ASSE 2015, 16º Simposio Argentino de Ingeniería de Software.

Marco para evaluar garantía en desarrollos de software

Pedro E. Colla

Maestría en Sistemas Embebidos – Instituto Universitario Aeronáutico
Av. Fuerza Aérea Km 8 ½ - (5000) Ciudad de Córdoba – Córdoba - Argentina (pcolla@iua.edu.ar)

Resumen. Este artículo expone un marco simplificado para estudiar los costos de garantía en el desarrollo de software. Se proponen métodos para obtener los parámetros requeridos por los modelos de confiabilidad citados en la bibliografía desde métricas de proceso comúnmente encontradas en la línea de base de organizaciones desarrolladoras. El marco propuesto es validado mediante técnicas de simulación por el método de Montecarlo para explorar la magnitud de los resultados y la sensibilidad a los parámetros utilizados. Se extraen conclusiones preliminares y se identifican líneas de trabajo futuras.

Palabras Claves. Garantía de software, proceso de desarrollo de software, SEI-CMMI™

1 Introducción

Se espera de las organizaciones desarrolladoras de software que para ser competitivas entreguen productos en tiempo y dentro de presupuesto, además de libres de falla, a sus clientes. Incluso cuando la aplicación sea compleja. Estos son requisitos que hasta hace no mucho tiempo no iban más allá de las buenas intenciones en la industria como práctica generalizada pero que actualmente satisfacerlos puede ser la diferencia para una organización entre ser exitosa y no serlo.

El ciclo de vida para el desarrollo de software (SDLC por sus siglas en inglés *software development life cycle*) debe por lo tanto incluir actividades distribuidas en *fases* o *etapas* tendientes a alcanzar estos objetivos. A pesar de los avances significativos en la tecnología de desarrollo, y el estado del arte en las disciplinas de ingeniería de software, el potencial para introducir defectos durante el ciclo de vida sigue siendo importante. Puesto que se puede afirmar que en el presente estado del arte no es posible producir software completamente libre de defectos, el re trabajo resultante para solucionarlos junto con la pobre gestión de cambios suelen ser las causas raíz de retrasos en las entregas y excesos de presupuesto en los proyectos.

Los clientes, por su parte, demandan que como parte de los servicios provistos se incluya el compromiso para garantizar el software entregado a partir de solucionar los defectos encontrados durante su uso durante un tiempo acordado, denominado *período de garantía*. La creciente integración de sistemas embebidos en bienes industriales y de consumo, los que usualmente requieren proporcionar garantías como parte de su estructura de comercialización, profundizan la necesidad proveer un marco para la comprensión sobre como proveer garantías sobre el software que contienen. La garantía a proveer puede incluir desde la reparación a costo del proveedor del defecto hasta el reconocimiento de multas que bajo determinadas condiciones reparen al cliente de los impactos en su negocio que las fallas pudieran ocasionar.

Para implementar estos mecanismos los proveedores de software, por su parte, enfrentan una situación de mucha competencia que les impide simplemente agregar una estimación razonable de los costos de garantía en que incurrirán; la matriz financiera de la cuestión favorece abordar las causas raíz. Por lo tanto es necesario entregar los productos de software con la menor cantidad posible de defectos que pudieran aparecer durante el período de garantía.

Para abordar la cuestión del pronóstico de los defectos Musa et.al (Musa, 1987) afirma que la respuesta tradicional ha sido que se debe realizar un proceso de *validación y verificación* (test) exhaustivo de los componentes del software contra un conjunto de requerimientos dado. Esta postura no aborda la cuestión fundamental sobre el esfuerzo que demandará y por cuanto tiempo, ambos elementos cruciales para determinar la viabilidad económica de hacerlo.

La bibliografía expone modelos estadísticos para estudiar este problema que son teóricamente robustos y con significativa validación empírica Tal et. al. (Tal, 2002). Estos proveen herramientas para permitir el planeamiento y monitoreo de los procesos de test hasta que se alcanzan determinados objetivos de defectos remanentes al momento de la liberación que satisfagan un razonable equilibrio entre los objetivos técnicos de prestaciones y confiabilidad, así como las metas financieras del proyecto.

