Model verification and validation based on CMMI

Modelo de verificación y validación basado en CMMI

O R Puello

Keywords:
Quality, CMMI, Model, Process, Validation, Verification.

Abstract

The growing concern for quality in the software industry have as its main objective the systematic development of products and services of better quality and fulfilling the needs and expectations from customers. This article presents an overview to the CMMI model at levels 2 and 3. Then and on the basis of the "Methods A the process of verifying and validation of SOFTWARE based on the standard CMMI" research presents a model to verify and validate software describing planning, activities and strategies to apply these processes during the software life cycle.

Palabras clave:
Calidad, CMMI, Modelo, Proceso, validación, verificación

La creciente preocupación por la calidad en la industria del software tiene como objetivo principal el desarrollo sistemático de productos y servicios de mejor calidad y el cumplimiento de las necesidades y expectativas de los clientes. En este artículo se presenta una introducción al modelo CMMI en sus niveles dos y tres. Seguidamente y basándose en la investigación “Metodología al proceso de verificación y validación de software basado en el estándar CMMI”, se plantea un modelo para verificar y validar software describiendo la planificación, actividades y estrategias para aplicar estos procesos durante el ciclo de vida del software.

I. INTRODUCCIÓN

Ahora, más que nunca, las industrias desean entregar productos y servicios mejores, más rápido, y más baratos. Al mismo tiempo, en el ambiente de alta tecnología del siglo XXI, casi todas las compañías se han encontrado con requerimientos en los productos y servicios cada vez más complejos. Hoy, una sola empresa generalmente no desarrolla todos los componentes de un producto o su mantenimiento. Normalmente, algunos componentes se construyen en la compañía y otros se adquieren; después todos estos se integran en el producto final o en el mantenimiento requerido. Las organizaciones deben estar preparadas para poder manejar y controlar tanto el proceso de desarrollo como el de mantenimiento.

En el mercado, existen modelos, estándares, metodologías, y pautas de madurez que pueden ayudar a una compañía para optimizar la forma en que hacen negocios. Sin embargo, la mayoría de los enfoques disponibles de la mejora se centran en una parte específica del negocio y no llevan un enfoque sistémico a los problemas que la mayoría de las industrias están haciendo frente. Centrándose en perfeccionar un área del negocio, estos modelos desafortunadamente han perpetuado los sistemas obsoletos y las barreras que existen en las compañías. El modelo CMMI proporciona una oportunidad de evitar o de eliminar estos sistemas obsoletos y barreras a través de los modelos integrados interdisciplinarios. CMMI para el desarrollo consiste en las mejores prácticas de las actividades de desarrollo y mantenimiento, encaminándolas a abarcar todo el ciclo de vida del producto hasta su entrega y mantenimiento. [1]

Este artículo se centra en el desarrollo de un modelo para verificar y validar software utilizando las mejores prácticas que propone el modelo CMMI. A continuación se contextualiza el mismo a partir de las definiciones de cada una de estas áreas de proceso y su incidencia en el costo, duración y demás factores que pueden influir en el ciclo de vida de un proyectoCMMI

Sus primeros orígenes hay que buscarlos en 1984, cuando el congreso del gobierno americano aprobó la creación de un organismo de investigación para el desarrollo de modelos de mejora para los problemas en el desarrollo de los sistemas de software y evaluar la capacidad de respuesta y fiabilidad de las compañías que suministran software al Departamento de Defensa. En este momento se crea el SEI (Instituto de Ingeniería del Software), fundado por el Departamento de Defensa Americano y la Universidad CarnegieMellon. [2]

En 1985 el SEI empieza a trabajar en un marco de madurez de procesos que permita evaluar a las empresas productoras de software. Esta investigación da lugar al "Modelo de Madurez de las Capacidades (CMM)" y en 1991 el Instituto publica la versión 1.0 del Modelo de Madurez de las Capacidades para el Software (SW-CMM, CapabilityMaturityModelfor Software). [2]

