Simplified framework to evaluate software development warranty

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. This article addresses a simplified framework to evaluate the warranty costs of a software development process. The approach uses parameters required by the models from metrics commonly found associated with a software development project. Methods are proposed to extract and apply organizational baselines. The proposed framework is validated using simulation techniques based on the Monte Carlo method, allowing for the assessment of the likely distribution of the results and the sensitivity with the parameters used. Preliminary conclusions are extracted and future lines of work identified.

Keywords: Software warranty, software development process, SEI-CMMI™

1 Introduction

Software development organizations are required to deliver to their customers on time, within budget and without defects in order to be considered competitive.This includes cases where the application to be developed provides complex features and other business constraints less than ideal needs to be considered in terms of restrictions of different sort.

The fulfillment of these requisites, often with contradictions among them, used to be little more than an act of goodwill based on the best effort among all the parties involved. However, the current marketplace requirements might define the fulfillment of these conditions as the condition making the difference between an organization to be successful or not.

The *software development life cycle* (SDLC) must then include activities distributed in the different project*phases* or *stages* to achieve the goals required by customers.However, and despite significant advances in the state of the art of software engineering technology, the chances ofintroducing defects during the development activities are still significant. At the current state-of-the-art it is just not possible to produce defect-free software. In fact the rework required to solve the defects found during the process as well as inadequate change management practices can usually be traced back as the root cause for most project troubles and failures.

At the same time, customers demand a commitment from the development organization to provide support, fix and mitigate defects found withinof the warranty period after the software has been delivered.

Given the increasing integration of embedded systems into industrial and manufactured goods, which customarily requires warranty arrangements to be provided as part of their commercialization offering, it is necessary to understand how the warranty on their, now integrated, software components.

The warranty scope ranges from the full repair at the vendor expense of the *reported unique defects* or *CRUDs* (*Customer Reported Unique Defects*) up to the recognition of penalties or compensations for the customer to offset the expenses arising because of the impact the defects found made on their business

To implement such mechanisms the software development vendors face a heavy competitive situation where the classical approach to add the projected expenses to fulfill the warranty into the cost is not feasible;simply because the financial profile of the market demands might force an organization using such practice out of business. Therefore, the financial matrix of the current market demands favors to address the root cause, which is to deliver software with the minimum of defects with potential to show up during the warranty period.

Among the pioneers to address the problem to forecast the software development defects were Musa (Musa, 1987), whose models proposed the introduction of a comprehensive *validation* and *verification* process (testing) against a requirement set. This approach provides little insights into the fundamental question of the effort and time demanded to achieve a given defect profile, both crucial elements to define the economic viability of a given development and testing strategy.

Robust statistical models with significant empirical validation described at the bibliography exposes the main components required to study this problem such as the one proposed by Tal(Tal, 2002). Tools provided by such modeling allows for the planning, monitoring and controlling of the testing processes to achieve any given latent defect profile at release time which can in turn provide a reasonable balance