Se han presentado en tal sentido estudios que permiten establecer criterios de liberación a producción para balancear el tiempo y esfuerzo dedicado al test y la confiabilidad resultante del software luego de su liberación. Okumoto et al. (Okumoto, 1980) estudió el problema del tiempo óptimo de liberación bajo restricciones de costo y confiabilidad. Yamada et al. (Yamada, 1987) sugiere criterios para establecer el óptimo de liberación considerando costos de garantía y requisitos de confiabilidad. Yang et al. (Yang, 2000) estudió el perfil de confiabilidad operativo en modelos de evaluación de software. Tal et al. (Tal, 2002) propuso criterios estadísticos que optimizan la confiabilidad del software liberado. Jain et al. (Jain, 2001) investigó consideraciones de modelos híbridos para la predicción de costos totales de garantía a ser reservados incluyendo efectos del valor tiempo del dinero. Popstojanova et al (Popstojanova, 2001) estudiaron las consideraciones de arquitectura a ser tenidas en cuenta al momento de evaluar la confiabilidad de un sistema basado en software. Yamada et al. (Yamada, 1993) propone estrategias de implementación de liberación óptima de acuerdo al ciclo de vida y la tasa de descuento financiera del proyecto. Prince Williams et al. (Williams, 2007) estudió el problema de estrategias de prueba y liberación óptimas para garantías por tiempos específicos. Pham (Pham, 2003) propone el modelado del costo total de producción de software bajo garantía teniendo en cuenta imperfecciones en el proceso de test y corrección de los defectos durante el ciclo de vida para distintos modelos de ciclo de vida y penalidades. Xie et al. (Xie, 2003) realizó aportes sobre los efectos de la depuración imperfecta de defectos y su impacto en el costo de desarrollo, lo que a su vez afecta la determinación de la estrategia óptima de liberación. Bhaskar (Bhaskar, 2006) realiza aportes sobre el modelado de costos basados en la criticidad de la falla y el costo de su ocurrencia durante diferentes fases del desarrollo en una estrategia de liberación consistente en sucesivas *liberaciones* (*releases* en inglés).

Rinsaka (Rinsaka & Dohi, 2005) estudia el problema de determinar el período óptimo de garantía bajo diferentes circunstancias operacionales. Lai (Lai, 2011) aporta un estudio de cuando liberar el software desde la perspectiva de los modelos de confiabilidad. Por su parte Bohun et.al (Bohun, 2004) realiza aportes que permiten estudiar la optimización conjunta de los costos asociados a calidad y garantía a partir de satisfacer los componentes de un vector de requisitos de calidad dados.

Por su parte es un hecho conocido en la industria (Westland,J.C., 2002) que cualquier estrategia basada en contener los defectos una vez que el aplicativo ha sido liberado a producción no solo creará riesgos para la cadena de valor que soporte, sino que enfrentará costos para la detección y remoción de los defectos que pueden ser órdenes de magnitud superiores a que fueran removidos en su ambiente original de desarrollo.

Sin embargo, los modelos aplicados por los investigadores tienen dificultades en su aplicación práctica pues el óptimo para la liberación de un software está fuertemente

influenciado por decisiones pragmáticas relacionadas con el calendario comprometido, necesidades del negocio subyacente y factores presupuestarios antes que la evaluación objetiva de la calidad del software bajo desarrollo. Los modelos propuestos requieren para su cálculo, además, valores de parámetros que no siempre las organizaciones tienen facilidad para calcular a partir de las métricas habituales de gestión.
Debido a este factor en no pocas ocasiones la estrategia del cálculo luce lo suficientemente engorrosa que se opta por simplemente realizar el test hasta que el calendario lo permita, lo que no es sorprendente que produzca resultados poco satisfactorios.
Una estrategia racional para abordar esta cuestión consiste en modelar el problema de forma de obtener resultados cualitativos compatibles con la realidad y cuantitativos con una aproximación razonable a partir del uso de métricas organizacionales obtenidas durante la gestión del proyecto o provenientes de una *línea de base organizacional* (*baseline* en inglés) de métricas históricas. De esta manera se puede proyectar que cantidad de defectos futuros esperar, lo que permite planear cuales son las implicancias financieras y técnicas que será necesario afrontar durante el período de garantía pactado, asegurándose que las provisiones necesarias no restan competitividad al proveedor.
Por otra parte, e independientemente de la flexibilidad en el momento de la liberación, es necesario comprender qué factores del proceso de desarrollo es necesario gestionar para poder satisfacer requerimientos de garantía en el mercado que se haya elegido servir; de tal manera que se logre realizar estas entregas en forma rentable para la organización proveedora a partir de sus propias capacidades operativas.
La contribución de éste artículo es intentar integrar las distintas fuentes citadas con una perspectiva derivada de la experiencia que permita estudiar algunas cuestiones de estrategia en base a modelos simples y mediciones usualmente disponibles durante la gestión de un proyecto de desarrollo. Las preguntas de investigación pueden entonces establecerse como:

- *¿Cuál es el perfil de garantía en el software que produce una organización que puede afrontar en forma sustentable a partir de sus parámetros de calidad de desarrollo?*
- *¿Qué influencia tiene la complejidad del producto bajo desarrollo y cuál es su influencia en las características de la garantía a ser provista?*
- *¿Cuál es la relación entre el momento de liberación y los costos de totales incluyendo garantía? ¿Que posibles equilibrios pueden establecerse para ambos en función de las capacidades que tiene la organización?*

