ieee.es
Instituto Español de Estudios Estratégicos

Documento
Opinión

25/2014 11 marzo de 2014

*José Tomás Hidalgo Tarrero**

DIPLOMADOS EN INFORMÁTICA MILITAR, RECURSO ESTRATÉGICO

Visitar la WEB Recibir BOLETÍN ELECTRÓNICO

DIPLOMADOS EN INFORMÁTICA MILITAR, RECURSO ESTRATÉGICO

Resumen:

Existen diversos tipos de pruebas para comprobar la calidad y seguridad del software, cada una de las cuales tiene sus propios requisitos y su ámbito de aplicación. De entre ellas las pruebas de caja blanca y la revisión del código son las que van a dar al cliente una visión mejor de la calidad y seguridad del software, pero requieren por parte del cliente personal con la formación adecuada para hacerlas. Además requieren que ese personal esté familiarizado con el diseño y estructura del código fuente.
El software COTS (Commercial off the shelf) suele ser el más barato pero es muy difícil controlar por parte del cliente la calidad y seguridad del mismo, lo cual puede ser poco conveniente.
El desarrollo propio es el que más control puede proporcionar sobre el software, pero requiere personal propio formado.
El software hecho a medida puede proporcionar ciertos niveles de control, sobre todo en el caso de desarrollo conjunto entre la empresa y el cliente. También requiere personal propio con la formación adecuada.
Dentro de las Fuerzas Armadas hay oficiales y suboficiales con los cursos de Diplomado en Informática Militar y Programador de Informática, las denominaciones pueden variar en cada uno de los Ejércitos y Armada, o con titulación universitaria en Ingeniería Informática, tanto superior como técnica, o Formación Profesional en informática, tanto media como superior. Este personal tiene una importancia estratégica porque son los que podrían ejercer el control de calidad y seguridad del software tanto de desarrollos propios como hecho a medida.
El proceso lógico para demostrar que los diplomados en informática militar son un recurso estratégico tiene cinco pasos. Primero se establecerá la necesidad de hacer controles de calidad y seguridad al software; a continuación, se describen los tipos de controles de calidad y seguridad más comunes que se hacen al software; en el siguiente paso se explica cuándo se pueden hacer esos controles y qué se necesita. En el cuarto paso se examinan los tres sistemas básicos de obtención del software desde el punto de vista de los controles de calidad y seguridad para en el quinto paso analizarlos desde el punto de vista del mantenimiento. Finalmente se concluye que los diplomados en informática militar son un recurso estratégico.

Abstract:

There are several types of tests to check the quality and security of software, each of which has its own requirements and scope. Among them the white-box testing and code review are those that will give the customer a better view of the quality and safety of software, but they require the client side to have properly trained staff to do them. They also require that that staff is familiar with the design and structure of the source code.
The COTS (Commercial off the shelf) software is usually the cheapest but it is very difficult to control, by the customer, the quality and safety of it, which can be inconvenient.
Own development can provide the maximum control of the software, but requires properly trained personnel.
Tailored software can provide certain levels of control, especially in the case of joint development between the company and the customer. It also requires own staff with appropriate training.
Within the Armed Forces there are officers and NCOs with Military Diploma in Software Engineering and Programmer Course, denominations may vary in each of the Armies and Navy, or university degree in Computer Engineering, Grade or Master, or apprenticeship in software engineering. These staffs are of strategic importance because they are the ones that could exercise control of software quality and safety of both internal and tailored developments.
The logical process to demonstrate that graduates in military computer science are a strategic military resource has five steps. First it is the established the need of software quality and security controls, then the types of quality control and common security software are described, in the next step it is explained when these checks can be made and what is needed to carry them out. In the fourth step the three basic systems for obtaining the software are examined from the viewpoint of quality and safety control then in the fifth step it is analysed from the point of view of maintenance. Finally we conclude that graduates are a strategic resource.

Palabras clave:

Software, control de calidad, Pressman, Ciberdefensa.

Keywords:

Software, quality control, Pressman, Cyber Defence.