Tras este modelo empezaron a surgir otros para diferentes disciplinas: [2]

- P-CMM: People CMM
- SA-CMM: Software Acquisition CMM
- SSE-CMM: Security Systems Engineering CMM
- T-CMM: Trusted CMM
- SE-CMM: Systems Engineering CMM
- IPD-CMM: Integrated Product Development CMM

A mediados de la década del 90, el SEI decide unificar los modelos de ingeniería de software (SW-CMM), de ingeniería de sistemas (SE-CMM) y de desarrollo integrado de productos (IPD-CMM), embarcándose en un esfuerzo que da origen a una nueva generación llamada CMMI (CapabilityMaturityModelIntegration)

En Marzo del 2002 se liberó el modelo CapabilityMaturityModel® Integration (CMMISM), Versión 1.1, el cual puede servir de guía para mejorar los procesos organizacionales, además del desarrollo del Software.

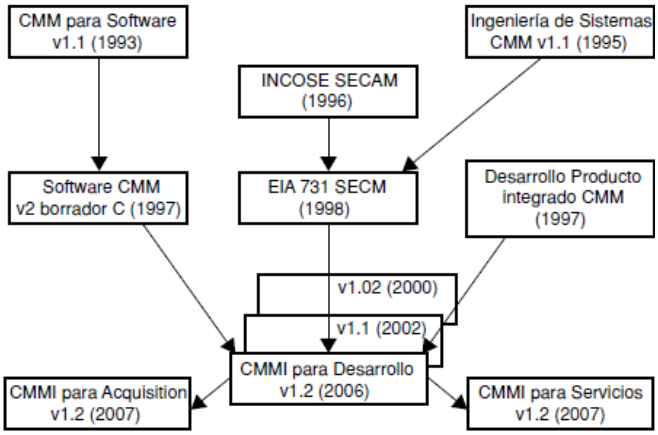

Fig. 1. Historia de los modelos CMM. *[3]*

El nuevo CMMI brinda un marco con una estructura común para todas las disciplinas (Ingeniería de Software, Ingeniería de Sistemas, etc.) e incorpora una forma de representación llamada Continua (tomada de IPD-CMM y SE-CMM), orientada a medir la mejora en los procesos de manera individual en vez de hacerlo de manera conjunta como la representación por niveles del modelo original (escalonado) [4]

El modelo CMMI define 22 áreas de proceso en la versión que integra desarrollo de software e ingeniería de sistemas (CMMI-SE/SW). Cada una de estas áreas de proceso se conforma como un conjunto de buenas prácticas en las actividades relacionadas con un determinado proceso del desarrollo del software. [2]

Entre los beneficios de implantar CMMI se pueden encontrar los siguientes: [5]

- Mejora alineación a objetivos de negocio.
- Mayor eficacia en la detección de errores a lo largo del ciclo de vida de los proyectos del Software, reduciendo drásticamente el número de errores que afecta directamente a los clientes y usuarios.
- Resultados más predecibles en los proyectos.
- Implementación de técnicas proactivas de gestión, mitigando los riesgos que afectan a los proyectos.
- Liberaciones de tensiones, malentendidos, y vacíos de responsabilidad en Proyectos Software.
- Disposición de información de gestión útil a la hora de tomar decisiones ya sean éstas relacionadas con la Gestión de Proyectos Software, bien con la mejora continua del Proceso Software.
- Mejora en la calidad del producto.
- Lograr la satisfacción del cliente.
- Menores costos de desarrollo.
- Mayor rapidez de respuesta.
- Mejora la productividad.

A. Verificación

El propósito de la Verificación (VER) es asegurar que los productos de trabajo seleccionados cumplen sus requerimientos especificados.