2 Modelos de Confiabilidad de Software

En los modelos propuestos por la literatura para pronosticar la mejora en la confiabilidad la experiencia empírica muestra que cualquier mejora en la capacidad de pronóstico resulta neutralizada por la inherente variabilidad de los distintos parámetros utilizados para su cálculo según se derivan de las métricas históricas encontradas en diferentes operaciones reales. Por lo tanto los modelos más simples tienden a dar explicaciones cualitativas y cuantitativas sorprendentemente ajustadas cuando son calibrados con métricas propias de la organización que las utiliza.
Todos los modelos pronostican que los defectos son expuestos como *errores* o *fallas* a medida que el software es ejecutado. El número remanente de defectos se reducirá en el tiempo a medida que ocurran correcciones. Esto no necesariamente es así, puesto que

las correcciones y cambios introducirán por su parte defectos. Sin embargo, en tiempos cortos como los involucrados en los períodos de garantía bajo consideración (semanas a pocos meses) se puede aceptar la simplificación que éste efecto no introduce cambios significativos en la matriz de comportamiento bajo estudio.
Los modelos de confiabilidad de software expresan el tiempo no en términos de calendario sino de *tiempo continuo de ejecución* (τ), el que tiene en cuenta las discontinuidades producidas por las actividades de corrección. Si bien en un período breve al comienzo del proceso de test el tiempo efectivo está limitado por las correcciones necesarias, la utilización de *tiempo calendario* es suficientemente práctica y sencilla de medir como para resultar ventajosa su adopción respecto a las distorsiones bajas que introduce al observar la totalidad del ciclo de test. Se asume que en la medida que las fallas permiten observar defectos se introducen correcciones que permiten hacer desaparecer los mismos. Asumiendo despreciable la introducción de nuevos defectos en este proceso el *número de defectos expuestos acumulados* o *perfil de defectos* (m) para un tiempo arbitrario (τ) desde el comienzo del proceso de test puede ser estimado por la siguiente expresión (Ec 1) (Musa, 1987):

$$m(\tau) = \mu_0\left(1 - e^{-\lambda_0 \tau}\right)$$

Ec 1

Donde la *cantidad total de defectos inyectados* (μ_0) y la *intensidad de detección de defectos* (λ_0) pueden ser obtenidas tanto por regresiones lineales simples de secuencias modestas de datos obtenidas en forma temprana durante el test así como de información histórica de la organización. En función de éste modelo es posible pronosticar los defectos detectados (y removidos) en cualquier tiempo arbitrario de ejecución del software.
Los modelos de perfiles de defectos no contienen hipótesis sobre el proceso utilizado para realizar el desarrollo. Sin embargo, las organizaciones de desarrollo pueden aprovechar las métricas históricas en sus líneas de base organizacionales para calibrar estos modelos y adquirir una perspectiva más amplia para comprender y pronosticar el comportamiento de sus proyectos.
Prácticamente cualquier organización con métricas básicas medirá el tamaño del software (S), con metodologías que capturen una buena correlación con la complejidad (Hummel & Burger, 2013). Típicamente se utilizará con propósitos de gestión la *productividad* (π) con la que opera bajo diferentes tecnologías y ambientes. El *esfuerzo total del proyecto* (E) puede entonces expresarse como (Ec 2):

$$E = \pi S^{\gamma} \cong \pi S$$

Ec 2

Donde el *factor de aprendizaje (γ)* suele ser usualmente muy cercano a la unidad si se acotan suficientemente las tecnologías utilizadas y contando con experiencia en ellas.
Se tomará en el presente artículo el concepto de costo como siendo intercambiable con el de esfuerzo, puesto que a todos los efectos prácticos será su principal componente.
Es relativamente simple para las organizaciones productoras de software determinar qué habilidad tienen en detectar defectos antes de liberar el software en base a colectar los reportes de *defectos escapados luego de una liberación* (μ_r) comparados con los *detectados durante el proceso de desarrollo* (μ_d); a esta capacidad suele definírsela

como *contención de defectos en fase* (PCE por las siglas en inglés *phase containment of errors*) el que es definido como (Ec 3):

$$PCE = \frac{\mu_d}{\mu_r + \mu_d}$$

Ec 3

Esta misma información puede relacionarse con el tamaño o complejidad de la aplicación, expresada en una métrica apropiada como por ejemplo *puntos de función* (Matson, Barrett, & Mellichamp, 1994) para determinar la *densidad de defectos al liberar* (δ_r) que puede expresarse como (Ec 4):

$$\delta_r = \frac{\mu_r}{S}$$

Ec 4

Las [Ec 3] y [Ec 4] pueden combinarse para estimar el *número total de defectos inyectados* (μ_0) mediante la relación (
Ec **5**):

$$\mu_0 = \frac{S\delta_r}{PCE}$$

Ec 5

Es usual observar que PCE sea razonablemente estable o que pueda ser gestionado dentro de rangos controlados para los proyectos de una misma organización que utilicen procesos y tecnologías similares, estando determinado mayormente por el proceso de calidad utilizado. Por su parte la densidad de defectos liberados también suele ser un factor razonablemente estable y susceptible de ser controlado por técnicas de gestión para un dado marco tecnológico y de proceso del desarrollo.

