Extended framework to evaluate software warranty costs

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)

Abstract. An extended framework to evaluate software warranty costs is discussed as part of an on-going research effort. A model is developed to integrate different aspects of the software development process and a proposal to populate the different parameters required out of commonly available software develepment process metrics is also made. As embedded systems are in common usage as part of other industry goods and services which requires the provision of warranty as part of their positioning in the market a better understanding on the ways to characterize the warranty of the embedded software is required. The proposed framework is callibrated and executed using a stochastic simulation execution using the Montecarlo method. The quantlitative behaviour of the dependent variables, the resulting quantitative magnitudes and the sensitivity of the results with different independent variables of the model is then explored. The significant effect of penalties and the strong influence of some product and process characteristics such as size, phase containment of defects and defect injected is noted, whereas the releatively low impact of financial factors such as the rate of discount are also noted. Finally preliminary conclusions and lines of work for future research are discussed.

Keywords. Software warranty, software development, embedded systems.

1 Introducción

En un contexto de alta competitividad se espera que las organizaciones desarrolladoras de software sean capaces de entregar sus servicios en tiempo y dentro de presupuesto, además de libres de fallas, incluso al punto de ser un factor clave de éxito. Al mismo tiempo, y crecientemente, no solo se trata de entregar el producto o servicio, sino también se debe proveer garantías de su funcionamiento, tanto en la forma de reparar cualquier falla como incluso de compensar a la organización compradora por los impactos ocasionados por ellas mediante mecanismos de multa acordados contractualmente. 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. En tal sentido no es posible producir software completamente libre de defectos, pero el esfuerzo (re-trabajo) resultante para solucionar defectos expuestos junto con la pobre gestión de cambios suelen ser las causas raíz de retrasos en las entregas, excesos de presupuesto en los proyectos además de fuente de insatisfacción en clientes.

Los clientes, por su parte, demandan que como parte de los servicios provistos se incluya el compromiso de solucionar los defectos del software entregado encontrados durante durante un tiempo acordado, denominado *período de garantía* (τ_w). 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 gestionar este aspecto en el software que contienen. La garantía a proveer puede incluir tanto la reparación a costo del proveedor del defecto hasta el reconocimiento de multas que bajo

978-1-5386-3057-0/17/$31.00

determinadas condiciones compensen al cliente por los impactos en su negocio que las fallas pudieran ocasionar.
Para los proveedores de software no es una opción simplemente calcular los costos extras que les permitan afrontar las garantías requeridas y establecer contingencias para las multas resultantes pues enfrentan una situación de mucha presión competitiva sobre sus precios; la matriz financiera de la cuestión favorece entonces abordar las causas raíz y entregar los productos de software con la menor cantidad posible de defectos, en particular aquellos que pudieran aparecer durante el período de garantía.
El modelado de defectos fue abordado entre otros por Musa et.al (Musa, 1987) quien aportó modelos cuantitativos que permiten anticipar el comportamiento de exposición de los defectos latentes en la medida que el software es ejecutado. Otros autores producen refinamientos con robusta validación empírica (Tal, 2002) así como el uso de los mismos para estudiar el problema del tiempo óptimo de liberación (Okumoto, 1980)(Yamada, 1987) (Yang, 2000), consideraciones de índole financiera (Jain, 2001) y la implicancia de decisiones sobre aspectos de la arquitectura (Popstojanova, 2001). Estos enfoques pueden ser utilizados para proponer estrategias de prueba y liberación óptima (Williams, 2007), incluso teniendo en cuenta las imperfecciones en el proceso de test (Pham, 2003) (Xie, 2003) o estrategias diferentes según la criticidad de los defectos encontrados (Bhaskar, 2006). Esto lleva a poder estimar los períodos óptimos de garantía bajo diferentes condiciones (Lai, 2011) (Bohun, 2004), incluyendo la consideración de los costos superiores de detección y remoción de defectos a partir de la liberación a producción (Westland,J.C., 2002) lo que alentará procurar que sean lo menos posibles.
Los modelos propuestos por la literatura pueden tener dificultades en su aplicación práctica pues el momento óptimo para la liberación de un software está fuertemente influenciado por decisiones pragmáticas relacionadas con el calendario comprometido, necesidades del negocio subyacente, compromisos comerciales y factores presupuestarios antes que la evaluación objetiva de la calidad del software bajo desarrollo. Estos modelos propuestos requieren para su cálculo, además, parámetros que no siempre las organizaciones tienen facilidad para calcular a partir de las métricas organizacionales habituales.
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.
En un trabajo anterior (Colla, 2015) se estudió como el *costo total de garantía* (C_t) podía ser influenciado primariamente por el test que ocurre hasta el *tiempo de la liberación* (τ_r), es decir la validación y verificación realizada para disminuir la cantidad de defectos latentes liberados (μ_r), de los que una parte aparecerán como *defectos en garantía* ($\Delta\mu$) mientras dure la misma. Sin embargo extender el ciclo de desarrollo como estrategia para reducir los defectos latentes involucra mas costos de desarrollo, los que en condiciones de arbitraje competitivo solo pueden deteriorar el margen de la empresa proveedora y su viabilidad comercial.