El área de proceso de Verificación implica: preparación de la verificación, realización de la verificación e identificación de acciones correctivas. La verificación incluye la verificación del entregable final y de los subproductos parciales frente a todos los requerimientos seleccionados, incluyendo los requisitos del cliente, del artefacto y del componente. En todas las áreas de proceso, donde se usan los términos producto y componente de producto, su significado previsto engloba también a los servicios y a sus componentes. La verificación es inherentemente un proceso incremental, debido a que ocurre durante todo ciclo de vida de desarrollo y de los productos de trabajo, comenzando con la verificación de los requerimientos, progresando a través de la verificación de los entregables parciales según van evolucionando y culminando en la verificación del producto finalizado. La verificación de los productos de trabajo incrementa substancialmente la probabilidad de que el entregable final cumpla con los requerimientos del cliente, del producto y de los componentes del mismo. Las áreas de proceso de Verificación y de Validación son similares, pero tratan aspectos diferentes.

B. Validación

El propósito de Validación (VAL) es demostrar que un producto o componente de producto se ajusta a su uso previsto cuando se sitúa en su entorno previsto.

Las actividades de validación pueden aplicarse a todos los aspectos del producto en cualquiera de sus entornos previstos, tales como los servicios de operación, de formación, de fabricación, de mantenimiento y de soporte. Los métodos empleados para lograr la validación pueden aplicarse a los entregables parciales, así como al producto y sus componentes. (En todas las áreas de proceso, donde se usan los términos de producto y de componente de producto, su significado previsto engloba también a los servicios y a sus componentes). Los productos de trabajo (p. ej., requerimientos, diseños y prototipos) deberían seleccionarse en base a cuáles son los mejores antecedentes acerca de cómo el producto y sus componentes satisfarán las necesidades del usuario, y así la validación se realiza temprana e incrementalmente en todo el ciclo de vida del producto. [3]

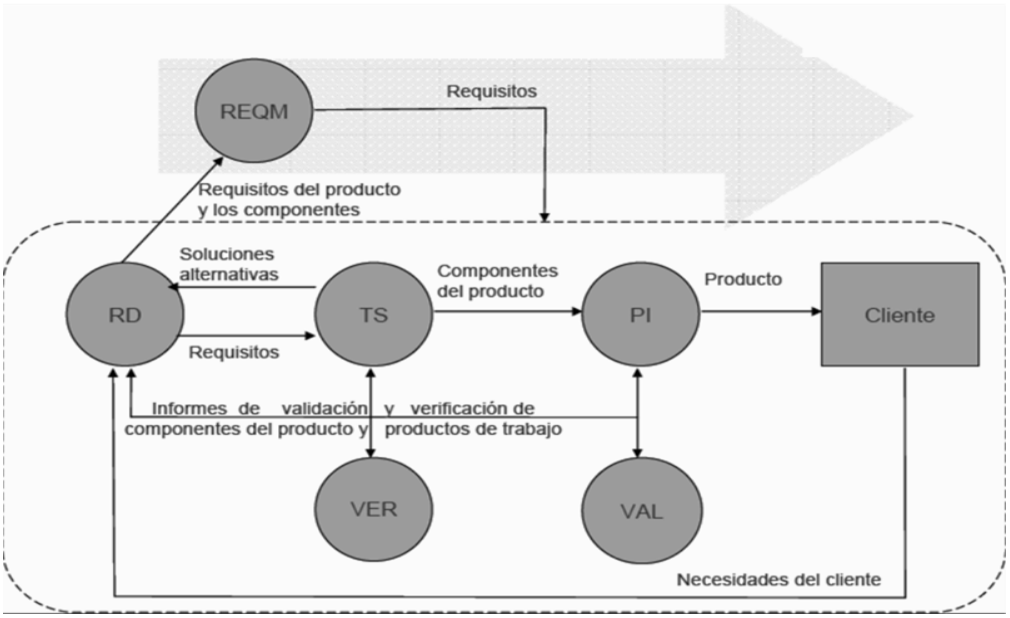

Fig. 2. . Relación entre los procesos de Ingeniería [6]

Las áreas de procesos de verificación y validación se encuentran ubicadas dentro del mapa de procesos de CMMI en la categoría de ingeniería, en la cual además se encuentran otras áreas de proceso que se presentan en la Tabla 1.

