Artículo de Investigación. Revista Killkana Técnica. Vol. 1, No. 3, pp. 25-32, Septiembre-Diciembre, 2017.
p-ISSN 2528-8024 / e-ISSN 2588-0888. Universidad Católica de Cuenca

Verificación y Validación de Software

Software Verification and Validation

Leopoldo Pauta Ayabaca[1]* **y Santiago Moscoso Bernal**[2]
[1]Unidad Académica de Tecnologías de la Información y Comunicación
[2]Unidad Académica de Ingeniería, Industria y Construcción
Departamento de Gestión de Calidad
Universidad Católica de Cuenca, Ecuador
*spauta@ucacue.edu.ec

Resumen

El proceso de verificación y validación (V&V) aborda todas las etapas del ciclo de vida del software de manera determinante, siendo utilizado para establecer si determinada tarea o producto, cumple con las necesidades del usuario y los requisitos establecidos para su desarrollo. V&V apoya al proceso de construcción proporcionando una valoración objetiva de los productos y los procesos que forman parte del ciclo de vida de desarrollo del software. Estos procesos se apalancan en estándares como ISO/EEC 15288:2008 e ISO/EEC 12207:2008 que permiten aportar al software el concepto de calidad, estableciendo si los requisitos son correctos, completos, precisos, consistentes y verificables. Las pruebas son parte de un proceso más amplio de verificación y validación de software V&V, y se soporta en los estándares IEEE1008 e ISO / IEC 29119. Las pruebas de software nacen por la necesidad de garantizar un producto de calidad, descubriendo defectos que podrían contener los programas antes de la implantación, y demostrar que un programa hace lo que se pretende que haga. El artículo busca diferenciar entre conceptos de Verificación, Validación y Pruebas, su tiempo de aplicabilidad en las diferentes fases del desarrollo de un software de calidad, y los diferentes estándares que pueden ser aplicados en función de estos conceptos.

Palabras clave: Proceso de verificación y validación V&V, ciclo de vida del software, requisitos, ciclo de vida de desarrollo del Software, pruebas, estándar.

Abstract

The verification and validation (V&V) process addresses all stages of the software lifecycle in a decisive way. It is used to determine if a given task or product fulfills the user's needs and meets the requirements established for its development. V & V supports the construction process by providing an objective assessment of the products and processes which are part of the software development lifecycle. These processes are based on ISO / EEC 15288:2008 and ISO / EEC 12207:2008 standards, which provide the software with a quality concept, establishing whether the requirements are correct, complete, precise, consistent and verifiable. Testing is part of a broader software verification and validation process V & V, and is supported by the IEEE1008 and ISO / IEC 29119 standards. The software testing originated from the need to guarantee a quality product, uncovering the defects the programs may contain prior to the implementation, and the necessity to demonstrate that a program does what it is intended to do. The article aims to differentiate between Verification, Validation and Testing concepts, their applicability time in different phases of the development of a quality software, and the different standards that can be applied based on these concepts.

Key words: V&V Verification and validation process, software lifecycle, requirements, software development lifecycle, testing, standard.

I. Introducción

El aseguramiento de la calidad es en la actualidad uno de los tópicos de investigación más importantes dentro del área de ingeniería de software. La falta de calidad de los sistemas que se desarrollan es uno de los mayores contribuyentes a la llamada "crisis" del software actual [1].

El desarrollo del software (sw) actualmente ha tenido un crecimiento muy importante y su implementación en diferentes áreas y productos ha crecido a pasos agigantados a tal punto que en estimaciones realizadas, se calculaba que en el 2011 la cantidad de líneas de código de los teléfonos móviles sobrepase fácilmente las 10 millones de líneas, el software de un automóvil bordeara las 100 millones de líneas [2],el sistema de control de vuelo del Boeing 787 alrededor de 6.5 millones de líneas [3],como un ejemplo a nivel de productos. Actualmente conocemos que las nuevas tendencias tecnológicas dependen en gran medida del software, tales como: los sistemas financieros, los sistemas de gestión, los sistemas médicos, entre otros, están presentes en la gran mayoría de actividades del ser

humano. Al ser parte de las actividades humanas, estos sistemas demandan software de calidad con un menor número de fallas posibles, de tal manera que permitirán una reducción de riesgos en sus actividades económicas, propiciando mayores ganancias y optimización del tiempo.

A nivel mundial existen múltiples metodologías, técnicas y herramientas modernas de ingeniería de software que permiten mejorar la calidad de los productos de software desarrollados. Algunas de ellas son: el Programa de Certificación de Pruebas de Software (Certified Software Tester, en sus siglas en inglés CSTE) [4],los estándares de ingeniería de software del IEEE [5],el Modelo de Capacidad/Madurez (CMM) y el estándar ISO 9001 [6],.

Bajo este contexto, podemos traer a colación algunas preguntas. ¿Por qué los problemas de software no fueron detectados?, ¿Cómo puede ser remediada esta situación? [7]. Según Black, los problemas que presenta el software se debe a que una de las áreas más desatendidas en el desarrollo de software, son las actividades de la V&V y las pruebas del software.