between the technical requirements of the software behavior in terms of reliability as well as the financial goals of the project to be met by the vendor.
Some authors addressed the problem of how to establish a release criteria balancing the time and effort consumedin testing and the resulting reliability to be expected after the release, being it the main driver for the warranty costs experienced.Okumoto(Okumoto, 1980)studied the problem from the perspective of the optimal release time under any given reliability and cost constraint. Yamada (Yamada, 1987)suggested a criteria to establish the optimum release time considering warranty cost and reliability restrictions. Yang (Yang, 2000)studied the reliability profile using software evaluation models. Tal (Tal, 2002)proposed a statistical set of criteria optimizing the reliability of the released software. Jain (Jain, 2001)documents a researchabout hybrid models to predict the total operating cost to provide a given warranty requirement including opportunity cost considerations.Popstojanova(Popstojanova, 2001)addressed the problem from the architectural viewpoint in order to derive the factors to consider in orderto evaluate the reliability of a given software based system. Yamada (Yamada, 1993)proposedan optimal release strategies depending upon the software lifecycle and the financial opportunity cost involved for a given industry. Prince Williams (Williams, 2007)studied the problem from the perspective of adopting testing strategies to address specific warranty periods. Pham (Pham, 2003) proposed that the total cost of production exploitation of the software under warranty should take into consideration the imperfect nature of the software test and correction process based on the lifecycle model and some of the penalties involved.Xie(Xie, 2003)provided further insight on the impact of imperfect debugging and its impacts on the total development cost derived from the adoption of a given release strategy.Bhaskar(Bhaskar, 2006)gave contributions considering the criticality of the defect based on the impact of the failure and the cost to address it during different development phases, adopting a defined release strategy to maximize the return.
Rinsaka&Dohi (Rinsaka & Dohi, 2005)explored the problem of defining the optimum warranty scheme under different operational circumstances. Lai (Lai, 2011)studiedthe optimal timing for releasing a givensoftware from the perspective of the reliability models.In turnBohun (Bohun, 2004)attempted to simultaneously optimize the costs associated with quality while satisfying multiple quality requisites.

Any strategy based on containing the defects after the release has been made faces severe issues because of the impacts on the underlying value chain it supports, increased costs and additional effort to solve and remove defects on a production environment(Westland,J.C., 2002).
However, the models proposed by the different authors have some challenges for their practical use as the software's optimal release time is often influenced by pragmatic business decisions related to the committed calendar, needs from the underlying business and budgetary constraints rather than the pure management of defects to satisfy some quality parameters. At the same time the proposed model typically requires parameters involved in their computation which might not be available from the development organizations as part of their usual organizational metrics baseline.
The complexity involved in the usage of some models often looks cumbersome enough for the organization to adopt in turn a simpler strategy to test till the calendar allows and then release, whose poor results do not come as a surprise.
A rational strategy to address this conundrum might be to model the problem using a simplified framework so that qualitative results can be evaluated, still consistent quantitatively with the real operation, and obtained using metrics available during the project execution or being part of their historical metrics baseline.
With this approach, an approximate number of future defects to be expected can be projected and thus the financial and technical implications that will be faced because of them afterwards can be evaluated. With such insight, the best balance can be found under a given competitive context to optimize the vendor outcome.
At the same time, and independent of the release decisions, it is important to understand which are the parameters most relevant to manage in order to satisfy the warranty requirements of the market the organization chooses to serve in such a way that a profitable delivery can be produced.
The main contribution of this article is to integrate different sources found in the bibliography with a perspective derived from the experience in order to address some strategic questions based on simple models and measurements usually available during the normal management of a development project.
The research questions can be stated as follows:

- *What is the warranty profile allowing a given organization to achieve a sustainable situation from their current quality parameters during the development process?*
- *What is the influence of the software product complexity and the issues associatedwith the warranty to be provided forit?*
- *What is the relation between the release time and the total cost including warranty? Which possible equilibriums can be established based on the organizational capabilities?*

2 Software reliability models

The previously discussed models proposed by the literature are useful to forecast the reliability improvement as long as the proper parameters are used on them. However, practical experience shows that any improvement in the forecast capability by increasing the complexity of the model might be neutralized by the wide dispersion in the typical values and variability of the parameters needed. Because of that, simpler models tend to offer a good trade-off between good qualitative and quantitative results as long as they are calibrated with the metrics from the organization using them.

All models forecast defects exposed as "*faults*" or "*errors*" while the software is executed. Assuming the number of defects injected by the development process is unknown but finite it is reasonable to expect that as corrections are made the total number of defects will be reduced. This is not necessarily true as corrections and changes made would inject their own share of defects. A good approximation can be obtained assuming that under short timeframes, such as the ones involved during the warranty period (few days to weeks) the impact of the defect injection during correction might be negligible as a first approximation. Further discussion and validation is needed on this regard, but it is a reasonable first assumption for organizations with a reasonably mature development process in place.