CONTROL DE LA CALIDAD Y SEGURIDAD DEL SOFTWARE

En los años 60 se produjo la primera gran crisis del software motivada, entre otras causas, por la falta de calidad del mismo y por el imparable aumento de su complejidad y de la del hardware. Para paliar los efectos de esa primera crisis, en la década de los 70 nació una nueva especialidad de la ingeniería que es la Ingeniería del software cuyo objetivo es aplicar técnicas de ingeniería en el desarrollo del software

En la ingeniería del software se establecen una serie de factores de calidad del software, aunque no todos los investigadores están de acuerdo en todos ellos. Todos los factores de calidad citados por Pressman[1], Meyer[2] o Cerrada[3], entre otros, implican la posibilidad de efectuar pruebas para constatar en qué medida se encuentran dichos factores en el software. Es interesante destacar que de entre todos ellos corrección, grado de cumplimiento de las especificaciones, y fiabilidad, o ausencia de fallos, son los que más frecuentemente se suelen medir.

Es evidente que el software debe hacer lo que está en las especificaciones, todo lo que está en las especificaciones y nada más que lo que está en las especificaciones. Para comprobar el grado de corrección y fiabilidad se diseñan pruebas; estas pruebas comprueban solo la funcionalidad, que hace todo lo que está en las especificaciones, y miden la tasa de fallos.

Hasta ahora hemos visto que los productos software deben pasar unos controles de calidad similares en concepto a los de cualquier otro producto manufacturado. Pero no hay que olvidar que existe un nuevo espacio de confrontación, el ciberespacio, y por tanto, el software tiene que ser seguro no sólo en su operación normal, como cualquier otro producto manufacturado, sino que también debe ser operar correctamente en situaciones no previstas o anómalas entre las que se encuentran los ataques informáticos. También hay que comprobar que no hace cosas no previstas en las especificaciones. Si tenemos en cuenta lo que dice Howard[4], el software seguro es software de calidad, el título de este epígrafe contiene una redundancia pero se ha preferido mantenerla por claridad. Esto se apoya en lo expuesto por Allen[5], la seguridad debe ser una característica más del software que

[1] **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico,* segunda edición. McGraw-Hill, 1992, 481-482.
[2] **Meyer, Bertrand.** *Object-oriented Software Construction.* Prentice Hall International, 1988, 4-7.
[3] **Cerrada Somolinos, José A., Collado Machuca, Manuel E., Gómez Palomo, Sebatián R. y Estívariz López, José F.** *Introducción a la ingeniería del software.* Madrid : Editorial universitaria Ramón Areces, 2007, 27-28.
[4] **Howard, Michael et alt.** *Writing Secure Code,* Segunda edición. Microsoft Press, 2003, 4.
[5] **Allen, Julia H.; Barnum, Sean; Ellison, Robert J.; McGraw, Gary; y Mead, Nancy R.** *Software Security Engineering: A Guide for Project Managers,* Addison-Wesley Professional, 1st edition, 2008, 73-78.

debe estar en el mismo desde las primerísimas etapas de su ciclo de vida; por tanto, debe estar en las especificaciones y su comprobación debe ser parte consustancial de las pruebas de calidad.

Como se verá en los siguientes epígrafes, la parte funcional suele ser más fácil de probar que la parte de seguridad; si las especificaciones están bien hechas y son claras, basta con probar que acepta y rechaza las entradas que debe aceptar y rechazar respectivamente y que el resultado de las entradas aceptadas es el previsto. Dicho muy fácil pero que requiere tiempo de planeamiento, preparación y ejecución.

CONTROL DE LA CALIDAD Y SEGURIDAD DEL SOFTWARE DESPUÉS DEL DESARROLLO

Este tipo de control es el que suele realizar el cliente. En la mayoría de los casos el cliente no ha participado en el desarrollo, excepto en la redacción de los requisitos o especificaciones, y lo que se puede hacer es probar la corrección y fiabilidad con pruebas llamadas de caja negra porque en ellas solo se comprueba que el software acepta y rechaza las entradas que debe aceptar y rechazar y que los resultados de las entradas aceptadas son los previstos; es decir, según Pressman[6] y Cerrada[7] solo se prueba la funcionalidad del software. Qué hace internamente el software es indiferente en este tipo de pruebas ya que se trata como una caja negra, de ahí el nombre de este tipo de prueba, y opaca que no deja ver lo que hay en su interior. Por ejemplo, un software que hace sumas de números enteros se prueba que solo acepta números enteros y que rechaza cualquier otra cosa y que el resultado de entrar 2 y 3 es 5 y el de entrar 2 y -3 es -1. Estas pruebas de caja negra son las únicas posibles cuando el probador no ha participado en el desarrollo del software[8], sobre todo en el caso de grandes desarrollos.