El *tiempo total de desarrollo del proyecto* (τ_0) puede obtenerse a partir del esfuerzo total del proyecto por una relación del tipo dado por la (Ec 6) (Walston, 1977).

$$\tau_0 = KE^{\beta}$$

Ec 6

Donde las constantes de *eficiencia de calendario* (K) y *factor de aprendizaje* (β) se obtienen por calibración con información histórica de cada organización.

El *tiempo total de test al momento de la liberación* (τ_r) se observa en la práctica que guarda usualmente relación con el tiempo total del proyecto de tipo, la que puede expresarse como (Ec 7):

$$\tau_r = \upsilon\tau_0$$

Ec 7

Donde la *proporción de tiempo de test* (υ) tiende a ser razonablemente constante para proyectos de similar complejidad por lo que se puede establecer valores y referencias basados en la historia. Combinando las ecuaciones (Ec 2 y Ec 6) se puede establecer el tiempo medio de test en función del tamaño/complejidad del desarrollo (Ec 8)

$$\tau_r = \nu K(\pi S)^{\beta}$$

Ec 8

Es decir, se puede estimar el tiempo medio de test esperado dado el tamaño del proyecto y con anclaje en las métricas históricas; este tiempo, instanciado a cada proyecto en particular, está sujeto a cierta flexibilidad producto de la gestión y termina siendo un factor decisivo al momento de establecer la habilidad que el resultado del desarrollo pueda satisfacer las necesidades de garantía. En efecto, el *número de defectos al finalizar el test* (μ_r) estará determinado por (Ec 1) por lo que puede expresarse como:

$$\mu_r = \mu_0(1 - e^{-\tau_r \lambda_0})$$

Ec 9

Juntando las (Ec 3y Ec 8) es posible escribir (Ec 10):

$$\lambda_0 = -\frac{1}{\tau_r}\ln[1 - PCE]$$

Ec 10

Con lo que se completa la caracterización del comportamiento de defectos en base a un modelo simple calibrado con datos históricos organizacionales o de gestión del proyecto.

3 Evaluación de costos de proceso de garantía

En la sección precedente el tiempo de test (τ_r) se establece como un parámetro organizacional obtenido a partir de series históricas que se asumen representativas y es determinado fundamentalmente por el tamaño del proyecto. Al hacerlo se asume que las decisiones sobre los proyectos en el pasado han contribuido a determinar la capacidad organizacional de establecer resultados en términos de su capacidad para inyectar, detectar y liberar defectos.

Sin embargo, en la práctica el tiempo de test es una decisión de gestión en cada proyecto; el equilibrio que propone el modelo simple es reducir costos reduciendo el tiempo de test a costa de dejar más errores latentes en el software liberado; o por el contrario reducir los errores latentes extendiendo el tiempo de test. A los efectos de la evaluación el tiempo de test será considerado un atributo de calidad sujeto a gestión y por lo tanto será uno de los factores a estudiar en términos de identificar como optimizar el costo total.

Al efecto el costo total de garantía estará relacionado en el costo de realizar el test de la aplicación hasta un punto en el que es liberado, para posteriormente sostener los términos de la garantía durante el período convenido; siguiendo el modelado propuesto por Rinsaka (Rinsaka & Dohi, 2005) utilizando la distribución de Rayleigh propuesta por Goel.

Por lo tanto para estimar el *costo de test* (C) es necesario identificar los *atributos de calidad (q)* que contribuyen al mismo, puede estimarse como que el mismo estará dado por la (Ec 11):

$$C(q) = C_0 + C_1\tau_r^{\alpha} + C_2\mu_r$$

Ec 11

Siendo los términos el *costo fijo de establecimiento del ambiente de test* (C_0), el *costo por unidad de tiempo de ejecución del test* (C_1) y *el tiempo total de test* (τ_r) como va-

riable sujeta a gestión y afectado por un *factor de aprendizaje* (α). Finalmente se incluye el costo de remover los defectos encontrados, el que resultará del *costo promedio de remoción* (C_2) y del número pronosticado de *defectos a la finalización del test* (μ_r) que dependerá del tiempo de test según la relación dada por (Ec 1) discutida previamente. El costo de solución por defecto puede ser calculado para un proyecto en particular o como línea de base histórica como la relación promedio histórica entre el esfuerzo utilizado en re trabajo de defectos, denominado en ocasiones *costo de calidad pobre* (CoPQ por sus siglas en inglés cost of poor quality), y el número de defectos encontrados hasta el momento de la liberación para ese proyecto. La proporción entre el CoPQ y el esfuerzo total del proyecto es usualmente una magnitud organizacional sujeta a gestión y que se puede controlar dentro de límites acotados mediante intervenciones en el proceso de desarrollo utilizado de forma de mejorar su madurez y las capacidades organizacionales de la organización (Knox, 1993).
Por su parte la bibliografía (Westland,J.C., 2002) y la experiencia empírica muestran que la relación entre el esfuerzo de corregir un defecto una vez en ambiente productivo respecto de hacerlo en el ambiente de desarrollo es un multiplicador, que denominaremos *complejidad de producción* (κ). Esta relación se origina en la mayor complejidad de acceso, necesidad de reproducir el contexto del defecto y procedimientos de puesta en producción mediante control de cambios. Por eso se considerará que el esfuerzo de corregir los defectos durante el período productivo (C_w) estará dado por (Ec 12):

