Aplicación de los procesos de nacimientos puros en confiabilidad de software

Gabriel Pena, Gonzalo Cozzi, and Néstor R. Barraza*

Universidad Nacional de Tres de Febrero, Caseros, Argentina
nbarraza@untref.edu.ar

Resumen La aplicación de los procesos de nacimiento en Confiabilidad de Software data de los comienzos de la especialidad. Si bien numerosos modelos han surgido en las últimas décadas, los procesos de nacimiento han sido aplicados ya sea para realizar analogías con otros modelos, para realizar simulaciones computacionales de fallas en software, o para proponer nuevos modelos. En este trabajo se presenta un resumen de la aplicación de los procesos de nacimiento en Confiabilidad de Software y se muestran simulaciones que permiten visualizar futuros desarrollos. Los resultados obtenidos de la simulación se comparan con reportes de fallas reales.

Keywords: Confiabilidad de Software, Procesos de nacimiento, Proceso de Yule, Proceso de Polya

1. Introducción

A medida que ha crecido la demanda de sistemas informáticos en prácticamente todos los desarrollos tecnológicos, la posibilidad de crisis por fallos en ellos se ha incrementado dramáticamente, crisis que van desde pequeñas molestias hasta pérdida de vidas. Y es el software el principal protagonista, tanto en los enormes avances logrados como en la mayoría de los fallos. Es en respuesta a numerosos incidentes del pasado, y a los que pudieran ocurrir a futuro, que dentro de la Ingeniería de Software se incluye como capítulo fundamental el relativo a la confiabilidad, y con él todas las técnicas cuyo objetivo es garantizar la máxima reducción de riesgos durante la vida operativa del software. La confiabilidad de software, se define como la probabilidad de operar software sin ocurrencia de fallos durante un lapso determinado de tiempo en un entorno específico. Entre todos los atributos de calidad del software, la confiabilidad se considera como un factor clave, ya que es fundamental para la prevención de incidentes. Por lo tanto, también se constituye en un factor clave para la satisfacción del usuario de software.

Los modelos de Confiabilidad de Software son procesos estocásticos que modelan la detección de fallas en software a lo largo del tiempo. Estos modelos son útiles para predecir el tiempo medio entre fallas, o la evolución del número de

* El autor también se desempeña en la Facultad de Ingeniería. UBA.

fallas a lo largo del tiempo, con lo cual se pueden dimensionar los recursos necesarios de testing, realizar ajustes en el desarrollo y tal vez lo mas importante es que se puede estimar el tiempo de entrega del producto de software, o la confiabilidad con la que será entregado. Los modelos mas conocidos son los basados en los procesos de Poisson no homogéneos. Dado que los procesos de nacimiento son procesos estocásticos de crecimiento de una población, los mismos pueden ser utilizados para modelar la producción de fallas en software, como se ha venido desarrollando en las últimas décadas, [5], [4], [7], [9], [11], [15]. Muchos de los modelos de Confiabilidad de Software propuestos son casos particulares de procesos de nacimientos, como ocurre con los procesos de Poisson no homogéneos. El análisis en general de los procesos de nacimientos puros permite formular nuevos modelos de confiabilidad con características interesantes tales como tasas de fallas crecientes o decrecientes o con forma S (S-shaped). En este trabajo se presenta un resumen de la aplicación de los procesos de nacimientos puros y se muestran simulaciones con distintos casos de tasa de nacimientos que pueden resultar muy útiles para modelar fallas en software. Este trabajo está organizado de la siguiente manera: en la sección 2 se da una introducción a los procesos de nacimientos, en la sección 3 se presenta una lista de los modelos basados en los procesos no homogéneos de Poisson vistos como procesos de nacimientos, en la sección 4 se analiza la aplicación de los procesos de nacimientos en Confiabilidad de Software, en la sección 5 se muestran simulaciones de procesos de nacimientos puros y se comparan los resultados con reportes de fallas reales, finalmente, en la sección 6 se presentan las conclusiones.