Es evidente que estas pruebas deben realizarse no solo en la entrega del producto final sino también en todas las unidades que compongan el mismo; pero también es cierto que no puede asegurarse que el software haga cosas que no estén en las especificaciones. Para tener un control más exhaustivo de la calidad y seguridad del software estos controles deben hacerse durante el desarrollo.

Dentro de estas pruebas finales, las de seguridad consisten en atacar realmente el sistema, tratando de descubrir vulnerabilidades como lo haría cualquier atacante. Si los ataques fallan no quiere decir que el sistema no vaya a fallar en el siguiente ataque. Además, es muy improbable, por no decir imposible, que estas pruebas de

[6] **Pressman.** *op. cit*, 537.
[7] **Cerrada Somolinos, et alt.** *op. cit*, 274.
[8] **Ibid**.

caja negra descubran la existencia de funciones no documentadas embebidas dentro del software analizado.

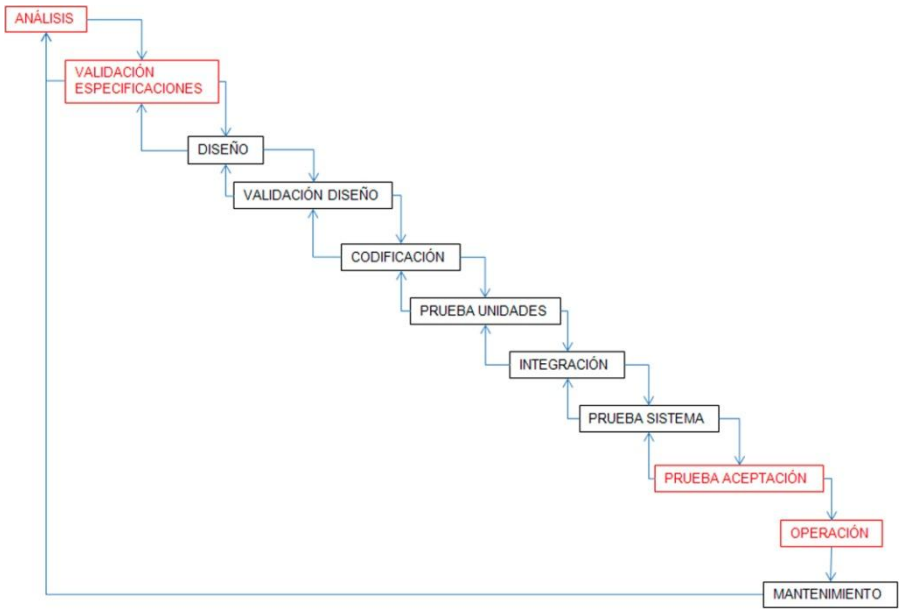

Figura 1

La figura 1 muestra en rojo las fases, pruebas y validaciones en las que suele participar el cliente cuando controla la calidad y seguridad después del desarrollo. Por simplicidad se ha usado en todas las figuras el gráfico de la metodología clásica o en cascada, esto no implica que no se empleen otras metodologías. Así mismo se han omitido muchas de las conexiones por claridad del gráfico en detrimento de la completitud del mismo.

CONTROL DE LA CALIDAD Y SEGURIDAD DEL SOFTWARE DURANTE EL DESARROLLO

Durante el desarrollo, aparte de las ya mencionadas pruebas de caja negra, se deben hacer otros tipos de pruebas, principalmente las pruebas denominadas por unos autores de caja blanca[9] y por otros de caja transparente y una actividad que podría considerarse como una variante de las mismas que es la revisión del código. Ninguna de las pruebas mencionadas son excluyentes, todas son complementarias y todas deberían hacerse para tener un mínimo de seguridad de que el software es seguro y de calidad.

[9] Las pruebas de caja blanca consisten en recorrer todos los caminos lógicos de la unidad que se está probando.

Las pruebas de caja blanca según Pressman[10] y Allen[11], se orientan principalmente a la prueba de las diferentes unidades que componen el producto software. Deben proponerlas quienes hayan participado en el desarrollo (Cerrada[12]) porque es preciso conocer en detalle la estructura del software analizado.