$$C_w = \kappa C_2$$

Ec 12

La cantidad de defectos (μ_w) que se pronostica detectar para un tiempo de garantía (τ_w) dado será utilizando (Ec 1):

$$\mu_w = m(\tau_w + \tau_r) - m(\tau_r)$$

Ec 13

El costo de garantía W(q) resultará entonces como (Ec 14):

$$W(q) = \kappa C_2 \mu_w$$

Ec 14

4 Modelado de Costo total

Para optimizar el *costo total* (C_t) es necesario encontrar el mínimo combinado para los costos de test (C(q)) y costo de garantía (W(q)) para un dado conjunto de especificaciones de calidad (q), es decir:

$$C_t = \min_q (C(q) + W(q))$$

Ec 15

A los efectos de este planteo propuesto del problema se considerará como especificaciones de calidad (q) los valores de *complejidad total del aplicativo* (S), *densidad de defectos inyectados por el proceso de desarrollo* (δ_0), la *efectividad de contener defectos antes de la liberación* (PCE), el *tiempo total de test* (τ_r) y el *tiempo de garantía a ser cubierto* (τ_w), es decir:

$$q = \{S, PCE, \delta_0, \tau_w, \tau_r\}$$

Estas especificaciones permiten capturar los principales parámetros de proceso y de gestión financiera del proyecto como para poder evaluar su influencia. El costo total mínimo ocurrirá para el tiempo de liberación τ_r tal que se verifiquen:

$$\frac{\partial C_t(\tau_r)}{\partial \tau_r} = 0 \qquad \frac{\partial^2 C_t(\tau_r)}{\partial \tau^2{}_r} > 0$$

Ec 16

5 Método de investigación

El modelado discutido en las secciones previas has sido validado parcialmente mediante su empleo práctico en la gestión cuantitativa de diferentes proyectos piloto. Los mismos configuran una línea base de distribuciones para los diferentes valores de interés. El número de proyectos involucrados excede una docena en condiciones competitivas de producción de software por lo que las distribuciones se asumen representativas.

Variable	Símbolo	U.M.	Min	Med	Máx
Tamaño/Complejidad de Código	S	PF	10	100	250
Contención de Defectos en Fase	PCE	%	0,5	0,8	0,95
Densidad Defectos (Inicial)	δ_0	Defectos/PF	0,5	1	10
Complejidad ambiente producción	κ		7	8	10
Período de Garantía	τ_w	Meses	0.5	1	6
Relación tiempo test/proyecto	ν		0.1	0.2	0.4
Productividad Desarrollo	π	Horas/PF	8	15	25
Los otros valores utilizados para establecer la simulación fueron α=1.05, C_0=1 Staff/Mes,C_1=2,09,C_2=0,01,K=0.66 ,β=0.5. PF=*Puntos de Función*					

Figura 1 Tabla de parámetros organizacionales utilizados en el modelado

El resultado de la aplicación de las ecuaciones precedentes (Ec 15 y Ec 16) es susceptible de ser resuelta por métodos analíticos para un determinado conjunto de valores de los parámetros; pero la utilidad de esta solución es limitada debido a la naturaleza simplificada de los modelos utilizados y a la dispersión de los parámetros incluso dentro de una misma organización incluso siendo los mismos estadísticamente estables y los procesos involucrados considerados estadísticamente capaces.

Entonces el modelado sistémico es verificado y validado mediante técnicas de simulación de forma de estudiar la magnitud de los resultados que pueden ser esperados al utilizar parámetros con rangos de valores obtenidos de líneas de base de métricas reales reportadas por organizaciones, datos de la industria y estimaciones basadas en juicio de experto teniendo en cuenta las dispersiones esperables en los mismos para proyectos típicos. La tabla (Figura 1) da los valores utilizados para los distintos parámetros; cada organización puede reemplazarlos con otros que surjan de sus propias líneas de base de métricas sin que el modelo pierda generalidad.

La validación realizada muestra que los resultados son consistentes con la experiencia práctica y por lo tanto pueden ser utilizados para realizar análisis y extraer conclusiones en las cuestiones bajo investigación.

Para tener en cuenta los rangos de valores posibles en los distintos parámetros se recurre a la utilización de simulaciones utilizando el método de Montecarlo, lo que por su parte permite estudiar las relaciones y sensibilidad del resultado a las distintas variables. Se

adoptan distribuciones triangulares (Sargent, 1998) para las distintas variables de entrada reflejando valores mínimos, medios y máximos; esta distribución es la indicada como estrategia de modelado cuando no hay una distribución claramente definida.