2. Procesos de nacimientos puros

Los procesos de nacimientos puros son descriptos por la siguiente ecuación general, ver [8]:

$$P_r'(t) = -\lambda_r(t)P_r(t) + \lambda_{r-1}(t)P_{r-1}(t) \tag{1}$$

donde $P_r(t)$ es la probabilidad de que haya r individuos en la población en el instante t. El proceso (1) es también un proceso de Markov donde el número de individuos corresponde al estado del sistema $S_r(t)$.

La probabilidad de no tener nacimientos en un intervalo mayor que $t-s$ dado que el sistema esta en el estado r en el instante s esta dada por la exponencial:

$$P(\text{no nacimientos en } t \geq t-s) = e^{-\int_s^t \lambda_r(t)dt} \tag{2}$$

la última expresión es la bien conocida ley de tiempo de espera exponencial. Distintos casos particulares devienen dependiendo que $\lambda_r(t)$ dependa solo de r, t o de ninguno como se explica en las secciones siguientes.

3. Procesos no homogéneos de Poisson

El proceso de Poisson es un caso particular de proceso de nacimientos, aparece cuando $\lambda_r(t)$ es constante en el tiempo $\lambda_r(t) = \lambda$, los casos de Poisson no

homogéneos aparecen cuando la tasa de nacimientos es una función no lineal del tiempo $\lambda_r(t) = \lambda(t)$:

$$P((N(s) - N(t)) = r) = \frac{\mu(t,s)^r}{r!} e^{-\mu(t,s)} \tag{3}$$

donde:

$$\mu(t,s) = \int_s^t \lambda(t)dt \tag{4}$$

Diferentes modelos de Confiabilidad en Software basados en procesos no homogéneos de Poisson han sido propuestos en la bibliografía, como se muestra en la tabla 1.

Tabla 1. Modelos no homogéneos de Poisson

Modelo	$\mu(t,0)$
Goel-Okumoto	$a(1 - e^{-bt})$
Musa-Okumoto	$\frac{1}{\theta}\ln(\mu_0\theta t + 1)$
Delayed S-shaped	$a(1 - (1 + b\ t)\mathrm{e}^{-b\ t})$
Log Power	$\alpha \ln^{\beta}(1 + t)$
Gomperts	$a(b^{c^t})$
Yamada Exponencial	$a(1 - \mathrm{e}^{-b\ c\ (1-\mathrm{e}^{(-d\ t)})})$

4. Procesos de nacimientos puros en Confiabilidad de Software

Los procesos de nacimientos puros han sido largamente aplicados en Confiabilidad de Software desde hace varias décadas. Originalmente, en [11], se generaliza el modelo de Jelinski & Moranda ([10]) y se propone una tasa de fallas que depende proporcionalmente del número de fallas encontrado, y a su vez es una función no lineal del tiempo. En [7], [13] y [15], se analiza el modelo de Moranda como un proceso de nacimientos puros con una tasa de fallas que depende geométricamente del número previo de fallas y es independiente del tiempo. En [9], se propone una simulación de producción de fallas con procesos de Poisson no homogéneos continuos en el tiempo que se analizan a su vez como procesos de nacimientos donde la tasa de fallas depende no linealmente del tiempo, lo que da lugar a los llamados procesos no homogéneos de Markov continuos en el tiempo (NHCTMC). En [4] y [5] se propone un proceso de nacimientos puro donde la tasa de fallas depende no linealmente del número de fallas detectadas previamente y no depende del tiempo, se propone además utilizar el estimador empírico de Bayes para estimar la tasa de fallas.

De la expresión (2), se puede deducir que toda vez que se modele la producción de fallas como un proceso de nacimientos puros, el tiempo medio entre fallas se calcula como:

$$mtbf(r,t) = \frac{1}{\lambda_r(t)} \tag{5}$$

Cuando se intenta modelar no solo el proceso estocástico de detección de fallas sino también la reparación, suelen utilizarse modelos basados en procesos de nacimientos y muertes, ver por ejemplo [11].