Durante y después del proceso de implementación, el programa que se está desarrollando debe ser comprobado para asegurar que satisface su especificación, y entrega la funcionalidad esperada por las personas que pagan por el software. La V&V tiene lugar en cada etapa del proceso del software, comienza con revisiones de los requerimientos y continúa con revisiones del diseño e inspecciones de código hasta la prueba del producto [8].

En este sentido, instituciones y organizaciones como la IEEE (Institute of Electrical and Electronics Engineers), la ISO (International Organization for Standardization), la IEC (Comisión Electrotécnica Internacional), entre otras, se han centrado en la definición de estándares que apoyen la V&V y pruebas. La versión más antigua de este estándar data de 1986, y describía el contenido del plan V&V para software, con las subsecuentes versiones (1998 y 2004) el enfoque cambia desde el plan V&V de software hacia los procesos de software V&V. Esta revisión expande el alcance de los procesos V&V para incluir sistemas y hardware así como también software. Adicionalmente, se alinea con la terminología y estructura para ser consistente con ISO/IEC 15288:2008 e ISO/IEC 12207:2008.

Lo anterior conlleva a centrarnos necesaria y obligatoriamente en la criticidad del software, que por lo tanto, debe ser llevada a cabo por personas con experiencia y conocimientos tanto del dominio del negocio como de las herramientas de prueba de software. Al igual que la definición de estándares ISO/IEC 29119 Software Testing, también se han creado centros para la Certificación en Pruebas de software (“Software Testing Certificatión”) del ISTQB (“International Software Testing Qualifications Board”), que permitirán en conjunto satisfacer las necesidades apremiantes de calidad del software, que se ha tratado de alcanzar desde los comienzos de la computación. Es así que [9] clasificaron las pruebas del software en las siguientes fases y metas:

i) **Periodo de Depuración**: (hasta 1956), aquí las pruebas eran asociadas a la depuración: no había una clara diferencia entre las pruebas y la depuración.
ii) **Periodo de demostración**: (desde 1957 a 1978), en este periodo la depuración y las pruebas fueron diferenciados, en este período fue demostrado que el software satisface los requisitos.
iii) **Periodo de destrucción**: (desde 1979 a 1982), esta fase tenía como propósito o meta la de encontrar errores.
iv) **Periodo de evaluación**: (desde 1983 a 1987), la intención de este proceso es que durante el ciclo de vida del software se satisfizo su especificación, para detectar y prevenir los defectos.
v) **Periodo de prevención**: (desde 1988 hasta la actualidad), fue el período de la prevención, en el cual las pruebas estarían para demostrar que el software satisface su especificación, para detectar y prevenir los defectos.

Podemos deducir entonces que existe la creciente percepción del software como un elemento del sistema y la importancia de los “costes” asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas. No es raro que una organización de desarrollo de software emplee entre el 30 y el 40 por ciento del esfuerzo total de un proyecto en las pruebas. En casos extremos, las pruebas del software para actividades críticas (por ejemplo, control de tráfico aéreo, control de reactores nucleares) pueden costar ¡de tres a cinco veces más que el resto de las fases de los procesos de la ingeniería del software juntos! [10].

Desde esta visión general del problema y del proceso de V&V y los procesos de pruebas del software, este articulo recolecta, identifica, y diferencia los conceptos, su aplicabilidad, en el Ciclo de Vida de Desarrollo de Software (CVDS) y sus implicaciones en la búsqueda de la calidad del software.

II. Conceptos: verificación, validación y pruebas

A. *Aseguramiento de la calidad del software*

La ingeniería de software es la disciplina que desarrolla y utiliza metodologías, métodos y herramientas para desarrollar software de buena calidad. Uno de los componentes claves de todo proceso de desarrollo de software, son las actividades que se llevan a cabo para asegurar la calidad del software que se produce. Este conjunto planeado y sistemático de actividades que aseguran que el proceso y los productos cumplen con los requerimientos técnicos establecidos, se conoce como aseguramiento de la calidad del software (ACS). Un software de calidad es aquel que: “. . . concuerde los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas (por ej.: fácil de mantener) que se esperan de todo software desarrollado profesionalmente” [11].

De acuerdo a [12] un componente importante del ACS son las actividades de verificación y validación (V&V) del software que se realizan durante las diferentes fases que componen el ciclo de desarrollo de los sistemas. El proceso de V&V provee una evaluación objetiva de los productos y del proceso desarrollados durante el ciclo de vida del software. Esta evaluación demuestra que los requerimientos del software y del sistema son correctos, completos, precisos, consistentes y fáciles de probar. Otros objetivos de V&V son:

a) Facilitar la detección y corrección temprana de errores.
b) Mejorar la administración del proceso y los riesgos del producto.
c) Apoyar el proceso del ciclo de vida para asegurar el cumplimiento de los requerimientos de rendimiento, cronograma y presupuesto del programa.