La Fig. 2 representa gráficamente la relación de las áreas de proceso de ingeniería donde cabe resaltar la relación que existe entre el proceso de PI y el cliente al cual se le entrega un producto ya terminado para su aprobación, esta categoría recoge las necesidades del cliente a través de los requisitos los cuales se tienen en cuenta a través de todas a las áreas de proceso con el fin de producir lo que el cliente solicito, esta categoría se soporta en el área de verificación con el fin de ir revisando si lo que se está produciendo es lo correcto y al final utiliza la validación para comprobar que lo se construyo es lo que el cliente pidió. Aunque no se muestra en esta figura la categoría de ingeniería se soporta en todas las demás para poder cumplir con la correcta elaboración del producto.

Tabla 1. Áreas de Proceso de la Categoría Ingeniería [Autoría Propia]

Area de Proceso	*Categoria*	*Nivel de Madurez*
Integración del Producto – PI	Ingeniería	3
Requerimientos de Desarrollo – RD	Ingeniería	3
Gestión de Requerimientos – REQM	Ingeniería	2
Soluciones Tecnicas – TS	Ingeniería	3
Validación – VAL	Ingeniería	3
Verificación – VER	Ingeniería	3

C. Beneficios de verificar y validar

El objetivo último del proceso de verificación y validación es establecer la seguridad de que el sistema software esta “hecho para un propósito”. Esto significa que el sistema debe ser lo suficientemente bueno para su uso pretendido. El nivel de confianza requerido depende del propósito del sistema, las expectativas de los usuarios del sistema y el entorno de mercado actual del sistema. Entre los beneficios de implementar estos procesos se pueden encontrar:

- Asegurar que los requerimientos del usuario se cumplan.
- Remover los defectos del producto fuera del ciclo de vida del proyecto, reduciendo el volver a trabajar sobre el mismo aspecto, y reduciendo los costos por la baja calidad.
- Asegurar que las necesidades de los usuarios son comprendidas y asegurar que los productos satisfagan el entorno previsto para el que fueron desarrollados.
- Mejorar la calidad de los procesos y los productos.
- Mejorar la productividad y el rendimiento.

Con base a la metodología "*Metodología al Proceso de Verificación y Validación de Software Basado en el Estándar CMMI"*, se planteara el siguiente modelo.

II. MODELO V&V

El modelo V&V ayuda a las organizaciones a revisar sus procesos de desarrollo basándose en cinco aspectos importantes: Capacitación del personal, Evaluación del producto, Verificación del producto, Validación del producto y Aseguramiento de la calidad del producto. De los cuales solo se describirán a continuación la planificación, las actividades y estrategias de verificación y validación.