5. Simulación de producción de fallas con tasas generales

En esta sección se presentan simulaciones de procesos de nacimientos con distintas funciones de tasa de fallas y se los compara con reportes de fallas reales. Las simulaciones fueron realizadas en Wolfram Mathematica 8 y consisten en generar nacimientos a intervalos dados por tiempos de espera aleatorios según 2.

5.1. Modelo geométrico

Esta simulación corresponde al modelo geométrico presentado en [13] como una forma a su vez de analizar el modelo de deseutratización de Moranda. La tasa de fallas está dada por:

$$\lambda_r = a\, b^r$$

De las simulaciones se observa que la curva de detección de fallas es muy sensible a los parámetros a y b, obteniendo en muchos casos tasas de fallas crecientes con el tiempo. En [13] se eligen como ejemplo los valores $a = 2$, $b = 0{,}4$. En la fig. 1 se muestra el resultado de la simulación de este modelo hasta el día 150, más extendido que los datos mostrados en ([13], fig. 3), cuyos datos se muestran hasta el día 30. Se puede observar la coincidencia de ambas curvas en ese rango de tiempo.

5.2. Modelo de Yule

En esta sección se presenta la simulación del proceso de Yule (ver [8]), en el cual la tasa de nacimientos es proporcional al número de individuos en la población, lo cual corresponde al número de fallas ya detectado. Dado que este caso presenta una disminución del tiempo medio entre fallas, sólo puede ser utilizado en una primera etapa del proyecto, donde se espera un aumento en la tasa de fallas, o bien cuando se produce un agregado de nuevo software. Usualmente, los casos de tasas de fallas crecientes inicialmente, se modelan con el modelo S-shaped, ver por ejemplo [16, pp. 393-402] y [1].

Seguidamente, se muestra la simulación del proceso de Yule con una tasa de fallas dada por:

$$\lambda_r = 0{,}001\, r$$

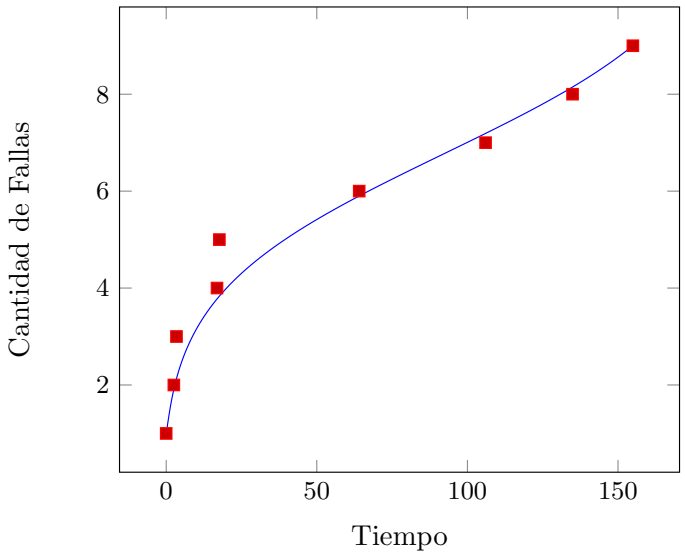

Figura 1. Modelo geométrico

y se compara con el número de fallas reportado por el proyecto Bugzilla gnome v. 2.2 desde el 07-01-2014 al 07-31-2015[1], como se muestra en la fig. 2.

Si bien se observa un apartamiento de la curva del modelo con los datos reales, se puede ver que la simulación sigue la curvatura. Esto último se puede observar al comparar los tiempos reales entre fallas con el obtenido de la simulación, lo cual se muestra en la fig. 3. Se puede observar que el resultado de la simulación sigue el decrecimiento de la confiabilidad de los datos reales de las primeras 10 fallas detectadas, luego, ambas curvas muestran una tasa de fallas que se mantiene en alrededor de una falla cada 5 días entre las fallas 10 y 25, a partir de la falla número 25, los datos reales muestran un ligero crecimiento de la confiabilidad, lo cual no se observa en la simulación, excepto por el caso aislado de la falla número 29.

