ver tabla 1).

Nivel de capacidad		**Atributos de los procesos**
0	Incompleto	
1	Realizado	Grado de realización
2	Gestionado	Gestión de la realización Gestión de productos de trabajo
3	Establecido	Grado de definición Grado de institucionalización
4	Predecible	Medida del proceso Control del proceso
5	Optimizado	Grado de gestión del cambio Grado de optimización

Tabla 1 Niveles de capacidad y atributos de los procesos de ISO 15504 [2] [3]

Para evaluar los procesos se utilizan estos atributos para cada proceso que, de cumplirse, le proporcionan mayor o menor nivel de madurez al proceso. Y para evaluar estos atributos se utilizan indicadores que detallan mucho más los atributos facilitando la evaluación de cada proceso. Una vez evaluados los procesos se obtiene lo que se conoce como perfiles de madurez.

Existen otros proyectos a escala internacional para la evaluación de procesos (como CMM y CMMI, desarrollados por el *Software Engineering Institute*) que, al igual que SPICE, están más orientados a la selección o evaluación de proveedores que a la certificación de productos críticos.

A continuación, se presentan los diferentes requisitos de seguridad (‘safety’) y fiabilidad en diferentes dominios, para analizar qué se certifica respecto a los diferentes niveles de criticidad, para luego compararlos con los modelos de evaluación de procesos existentes y definir sus diferencias.

3 Niveles de criticidad de productos software y estándares de desarrollo

Los niveles de criticidad de un sistema software se asignan según la severidad y frecuencia del mal funcionamiento del software durante su operación. Son los denominados riesgos del producto o sistema, de forma que cuanto más severos y frecuentes sean los efectos de sus fallos más altos es el riesgo y más alto será su nivel de criticidad del mismo.

La clasificación del software en diferentes niveles de criticidad puede proporcionar una base para la definición de exigencias más o menos estrictas respecto a los procesos y actividades de desarrollo de software: a mayor criticidad, mayor exigencia.

Los objetivos de esta clasificación son los siguientes:

- Asegurar que los procesos y actividades que se utilizan son lo suficientemente completos para que se adecuen a las necesidades de desarrollo de software más o menos crítico.
- Eliminar costes innecesarios, de modo que el software de baja criticidad no sea desarrollado con excesivos, costosos y sofisticados procesos.
- Que los procesos de software usados en distintos proyectos sean lo más uniformes posibles, para cada nivel de criticidad.

Los niveles de criticidad son diferentes según cada dominio de aplicación y las características de seguridad que se apliquen respectivamente.

A continuación se hace un pequeño análisis de los estándares más representativos de los distintos dominios de aplicación (sistemas electrónicos, aviónica, etc.) con el objetivo de realizar un estudio de las distintas exigencias que estos recomiendan en función de los diferentes niveles de criticidad específicos de cada uno de ellos.

Sistemas Electrónicos

Como cualquier estándar general de seguridad (*safety*), el estándar IEC 61508 [10] define una serie de niveles de criticidad a partir de la probabilidad media de fallos y recomienda el uso de métodos y técnicas más o menos estrictos según el nivel.

En la tabla siguiente se presenta un ejemplo de técnicas y recomendaciones asociadas a los diferentes niveles de criticidad definidos en este estándar.

Método o técnica	**Categoría**	**SIL1**	**SIL2**	**SIL3**	**SIL4**
Uso de estándares de código	Estándares de código	HR	HR	HR	HR
Clases de equivalencia y pruebas de partición	Pruebas de caja negra	R	HR	HR	HR
Pruebas de estrés	Pruebas de funcionamiento	R	R	HR	HR
Análisis de flujo de control	Inspecciones	---	R	R	HR
Leyenda: 'HR' – altamente recomendado; 'R' – recomendado; '---' no es necesario					

Tabla 2 Técnicas y recomendaciones paracada nivel de criticidad para sistemas electrónicos[10]

Locomoción

El estándar EN 50128 [9] introduce también diversos aspectos de criticidad para el desarrollo de software en el dominio ferroviario. Está basado en el IEC 61508 anteriormente mencionado.