Also, software reliability models express time not in terms of regular *calendar time* but in terms of *continuous execution time*(τ), which accounts for the time discontinuities introduced while the correction activities are made. Although at the very beginning of the validation and verification process the effective run time is limited by the stop time imposed by the corrections made. However, the usage of *calendar time* in the model is simple and effective enough to become useful despite the small distortions introduced in the results over the entire test cycle time. As the faults are experienced allowing defects to be exposed, then corrections are made which fix them.Assuming as negligible the introduction of new defects during the correction process the *accumulated number of defects* or *defect profile* (μ) for a given time (τ) since the start of the testing can be estimated by the following expression (Ec1) (Musa, 1987):

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

Ec1

The *total number of injected defects*(μ_0) and *the defect detection intensity*(λ_0) can be obtained either using linear regression over simple testing sequences early in the process or using historical organizational data. Using this model the total number of defects found, and removed, at any given arbitrary software execution time can be evaluated.
Defect profile models will not include hypothesis around the process used to develop the software under test. However, the development organization could leverage their historical metrics or quantitative management baselines to calibrate these models and acquire a broader perspective to understand and forecast the behavior of their projects.
Almost all development organizations would capture the *size of the software* (S) they produce, usually by adopting counting mechanisms with methodologies showing a good correlation with complexity (Hummel & Burger, 2013). The *productivity*(π) shall then be used to manage the efficiency of the projects under different technologies and environments.The *total project effort* (E) might then be expressed as (Ec2):

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

Ec2

The scaling exponent operates as a *learning factor (γ)*which isusually very close to the unity when the organization performsits projects using a relatively small number of technologies, given the organization has experience with them.
For all practical purposes the cost and effort of a development project would be assumed as equivalent, as typically are highly correlated be-

cause a higher proportion of the total cost is defined by the effort expended. Give the previous considerations the software producing vendor organization can determine the number of defects prior to release starting from the collection of field reports about *defects escaped after release* (μ_e) compared with the *defects detected during the development process* (μ_r); a ratio usually called *phase containment of errors* (PCE) can be then defined as (Ec3):

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

Ec3

This very same information can be correlated with the size or the complexity of the application, using suitable metrics such as *function points*(Matson, Barrett, & Mellichamp, 1994)to define the *defect density at release* (δ_e) which can be expressed as (Ec4):

$$\delta_e = \frac{\mu_e}{S}$$

Ec4

Combining [Ec3] and [Ec4] the *total number of injected defects(μ_0)*can be estimated using *(*Ec5*)*

$$\mu_0 = \frac{\mu_r}{\mathrm{PCE}} = \frac{S\delta_e}{[1 - \mathrm{PCE}]}$$

Ec5

Empirical observation shows the PCE as being reasonably stable for a given organization. Typically it can be controlled using quantitative management methods to manage their value to fall within limits exhibiting a *stable* and *capable* behavior for projects within an organization using similar processes and technologies.
The *total development cycle time*(τ_0) can be obtained from the total effort using a relation as per (Ec6) (Walston, 1977).

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

Ec6

The calibration constants for the relation can be interpreted as the *calendar efficiency* (K) and a *learning factor* (β) and might be derived from historical information at each organization.
The *total test at release* (τ_r) is related with the total project time with a relation such as (Ec7):

$$\tau_r = \upsilon \tau_0$$

Ec7

The *test time proportion* (ν) can be observed as reasonably constant across projects of similar complexities so can also be calibrated from historical data. Combining equations (Ec2 andEc6) the average expected testing time to release as a function of the size/complexity can be evaluated (Ec8)

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

Ec8

In other words, the *expected average testing time till release* can be estimated given the size of a particular project.The actual results for a project instance might show flexibility reacting to the management actions in order to satisfy the requirements and constraints of the warranty in the particular case.As such, the *number of defects at the end of the test* (μ_r) will be defined using (Ec1)and expressed as:

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

Ec9

Joining (Ec3y Ec8) it is possible to express (Ec10):

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

Ec10

allowingfor the characterization of the defect behavior using a very simple model calibrated using commonly available historical data or metrics collected during the project management activities.