5.3. Modelo de Contagio de Polya

En general, los modelos de Confiabilidad de Software proponen una tasa de fallas decreciente modelando un crecimiento de la confiabilidad, dado que se espera que el número de fallas existente en el software disminuya. No obstante, hay casos al comienzo del proyecto o situaciones como el agregado de nuevo software que hacen que temporalmente la tasa de fallas se incremente. Existen pocos modelos que consideren una tasa de fallas creciente, uno de ellos es el

[1] https://bugzilla.gnome.org/

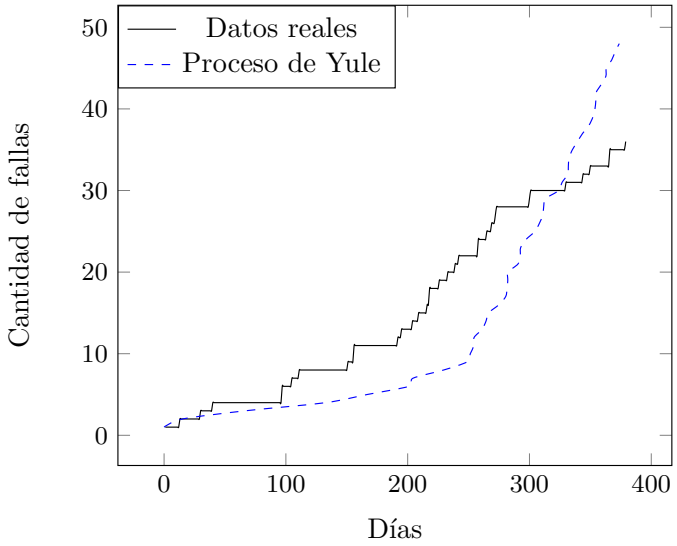

Figura 2. Datos del proyecto Bugzilla gnome v 2.2

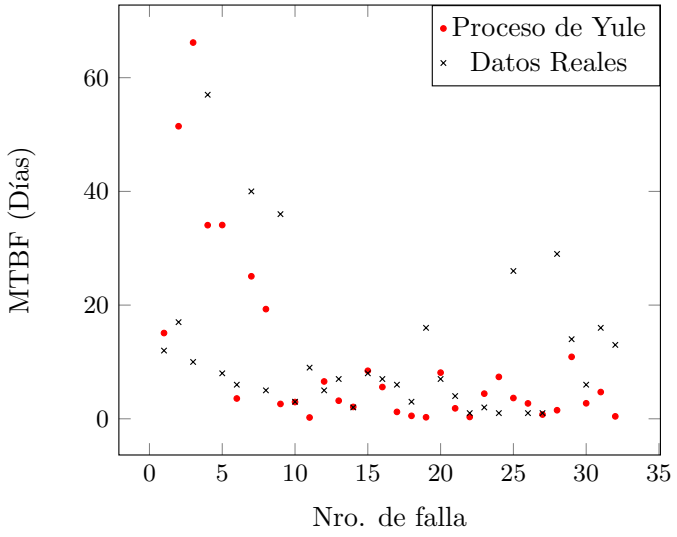

Figura 3. Datos del proyecto Bugzilla gnome v 2.2. (Tiempo entre fallas) MTBF.

propuesto por Shick y Volverton, ver por ejemplo un análisis de este modelo en [6].

El proceso estocástico de Polya puede ser utilizado para modelar casos de tasas de fallas creciente. Este proceso estocástico surge del modelo de urna de Polya y ha sido aplicado en estudios de Confiabilidad y seguridad, ver por ejemplo [14]. El modelo de urna de Polya es un modelo de contagio y consiste en realizar extracciones con reposición en una urna compuesta de bolas de cierto color de manera que se repone la bola extraída mas un dado número de bolas del mismo color, aumentando la probabilidad de extraer el mismo color en la siguiente extracción, ver por ejemplo [8]. Al considerar que se realizan extracciones en determinados intervalos de tiempo, se obtiene como caso límite el proceso estocástico de Polya, análogamente como se obtiene el proceso de Poisson a partir de la distribución de Poisson. El proceso estocástico que da la probabilidad de un cierto número de fallas en función del tiempo está dado por la siguiente expresión, (ver por ejemplo [3] y [8]):

$$P_r(t) = \frac{\Gamma(\frac{\lambda}{\rho} + r)}{r!\ \Gamma(\frac{\lambda}{\rho})} \left(\frac{\rho t}{1+\rho t}\right)^r \left(\frac{1}{1+\rho t}\right)^{\frac{\lambda}{\rho}} \tag{6}$$