B. *Verificación*

Algunos de los conceptos obtenidos expresamos a continuación:

Verificación, a la facilidad con la que puede demostrarse el correcto funcionamiento del software, mediante su prueba [13].

Verificación, se entiende por la capacidad de un software de ejecutar las funciones definidas en las especificaciones [14].

También, Unhelkar Behuvan presenta una definición acerca de lo que es verificación:

Verificación, comprende un conjunto de actividades separadas que aseguran que el modelo es correcto [15].

Se puede decir entonces que no se podrá Verificar la conformidad de un software sin que sus funcionalidades hayan sido correctamente especificadas. *La verificación se refiere al conjunto de actividades o pruebas que aseguran que el software implementa correctamente las especificaciones definidas para este sistema*.

C. *Validación*

Algunos de los conceptos obtenidos de validación los expresamos a continuación:

Validación, mientras que la validación se refiere a un conjunto diferente de pruebas que aseguran que el software construido se ajuste a los requisitos del cliente [10].

Validación, la validación del software es el proceso de comprobar que el sistema está acorde a las especificaciones y que cumple con las necesidades reales de los usuarios del sistema [16].

Validación, como señala (Boehm, 1981), el software está verificado si estamos construyendo el producto correctamente. Y es válido si estamos construyendo el producto correcto [14].

Analizando las definiciones anteriores, podríamos indicar que la *Verificación* es una actividad para gente que se encuentra inmersa en el desarrollo del software, que se centran en el modelo del programa, mientras que la *Validación* involucra al usuario con el fin de determinar si el programa cumple con sus necesidades.

En este sentido, [17] comenta que la *Verificación* permite garantizar la consistencia entre los productos desarrollados (i.e. los prototipos y la versión final del software) y sus requerimientos predeterminados en la fase de especificación. Así mismo, indica que la Validación debe satisfacer los requerimientos del sistema, es decir, si se está desarrollando el producto correcto.

D. *Pruebas de software*

Las *Pruebas* son una técnica dinámica de V&V [8].

Las *Pruebas* del software, consisten en un proceso crítico para asegurar que el software sea entregado al cliente libre de defectos, y debería ser tratado como tal [7].

Prueba, se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas. Los resultados de la ejecución se observan y registran con el fin de realizar posteriormente una evaluación de algún aspecto.

Dijkstra comenta que las pruebas solo pueden demostrar la presencia de errores, no su ausencia [18]. Esto conlleva a que las pruebas cumplan los siguientes objetivos:

- Las pruebas podrían ser llevadas a cabo para encontrar defectos, proporcionando a los programadores la información que ellos necesitan para corregir los mismos.
- Las pruebas podrían ser llevadas a cabo para brindar confianza a la gente en cuanto al nivel de calidad del sistema. Así mismo las pruebas podrían ser llevadas a cabo para prevenir defectos.
- Los objetivos de las pruebas pueden cambiar dependiendo de la fase o del nivel de pruebas con el que estemos involucrados [7].

Resumiendo, podemos concluir que la calidad del software estará fundamentada en las pruebas como un elemento crítico que se encuentra presente en todas las fases del desarrollo del software.

III. V&V y pruebas como solución al mejoramiento de la calidad del software

Lo anterior conlleva a realizar la pregunta: ¿Cómo realizar la Verificación y la Validación?. Los procesos de V&V proporcionan una evaluación objetiva de los productos y procesos en los CVDS. Esta evaluación objetiva demuestra si los requisitos son correctos, completos, precisos, consistentes y verificables. Un sistema de software es verificable, si sus propiedades pueden ser verificadas fácilmente. Por ejemplo, la navegabilidad o el performance de un sistema son propiedades que interesa verificar, el diseño modular, prácticas de codificación disciplinadas, entre otros, contribuyen a la verificabilidad de un sistema. ¿Cómo determinar si el sistema está cumpliendo a cabalidad con las necesidades del usuario?, si el sistema está respondiendo en su transaccionalidad, en sus salidas, etc., son consideraciones entre otras que se deben tomar en cuenta. Para cumplir y

desarrollar software de calidad, se han creado estándares internacionales como la IEEE Std 1012TM-2012 que define los procesos de V&V en términos de actividades específicas y tareas relacionadas. Estas series de normativas cambiantes se han ido incrementando, sustituyendo y mejorando tal es el caso de: ISO/EEC 15288:2008 reemplaza al estándar IEEE/EIA Std 12207.0:1996. ISO/EEC 15288:2008 es el estándar para los procesos del sistema. ISO/EEC 12207:2008 es el estándar para los procesos de software.