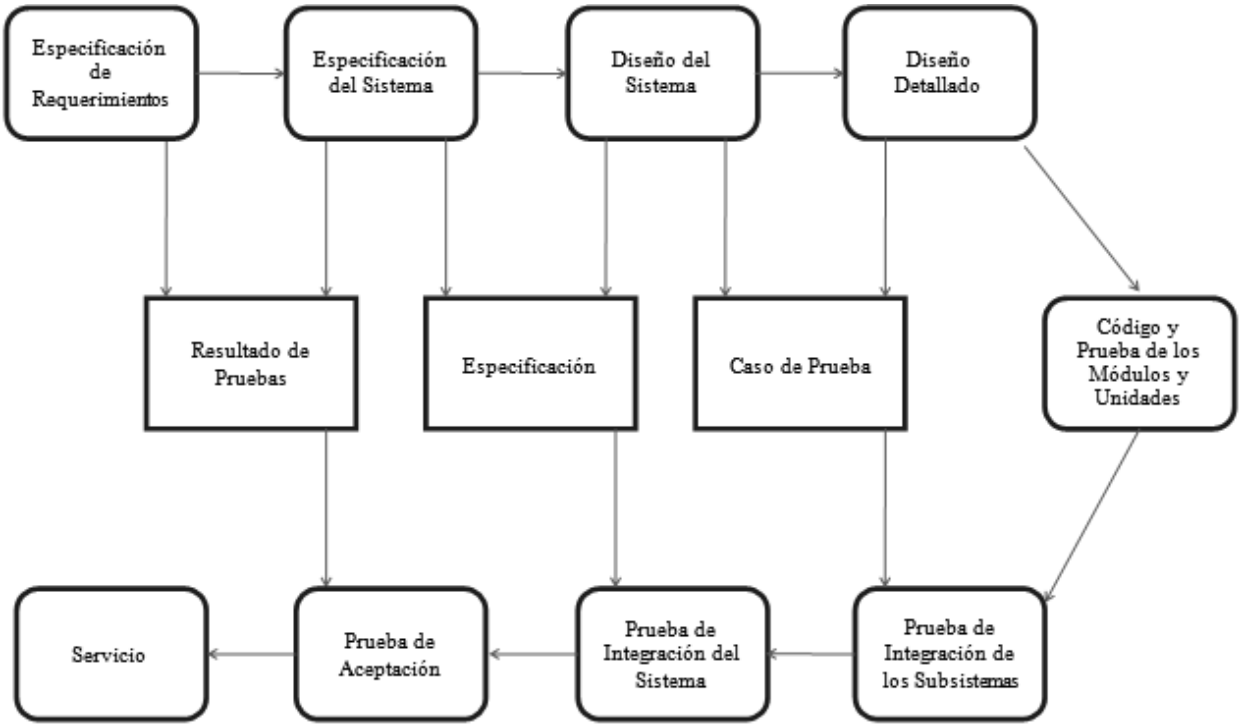

Fig. 3. . Planes de Pruebas como un Enlace entre las Pruebas y el Desarrollo [7]

A. Planificación de v&v

La verificación y validación es un proceso caro. Para algunos sistemas, tales como los sistemas de tiempo real con restricciones no funcionales complejas, más de la mitad del presupuesto para el desarrollo del sistema puede invertirse en V & V. Es necesaria una planificación cuidadosa para obtener el máximo provecho de las inspecciones y pruebas y controlar los costos del proceso de verificación y validación.

Debería comenzarse la planificación de la verificación y la validación del sistema en etapas tempranas del proceso de desarrollo. El modelo de proceso de desarrollo del software mostrado en la Fig. 3 se denomina a veces modelo V. Es una instanciación del modelo genérico en cascada y muestra que los planes de pruebas deberían derivarse de la especificación y diseño del sistema. Este modelo también divide la V& V del sistema en varias etapas. Cada etapa esta conducida por las pruebas que tienen que definirse para comprobar la conformidad del programa con su diseño y especificación.

La planificación de las pruebas está relacionada con el establecimiento de estándares para el proceso de las pruebas, no solo con la descripción de los productos de las pruebas. Los planes de pruebas, además de ayudar a los gestores a asignar recursos y estimar el calendario de la pruebas, son de utilidad para los ingenieros del software implicados en el diseño y la realización de la pruebas del sistema. Estos ayudan al personal técnico a obtener una panorámica general de las pruebas del sistema y ubicar su propio trabajo en este contexto.

B. Actividades de v&v

Utilizando una de las características del proceso unificado de desarrollo de software, que se denomina Guiado/manejado por casos de uso, la cual se presenta gráficamente en la Figura4.