El proceso estocástico de Polya resulta interesante dado que depende de dos parámetros, la tasa de nacimientos (fallas) λ y la tasa de contagio ρ, de manera que reemplazando (6) en la ecuación (1). se puede verificar que $\lambda_r(t)$ queda expresado como:

$$\lambda_r(t) = \rho\ \frac{r + \frac{\lambda}{\rho}}{1 + \rho\ t} \tag{7}$$

De (7) se puede observar que la tasa depende de un factor cuyo numerador corresponde a un contagio y su denominador corresponde a un término de crecimiento de la confiabilidad, que hace disminuir la tasa con el tiempo. El ajuste de algún caso real con el modelo de Polya puede resultar interesante dado que permitiría concluir que existe algún fenómeno de contagio en las fallas del software, información que podría utilizarse para analizar como fueron causadas esas fallas en el proceso de desarrollo de software o como han sido detectadas en el proceso de debugging y testing. Otro dato interesante para hacer notar de la expresión (7) es que de acuerdo con la expresión (4), el valor medio del número de fallas resulta:

$$\mu_r(t, 0) = \left(\frac{\lambda}{\rho} + r\right)\ \ln(1 + \rho\ t) \tag{8}$$

de la expresión (8) y de la tabla 1 se observa que el factor correspondiente al crecimiento en la confiabilidad del proceso de Polya coincide con el del modelo de Musa-Okumoto. Esto da lugar a proponer modelos cuya tasa de fallas contenga un factor de contagio similar al de Polya y un factor de crecimiento de la confiabilidad como alguno de los propuestos en la tabla 1.

La simulación del proceso de Polya con parámetros $\rho = 0{,}5$, $\lambda = 0{,}001\ \rho$ se muestra en la fig. 4 y se compara con el proyecto T1 reportado en [12]. Este proyecto corresponde a un sistema en tiempo real de un total de 21700 instrucciones desarrollado por 9 programadores donde las fallas fueron registradas en las etapas de testing y operación. Si bien los datos de este proyecto son de hace varias décadas y en ese lapso se produjo un gran avance en técnicas de Ingeniería de software, se considera relevante su utilización dado que el mismo fue muy analizado en la literatura y además el resultado presentado en este trabajo puede compararse con el realizado en [1], donde se modela el incremento inicial de la tasa de fallas con el modelo S-shaped (ver [2], fig. 8)[2]. Si bien el modelo S-shaped ajusta mejor los datos de fallas para este proyecto, se puede observar que el proceso de Polya muestra un salto en el número de fallas en la semana 8, al igual que los datos reales. En general, el proceso de Polya tiene esta característica, la de provocar un cambio abrupto en algún instante de tiempo, que no es el caso del proyecto T1, el cual tiene una tasa casi constante al principio y un cambio mas gradual en la semana 8.