En los procesos de V&V, deben considerarse los estándares de desarrollo establecidos como por ejemplo la IEEE Std 1012TM-2012., que abarcan:

a) Apoyo en adquisición V&V.
b) Planificación de suministros V&V.
c) Planificación del proyecto V&V.
d) Administración de la configuración V&V.
e) Definición de requerimientos del sistema (stakeholder) V&V.
f) Sistema de análisis de requisitos V&V.
g) Diseño de la arquitectura del sistema V&V, concepto de software V&V, concepto de hardware V&V.
h) Implementación del sistema V&V, Actividades V&V de hardware y software.
i) Diseño V&V, implementación/fabricación V&V, test de integración V&V, test de calificación V&V, test de aceptación V&V.
j) Integración del sistema V&V.
k) Todas las actividades del ciclo de vida del estándar IEEE 1012.
l) Sistema de transición V&V, software de instalación y transición V&V, transición de hardware V&V.
m) Todas las actividades de validación del ciclo de vida del estándar IEEE 1012.
n) Funcionamiento del sistema V&V, funcionamiento de software V&V, funcionamiento de hardware V&V.
o) Mantenimiento del sistema, hardware y software V&V.

Traspaso de sistema, hardware y software [19].

El estándar ISO/EEC 12207:2008 adicionalmente tiene los siguientes procesos que corresponden a las actividades de implementación del sistema, hardware y software, como se indicó en la Fig. 1. Requisitos de sw, análisis de procesos de sw, proceso de diseño arquitectónico, proceso de diseño detallado de sw, proceso de construcción de sw, proceso de integración de sw, proceso de calidad de pruebas del software.

Dentro del proceso de V&V, existen dos aproximaciones complementarias para el análisis y comprobación de los sistemas:

a) Las inspecciones de software analizan y comprueban las representaciones del sistema tales como: el documento de requerimientos, los diagramas de diseño y el código fuente del programa. Puede usarse las inspecciones en todas las etapas del proceso (ver Fig. 2). Las inspecciones pueden ser complementadas con algún tipo de análisis automático del código fuente de un sistema o de los documentos asociados. Las inspecciones de software y los análisis automáticos son técnicas de V&V estáticas, ya que no se necesita ejecutar el programa desarrollado. Como se puede observar en la tabla I, en donde se revisa las algunas de las etapas del desarrollo de software, aplicados en una Metodología de desarrollo RUP (Rational Unifiqued Process).

b) Las pruebas en el proceso de desarrollo implican ejecutar los procesos que realiza el programa desarrollado con el fin de evaluar los diferentes estados que adquiere el sistema. Se examinan las salidas del software y su entorno operacional para comprobar que funciona tal y como se requiere. Las pruebas son una técnica dinámica de verificación y validación [8] tal como se indican en los gráficos de los modelos de desarrollo de software tomados de [16] en donde los modelos de Entrega Incremental y el modelo en Espiral de Boenh,

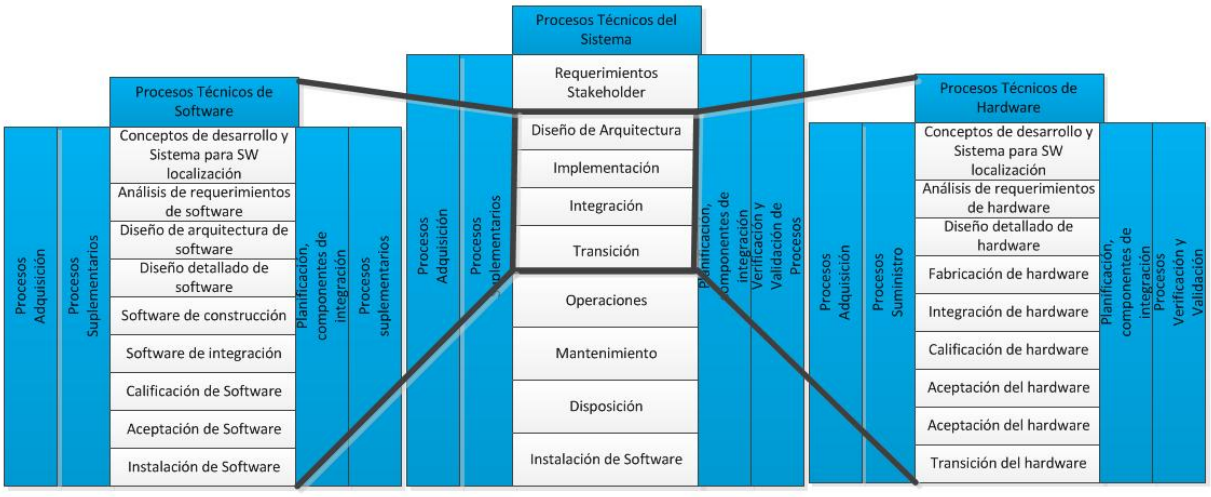

FIG. 1. V&V proceso de Sw, Sistema y Hw [19].