Incluso con software con pocas líneas de código y, al menos en apariencia, con poca complejidad es muy difícil que sean exhaustivas. Pressman[13] pone un ejemplo de una unidad con un solo bucle que se ejecuta un máximo de 20 veces y dentro del cual hay solo cinco predicados lógicos simples (instrucciones if...then…else o de bifurcación); esta unidad puede tener cerca de 100 billones de caminos posibles. No obstante hay técnicas y procedimientos como el cubrimiento lógico, las pruebas de bucles y la intuición (Cerrada[14]) que permiten simplificarlas.

Las pruebas de caja blanca nos pueden señalar la existencia de errores de diseño, lógicos, de codificación; nos pueden indicar si el software analizado hace lo que se supone que debe hacer, todo lo que debe hacer y solo lo que debe hacer y no hace alguna cosa no prevista.

La revisión de código es complementaria de la prueba de caja blanca; está muy bien descrita por Allen[15] y por Howard[16] y consiste en leer línea a línea el código fuente.

Lo ideal es hacer la revisión del código mediante equipos que tengan cuatro personas: moderador, lector, apuntador y autor; aunque es admisible una forma rápida consistente en el lector y el autor, es menos rigurosa pero sigue siendo útil. Por supuesto que existen y se usan herramientas para acelerar esta actividad, pero estas herramientas no pueden ni deben sustituir al equipo porque se basan en bases de datos de patrones y si hay algún patrón que no esté recogido en las bases de datos usadas pasará desapercibido; tampoco evalúan problemas de diseño ni de arquitectura. De lo dicho se deduce que el personal del equipo debe estar familiarizado con el software que se está analizando, no solo con el lenguaje usado, también con el diseño y estructura de aquel y la implementación, compilador y librerías utilizadas, de éste.

Una idea de la magnitud de esta tarea la da que el código fuente de Windows XP tiene unos 40.000.000 de líneas, sin contar líneas de comentarios; en una página DIN A4 caben unas 50 líneas de texto esto quiere decir que el código fuente de Windows XP ocuparía un libro de tamaño DIN A4 de 800.000 páginas o 400.000 hojas que

[10] **Pressman,** *op. cit*, 559.
[11] **Allen,** *op. cit*, 175.
[12] **Cerrada Somolinos, et alt.** *op. cit*, 280.
[13] **Pressman,** *op. cit*, 522.
[14] **Cerrada Somolinos, et alt.** *op. cit*, 280-284.
[15] **Allen, Julia H. et alt.** *op. cit*, 156-160.
[16] **Howard, Michael et alt.** *op. cit*, 615-626.

apiladas tendrían una altura de unos 40 metros o la altura aproximada del Palacio de Cibeles, antiguo Palacio de Comunicaciones, en Madrid.

Ocurre algunas veces que el software hace cosas al margen de las especificaciones; muchas de esas cosas son las llamadas puertas traseras que el desarrollador pone para facilitarle la tarea de desarrollo y de mantenimiento. También puede ocurrir que en determinadas aplicaciones el desarrollador escriba unas muy pocas líneas de código para que determinado evento o circunstancia sea desechada por el sistema y no aparezca; pensemos en un sistema informático que controla si pagamos algo y el desarrollador de dicho sistema ha introducido una pocas líneas de código que pongan que él siempre ha pagado. No es que haya muchos documentos que prueben que estas cosas han ocurrido, y de hecho es poco probable que los clientes a los que les ha pasado lo digan públicamente, pero podría ocurrir y hay rumores de que ha ocurrido. Descubrir esas líneas de código en un mar de cientos de miles o millones de líneas de código una vez que el sistema ha sido entregado al cliente y sin utilizar un equipo de revisión de código es tarea de titanes. Este hipotético caso de ejemplo tiene más probabilidades de evitarse mediante la revisión de código, para seguridad y calidad, durante todo el desarrollo y unidad por unidad. Según Garfinkel[17] la única forma de descubrir puertas traseras y similares es mediante revisión del código.

La revisión del código nos puede permitir además descubrir problemas de eficiencia y sustituir algoritmos poco eficientes por otros más eficientes.

