Reliability Models Applied to Mobile Applications

Sonia Meskini, Ali Bou Nassif, Luiz Fernando Capretz
Department of Electrical and Computer Engineering
University of Western Ontario
London, Ontario, N6A 5B9, Canada
{smeskini, abounas, lcapretz}@uwo.ca

***Abstract*—: Smartphones have become the most used electronic devices. They carried out most of the functionalities of desktops, allowing various useful applications that suit the users' needs. Therefore, instead of the operator, the user has become the number one controller of the device and its applications and thus its reliability becomes an emergent need. We aim to investigate and evaluate the efficacy of Software Reliability Growth Models (SRGMs) when applied to Smartphone application failure data and check whether they achieve the same success as in the desktop/laptop area. We selected three of the most used SRGMs and applied them to three different Smartphone applications. None of the selected models were able to account for the data satisfactorily. Their failure is traced back to the specific features of mobile applications compared to desktop applications. Thus, a suitable model for Smartphone applications is still needed to improve their reliability.**

Keywords—Smartphone applications; software reliability; NHPP; Musa-Basic; Musa-Okumoto model; SRGM model.

I. INTRODUCTION

Smartphones are so important nowadays and they are overselling PCs. The high usage and trust in these devices and their applications make their reliability a critically important goal to achieve. Thus, manufacturers are competing to release the most reliable devices and they successfully achieved high reliability in terms of hardware by applying traditional and enhanced Hardware Reliability Growth Models (HRGM) [1]. These HRGMs have been useful for classic mobile phones.

Today's cellular phones, called Smartphones, are almost like pocket PCs, meaning that their functionalities far exceed those of the classic mobile phones. This kind of change requires Smartphone application developers to pay more attention and spend more effort on the reliability and security of their applications. The rapid increase in developing smart phones relies on Software Product Lines (SPL). In SPL, software is not re-developed from scratch where similar products can be developed using common set of core assets [2][3].

Software Reliability deals with *"the probability that software will not cause the failure of a system for a specified time under a specified condition. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection"* [4]. Reliability is an important attribute of software quality in addition to other attributes such as performance, usability and fault prediction [5]. Software testing techniques can impact reliability measurement [6]. Software Reliability Growth Models (SRGMs) have been successfully applied to desktop (classic/standard) applications to assure high reliability, as was the case with Hardware Reliability Growth Models applicability to both desktops/laptops and mobile phone hardware [1]. We suspect SRGM is similarly applicable to both desktops/laptops and Smartphones. To the best of our knowledge, there is no published work that addressed the application of the usual SRGMs to Smartphone applications. In the present paper, we attempt to answer the research question "Is it possible to successfully apply SRGMs to Smartphone applications as it is the case for desktop applications? How accurate are they and what are the challenges?"

The rest of the paper is organized as follows: in Section II we stress the need for the reliability of Smartphone applications. In Section III we provide a short roadmap of the existing models that we will use. We devote section IV and Section V to describe our dataset and experiments. Finally, in Section VI, we evaluate the application of the existing models to Smartphone application failure data. We conclude, in Section VII, and outline our future work.

II. RELIABILITY FOR SMARTPHONE APPLICATIONS

Smartphones have become personal devices that are used almost anywhere, at any time and for everything; checking e-mails, gas prices, browsing the Internet, banking and even for health services. These high expectations and trust placed on the mobile environment make it useful more than just a phone and exceeds the functionalities of only sending SMS or making voice calls. A few years ago, Smartphones use was only limited to business. Lately, thanks to network and mobile technology improvements and progress, those smart devices started to gain tractions and they got a remarkable acceptance in the users market. Since then, it has been a significant increase in the Smartphone technology. Thus, countless Smartphone applications have been designed and developed. In step with this, the market has been rapidly growing, and market analysis confirmed that it will continue growing to the point that it will exceed the desktop/laptop growth and oversell many other electronic devices in particular laptops [7].

As a result, to assure the continuing growth in this competitive market, there are various types of customers' needs to be satisfied because, based on previous studies, those needs differ from one region to another and from one user to another, thus, here is where the reliability of Smartphone applications will play an important role to keep the trust in one's device.

Nevertheless, the quality and achievement levels of Smartphone applications are falling since countless applications have been increasingly developed. This is a consequence of the “time to market” strategy that Smartphone applications development companies are adopting nowadays, which let the developers overlook development phases (especially the design phase which is considered as the most important stage) of the Development Life Cycle (DLC) of the application to meet project deadlines [8]. However, multiple defects and bugs might be avoided during the design phase. Hence, following this strategy could result in detecting many problems later on that might need more effort and time to be solved than those needed to develop the entire application.