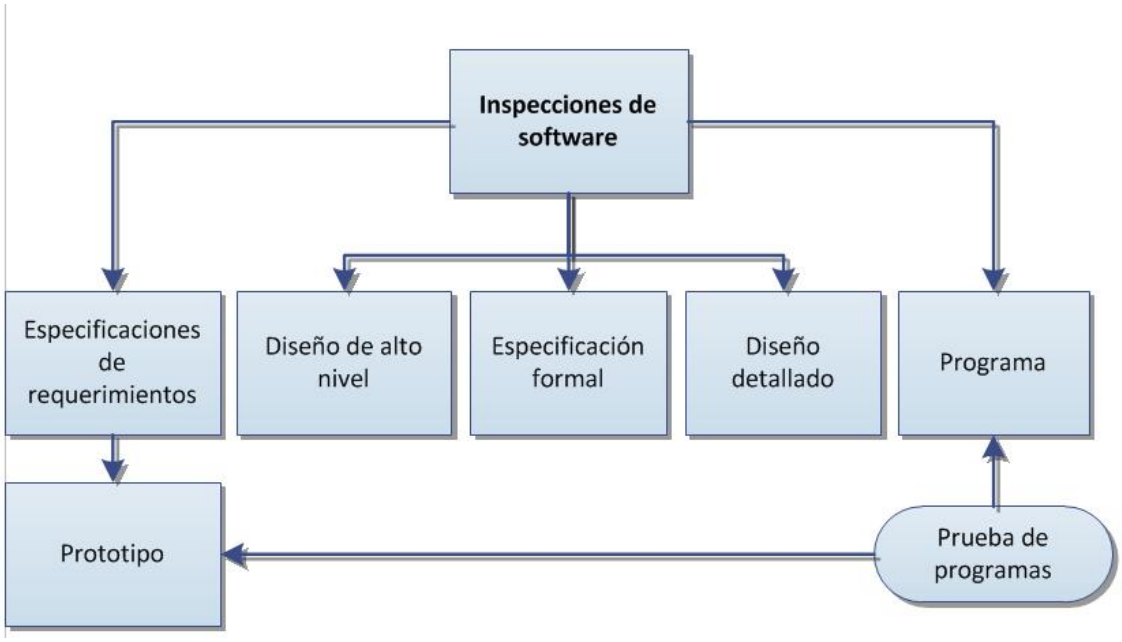

FIG. 2. V&V estática y dinámica [8].

que en un bucle constante ejecuta pruebas en fases criticas de desarrollo, como se lo puede observar en las figuras 3 y 4.

TABLA I
INSPECCIONES DE SOFTWARE

ETAPA	SUCESO
Modelo de Negocios. Casos de Uso. Tarjetas de Descripción. Conjunto de Requisitos.	- No llega a definir claramente los requisitos. - No representa los procesos organizacionales en análisis.
Modelado Funcional. Planteamiento de escenarios. Diagrama de clases.	- No define todos los escenarios activamente, implica la posibilidad de pérdida de clases. - No relaciona adecuadamente las clases con herencia y asociación.
Modelado Dinámico. Diagramas de Colaboración. Diagramas de Actividades. Tarjetas CRC.	- Se plantea inadecuadamente los diagramas de colaboración en el envío de los mensajes. - No concuerdan las clases del diagrama de clases en los diagramas de colaboración. - No existen las tarjetas de colaboración.
Modelado de Datos. Modelado Conceptual. Modelado Lógico. Modelado Físico. Diccionario de datos.	- El modelo conceptual, no representa adecuadamente los procesos organizacionales. - La cardinalidad no está adecuadamente implementada. - No existe el diccionario de datos.

A. *Niveles de pruebas*

Existen distintos niveles y tipos de pruebas:

i) **Test: Unitario**
Objetivo: Detectar errores en los datos, lógica, algoritmos.
Participantes: Programadores.
Ambiente: Desarrollo.
Método: Caja Blanca.

ii) **Test: Integración**
Objetivo: Detectar errores de interface y relaciones entre componentes.
Participantes: Programadores.
Ambiente: Desarrollo.
Método: Caja Blanca, Top Down, Botton Up.

iii) **Test: Funcional**
Objetivo: Detectar errores en la implementación de requerimientos.
Participantes: Testers, analistas.
Ambiente: Desarrollo.
Método: Funcional.

iv) **Test: Sistema**
Objetivo: Detectar fallas en el cubrimiento de los requerimientos.
Participantes: Testers, analistas.
Ambiente: Desarrollo.
Método: Funcional.

v) **Test: Aceptación**
Objetivo: Detectar fallas en la implementación del sistema.
Participantes: Testers, analistas, clientes.
Ambiente: Productivo.
Método: Funcional.

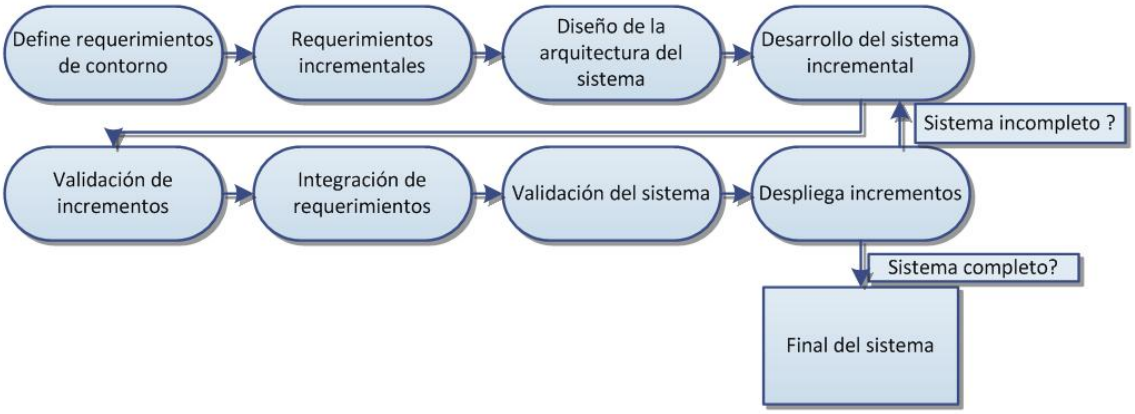

FIG. 3. Modelo de Entrega Incremental [16].