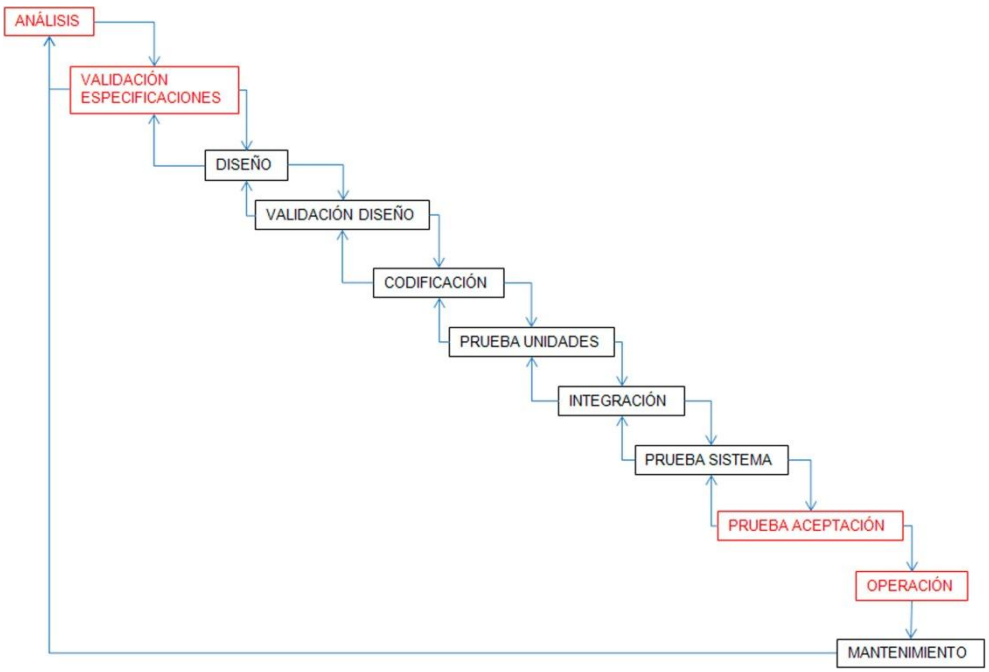

Figura 2

[17] **Garfinkel, Simson y Spafford, Gene.** *Practical UNIX & Internet Security,* Second Edition, April 1996, 12.3.5.2.

La figura 2 muestra en rojo las fases típicas donde el cliente suele intervenir cuando controla la calidad y seguridad durante todo el proceso de desarrollo.

DESARROLLO INTERNO VERSUS COTS Y VERSUS SOFTWARE A MEDIDA DESDE EL PUNTO DE VISTA DE LA SEGURIDAD

Ha habido, y continuará por siempre, un debate sobre si desarrollar el propio software, comprar software comercial, el famoso COTS (Commercial off the shelf) o comprar software a medida.

Desde el punto de vista del control propio de la calidad y seguridad del software, el software COTS es sobre el que menos control se tiene. Los fabricantes nunca suelen dar acceso a todo el código fuente de este software; además aunque tengamos personal con la formación adecuada, no conocerán en detalle ni el diseño ni la estructura de las diversas unidades que componen ese software por lo que solo podremos hacer pruebas de caja negra y ya hemos visto que las pruebas de caja negra solo nos permiten afirmar si el software hace todo lo que dicen las especificaciones que debe hacer, pero no lo que no debe hacer. Por supuesto, suele ser mucho más económico que los desarrollos propios y que el software a medida, pero es muy difícil que podamos saber alguna vez si hace cosas no previstas. De hecho, no es infrecuente encontrar en Internet que tal o cual programa comercial cuando se introducen tales o cuales caracteres muestra tal o cual cosa o permite ver aquello o lo otro. Esto, que en principio puede parecer gracioso, no deja de ser una vulnerabilidad y siempre deja la duda de si hay otras cosas que haga y esto sin entrar en que haya ataques externos; no hay ningún virus es la propia funcionalidad del programa que estamos utilizando y que desconocemos porque, obviamente, no está documentada, por lo que ningún antivirus nos puede proteger de ello.

Ejemplos de software COTS son el sistema operativo y el paquete ofimático que se utilizan en todos los ordenadores personales del Ministerio de Defensa y de otras administraciones públicas.

El desarrollo propio nos permite tener un control exhaustivo de la calidad y seguridad del software, pero para eso necesitamos tener personal propio con la formación necesaria en todos los niveles del proyecto. Hay diversos sistemas de las Fuerzas Armadas que son ejemplos de desarrollo propio.

El comprar software a medida es la única vía cuando no se puede desarrollar internamente ni existe software comercial que satisfaga nuestros requisitos. El control sobre la calidad y seguridad del mismo dependerá del contrato firmado con la

empresa que lo desarrolla y de nuestras capacidades. También tenemos ejemplos de este sistema en las Fuerzas Armadas.