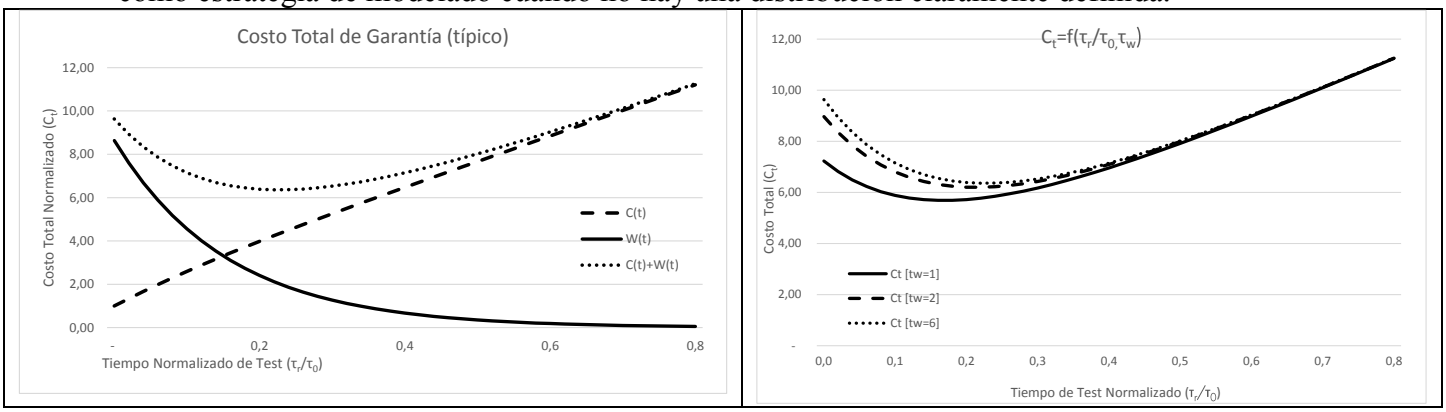

Figura 2 Costo total de garantía (típico) en función del tiempo de test normalizado

Como variable objetivo de la simulación se utiliza el tiempo de test que minimiza el costo total para una determinada constelación de parámetros.
Ese se expresa por comodidad y generalidad en forma *normalizada al tiempo total del proyecto* (τ_r/τ_0) de forma de independizar las magnitudes absolutas obtenidas del tamaño o la complejidad de cada proyecto en especial. El resultado del cálculo con un conjunto de parámetros dentro de los rangos simulados (Figura 2 izq) muestra el balance entre los costos de test y de provisión de garantía para diferentes duraciones del esfuerzo de test así como el óptimo tiempo de test que determina el costo mínimo. Por su parte, para una dada configuración de parámetros es posible ver (Figura 2 der) como la extensión del período de garantía modifica el tiempo de test necesario para obtener el costo óptimo.
Para permitir que la simulación incluya un número significativo de casos se utilizan corridas de simulación con 5000 intentos, la experiencia muestra un adecuado balance entre la convergencia de los resultados y el tiempo de ejecución con esta configuración. Para capturar las relaciones de sensibilidad mutua entre las distintas variables y los efectos de dispersión de los parámetros se presenta el resultado de una corrida típica donde es posible observar la interacción entre los diferentes factores de costo de forma de identificar el costo óptimo y los rangos para los que estos ocurren.

6 Influencia de las variables

Realizada la simulación se obtiene la distribución esperada y la sensibilidad a los diferentes parámetros constitutivos de la especificación de calidad adoptada cuyos resultados pueden observarse en la Figura 3 (izq), la distribución de posibles valores de tiempo de test para obtener el costo óptimo puede verse también en la Figura 3 (der).
El parámetro que más influencia el resultado es la densidad inicial de defectos (δ_0), este factor está fuertemente determinado por las prácticas de ingeniería de software utilizadas para el desarrollo; en particular gestión de requerimientos, aplicación de mecanismos de test estático (inspecciones) y la utilización de técnicas de patrones y reutilización en la construcción de código.