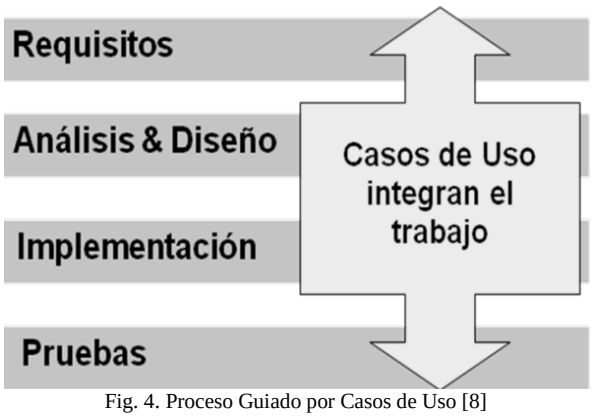

Fig. 4. Proceso Guiado por Casos de Uso [8]

Se pueden diferenciar distintos tipos de actividades para cada una de las etapas del ciclo de vida del software, en la Tabla2 se ejemplifican algunas de ellas para los procesos de verificación y validación.

Tabla 2. Ejemplo de actividades de verificación y validación

		Procesos	
		Verificación	**Validación**
Etapas del ciclo de vida	**Requerimientos**	Inspección 100% de las especificaciones de los requerimientos del sistema y de las especificaciones de los requerimientos del software	Casos de uso; Revisiones del usuario; Prioridades del cliente
	Diseño	Revisión de pares del 100% de los diseños	Decisión análisis y resolución
	Implementación	Inspección 100% de la implementación critica	Simulación
	Pruebas	Confiabilidad/ Estadísticas de pruebas	Aceptación de las pruebas
	Lanzamiento	Verificación de cambios	Usuarios/ clientes retroalimentan las revisiones

C. Estrategias de v&v

Como gran porcentaje (entre un 80 y 90%) de los defectos del software se encuentran en las fases previas a las pruebas, a continuación se presentan algunas estrategias para reducir los mismos.

1. Verificación formal de programas y modelos

La verificación formal de programas consiste en un conjunto de técnicas de comprobación formales que permiten demostrar si un programa funciona correctamente (se valida a través del propio código del programa, a partir de una abstracción o de una representación simbólica), en el que partiendo de un conjunto axiomático, reglas de inferencia y algún lenguaje lógico (como la lógica de primer orden u otra de preferencia una lógica sólida y completa), se puede encontrar una demostración o prueba de corrección de un programa. La lógica de Hoare es un ejemplo de este tipo.

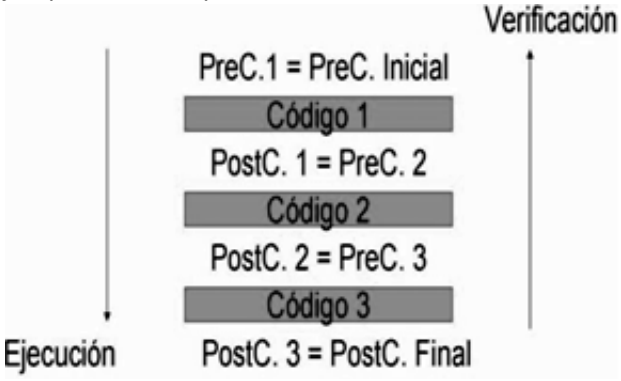

Fig. 5. Un ejemplo de verificación formal [11]

2. Testing

Generalmente, el objetivo de las pruebas del software es convencer a los desarrolladores del

sistema y a los clientes de que el software es lo suficientemente bueno para su uso operacional. La prueba es un proceso que intenta proporcionar confianza en el software.

Un modelo general del proceso de prueba se muestra en la Fig. 6. Los casos de prueba son especificaciones de las entradas para la prueba y la salida esperada del sistema mas una afirmación de lo que se está probando. Los datos de prueba son las entradas que han sido ideadas para probar el sistema. Los datos de prueba a veces pueden generarse automáticamente. La generación automática de casos de prueba es imposible. Las salidas de las pruebas solo pueden predecirse por personas que comprenden lo que debería hacer el sistema.