Para abordar sistémicamente esta cuestión será necesario que el proveedor mejore su capacidad de evitar inyectar defectos latentes (μ_0) durante el ciclo de desarrollo del software que produce, además de implementar metodologías que disminuyan su nivel de re trabajo medida como el *costo de calidad pobre* (CoPQ del inglés *cost of poor quality*) e incrementar eficacia del proceso para remover defectos (PCE del inglés *phase containment of errors*). Todos factores susceptibles de ser controlados mediante iniciativas de despliegue de tecnologías para el desarrollo y mejoras de procesos de software (SPI, por sus siglas en inglés *Software Process Improvement*).

Por su parte las organizaciones compradoras pueden estar tentadas de imponer condiciones de garantía arbitrariamente altas, tanto en tiempos de cobertura como en niveles de multa, inconsistentes con lo que el estado del arte en ingeniería de software puede técnicamente sostener; si esto ocurre solo logrará que sus costos se incrementen en la medida que el proveedor deba realizar provisiones de esos costos adicionales y por lo tanto si todas las organizaciones oferentes enfrentan el mismo problema el mercado no inducirá mejoras mediante el mecanismo de arbitraje competitivo y eso solo producirá que el comprador vea incrementar sus costos.

Este equilibrio entre intereses contrapuestos entre el proveedor y el comprador de software se puede alcanzar mediante la aplicación de conceptos de ingeniería de software los que derivan en una mejora de performance técnica en el proceso de desarrollo de manera que se inyecten menos defectos latentes, la que por su parte influye sobre la competitividad económica de la organización desarrolladora primero y luego sobre su diferenciación competitiva al mejorar los términos en los que le es posible comprometerse a entregar garantía sobre el software liberado de forma que la organización compradora tenga mejores precios y menos impactos en su operación.

En el ya citado artículo anterior de éste esfuerzo, el foco estuvo puesto en efectos directos y primarios que influencian este comportamiento, pero es de interés avanzar sobre factores adicionales de manera de comprender mejor su influencia.

Estos factores serán, preliminarmente, la consideración de cómo el costo financiero y el impacto que la aplicación de multas por parte de los clientes pueden influir los comportamientos de los proveedores en las distintas etapas del desarrollo y garantía

El proveedor seguramente abordará como primer estrategia reducir los defectos liberados incrementado el *tiempo de test* (τ_r) para de esa forma reducir la cantidad de *defectos encontrados durante la garantía* ($\Delta\mu$) y por lo tanto el costo de su reparación. Sin embargo los costos del test son más tempranos que los de solución de garantía y por lo tanto financieramente más relevantes, el equilibrio estará dado por la magnitud del *costo de oportunidad* (r_{TEA}) a la que el proveedor descuenta el proyecto.

El proceso de construcción de software introducirá defectos tanto durante el proceso de reparación de otros defectos durante el test como cuando se solucionen defectos bajo garantía; lo que por su parte aumentará el número total de defectos que es probable aparezcan durante la garantía y por lo tanto el costo de proveer la misma. Sin embargo el número de defectos que es esperable introducir por este mecanismo se asume bajo, toda vez que las intervenciones para solucionar problemas de garantía seguramente serán de funcionalidad, tamaño y complejidad modestas por lo que la posibilidad de introducir nuevos defectos escapados en número significativo por éste mecanismo será posiblemente reducida, la revisión de ésta hipótesis será un aspecto a validar en el futuro.