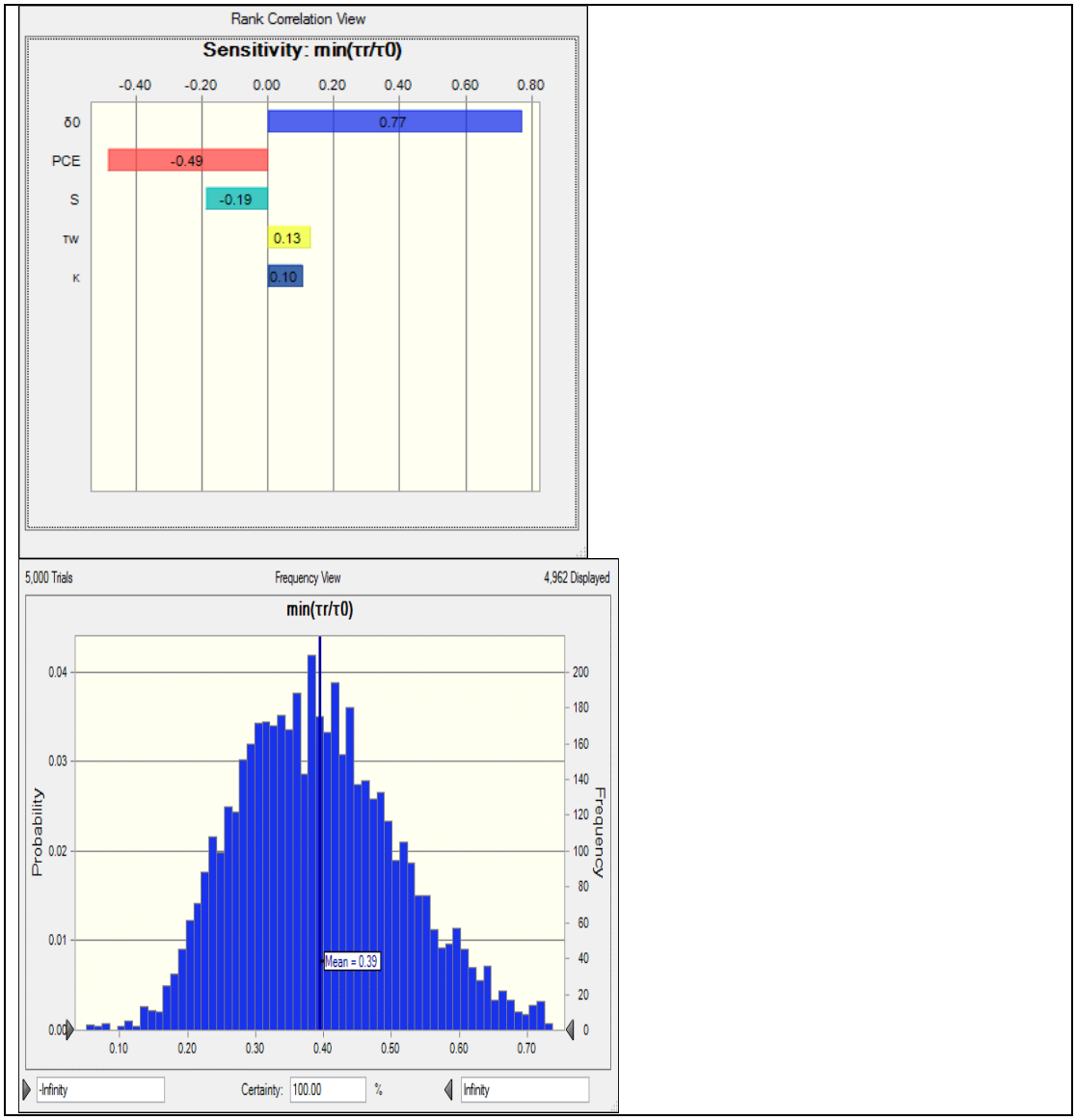

Figura 3 Sensibilidad y distribución resultante (τ_r/τ_0) para simulación típica

En segundo orden de importancia en cuanto a sensibilidad del resultado aparece la contención de defectos durante el test (PCE) que mide de alguna manera la eficacia del proceso de testeo y está determinado por la utilización de técnicas de cobertura, prácticas de gestión y metodología de test.

Es decir, los dos principales componentes que definen la capacidad de la organización para obtener resultados óptimos en cuanto a la provisión de la garantía requerida en su proceso de test están determinados fuertemente por las características del proceso y la tecnología utilizados para desarrollar el software, resultado que de ninguna forma es sorprendente y debería ser la base conceptual que facilite la realización de inversiones en los sistemas de calidad utilizados.

El tamaño/complejidad (S) del proyecto desarrollado tiene influencia en la determinación del tiempo óptimo de test sugiriendo la conveniencia de realizar componentes de software más pequeños. Por su parte el período de garantía (t_w) define en parte la necesidad de extender el período de test para capturar la mayor cantidad de defectos antes de la liberación, aunque la sensibilidad del resultado a su valor resulta sorpresivamente pequeña. Finalmente, la magnitud de la complejidad del ambiente productivo juega un

rol en la determinación del costo óptimo pero su valor sugiere que las conclusiones de la simulación se mantienen sobre una gama de variación muy amplia de este parámetro.