3 Warranty cost evaluation

In previous sections the *testing time to release* (τ_r) is assumed as derived from the organizational metrics baseline and built from historical series capturing representative past efforts and driven mostly by the project size.In this way past project decisions contributed to define the organizational capability in order to achieve results today in their particular technology and business context in terms of their performance avoiding the injection, performing the detection and removing defects as efficiently as possible.

However, on a project to project basis the total testing time to release is often a management decision rather than a token given by a fixed technicaldecision. The modeling approach discussed in this article helps to evaluate the tradeoff of reducing the costs associated with testing time at the expense to release with a higher number of latent defects and therefore potentially face higher warranty costs, or conversely, to improve the warranty costs profile by releasing with as few latent defects as possible with the potential to appear during the covered warranty time by extending the testing time.The underlying assumption on this article is that the testing time is a quality attribute subject to management decisions and therefore one of the potential candidate factors to be operated upon in order to optimize the total cost.

The total cost of warranty will then be related with the cost of testing the application until the release time and to provide the services within the agreed upon warranty time thereafter. A good modeling for such tradeoff has been presented by Rinsaka&Dohi(Rinsaka&Dohi, 2005) using the Rayleigh distribution proposed by Goel-Okumoto.

At the same time the different elements of cost modeling will occur at different stances of the project lifecycle, and therefore it is relevant to consider the financial impact produced depending on the time of occurrence.To account for this factor the time and risk cost of the money, or opportunity cost, needs to be factored (Brealey R., 2015) through the discounted cash flows using a *discount rate* (r_{TEM}) of the different expenditures during the lifecycle.

To estimate the total testing cost (C) it is necessary to identify the quality attributes (q) driving it, a proposed value can be obtained from (Ec11):

$$C(q) = C_0 + \frac{C_1 \tau_r^{\alpha} + C_2 \mu_r}{(1 + r_{TEM})^{\tau_r}}$$

Ec11

The *testing environment setup fixed costs* (C_0) typically happens at the beginning of the project and thus notin need of being discounted, the *cost per testing execution time unit*(C_1) and the total testing time to release (τ_r) affected by a scaling factor representing by a*learning factor* (α) are assumed to be subject to management. Finally, the cost to remove the defects found can be obtained using the *average removal cost per defect* (C_2) and the expected *number of defects to be found during testing* (μ) defined by the testing time as shown by the relation (Ec9).The cost to remove defects can, in turn, be estimated for a given organization either by using historical data or by tracking the rework effort or *cost of poor quality*(CoPQ)and dividing it by the number of defects found in the recent history of the project.The proportion between the CoPQand the total project effort is commonly tracked by organizations performing process improvement actions to keep it within defined limits depending on the maturity of their operation (Knox, 1993).
The available bibliography(Westland,J.C., 2002)and practical observation support the relation between the effort to correct a defect once the release has been made to the production environment as compared with performing the same modification in the development environment. The access complexity, higher level of scrutiny and authorizations involved, limited time windows to perform and difficulty to recreate the conditions to properly investigate the defectuntil the solution make the effort to fix be higher. The model proposed to take account of this factor by raising the needed effort to fix once in production by a multiplier named *production complexity factor* (κ) reflecting the cost increase between operating in a development or production environment.Then the *cost to correct defects during the production time*(C_w)after therelease while the warranty coverage applieswill be given by (Ec12):

$$C_w = \kappa C_2$$

Ec12

The *total number of defects during warranty*(μ_w) forecasted during the *warranty period* (τ_w) can also be obtained using (Ec1):