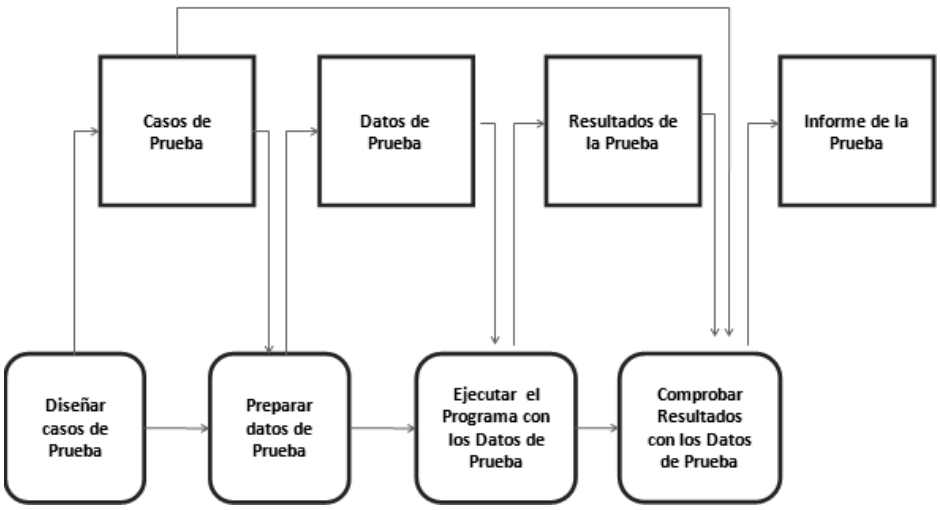

Fig. 6. Un Modelo de Proceso de Pruebas del Software [7]

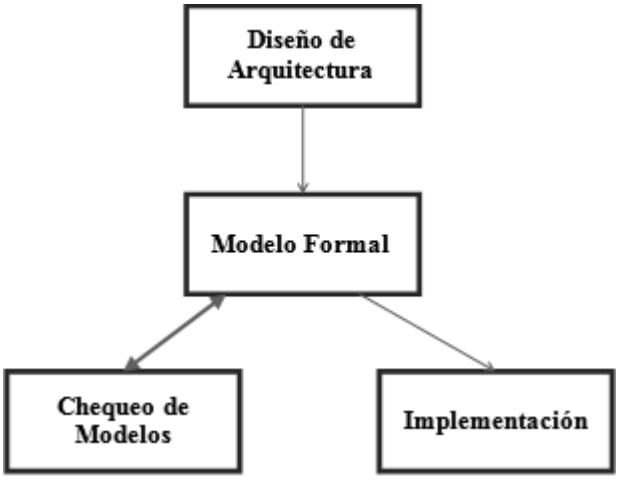

Fig. 7. Un ejemplo de modelchecking [12]

Las pruebas exhaustivas, en las que cada posible secuencia de ejecución del programa es probada, son imposibles. Las pruebas, por lo tanto, tiene que basarse en un subconjunto de posibles casos de prueba. Idealmente, algunas compañías deberían tener políticas para elegir este subconjunto en lugar de dejar solo al equipo de desarrollo. Estas políticas podrían basarse en políticas generales de pruebas, tal como una política en la que todas las sentencias de los programas deberían ejecutarse al menos una vez.

3. Chequeo de modelos (modelchecking)

Es un método para verificar formalmente estados-finitos de sistemas de forma simultánea. Especificaciones sobre el sistema se expresan como fórmulas de lógica temporal, y se utilizan algoritmos simbólicos eficientes para atravesar el modelo definido por el sistema y comprobar si la especificación se mantiene o no. Espacios-estado muy grandes a menudo se pueden recorrer en minutos. Esta técnica ha sido aplicada a muchos sistemas complejos como los encontrados en los futurebus+ y los protocolos de los buses locales PCI.