7 Conclusiones

- *¿Cuál es el perfil de garantía en el software que produce una organización puede afrontar en forma sustentable a partir de sus parámetros de calidad de desarrollo?*
  Los principales determinantes son la densidad original de defectos (δ_0) y la contención de defectos en desarrollo (PCE) los que para un dado contexto tecnológico y organizacional pueden optimizarse mediante la adopción de procesos maduros de calidad, mejores prácticas de desarrollo, herramientas de soporte al ciclo de vida y reutilización de componentes.
- *¿Qué influencia tiene la complejidad del producto bajo desarrollo y cuál es su influencia en las características de la garantía a ser provista?*
  La influencia muestra ser moderada, se visualiza la conveniencia de abordar la construcción utilizando los componentes de la menor complejidad posible; indirectamente alienta el reutilización de componentes desarrollados (y testeados) previamente.
- *¿Cuál es la relación entre el momento de liberación y los costos de totales incluyendo garantía, que posibles equilibrios pueden establecerse para ambos en función de las capacidades que tiene la organización?*
  Se identifica un tiempo óptimo de liberación para cada combinación de los requisitos de calidad, siendo uno de ellos el tiempo de garantía (t_w). Dados el resto de los factores constantes la duración de la garantía incrementa el tiempo necesario de test para un costo óptimo.

8 Trabajo Futuro

Un número de cuestiones se presentan como interesantes para establecerse como posibles líneas de investigación futuras, entre ellas:

- Estudiar el efecto en los resultados de la aplicación de multas e incentivos.
- Variaciones en el análisis producto de la introducción de defectos durante la garantía.
- Posibilidad de utilizar como estrategia competitiva el ofrecer garantías más extensas que los que los competidores pueden ofrecer en función de las fortalezas identificadas valuando las mismas como opciones reales para los clientes.
- Introducir en el análisis el costo del dinero para el caso de los proyectos más prolongados o con incertidumbres comerciales y tecnológicas significativas.
- Identificar la influencia de utilizar automatización en las pruebas como modificador del costo óptimo de garantía.

9 Referencias

Bhaskar, T. (2006). A cost model for N-version programming with imperfect debugging. *Journal of the Operational Research Society*, Vol. 57 (8), pp. 986-994.

Bohun, C. (2004). Modelling Quality and Warranty Cost (Chapter 2). En *Canadian Applied Mathemathics Quarterly V12 N1* (págs. pp. 37-66).

Hummel, O., & Burger, S. (2013). A pragmatic means of measuring the complexity of source code ensembles. *IEEE WETSoM 2013*, pp. 76-79.

Jain, M. (2001). Cost analysis for repairable units under hybrid warranty. Recent developments in Operational Research. *Narosa Publishing House*, pp. 149-165.

Kan, S. (2002). *Metrics and Models in Software Quality Engineering (2nd Edition)*. Addison-Wesley Professional.

Knox, S. (1993). Modeling the Cost of Software Quality. *Digital Technical*, pp 9-16.

Lai, R. (2011). A study of when to release a software product from the perspective of software reliability models. *Journal of Software V6 N4*, pp. 651-661.

Matson, J., Barrett, B., & Mellichamp, J. (1994). Software development cost estimation using function points. *Software Engineering, IEEE Transactions on 20.4*, pp. 275-287.

Musa, J. (1987). Software Reliability: Measurement, Prediction, Application. NY: McGraw-Hill.

Okumoto, K. (1980). "Optimum release time for software system based on reliability and cost criteria". *Journal of System and Software*, Vol. 14, pp. 315-318.

Pham, H. (2003). A software cost model with imperfect debugging, random life cycle and penalty cost. *International Journal in System Science*, V27 pp 455-463.

Popstojanova, K. (2001). Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*, Vol. 45, pp. 179-204.

Rinsaka, K., & Dohi, T. (2005). Determining the optimal software warranty period under various operational circumstances. *The Int Journal of Quality & Reliability Management*, pp 715-730.

Sargent, R. (1998). Verification and Validation of Simulation Models. *Proceedings of the Winter Simulation Conference*.

Tal, U. (2002). An optimal statistical testing policy for software reliability. *Demonstration of safety critical systems*, Vol 137 (3), pp. 544-557.

Walston, C. (1977). A method of programming measurement and estimation. En V. Basili, *Models and Metrics for Software Management and Engineering* (págs. pp. 10-29). IEEE Computer Society Press (EHO-167-7).

Westland,J.C. (2002). The cost of errors in software development: evidence from industry. *The Journal of Systems and Software (62 1-9)*, pp. 1-9.

Williams, D. P. (2007). Study of the Warranty Cost Model for Software Reliability with an imperfect Debugging Phenomenon. *Turk Journal of Electronic Engineering*, V15 N3 pp 369-381.

Xie, M. (2003). A study of the effect of imperfect Debugging on software development cost model. *IEEE Trans on Software Engineering*, V29(5), pp. 471-473.

Yamada, S. (1987). "Optimal software release policies with simultaneous cost and reliability requirements". *European Journal of Operational Research*, V31/1, pp. 46-51.

Yamada, S. (1993). Optimal software release problems with life-cycle distribution and discount rate. *Trans. IPS Japan (in japanese)*, Vol. 34(5), pp. 1188-1197.

Yang, B. (2000). A study of operational and testing reliability in software reliability analysis. *Reliability Engineering and System Safety*, Vol. 70, pp. 323-329.