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

Ec13

The *total cost of warranty* W(q), will also be affected by financial considerations related to the opportunity cost of the money and therefore needs to be discounted using the opportunity costwhich results in (Ec14):

$$W(q) = \frac{\kappa C_2 \mu_w}{(1 + r_{TEM})^{\tau_r + \tau_w}}$$

Ec14

4 Modeling the total delivery cost

To optimize the *total delivery cost*(C_t) a combined minimum between the *testing cost* (C(q)) and *the warranty cost* (W(q)) for a given set of *quality attributes*(q), so the following condition is met (Ec15):

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

Ec15

For the scope of this article the *quality attributes*(q) are considered to be the *size/complexity*(S), *injected defect density during development* (δ_0), *phase containment of defects prior to release* (PCE), the *total testing time* (τ_r) and *the warranty time* (τ_w)while the solution of defects will be honored, so:

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

Capturing the main project processes and financial management parameters their influence can be evaluated. The condition for the minimum total cost will be satisfied at release time(τ_r) when the first and second derivative of the total cost as a function of the time to release satisfy the following conditions (Ec16):

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

Ec16

5 Research method

The model discussed in previous sections has been developed and partially validated by using it as a quantitative management resource for different pilot projects. The impact of different parameter distributions and ranges of values expected to be explored and also validated. As a result, the final model distribution and value ranges can be configured as a baseline for the different factors of interest. Data from over a dozen projects performed under a mature software production process aimed to deliver to a very competitivemarketplace, suggests the values used to be representative and relevant for the purpose.

Applying the equations (Ec 14) analytically, the optimal result for a given set of parameters might be found, but the usefulness of such approach is limited because of the simplified nature of the modeland the dispersion typically found in the parameters used, even when derived from stable and capable processes from a single organization. Conversely, an analytical solution approach hides the fabric of correlations and sensitivity between the different parameters involved in terms of their influence over the final outcome, which is a very rich set of information to be analyzed.

To overcome this hurdle the systemic model proposed has been verified and validated using stochastic simulation techniques where the results shows the values and distributions for results that can be expected when driven by ranges and distributions of the input parameters.

The table of values (seeFigure 1) shows the actual ranges and distributions used in the simulation supporting this article. This dataset is derived from the set of pilot projects involved; however, each organization can replicate the methodology just using their own set of data from their organizational history. The proposed model has been created using generic considerations and no assumptions were made about the underlying development process used, with appropriate validation it is likely that with the proper parameters it can be used in a wide number of development scenarios.

The interrelation between the organizational parameters, modeled as stochastic variables, allows for complex interrelations between them to emerge and the sensitivityof the influence to the final outcome to be understood.

Validating the results, mostly using expert judgment, shows a remarkable consistency withtypical results obtained from practical experience. Therefore the results could be used to perform analysis and extract conclusions on the research topics.

In order to account for the ranges and distributions of the different parameters simulation techniques involved, the MonteCarlo method are used. Triangular distributions are adopted (Sargent, 1998) for all parameters where only a minimum, average and maximum values are available, this distribution is recommended when there is no other clearly defined distribution for a given variable, a best fit distribution of observed data can be used when available.

The goal of the simulation is to find the total test time minimizing the total cost for a given constellation of parameters.