Hay una variante de la adquisición de software a medida y es el desarrollo compartido entre la empresa desarrolladora y el cliente. Aquí parte del equipo de desarrollo lo pone la empresa desarrolladora y la otra parte el cliente. Todo el equipo trabaja junto en el mismo lugar, no hay partes opacas y se dan todas las circunstancias favorables para que se desarrollen pruebas de caja negra, caja blanca y revisión del código. Para utilizar esta variante es necesario que el cliente tenga personal con la formación adecuada y suficiente para hacer el desarrollo, pero no en la cantidad suficiente como para hacerlo en solitario; por esta razón se contrata a una empresa que complementará a nuestro equipo de desarrollo; normalmente cada una de las partes pone un 50% del equipo. Este sistema también se ha empleado en las Fuerzas Armadas aunque no es el más común; de hecho solo se ha divulgado un caso, lo cual no quiere decir que haya más, fue el Ejército del Aire que lo empleó en el programa ALERCAN. Con este sistema el control sobre la calidad y la seguridad del software se puede considerar igual que en el caso de desarrollo propio.

DESARROLLO INTERNO VERSUS COTS Y VERSUS SOFTWARE A MEDIDA DESDE EL PUNTO DE VISTA DEL MANTENIMIENTO

Una vez terminado el desarrollo, pruebas e instalación, viene la fase de mantenimiento que será correctivo y perfectivo. Pressman[18] dice “en tono jocoso (sic)”, aunque no se alcanza a percibir la gracia, que lo mejor para esta fase es que la desarrollase la misma empresa y, a ser posible, las mismas personas que hicieron el desarrollo; en muchas ocasiones estas son las Horcas Caudinas que hay que pasar porque los manuales de mantenimiento no son lo buenos que debieran y el código fuente no tiene los comentarios, ni en calidad ni en cantidad, que faciliten la labor de mantenimiento. También se da con cierta frecuencia que el diseño no está bien estructurado y es difícil seguir el código ni entender su lógica sin los manuales. En estas condiciones, que se agravan cuando quien hace el mantenimiento no tiene acceso a documentos que expliquen la lógica del código, es fácil que al hacer mantenimiento, tanto correctivo como perfectivo o evolutivo, se introduzcan nuevas vulnerabilidades que reducirán el nivel de seguridad del software. Pressman[19] apunta incluso que la solución menos mala podría ser hacer nuevo software que sustituya al

[18] **Pressman,** *op. cit*, 592.
[19] **Pressman,** *op. cit*, 603.

viejo. La reingeniería[20] [21]es, en cierto modo, sustituir el software viejo por otro nuevo basado en el viejo y hecho usando las técnicas y metodologías de la ingeniería del software.