In step with this, in order for companies to be competitive, it is important to study the market, based on surveys and other analysis, to understand that locations are one of the important factors in the variety of users’ requirements. Therefore, a competitive Smartphone application must meet these requirements before, during and after the DLC.

However, some of those requirements might be a challenge for developers in the design stage and their resulting failures might be difficult to solve in the execution stage since it is sometimes difficult to identify the cause of the failure and fix it. The reason in that there are various factors that could result in the unreliability of the application or its failure such as the nature of the technology used, the platform, the version of the OS, and many other internal or external causes [9][10].

In the following, we will provide a short roadmap on the most famous SRGMs that have been successfully applied in classic (standard) desktop/laptop applications and check whether they will achieve the same success in the mobile area.

III. Existing Reliability Models

In this section we will shortly present three different SRGMs that will be used in our experiments; the Non-Homogenous Poisson Process (NHPP) model, the Musa-Basic model and the Musa-Okumoto model.

As a matter of fact, hundreds of models exist but those models were chosen because according to Software Reliability modeling survey [11][12], they are the most useful and successful models in the computer applications domain. Hence, one of our research goals is to check if those models will achieve the same success when applied to Smartphone applications failure data as it was the case for the desktop and Hardware Growth Reliability Models [1].

The purpose behind developing models is the measurement, estimation and prediction of Software Reliability which has become an important major target for companies because it was shown reliability has a significant effect in each stage of the DLC of an application; from the design to the maintenance [2].

An SRGM usually results in a set of mathematical equations that fit accurately the collected failure data [12]. Any model relies on simplifying assumptions. However, some of these assumptions may not be useful in real situations. Table 1 from [2] presents an assessment of the most used assumptions and their conformity to real observations.

Due to the limited space of this paper we are not going to present all the assumptions of the different models used later but only the basic ones that are shared by all the models. The detailed list of the different assumptions of each model can be found in [11].

The common assumptions are: *1) “The rate of fault detection is proportional to the current fault content of the software, 2) The fault detection rate remains constant over the intervals between fault occurrence, 3) A fault is corrected instantaneously without introducing new faults into the software, 4) The software is operated in a similar manner as that in which reliability predictions are to be made, 5) Every fault has the same chance of being encountered within severity class as any other fault in that class, 6) The failures, when the faults are detected, are independent”* [11].

A. The NHPP Model

The Non-Homogeneous Poisson Process (NHPP) model was presented by Amrit Goel and Kazu Okumoto in 1979 [11]. In this model, the failure event is modeled by an NHPP distribution where it is assumed that there exists a mean value function giving the expected number of failures up to a given time. It was successfully used as a Hardware Reliability Growth Model. Moreover, for this model the expected number of failure varies with time, thus, it suits the calendar and execution time data [13].

Because of its simplicity and easy implementation, there were several models that have been developed based on the NHPP model. In addition to the above mentioned assumptions, termed the standard assumptions, the added assumption is that the probability distribution obeyed by the random variable N(t) follows a Poisson Process, i.e. is given by :

$$\boldsymbol{Pr[N(t) = n] = \frac{\mu(t)^n}{n!} e^{-\mu(t)}} \qquad (1)$$

where N(t) is the cumulative number of failure by time t and μ(t) is the mean value of N(t) or the expected cumulative number of failure :

$$\boldsymbol{\mu(t) = E(N(t))} \qquad (2)$$

The instantaneous failure intensity is defined as:

$$\boldsymbol{\lambda(t) = \frac{d\mu(t)}{dt}} \qquad (3)$$

The NHPP model implemented in the RGA7 tool is the NHPP-Crow model [14] where the probability density function for the failure time is given by:

$$\boldsymbol{f(t) = \lambda(t)e^{-\mu(t)} = \lambda\beta t^{\beta-1}e^{-\lambda t^{\beta}}} \qquad (4)$$

where $\lambda > 0$; $\beta > 0$ are the two parameters of the model.

Finally, to implement this model, either the fault counts or the time between failures are required [11]. A detailed study of NHPP model can be found in [11][13].