Aviónica

La actividad de desarrollo del software en el dominio de la aviónica según el estándar DO-178B [6] empieza con una evaluación de la seguridad (*safety*) del sistema. Durante esta evaluación, el efecto de un fallo sobre el funcionamiento total del avión es estudiado y analizado, clasificando al software en cinco niveles de criticidad (A, B, C, D, E) según el tipo de efecto del fallo (catastrófico, arriesgado, mayor envergadura, menor envergadura, sin efecto). Además, para cada una de estas clases de software se indican las actividades que se tienen que satisfacer para conseguir la certificación según el estándar DO-178B [6] (ver ejemplo en tabla 3)

Actividad	**Clases de software**			
	A	**B**		
El código fuente cumple los requisitos de bajo nivel.	●	●	❍	
El código fuente implementa la arquitectura del software.				
El código fuente es verificable.	❍	❍		
El código fuente es acorde a los estándares.	❍	❍	❍	
La cobertura de las pruebas de todas las estructuras internas del software es completa	●	●		
LEYENDA	● → La actividad debería realizarse por personas u organizaciones independientes. ❍ → La actividad debería satisfacerse. (en blanco) → El cumplimiento de la actividad queda a elección del usuario.			

Tabla 3 Ejemplo de actividades del proceso de codificación y pruebas del software en el dominio de la aviónica [10]

Automóvil

En la industria del automóvil, el estándar MISRA [14] define los niveles de control (*controllability categories*) como la capacidad de los ocupantes, y no sólo del conductor, de preservar la seguridad (*safety*) después de un fallo en el vehículo.

MISRA clasifica el software en 4 niveles de criticidad (1-4) y, a su vez, para cada una de estas clases de software define las actividades que se tienen que satisfacer [6] (ver tabla 4).

Espacio

Cualquier desarrollo de sistemas espaciales en Europa sigue los estándares ECSS (*European Cooperation for Space Standardization*). El estándar ECSS-Q-80B [11] contiene requisitos para la calidad de software y está apoyado por los estándares ECSS-Q-40B [12] y ECSS-Q-30B [13] en cuestiones de seguridad (*safety*) y fiabilidad respectivamente. Este estándar contiene requisitos específicos para software crítico. Es importante destacar que no se refieren a clases de software en particular, aunque sí especifican requisitos para software crítico, añadidos a los demás requisitos para cualquier clase de software.

El estándar ECSS-Q-80B [11] propone una serie de actividades que permitan asegurar la fiabilidad del software crítico, como por ejemplo el uso de técnicas de programación defensivas, la inspección completa del código fuente o la prohibición del uso de características que puedan proporcionar resultados impredecibles.

Como se ha visto en los apartados anteriores cada estándar en cada dominio define exigencias con diferentes grados de rigor según los niveles de criticidad del software. La pregunta ahora radica en cómo relacionar estas exigencias de diferente índole con los perfiles de madurez de los procesos.

4 Perfiles de madurez y criticidad del producto software

El alcance y los objetivos de los modelos de evaluación de procesos y de los diferentes estándares de desarrollo de software crítico son muy diferentes. Por tanto es necesario ahondar más en la adaptación de los métodos de evaluación de procesos software, para dar lugar a los perfiles de madurez para la evaluación /certificación de software crítico.

Actividad	**Clases de software**			
	1	**2**	**3**	**4**
Lenguajes de codificación y compiladores	Utilización de un lenguaje estructurado	Utilización de un subconjunto restringido de lenguaje estructurado. Utilización de compiladores validados	Como para el nivel 2	Uso de compiladores certificados independientemente con reglas formales de sintaxis y semántica
Pruebas	Mostrar que se cumplen los requisitos. Plan de pruebas repetible.	Pruebas de caja negra	Pruebas de caja blanca a todos los módulos de código – midiendo la cobertura. Pruebas de estrés. Análisis estático de sintaxis.	100% pruebas de caja blanca a los módulos. 100% pruebas de los requisitos. 100% pruebas de integración.