Finalmente, la provisión de servicios de desarrollo que asegure que los defectos son solucionados durante un período de garantía no mitiga los impactos que su aparición tiene en la operación del cliente del software en términos de costos adicionales, pérdidas de ingreso o impactos de imagen. Es natural, entonces, que el cliente desee protegerse de éstos impactos mediante la aplicación de estrategias de penalización que provean un

incentivo muy fuerte a que la cantidad de defectos que aparezcan durante el período de garantía sean realmente mínimas.
La contribución de éste artículo es explorar estas cuestiones mediante las siguientes preguntas de investigación:

- *¿Como varía el tiempo óptimo de reléase (τ_r) para distintos valores de multas aplicadas por la organización compradora?*
- *¿Como varía el tiempo óptimo de reléase para distintos valores del costo de oportunidad (r_{TEA}) que el proveedor asigna al proyecto?*
- *¿Cual es la sensibilidad de los distintos factores bajo estudio en el resultado final?*

2 Marco teórico

Se asume que en la medida que las fallas permiten observar defectos se introducen correcciones al código para repararlos; asumiéndose despreciable la introducción de nuevos defectos al hacerlo. Entonces el *número de defectos expuestos acumulados* o *perfil de defectos* (μ) para un tiempo arbitrario (τ) desde el comienzo del proceso de test puede ser estimado por la siguiente expresión (Ec 1) (Musa, 1987):

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

Ec 1

Donde la *cantidad total de defectos inyectados* (μ_0) y la *intensidad de detección de defectos* (λ_0) pueden ser obtenidas por regresión lineal de secuencias, incluso modestas, de datos obtenidas en forma temprana durante el test así como de información histórica en las líneas de base organizacionales. Usando éste modelo simple es posible pronosticar en forma aproximada los defectos detectados y removidos en cualquier tiempo arbitrario de ejecución del software.
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), lo que se utilizará como punto de partida para obtener el *esfuerzo total del proyecto* (E) del proyecto. Ambos se relacionarán típicamente mediante el uso de la *productividad* (π) con la que opera la organización bajo diferentes tecnologías y ambientes. Se pueden combinar éstos factores (Ec 2):

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

Ec 2

Donde el *factor de aprendizaje (γ)* suele ser en la práctica muy cercano a la unidad si se acotan suficientemente las tecnologías utilizadas y apalancando la experiencia en ellas. El costo del proyecto es intercambiable con el de esfuerzo, puesto que a todos los efectos prácticos será su principal componente. Es usual para las organizaciones productoras de software determinar líneas de base sobre métricas organizacionales sobre su habilidad para en detectar defectos antes de liberar el software. Para ello se colectan los reportes de *defectos escapados luego de una liberación* (μ_e) de sus proyectos comparados con los *detectados durante el proceso de desarrollo* (μ_r); 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):

$$PCE = \frac{\mu_r}{\mu_r + \mu_e} = \frac{\mu_r}{\mu_0}$$

Ec 3

Es usual observar que PCE sea razonablemente estable para una dada mezcla de tecnologías, ámbitos funcionales y procesos de desarrollo en una organización y por lo tanto que sea gestionado mayormente mediante acciones del proceso de desarrollo utilizado.
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 4) (Walston, 1977).

$$\tau_0 = KE^{\beta} \text{ y } \tau_r = \upsilon\tau_0$$

Ec 4

Donde las constantes de *eficiencia de calendario* (K) y *factor de aprendizaje* (β) se obtienen por calibración con información histórica de cada organización y guardando el *tiempo total de test al momento de la liberación* (τ_r) relación con el tiempo total del proyecto (τ_0) a partir de la *proporción de tiempo de test* (υ) la que tiende a ser aproximadamente constante para proyectos de similar complejidad por lo que se puede establecerse sus valores a partir de información histórica. Es decir, se puede estimar el tiempo medio de test esperado para un dado tamaño del proyecto y con anclaje en las métricas históricas; este tiempo, instanciado como lineamiento de cada proyecto en particular, luego sujeto a cierta flexibilidad producto de la gestión y terminará siendo un factor decisivo al momento de establecer la habilidad que el resultado del desarrollo pueda satisfacer las necesidades de garantía. Ordenando las Ec1 y Ec 3 es posible escribir para el momento donde se completa el tiempo de test (τ_r) (Ec 5):

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

Ec 5

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 propuesto 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, asumiendo que alterar cualquiera de los otros factores intervinientes requerirá procesos de intervención cuyos tiempos usualmente excederán a los del proyecto. Por lo tanto, el tiempo de test será considerado un atributo de calidad sujeto a gestión y 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.
Por lo tanto para estimar el *costo de test* (C) para un dado conjunto de *atributos de calidad (q)* que contribuyen al mismo, puede estimarse como que el mismo estará dado por la (Ec 6):

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

Ec 6

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 variable 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 estimado para un proyecto en particular o proyectado desde la línea de base histórica como la relación histórica promedio entre el esfuerzo utilizado en re trabajo de defectos y el número de defectos encontrados hasta el momento de la liberación para ese proyecto. La proporción que representa 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).