III. CONCLUSIONES

Después de analizar el modelo CMMI se puede concluir que presenta una base sólida del concepto de calidad y de la forma como se aplica al desarrollo de software. Se puede pensar en la necesidad actual que tiene la industria de software mediante la aplicación de estándares y modelos que permiten mejorar los servicios y productos.

El CMMI es un modelo de calidad basado en procesos, que ayuda a las organizaciones a optimizar los mismos, se desarrolla en un contexto internacional y permite a las empresas nacionales adquirir un importante factor de competitividad y eficiencia. Lo anterior es de vital importancia cuando se tiene el interés particular de internacionalizar los productos y servicios de software.

Los procesos de verificación y validación son de gran importancia y beneficio para las organizaciones que los implementen correctamente cuando se está desarrollando un producto ya que garantizan el aseguramiento de que los requerimientos del cliente están siendo cumplidos y que a además el producto está enmarcado dentro de los estándares de calidad que la organización ofrece, todo ello repercute en menores tiempos de entrega, disminución de costos y detención temprana de errores.

Los planes, actividades y estrategias presentadas en el modelo, sirven como base para el establecimiento de políticas en las organizaciones con el fin de mejorar y asegurar a través del uso de las mejores prácticas de CMMI los procesos de verificación y validación.

IV. REFERENCIAS

[1] CMMI Product Team. CMMI® for Development, Version 1.2.2006, Pág. 561.

[2] CMMI, [visitado: 30 de septiembre de 2009] http://3ecubo.es/articulos/41-articulos/72-cmmi.html?d959f61ad4cb28aaf91605cce54f7093=cb47788b6d9fd407f9ed22787dd60c3f

[3] Cátedra de Mejora de Procesos de Software en el Espacio Iberoamericano de la Universidad Politécnica de Madrid, CMMI® Guía para la integración de procesos y la mejora de productos, 2da ed. 2009 Pág.630

[4] Modelos de Procesos de Software [Visitado: 16 de Mayo de 2009]http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/ModelosProcesoSoftware.pdf

[5] Qualitrain - Beneficios del proceso de software bajo el modelo CMMI (Segunda parte) [Visitado: 30 de septiembre de 2009]http://www.qualitrain.com.mx/index.php/Procesos-de-software/Beneficios-del-proceso-de-software-bajo-el-modelo-CMMI-Segunda-parte/Page-4.html

[6] C. RiogoniBrualla. CMMI®: Mejora del Proceso en Fabricas de Software [Visitado: 10 de Septiembre 2009] http://www.mityc.es/dgdsi/es-ES/Servicios/Biblioteca%20Jornadas/Jornadas/s01CeciliaRigoni.pdf

[7] I. Sommerville. Ingeniería del Software, 7ma ed. Prentice Hall. 2005, Pág. 687

[8] P. Letelier Torres. Introducción a RationalUnifiedProcess (RUP). Departamento Sistemas Informáticos y Computación (DSIC), Universidad Politécnica de Valencia (UPV). [Visitado: 10 de Septiembre de 2009] http://www.dsic.upv.es/~letelier/pub/p16.ppt

[9] R. Pressman. Ingeniería del Software – Un Enfoque Práctico, 5ta ed. Mc Graw Hill.2002, Pág. 640.

[10] Modelos de Gestión de la Calidad del Software. [Visitado: 30 de septiembre de 2009] http://modelosdegestiondelacalidad.blogspot.com/2008/01/modelo-cmmi.html

[11] Sitio Web: http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=Verificaci%C3%B3n_de_programas:_Pruebas_Unitarias_y_JUnit [Visitado: 15 de octubre de 2009]

[12] Sitio Web: http://users.tkk.fi/rshursti/concurrent/x28.html [Visitado: 15 de octubre de 2009]