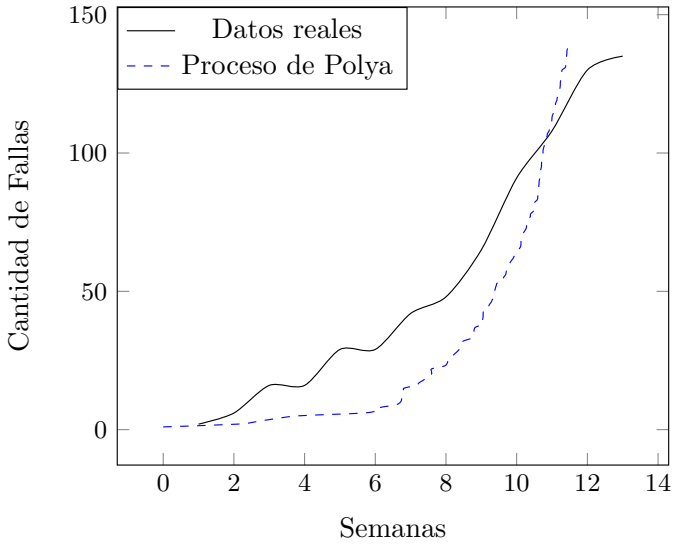

Figura 4. Datos reales y simulación de Polya del proyecto T1.

[2] Se aclara que en la referencia citada [2], se considera que los datos fueron reportados en el lapso de 21 semanas, no obstante, de acuerdo a los datos que disponen los autores según un reporte enviado en una comunicación privada desde "The Data and Analysis center for Software (DACS)", el último día de falla es el número 96, correspondiendo a la semana 14 considerando 7 días por semana.

En la fig. 5 se muestran los datos de tiempo entre fallas para el proyecto T1. Se puede observar que los tiempos entre fallas dados por el proceso de Polya siguen aproximadamente los datos reales. Análogamente a lo analizado en la fig. 3, en los datos reales se observan tres etapas, un decrecimiento de la confiabilidad hasta la falla número 60, una tasa de fallas casi constante hasta la falla número 130, y luego, un aumento de la confiabilidad. Se puede observar que la simulación sigue el comportamiento de los datos reales en las primeras dos etapas.

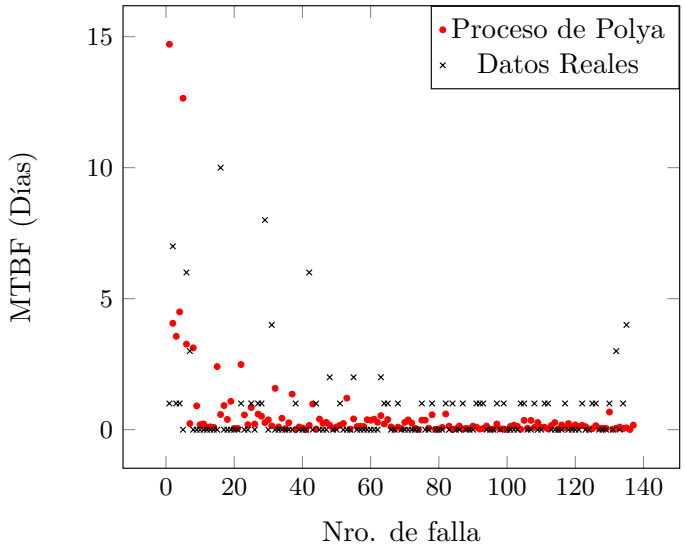

Figura 5. MTBF del proyecto T1.

6. Conclusión

En este trabajo se ha mostrado como los procesos de nacimientos puros han sido utilizados en Confiabilidad de Software en distintos casos desde hace varias décadas. Se han mostrado resultados de simulaciones de producción de fallas basado en algunos procesos conocidos como el de deseutratización de Moranda, el de Yule y el de Polya. En el caso del proceso de Yule, se ha comparado la curva obtenida con los datos de producción de fallas reportados en el proyecto Bugzilla gnome v 2.2. En el caso del proceso de Polya, se ha mostrado una aplicación a un conocido proyecto de software de la literatura de Confiabilidad y se comparan los resultados con la aplicación del modelo S-shaped. Si bien las curvas obtenidas muestran un apartamiento significativo de los datos reales, la curvatura sigue la de los datos reportados, lo cual se observa en los tiempos entre fallas. Estos casos

resultan además de interés dado que presentan la particularidad de una tasa de fallas creciente, lo cual permite utilizarlos al comienzo de la etapa de debugging o bien cuando se realiza el agregado de nuevo software. En trabajos futuros, se verificará el ajuste de estos modelos con mas datos de fallas reales.