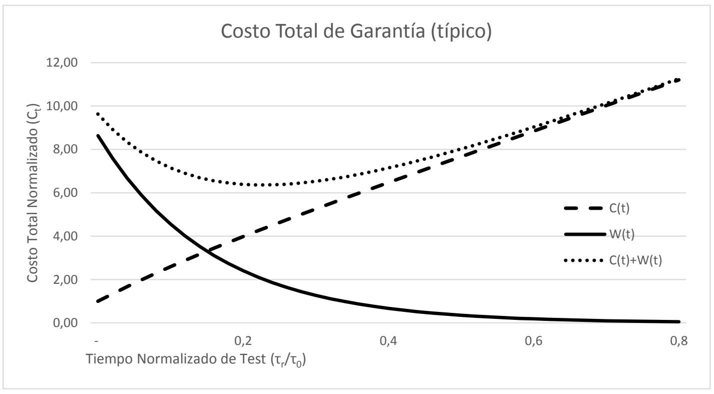
In order to accommodatethe comparison of a large set of projects the resulting release time is expressed in a normalized form compared with the total project time ($\tau_r/\tau 0$) withoutlosing generality but making the result independent of the absolute magnitudes of a given project.Once the simulation is performed with a given set of parameters a balance between the testing cost and warranty provisions can be identified under different testing durations (Figure 2 left) showingthe existence of avalue which minimizes the total cost. For a given set of parameters it is also possible (Figure 2 right) to observe how the extension of the warranty period modifies the testing time needed to drive the total cost to a minimum. The simulation is performed using 5000 stochastic trials, because experience suggests this number offers a suitable balance between the fast convergence of the results and execution time.

To capture the sensitivity relation between different variables and the impact from the expected dispersions in the parameters, the results from a typical simulation sessionare shown. Hence the sensitivity can be explored between the optimal cost and the parameter ranges where this result can be obtained.

Variable	Symbol	U.M.	Min	Med	Máx
Code size/complexity	S	PF	10	100	250
Phase containment of errors	PCE	%	0,5	0,8	0,95
Initial defect density	δ_0	Defects/PF	0,5	1	10

Production environment complexity	K		7	8	10
Warranty period	τ_w	Months	0.5	1	6
Discount rate (effective yearly)	r_{TEA}	% Year	1%	7%	15%
Discount rate (effective monthly)	r_{TEM}	% Monthly	0.00083	0.00565	0.01171
Testing to Project Time ratio	N		0.1	0.2	0.4
Development Productivity	Π	Hours/PF	8	15	25
Other values used during the simulation wereα=1.05, C_0=1 Staff/Mes,C_1=2,09,C_2=0,01,K=0.66 ,β=0.5. PF=Function Points					

Figure1Organizational parameters used during the simulation



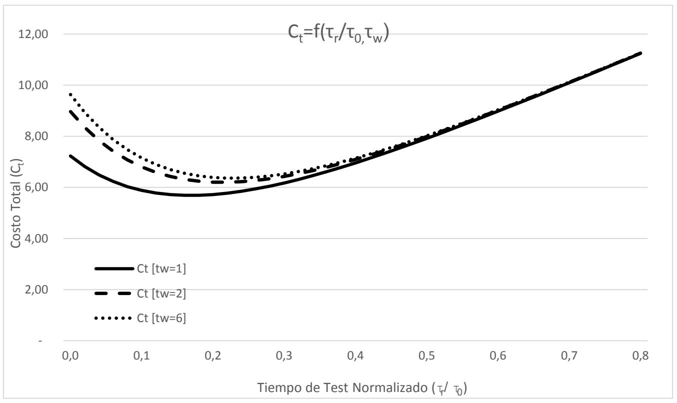


Figure2Typical warranty cost as a function of the normalized test time

6 Variable influence

Once the simulation is performed the resulting distribution and the sensitivity between parameters is obtained. The results can be seen inFigure3 (left), whereasthe distribution for the most likely optimum values for the testing time to achieve an optimal cost can be seen inFigure3 (right).

The parameter with greatest influence on the outcome is the initial density of defects (δ_0).This factor is strongly defined by the software engineering practices involved during the development stages; in particular requirements management, static testing techniques (inspections or peer reviews) and the usage of coding pattern oriented development during the code building phase.

The next parameter in importance is the phase containment of errors (PCE) which is a strong indicator of the test effectiveness and can be improved using coverage techniques, formal testing methodologies and project management disciplines.