B. Tipos de pruebas

Los principales tipos de pruebas que se pueden realizar a cualquier tipo de software son:

1. Pruebas unitarias

Comprueba si el código funciona y cumple con las especificaciones necesarias para su desempeño óptimo, se focaliza en verificar cada módulo con lo que mejora el manejo de la integración de lo más básico a los componentes mayores.

Para la elaboración de pruebas unitarias en java por ejemplo, se puede utilizar el JUNIT y CACTUS.

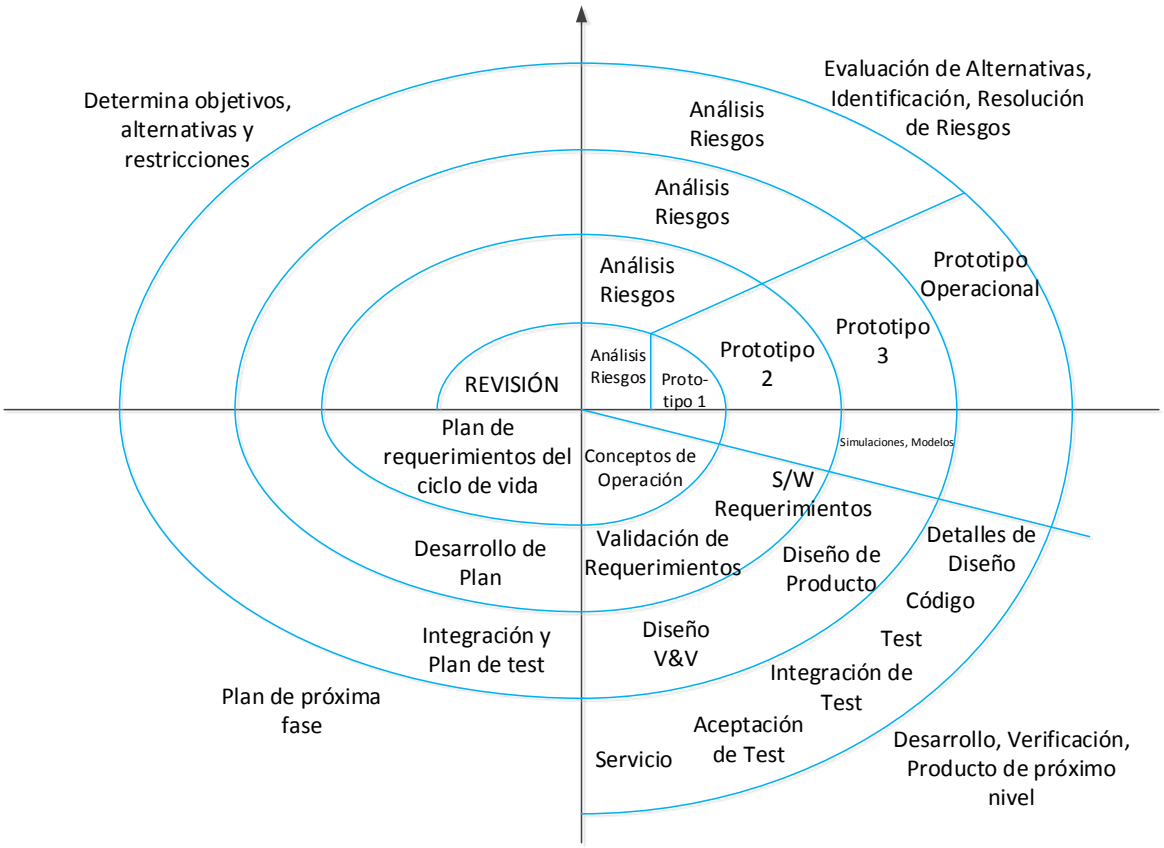

FIG. 4. Modelo en espiral de Boehm [20].

2. Pruebas de integración

Comprueba si los programas básicos funcionan correctamente luego de integrarlos, identificando los errores producidos por la combinación, definiendo si las interfaces entre los usuarios y las aplicaciones funcionan de una manera adecuada.

Existen dos técnicas definidas, top-down en donde se verifica que los módulos de nivel superior llaman a los de nivel inferior de manera correcta, y down-top donde se verifica que los módulos de nivel inferior llaman a los de nivel superior de manera correcta.