Tabla 4 Ejemplo de actividades del proceso de codificación y pruebas en el ámbito del automóvil [16]

Existen opiniones contrapuestas respecto al valor añadido de relacionar los perfiles de madurez respecto a los niveles de criticidad. Según el Departamento de Defensa de los EE.UU. el alcanzar un determinado nivel de madurez de procesos no garantiza un nivel adecuado de calidad en el desarrollo de los sistemas críticos. Sin embargo, los dominios del automóvil y del espacio tienen planes para establecer una relación entre los niveles de criticidad y los niveles de madurez.

En los últimos años, la implantación de sistemas de calidad basadas en la norma ISO9001 en diferentes empresas y organizaciones ha crecido, siendo ahora orientada a procesos. Sin embargo, para aquellas organizaciones que desarrollan sistemas de software crítico, esta norma presenta una serie de problemas pudiendo resultar insuficientes ya que:

a) Disponer de un sistema de calidad del software acorde con la ISO 9001 no garantiza más allá de poder desarrollar software con el nivel más bajo de criticidad (SIL1)
b) Obtener una certificación ISO 9001 en una organización correspondería a una madurez de procesos al nivel 3 para todos los procesos del ciclo de vida (tal y como se menciona en diferentes artículos y literatura existente)
c) La norma ISO 9001 no es específica para software, aunque se dispondrá de la ISO 90003; sin embargo, ésta tampoco es específica para software crítico, resultando todavía insuficiente.

Tanto ISO/IEC TR 15504 [2] como otros modelos de evaluación de procesos software, inciden en la determinación de la capacidad y la mejora de los procesos software en detalle, pero no aseguran las condiciones suficientes para garantizar el nivel de fiabilidad y seguridad (*safety*) necesarios del producto. Se haría pues necesario, modificar estos modelos para poder garantizar los aspectos relacionados con la criticidad del software. Estas adaptaciones son necesarias en relación con:

a) actividades adicionales para el desarrollo y la verificación de software crítico,
b) más detalle de los documentos y productos resultantes de cada actividad y,
c) exigencias más específicas según el nivel de criticidad respecto al personal que desarrolle el sistema,
d) exigencias específicas respecto a los métodos, técnicas y herramientas que se utilicen,
e) exigencias para la organización que asegure la fiabilidad y seguridad (‘*safety’*) del sistema.

Estas modificaciones afectan a casi todos los procesos y resultan ser exigencias a diferentes niveles de madurez y no sólo referentes a la definición de las actividades del proceso en sí.

5 Definición de los perfiles de madurez

La parte 8 de ISO/IEC TR 15504 [2] describe un método general para definir perfiles de madurez. Este método consiste en vincular los procesos con los niveles de criticidad mediante un análisis de la relación existente entre el riesgo de los procesos y el éxito de los proyectos. En la tabla 6, mostrada a continuación, se muestra un ejemplo de estos perfiles de madurez para los procesos de clientes y proveedores según el nivel de criticidad. Como puede comprobarse, para la clase A (la de mayor criticidad) se exige el nivel de madurez 4 que resulta, en cualquier caso, muy exigente. Su cumplimiento puede resultar por tanto inviable.

Proceso		**Criticidad**			
		A	**B**	**C**	**D**
CUS.2.2	Entrega	4	3	3	1
CUS.3	Análisis de Requisitos	4	3	3	1
CUS.4	Operación	4	4	3	2
CUS.4.1	Uso operacional	4	4	3	2
CUS.4.2	Suporte al cliente	4	4	3	2

Tabla 6 Ejemplo de definición de perfiles de madurez para diferentes niveles de criticidad [15]

El borrador del estándar ECSS-Q-80-02 [15] contendrá un ejemplo de los perfiles de madurez de referencia, para los diferentes proyectos de desarrollo de software de sistemas espaciales en Europa.