Agradecimientos

Los autores agradecen a la Universidad Nacional de Tres de Febrero por financiamiento correspondiente al subsidio no. 32/15 201 correspondiente al proyecto "Aplicación, Análisis y Desarrollo de Modelos Estadísticos de Confiabilidad de Software, Pruebas y Programación en a nube".

Referencias

1. Ahmad, N., Imam, M.Z.: Article: Software reliability growth models with log-logistic testing-effort function: A comparative study. International Journal of Computer Applications 75(12), 6–11 (August 2013), full text available
2. Ahmad, N., Khan, G.M., Rafi, L.S.: Analysis of an inflection s-shaped software reliability model considering log-logistic testing-effort and imperfect debugging. International Journal of Computer Science and Network Security 11(1), 161–171 (2011), http://paper.ijcsns.org/07_book/201101/20110125.pdf
3. Barraza, N.R.: Aplicaciones y Análisis de Fenómenos Cooperativos utilizando Mecánica Estadística, Contagio y Cadenas de Eventos Raros. Ph.D. thesis, UBA. Facultad de Ingeniería (1998), http://web.fi.uba.ar/~nbarraz/publications_files/tesis.pdf
4. Barraza, N.R.: A new homogeneous pure birth process based software reliability model. In: Proceedings of the 38th International Conference on Software Engineering Companion. pp. 710–712. ICSE '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2889160.2892645
5. Barraza, N.R.: A parametric empirical bayes model to predict software reliability growth. Procedia Computer Science 62, 360 – 369 (2015), http://www.sciencedirect.com/science/article/pii/S187705091502551X, proceedings of the 2015 International Conference on Soft Computing and Software Engineering (SCSE'15)
6. Bendell, A., Mellor, P.: Software Reliability: State of the Art Report 14:2. State of the art report, Elsevier Science (2014), https://books.google.com.ar/books?id=sESeBQAAQBAJ
7. Boland, P.J., Singh, H.: A birth-process approach to Moranda's geometric software-reliability model. IEEE Transactions on Reliability 52(2), 168–174 (June 2003)
8. Feller, W.: Introducción a la teoría de probabilidades y sus aplicaciones: Volumen I. Instroducción [sic] a la teoria de probabilidades y sus aplicaciones, Limusa (1980), https://books.google.com.ar/books?id=1z18rgEACAAJ
9. Gokhale, S.S., Lyu, M.R., Trivedi, K.S.: Incorporating fault debugging activities into software reliability models: a simulation approach. IEEE Transactions on Reliability 55(2), 281–292 (June 2006)
10. Jelinski, Z., Moranda, P.: Software reliability research. In: Freiberger, W. (ed.) Statistical Computer Performance Evaluation. pp. 465–484. Academic Press, New York (1972)

11. Kremer, W.: Birth-death and bug counting. IEEE Transactions on Reliability R-32(1), 37–47 (April 1983)
12. Musa, J.D., Iannino, A., Okumoto, K.: Software Reliability: Measurement, Prediction, Application. McGraw-Hill, Inc., New York, NY, USA (1987)
13. Sandhu, P.S., Kamra, A., Singh, H.: A recursive method for reliability computation of Moranda's geometric software reliability model. Proceedings of World Academy of Science, Engineering and Technology 26, 720–725 (2007)
14. Thompson, W.: Point Process Models with Applications to Safety and Reliability. Springer US (2012), https://books.google.com.ar/books?id=-JLbBwAAQBAJ
15. Vasanthi, T., Arulmozhi, G.: Reliability computation of Moranda's geometric software reliability model. Economic Quality Control 22(2), 261–272 (2007), http://EconPapers.repec.org/RePEc:bpj:ecqcon:v:22:y:2007:i:2:p:261-272:n:9
16. Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J.: Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30 – July 4, 2014, Brunów, Poland. Advances in Intelligent Systems and Computing, Springer International Publishing (2014), https://books.google.com.ar/books?id=NToqBAAAQBAJ