3. Pruebas de regresión

Comprueba si los cambios efectuados en una parte del programa afectan a otras partes de la aplicación.

4. Pruebas de humo (Smoke Testing o Ad Hoc)

Comprueba los errores tempranamente revisando el sistema constantemente, lo cual permite disminuir la dificultad en el momento de la integración reduciendo los riesgos.

5. Pruebas del sistema

Están enfocadas directamente a los requisitos tomados de los casos de uso según el giro del negocio, verificando el ingreso, procesamiento, recuperación de los datos y la implementación propiamente dicha.

El principal objetivo es que la aplicación cubra las necesidades del negocio, entre las cuales tenemos: prueba de funcionalidad, usabilidad, performance, documentación, procedimientos, seguridad, controles, volumen, esfuerzo (Stress), recuperación y múltiples sitios.

En referencia a los estándares debemos indicar lo siguiente: el estándar IEEE 1008 1987 se relaciona con el estándar ANSI/IEEE Std 829-1983, para documentación de pruebas, la misma que describe a la unidad de pruebas unitarias que está compuesta de 3 fases que son particionadas dentro de un total de ocho actividades básicas:

1) Realizar la planificación de pruebas.
   a) Planear el enfoque general, recursos y cronograma.
   b) Determinar las características a ser examinadas.
   c) Refinar el plan general.
2) Determinar el conjunto de pruebas.
   a) Diseñar el conjunto de pruebas.
   b) Implementar la refinación del plan y diseño.
3) Medición de las pruebas unitarias.
   a) Ejecutar los procedimientos de pruebas.
   b) Comprobar la finalización.
   c) Evaluar el esfuerzo de la prueba y la unidad.

El flujo de información que viaja hacia las fases (entrada y salida) se muestra en la Fig. 5:

IV. Conclusiones

Actualmente el incremento exponencial del software a nivel de productos y de sistemas informáticos complejos y grandes, obliga a los desarrolladores a tomar muy en serio los procesos de V&V y pruebas.

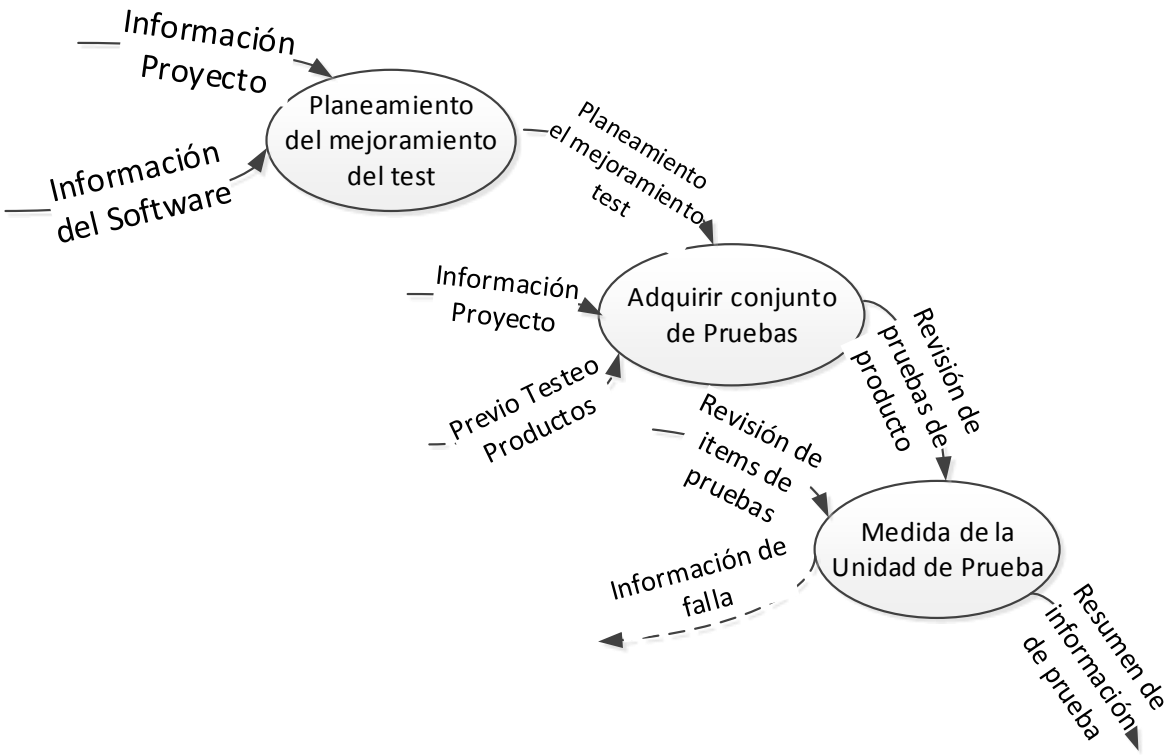

FIG. 5. Gráfico de pruebas unitarias [21].

Los estándares que se proponen no deben considerarse como definitivos y únicos, más bien deben ajustarse al tipo de software desarrollado.