B. Musa’s basic execution tim Model

The Musa-Basic model, also termed the exponential model, is given by the following mean value [11]:

$$\boldsymbol{\mu(t) = \beta_0^E \left(1 - e^{-\beta_1^E t}\right)} \qquad (5)$$

where $\boldsymbol{\beta_0^E}$: is the expected number of failures and $\boldsymbol{\beta_1^E}$ is the hazard rate or in other words “the amount that each fault contributes to the overall failure rate”.

This model is used especially for execution time data but it can also be applied to calendar time data by applying a conversion from calendar to execution time. The required data to build this model are either the time of failure or time between failures.

Based on the Software Reliability modeling survey from the Handbook of SRE [11], this model is considered as one of the most widely used models.

There are several similar models that have been developed. Moreover, Musa mentioned that *"the basic execution model generally appears to be superior in capability and applicability to other published models"* [15].

C. Musa-Okumoto logarithmic Poisson Model

The Musa-Okumoto model, also termed the logarithmic model, is one of the most extensively applied models [11][12]. Besides that, Musa himself confirmed that this model is more accurate comparing to the exponential model [15].

As for the previous models, the mean value is extracted from the model's proper assumptions [11] and given by:

$$\mu\ (t) = \beta_0^L\ Ln\left(1 + \ \beta_1^L t\ \right) \qquad (6)$$

where β_0^L is the expected number of failures and β_1^L is the hazard rate.

The required data to build this model are the same as for the exponential model.

As one of the best predictive models, the Musa-Okumoto model belongs to the selected models in the AIAA Recommended Practice Standard on Software Reliability [11][12]. Logarithmic models have been also used in software cost estimation models with high accuracy [16][17][18][19].

Further details on the Musa-Basic and Musa-Okumoto models can be found in [11]. Various models were proposed and developed; however, they may give different results and predictions for the same failure dataset. Besides that, one model applied on two different datasets may give good results for the first but confusing results for the second dataset. This makes it difficult to choose the best model to fit the data [13]. Hence, none of the models can be classified as absolutely perfect or better than the other. However, the models presented in this section are considered to be the most accurate applied models on a variety of software projects [12]. Thus, these three models are selected for our study.

IV. Datasets

We use Apple devices (iPhone, iPad, iPod Touch) crash files as well as a Windows Phone crash file as our "experimental" data.

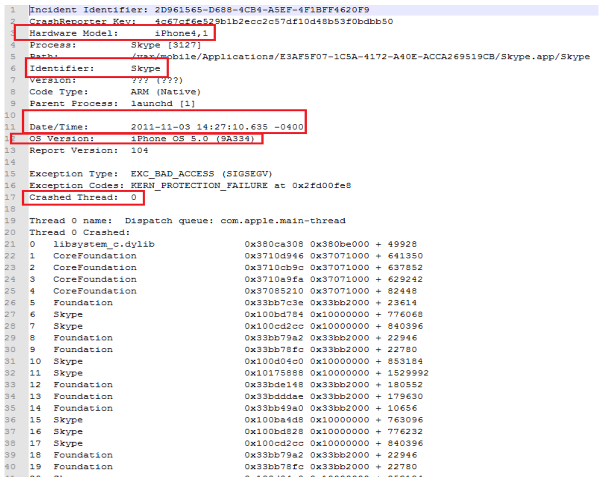
These crash files are not public, therefore confidential. Hence, we will focus more on the Apple devices crash files since it was easier to collect them from my personal device and through a survey that has been sent to different people from different parts of the world. There are those who gratefully accepted to send us their failure data whereas other didn't.

For the Windows Phone case, we could only get the crash file report of one application due to confidentiality policies. Collecting the data was, and still, a challenge.

Fig. 1 presents an example of the Apple devices crash log. We mentioned in each case:

- Name of the crashed application
- Type
- Hardware type (device an iPhone, iPad or iPod Touch) this information is needed to check whether the crash is of an application of the same device or same application from a different device.
- Date/Time of the crash (which is the most important information in the crash log for our research work)
- The version of the OS.