6 Conclusiones

Existen iniciativas para redirigir la certificación del software como parte de sistemas críticos desde la evaluación del producto a la de sus procesos de desarrollo para así agilizar estos tediosos procesos de certificación especialmente cuando los productos software de estos sistemas son nuevos y cada vez más complejos.

A la vista de lo expuesto anteriormente, para el desarrollo de sistemas de software crítico:

a) ni la familia de normas ISO 9000 ni los modelos de evaluación de procesos (Ej. ISO/IEC 15504 [2] [3]) resultan suficientes para asegurar la madurez necesaria de los procesos en las organizaciones;

b) las adaptaciones necesarias a los modelos de evaluación de procesos existentes no son despreciables, incluyendo la dificultad de tener que armonizar las exigencias en los diferentes dominios de aplicación;

c) es previsible la exigencia de un nivel de madurez elevado en diferentes procesos, pudiendo resultar inasumible para las organizaciones;
d) tener el nivel necesario de madurez de los procesos no asegura el nivel final requerido de seguridad y fiabilidad en los productos.

La literatura actual muestra un amplio abanico de posibilidades en cuanto a la relación que debe establecerse entre los niveles de madurez de los procesos y la criticidad del software.

A día de hoy, se sigue investigando en la relación existente entre los niveles de madurez del software y la criticidad del software en distintos dominios. Y aunque resulta imprescindible seguir estudiando esta relación con el fin de poder facilitar la certificación de estos sistemas, hay que mantener ciertas reservas respecto a que la certificación de estos sistemas sea posible sólo a través de la evaluación de sus procesos de ciclo de vida.

7 Bibliografía

[1] Commission on the Future of the U.S. Aerospace Industry. FAA. November 2002. FAA. www.faa.gov

[2] ISO/IEC TR 15504:1998 *Information Technology – Software Process Assessment*.

[3] ISO/IEC 15504:2003. *Information Technology – Process Assessment.*

[4] ISO/IEC 12207:1995/Amd 1:2002. *Information technology -- Software life cycle processes.* December 2002.

[5] ISO/IEC 12207:1995/Amd 1:2002 Corrigenda. *Information technology -- Software life cycle processes.*

[6] RTCA/DO-178B. *Software Considerations in Airborne Systems and Equipment Certification.* December 1992.

[7] *CMMI[SM] for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1) continuous representation (CMU/SEI-2002-TR-001) and staged representation (CMU/SEI-2002-TR-002)* Software Engineering Institute. December 2001.

[8] *CMMI[SM] for Software Engineering (CMMI-SW, V1.1) continuous representation (CMU/SEI-2002-TR-029) and staged representation (CMU/SEI-2002-TR-028).* Software Engineering Institute. August 2002.

[9] EN 50128:2001. *Railway applications. Communications, signalling and processing systems. Software for railway control and protection systems.* (debe leerse conjuntamente con EN50126 y EN

50129).

[10] *IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related system. Parts 1-7*. International Electro-technical Commission. IEC 61508. 1999.

[11] *ECSS-Q-80B Space product assurance – Software product assurance.* 10 October 2003.

[12] *ECSS-Q-40B Space product assurance – Safety.* 17 May 2002.

[13] *ECSS-Q-30B Space product assurance – Dependability.* 8 March 2002.

[14] *Draft ECSS-Q-80-03 Space product assurance – Guidelines for methods and techniques to support the verification and validation of software dependability and safety*. Draft D1.5, 25 July 2003

[15] *Draft ECSS-Q-80-02 Space product assurance – Software process assessment and improvement*. Draft 1.5.1. ECSS-Q-80-02 working group http://www.ecss.nl

[16] Jesty, K. Mobley, R. Evans, I. Kendall, "Safety Analysis of Vehicle-Based Systems". PMISRA. http://www.misra.org.uk/papers/SCSC00-SA.PDF