En los modelos de desarrollo de software, en algunas de sus representaciones gráficas no expresa adecuadamente las actividades de V&V y pruebas.

Las pruebas al final son un grupo de intentos cambiantes, acorde a la fase en la que este el desarrollo y al tipo de software que se esté desarrollando que se plasman desde la Verificación y Validación con la norma IEEE std 1012TM-2012.

Se debe procurar establecer la V&V buscando la concordancia entre las inspecciones estáticas con las implementaciones en la codificación, sobre todo en los procesos de aprendizaje del desarrollador.

En función del objetivo propuesto de determinar las normativas de V&V, se recomienda aplicar en las distintas etapas del ciclo de vida la norma IEEE std 1012TM-2012.

En el afán de la calidad del software se aplicará la consigna de ejecución paralela de Verificación y Validación en las distintas fases del ciclo de vida.

Referencias

[1] E. Yourdon, *Decline and fall of the American programmer*. Prentice Hall PTR, 1994.

[2] R. N. Charette, "Why software fails," *IEEE spectrum*, vol. 42, no. 9, p. 36, 2005.

[3] M. Mecham, "Boeing faces pretty tight 787 delivery schedule," *Aviation Week*, vol. 9, 2007.

[4] Quality Assurance Institute, "2006 guide to the CSTE common body of knowledge." Disponible en http://www.softwaretestinggenius.com/download/CBOK.pdf, 2006.

[5] *IEEE Software Engineering Standards Collection: 2003*. Inst of Elect & Electronic, 9 2003.

[6] I. ISO, "9000-3 guidelines for the application of iso 9001 to the development, supply, and maintenance of software," 1991.

[7] R. Black and G. R. Sandoval, *Fundamentos de Pruebas de Software (Spanish Edition)*. RBCS, Inc., 2011.

[8] I. Sommerville, *Ingenieria del Software*. Madrid: EducaciónPearson, 2005.

[9] D. Gelperin and B. Hetzel, "The growth of software testing," *Communications of the ACM*, vol. 31, no. 6, pp. 687–695, 1988.

[10] R. Pressman, *Ingeniería del Software. Un enfoque práctico. 5ta. Edición.* España: Madrid, Mc. Graw Hill, 2002.

[11] W. E. Lewis, *Software testing and continuous quality improvement*. CRC press, 2016.

[12] IEEE Computer Society, "IEEE Standard for Software Verification and Validation ." Disponible en https://people.eecs.ku.edu/~hossein/Teaching/Stds/1012.pdf, Mar. 1998.

[13] N. Paez and R. Wachenchauzer, "Utilización de programación orientada a aspectos en aplicaciones enterprise," mathesis, Facultad de Ingeniería, Universidad de Buenos Aires, Buenos Aires, 2007.

[14] F. A. Amo, L. M. Normand, and F. J. S. Pérez, *Introducción a la ingeniería del software*. Delta Publicaciones, 2005.

[15] B. Unhelkar, *Verification and validation for quality of UML 2.0 models*, vol. 42. John Wiley & Sons, 2005.

[16] I. Sommerville, *Ingeniería del Software*. México: Pearson Education, México, 9na. Edición ed., 2011.

[17] G. Mackenzie, *Verification and Validation of Moderm Software-Intensive Systems*. San Diego: Prentice Hall, 2000.

[18] O.-J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, "Structured programming," 1972.

[19] C. IEEE-1012, "IEEE std 1012 2012," 2012.

[20] B. W. Boehn, "Software Engineering. ICSE," 1979.

[21] IEEE-1008, "IEEE Standard for Software Unit Testing," July 1986.

***Recibido**: 07 de diciembre de 2017*

***Aceptado**: 30 de diciembre de 2017*

Segundo Leopoldo Pauta Ayabaca: *Ingeniero de Sistemas de la Universidad de Cuenca, Especialista en Docencia Universitaria en la Universidad Católica de Cuenca, Magister en Gestión de Base de Datos en la Universidad Técnica de Ambato, Diplomado en Metodologías de la Investigación UNAM [México], Diplomado en Pedagogías Innovadoras en la UTPL; autor de cinco libros y de varios artículos científicos, Auditor Internacional ISO:9001-2015. Actualmente es Jefe del Área de Gestión de Calidad y docente de Ingeniería de Sistemas en la Universidad Católica de Cuenca.*

Santiago Arturo Moscoso Bernal: *Ingeniero Eléctrico y Especialista en Docencia Universitaria en la Universidad Católica de Cuenca, Magister en Aprendizaje de la Física en la Universidad Nacional de Chimborazo, Master en Energías Renovables en la Universidad Europea del Atlántico (España), Diplomado en Metodologías de la Investigación UNAM [México],autor de dos libros y de varios artículos científicos, Auditor Internacional ISO:9001-2015 actualmente es Jefe del Área de Auditoria de Gestión del Dep. de Gestión de Calidad y docente de Ingeniería Eléctrica en la Universidad Católica de Cuenca.*
Correo electrónico: smoscoso@ucacue.edu.ec