The crash logs of Apple devices are transferred to a hidden folder located/created in the PC that is used for the synchronization of the device. It contains the crash logs of all the applications installed on the device as well as reports about the battery, memory and other features; however we are interested in the crash files. Thus, we ignored the other files.



```
Incident Identifier: 2D961565-D688-4CB4-A5EF-4F1BFF4620F9
CrashReporter Key:   4c67cf6e529b1b2ecc2c57df10d48b53f0bdbb50
Hardware Model:      iPhone4,1
Process:         Skype [3127]
Path:            /var/mobile/Applications/E3AF5F07-1C5A-4172-A40E-ACCA269519CB/Skype.app/Skype
Identifier:      Skype
Version:         ??? (???)
Code Type:       ARM (Native)
Parent Process:  launchd [1]

Date/Time:       2011-11-03 14:27:10.635 -0400
OS Version:      iPhone OS 5.0 (9A334)
Report Version:  104

Exception Type:  EXC_BAD_ACCESS (SIGSEGV)
Exception Codes: KERN_PROTECTION_FAILURE at 0x2fd00fe8
Crashed Thread:  0

Thread 0 name:  Dispatch queue: com.apple.main-thread
Thread 0 Crashed:
0   libsystem_c.dylib             0x380ca308 0x380be000 + 49928
1   CoreFoundation                0x3710d946 0x37071000 + 641350
2   CoreFoundation                0x3710cb9c 0x37071000 + 637852
3   CoreFoundation                0x3710a9fa 0x37071000 + 629242
4   CoreFoundation                0x37085210 0x37071000 + 82448
5   Foundation                    0x33bb7c3e 0x33bb2000 + 23614
6   Skype                         0x100bd784 0x10000000 + 776068
7   Skype                         0x100cd2cc 0x10000000 + 840396
8   Foundation                    0x33bb79a2 0x33bb2000 + 22946
9   Foundation                    0x33bb78fc 0x33bb2000 + 22780
10  Skype                         0x100d04c0 0x10000000 + 853184
11  Skype                         0x10175888 0x10000000 + 1529992
12  Foundation                    0x33bde148 0x33bb2000 + 180552
13  Foundation                    0x33bdddae 0x33bb2000 + 179630
14  Foundation                    0x33bb49a0 0x33bb2000 + 10656
15  Skype                         0x100ba4d8 0x10000000 + 763096
16  Skype                         0x100bd828 0x10000000 + 776232
17  Skype                         0x100cd2cc 0x10000000 + 840396
18  Foundation                    0x33bb79a2 0x33bb2000 + 22946
19  Foundation                    0x33bb78fc 0x33bb2000 + 22780
```

Fig. 1. Apple crash file

Besides that, the crash log is a long text file full of symbols and information that we don't need, however it contains useful information that we used to create our failure dataset. To achieve that, we developed a program in JAVA that we run each time we synchronize the devices or receive log folders from other users to update our dataset. The following algorithm allows extracting only the information we need.

1) *Begin*

2) *Open the folder that contains all the crash logs*

3) *Create "Concat.txt" that contains all the crash files*

4) *Create "Crash.txt" that contains only the information needed extracted from "Concat.txt" :*

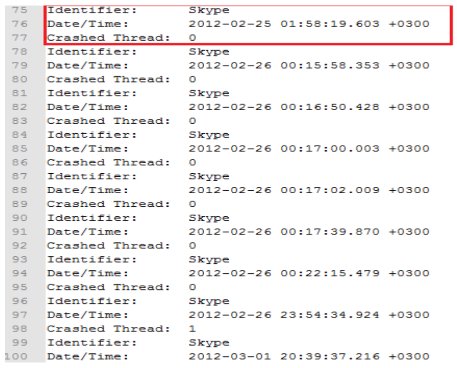
   a. *Identifier*

   b. *Date/Time*

   c. *Crashed Thread*

5) *End/Close*

Fig. 2 shows an example of the output file of the JAVA program developed for the extraction purpose, where *Identifier* is the name of the application. *Date/Time* is the date and time of the crash and *Crashed Thread* is the number of the thread that caused the crash.



```
Identifier:        Skype
Date/Time:         2012-02-25 01:58:19.603 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 00:15:58.353 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 00:16:50.428 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 00:17:00.003 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 00:17:02.009 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 00:17:39.870 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 00:22:15.479 +0300
Crashed Thread:    0
Identifier:        Skype
Date/Time:         2012-02-26 23:54:34.924 +0300
Crashed Thread:    1
Identifier:        Skype
Date/Time:         2012-03-01 20:39:37.216 +0300
```

Fig. 2. Output of the JAVA program

V. Experiments

The reliability demonstration of Smartphone applications is carried out through the traditional testing, failure data collection, and the application of the most used SRGMs for standard applications to observe and check the adequacy of these models, in the mobile area for Smartphone applications.