$$\mathbf{C_2} = \frac{\mathbf{CoPQ}}{\boldsymbol{\mu_r}}$$

Ec 7

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 8):

$$C_w = \kappa C_2$$

Ec 8

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

$$\Delta\mu_w = \mu(\tau_w + \tau_r) - \mu(\tau_r)$$

Ec 9

El costo de garantía W(q) resultará entonces como (Ec 10):

$$W(q) = \kappa C_2 \Delta\mu_w = C_3 \Delta\mu_w$$

Ec 10

Finalmente, la función de multa puede escogerse arbitrariamente pero a los efectos de éste análisis inicial se adopta la más simple posible, es decir una función lineal de los defectos encontrados, es decir

$$\mathrm{M(t)} = \mathrm{C_4}\ \Delta\mu_\mathrm{w} = K_4\mathrm{C_3}\ \Delta\mu_\mathrm{w}$$

Ec 11

Donde se asume que la *multa por defecto encontrado en garantía* (C_4) se calculará como un multiplicador del costo promedio de repararlo a partir del *coeficiente de multa* (K_4).

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)), el costo de garantía (W(q)) y la función de penalización que ocurrirá durante el mismo período de vigencia de la garantía M(q) para un dado conjunto de especificaciones de calidad (q) , es decir:

$$C_t = \min_q [C(q) + W(q) + M(q)]$$

Ec 12

Para tener en cuenta el impacto financiero es necesario obtener el *valor presente (VP)* de los distintos componentes de costo, los que se obtendrán mediante el descuento al costo de oportunidad (r_{TEA}) de los flujos de fondos respectivos al final de las etapas donde ocurren, según se muestra en (Ec 13):

$$VP\big(C(q)\big) = \frac{C(q)}{(1 + r_{TEA})^{\tau_r}} \qquad VP\big(W(q)\big) = \frac{W(q)}{(1 + r_{TEA})^{\tau_r + \tau_w}} \qquad VP\big(M(q)\big) = \frac{M(q)}{(1 + r_{TEA})^{\tau_r + \tau_w}}$$

Ec 13

Y por lo tanto el costo total considerando el impacto financiero vendrá dado por

$$VP(C_t) = \min_q [VP(C(q)) + VP(W(q)) + VP(M(q))]$$

Ec 14

A los efectos de este planteo propuesto del problema se considerará como especificaciones de calidad (q) los valores de *tamaño (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), el *tiempo de garantía a ser cubierto* (τ_w), el costo de oportunidad (r_{TEA}) y la magnitud de la multa (K_4), es decir:

$$q = \{S, PCE, \delta_0, \tau_w, \tau_r, r_{TEA}, K_4\}$$

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 (Ec 15):

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

Ec 15

5 Análisis del resultado

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 permiten obtener 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	ν		7	8	10
Coeficiente de multa por defecto	K_4		1,1	2	4
Tasa de descuento	r_{TEA}	Tasa ef. anual	1%	7%	15%

Período de Garantía	τ_w	Meses	0.5	1	6
Relación tiempo test/proyecto	N		0.2	0.4	0.6
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 discutidas en el marco teórico 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 distribución de los resultados y la sensibilidad a los diferentes parámetros propuestos que pueden ser esperados al explorar 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 de tipo 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.

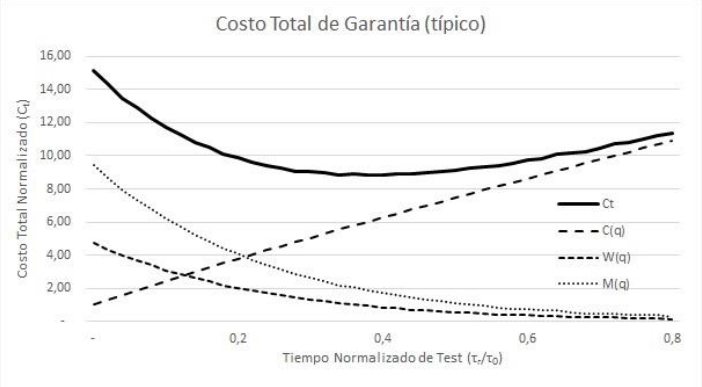

Figura 2 Costo total de garantía (típico) en función del tiempo de test normalizado

Como variable objetivo de la simulación se utiliza el tiempo de test (τ_r) 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) 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. Para permitir que

la simulación incluya un número significativo de casos se utilizan corridas de simulación con 10000 intentos, la experiencia muestra un adecuado balance entre la convergencia de los resultados y el tiempo de ejecución con esta configuración.
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 (der), la distribución de posibles valores de tiempo de test para obtener el costo óptimo puede verse también en la Figura 3 (izq.). El parámetro que más influencia el resultado es el tamaño (S) del proyecto seguido en importancia relativa el PCE y la magnitud de las multas (K_4).