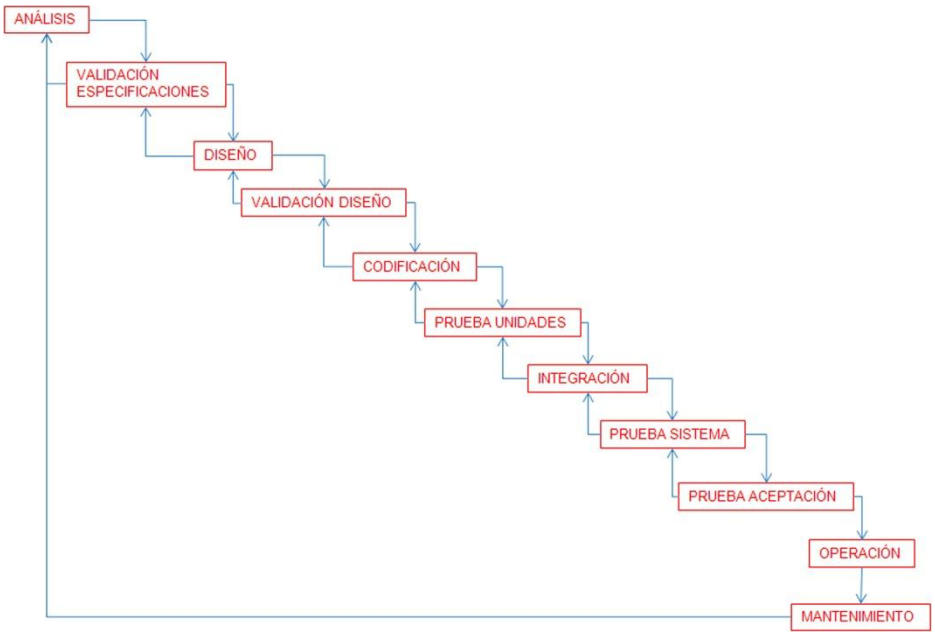

Figura 3

La figura 3 muestra en rojo la participación del cliente en los casos de desarrollo propio y de desarrollo conjunto con la empresa. Como se ve en estos casos el control de todas las fases, pruebas y validaciones puede ser total. En el caso del software COTS, en las figuras anteriores solo estaría en rojo la fase de operación; también el cliente podría hacer algunas tareas de mantenimiento dentro de la administración del sistema.

LOS DIPLOMADOS EN INFORMÁTICA MILITAR (DIM) SON UN RECURSO ESTRATÉGICO

Como hemos visto, para cualquier empresa u organismo los dos sistemas de obtención de sistemas software que permiten al cliente tener un control de cierto nivel sobre la calidad y sobre todo la seguridad del software implican tener en la organización personal con la formación adecuada.

En el caso de las FAS, ese personal es el que tiene la titulación de Diplomado en Informática Militar (DIM) y la de Programador de Informática (PIN); las

[20] **IEEE.** *SWEBOK v3.0*, 2014, 5-10.
[21] **Cerrada Somolinos, et alt.** *op. cit*, 24.

denominaciones varían en cada Ejército y Armada pero en lo sustancial las titulaciones tienen el mismo objetivo; además, los currículos aunque no iguales sí coinciden en dar gran importancia a la ingeniería del software, a la ingeniería de sistemas y a las redes o sistemas de transmisión de datos. Además, hay oficiales y suboficiales que tienen terminada la carrera de Ingeniería en informática, tanto superior como técnica, o módulos de Formación Profesional media y superior en informática. Aquí se les considerará a todos englobados en las denominaciones de DIM, en el caso de los diplomados e ingenieros, y PIN, en el caso de los programadores y técnicos de FP, para abreviar.

La situación actual es que cualquier sistema informático, aunque no esté conectado a ninguna red, es susceptible de ser atacado. Esto se vio muy claramente con el ataque del virus Stuxnet a la planta de enriquecimiento de uranio de Natanz, Irán, y que ha sido muy ampliamente difundido.

Además, tal y como ya se ha explicado, las pruebas del software son largas y caras por lo que encarecen el producto final el cual puede dejar de ser competitivo, si solo nos fijamos en el precio y en lo más llamativo de la funcionalidad. Este encarecimiento proporcionará motivos a los gestores de las empresas desarrolladoras y de las empresas clientes para aceptar una reducción de las pruebas a lo más llamativo de las especificaciones.

También hacer unas especificaciones completas que contemplen de manera adecuada no solo la funcionalidad de la aplicación sino también la seguridad, lleva su tiempo y, por tanto, encarece el producto final; otro motivo más para obviar, en la medida de lo posible la seguridad. Según Pressman[22], Cerrada[23] y el Institute of Electrical and Electronics Engineers[24] (IEEE), la participación del cliente en la redacción de las especificaciones es crucial y no debe limitarse a los dirigentes sino que debe incluir necesariamente a los usuarios finales y a los expertos del cliente. Además el IEEE[25] preconiza también los equipos de revisión de especificaciones en los que deben estar tanto los desarrolladores como el cliente representado tanto por la gerencia como por los usuarios finales del software.

Sin embargo las FAS deben velar, en general toda la administración, porque sus sistemas tengan el nivel de calidad y seguridad requerido y poderlo probar antes de recibir de conformidad el software. Así mismo es necesario que esas pruebas se extiendan a todas las fases del ciclo de vida del software.

[22] **Pressman,** *op. cit*, 153.
[23] **Cerrada Somolinos, et alt.** *op. cit*, 42.
[24] **IEEE.** *op. cit*, 1-4.
[25] **IEEE.** *op. cit*, 1-11 – 1-12.