Therefore, the main components for which the result shows a greater degree of sensitivity related to the optimal provision of a software warranty mostly defined by the development process and technology used by the organization. This result, far from being surprising, is aligned with the reported benefits of the adoption of robust software engineering and project management practices.In this regard, this result can be employed as leverage as the conceptual justification required to justify the investment to adopt, deploy and institutionalize a robust quality system as a way to achieve a competitive edge.

Finally, the magnitude of the production environment complexity is related to the optimal cost but the outcome shows little relation with it under a wide range of complexity values. The opportunity cost, as a way to measure the financial implications produced by the different stages of the warranty cycle, shows a negligible influence in the outcome, probably because the timeframes involved for the different cash flows and their magnitude results on financial impacts much smaller than the ones driven by other factors considered.

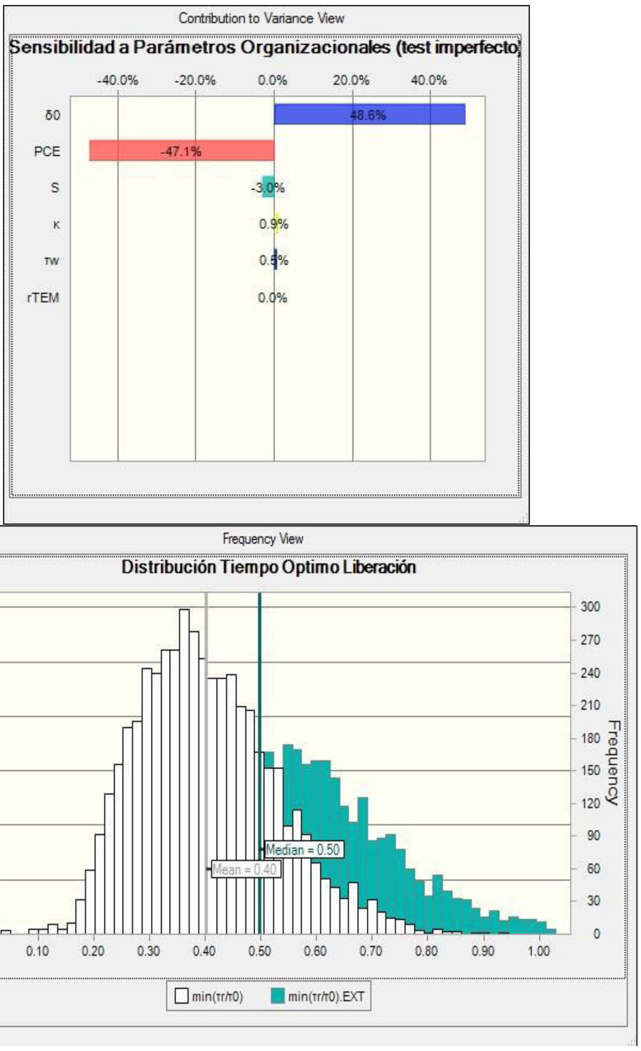


Figure3Variable sensitivity and normalized testing time (τr/τ0) for a typical simulation

7 Conclusions

The analysis performed enabled usto provide a preliminary answer to the research questions originally posted.

- *What is the warranty profile allowing a given organization to address a sustainable from their current quality parameters during the development process?*

  The main drivers are the initial defect density (δ_0) and the phase containment of errors (PCE) which for a given technological and organizational context can be optimized by adopting mature quality processes, good development practices, tools to support the development life cycle and component reuse as an effective way to both reduce the code size to develop and the defects injected.

- *What is the influence of the software product complexity and the issues associated of the warranty to be provided to it?*

  The project size and complexity (S) influences the definition of the optimum time to test suggesting the convenience to develop smaller components to improve the warranty profile.

- *What is the relation between the release time and the total cost including warranty? Which possible equilibriums can be established based on the organizational capabilities?*

  An optimal release time can be identified for each combination of the quality requisites, being the warranty time (t_w) one of them. All other factors being equal the duration of the warranty increases the testing time needed to achieve a minimum total cost.

8 Bibliography

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.