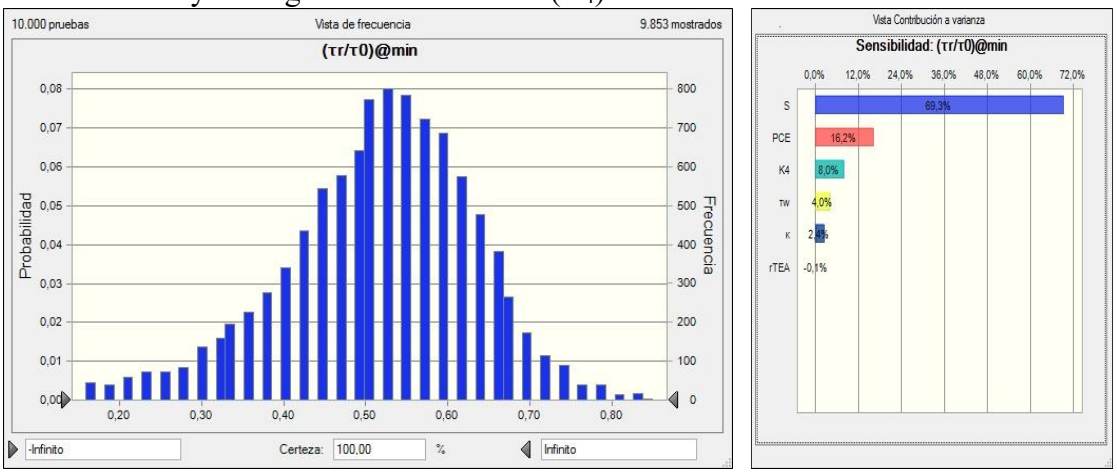

Figura 3 Sensibilidad y distribución resultante (τ_r/τ_0) para sesión de simulación típica

En siguiente orden de importancia en cuanto a sensibilidad del resultado aparece el tiempo total de garantía (τ_w) seguido por el factor de complejidad del ambiente productivo (κ). El costo de oportunidad muestra tener una influencia poco significativa en los resultados, y eso puede explicarse en los tiempos relativamente reducidos en los que operan las garantías consideradas.
Finalmente, la magnitud de la complejidad del ambiente productivo juega un rol en la determinación del costo óptimo equivalente al correspondiente al tiempo de garantía.

6 Conclusiones

- *¿Cómo varía el tiempo óptimo de reléase (τ_r) para distintos valores de multas aplicadas por la organización compradora?*
  La existencia de multas incentiva a la organización desarrolladora a reducir los defectos latentes y eso se consigue tanto mediante la mejora de evitar inyectar defectos primero y su capacidad de remover defectos posteriormente, una de cuyas estrategias posibles sugeridas por el modelo es incrementar el tiempo de test.
- *¿Cómo varía el tiempo óptimo de reléase para distintos valores del costo de oportunidad que el proveedor asigna al proyecto?*
  Prácticamente no tiene incidencia, eso puede comprenderse dado que los tiempos entre los distintos momentos de generación de costos considerados (testing y ejecución de garantía) es reducido mientras que las magnitudes relativas de los flujos involucrados es muy superior sobre tiempos relativamente breves.
- *¿Cuál es la sensibilidad de los distintos factores bajo estudio en el resultado final?*
  El análisis de la sensibilidad entre todos los componentes del modelo de simulación indica que el orden de importancia de los factores es tamaño (S), la efectividad para remover defectos (PCE), la magnitud de la multa (K_4), el tiempo de garantía *(τ_r)*,

complejidad del ambiente de producción (κ)y en último lugar costo de oportunidad (r_{TEA}). Lo que sugiere que la imposición de multas y extensión del plazo de garantía por parte de los compradores debería inducir un cambio de estrategia en los proveedores de software en cuanto a liberar con menos defectos sus productos y confirma que la mejor estrategia desde el punto de vista del proveedor es reducir el tamaño (complejidad) del producto liberado y mejorar la eficiencia de su proceso de calidad.

7 Trabajo Futuro

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

- Variaciones en el análisis producto de la introducción de defectos durante la garantía.
- Identificar la influencia de utilizar automatización en las pruebas como modificador del costo óptimo de garantía.

8 Referencias

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

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

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.

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

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

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

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

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.

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.

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.

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.

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.

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.

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.

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

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.

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).

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

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

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

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

Colla, P. (2015). Marco para evaluar garantía en desarrollos de software. *ASSE 2015, 16° Simposio Argentino de Ingeniería de Software.* (págs. pp 14-25). 44 JAIIO.