Para probar que el software que compran las FAS tiene ese nivel de calidad y seguridad es necesario hacer pruebas de caja negra, caja blanca y revisiones de código de calidad y seguridad en las que participen de manera activa personal de las FAS. Estas pruebas deben hacerse no sobre algunas partes del software, deben abarcar todas las unidades para tener un mínimo de garantía de que ese software no va a hacer algo oculto, ni existen puertas traseras, obviamente no documentadas; por supuesto también deben hacerse las pruebas de integración y de sistema. Esto debe ser extendido a absolutamente todo el software incluidos compiladores y sistemas operativos, sobre todo en los sistemas de mando y control. Ese personal de las FAS que debe participar en las pruebas obviamente tiene que tener la formación adecuada, DIM y PIN, y debería tener un conocimiento de la estructura, diseño e implementación de cada una de las unidades y de todo el sistema.

Es obvio que esas pruebas deben realizarse en todos los sistemas informáticos de las FAS y sobre todo en los sistemas de mando y control, incluyendo la totalidad de cada uno de estos sistemas, porque esas pruebas son las que darán la necesaria confianza al Mando en los mismos; teniendo en cuenta que los sistemas de mando y control son críticos y tienen importancia estratégica, el personal de las FAS que debería hacer las pruebas mencionadas anteriormente, y a ser posible el mantenimiento de los sistemas, es también un recurso estratégico.

Mantener un número suficiente de oficiales y suboficiales con las titulaciones mencionadas o, en el caso de las civiles, su equivalente de acuerdo con las nuevas denominaciones del Espacio Europeo de Educación Superior (EEES) puede ser vital y eso sin pensar en acciones de ciberdefensa, solo pensando en que el software que usemos tenga un nivel mínimo, alto, de calidad y seguridad.

Se puede pensar que emplear laboratorios certificados de acuerdo con los Common Criteria que certifiquen el software que se adquiere sería suficiente; sin embargo, los Common Criteria han tenido críticas importantes. En este sentido es necesario recordar un artículo de William Jackson[26] en el que, basado en entrevistas a ingenieros y gestores de sistemas informáticos, ponía de relieve que los Common Criteria realmente no aseguran que el software sea seguro, sino que en el proceso de desarrollo se han seguido unas determinadas pautas. Solo en el nivel 7 (EAL7) se presenta todo el código fuente de la parte del sistema que se pretende certificar y ya hemos visto que para tener un mínimo de control sobre la calidad y seguridad del software es necesario hacer pruebas de caja blanca y revisiones del código en las que es imprescindible que haya personal que conozca bien el código fuente presentado y el diseño y estructura de todo el sistema desde el nivel unidad.

[26] **William Jackson**. Under Attack: Common Criteria has loads of critics, but is it getting a bum rap? By William Jackson, http://gcn.com/Articles/2007/08/10/Under-attack.aspx?p=1. Consultado el 17/12/2013.

El control debe extenderse durante todo el ciclo de vida del software incluido el mantenimiento. La fase de mantenimiento siempre es la que dura más tiempo y cubre según Pressman[27] más del 70 % del coste total del software y esto puede aumentar dramáticamente si el personal que desarrolló el software no está disponible; si esta fase se deja exclusivamente en manos de empresas, tendrán siempre la tentación de aumentar los precios de ese mantenimiento porque será muy difícil que haya competencia real y esos contratos de mantenimiento corren peligro de convertirse en monopolios. Lo cual es otra razón más para considerar a los DIN y PIN un recurso estratégico, sobre todo en las actuales circunstancias de crisis económica y recursos económicos restringidos. Que el mantenimiento tanto correctivo como perfectivo como adaptativo lo desarrollen total o parcialmente ellos, evitará que las empresas contratadas en este último caso caigan en la tentación del monopolio.

Todo lo anterior demuestra que los DIM y PIN son imprescindibles si se quiere tener control propio de la calidad y seguridad del software que se adquiere. Que disponer de ellos en número suficiente proporcionará seguridad y ahorrará dinero del presupuesto. Que deberían estar involucrados en cualquier proyecto en el que el software sea importante.

Además, dado que son ellos quienes deberían hacer todas las pruebas y revisiones mencionadas, en representación de las FAS, al software de sistemas estratégicos y críticos, adquieren la categoría de recurso estratégico y crítico.

i

*José Tomás Hidalgo Tarrero**
Coronel EA

[27] **Pressman,** *op. cit*, 589-590.

BIBLIOGRAFÍA

Common Criteria Recognition Arrangement (CCRA). *Common Criteria for Information Technology Security Evaluation (CC) versión 3.1, edición 4*.

Common Criteria Recognition Arrangement (CCRA). *Common Methodology for Information Technology Security Evaluation (CEM) versión 3.1, edición 4*.