For this purpose we used two applications for iOS and one for Windows mobile phone to test the models with different platforms. We couldn't collect enough data from Android phones but we are still collecting to have enough data to test the models on Android applications. The first iPhone application is one of the most popular applications in communication, Skype, which has been tested and used for a year (from 01 Nov. 2011 to 11 Nov. 2012). Hence, the data have been collected during a year with some missing values due to the non-use of the application occasionally. Therefore, we were able to collect 46 data points. The second application is Vtok (an application for Google talk). This application has been continuously used during two months (from 19 Sep. 2012 to 25 Nov. 2012). Hence, we were able to collect failures every day (80 data points). Each of the above mentioned SRGM models was applied to Skype and Vtok failure data which represent two different situations: Skype used during a year but with some missing values, and Vtok application used for two months every day with the possibility of collecting more than one failure per day. This is an instance of testing the efficiency and goodness of the models in different situations with different types of data. On the other hand, the Windows phone application, for finding bicycle stations, was continuously used and tested for six months (from March 2012 to August 2012). The crash count of the application is illustrated by Fig. 3.

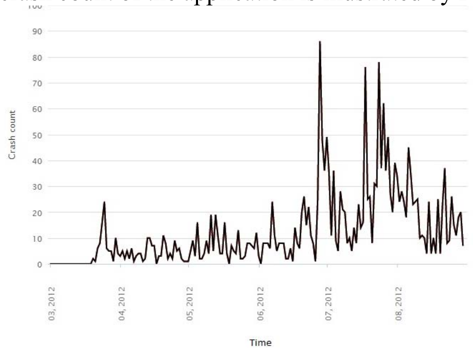


Fig. 3. Windows phone crash count

It is evident to note that June, July and August are the months with the highest crash rate. Since this application is developed for the purpose of locating bicycle stations, it is used during the summer period more than in winter, which explains the high crash rate during the hot season. This reflects the fact that the type of an application and its usage play an important role in its reliability. From this graph we extracted the failure data during six months (177 data points).

We used two Software Reliability tools for this application to check the results. The first tool is the RGA 7 from ReliaSoft and the second one is SMERFs (**S**tatistical **M**odeling and **E**stimation of **R**eliability **F**unctions for **S**oftware). We configured our tools as follows: "1" for the severity level of all failures and the unit selected is hour. As the time scales of the three applications are very different, we choose to normalize our data between $Z_i[0,1]$ using the following equations:

Y_i = value of the raw target variable Y for the training case Z_i = standardized value corresponding to Y

$$\textbf{\textit{Range = upper bound of Y – lower bound of Y}} \qquad (7)$$

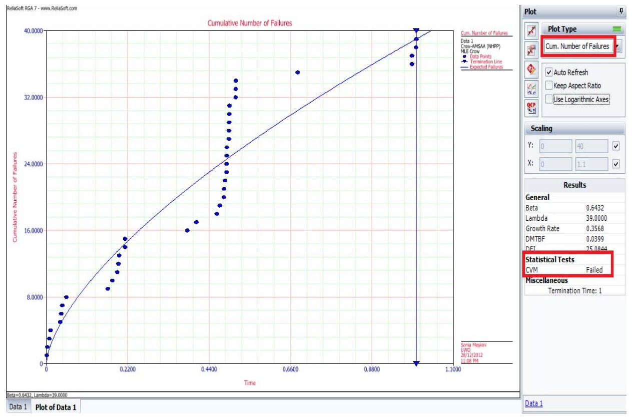
$$Z_i = \frac{Y_i - \text{lower bound of } Y}{\text{range}} \qquad (8)$$

As the RGA tool doesn't accept the zero value as a time to event, we entered 0.001 instead of 0 as the first value to be able to have results. For the severity, 1 was selected because the applications used are not going to cause harmful consequences if they fail. But it is not the case with other applications. When working with applications such as online banking and health, the severity of the failure has to be taken into consideration.

VI. Evaluation

Smartphone applications reliability is a challenge. Thus, it is a necessity for reliability methods being applied elsewhere to be evaluated and to assess their validity in the mobile area. One of the main goals of this work is to check if the most accurate and used SRGMs for desktop applications have the same accuracy when applied to Smartphone applications.

Fig. 4, 5 and 6, present respectively the cumulative number of failures per time, the failure intensity per time and the Mean Time Between Failure (MTBF) per time, for the Skype application when applying the NHPP model. The RGA tool indicates an evident failure.



Plot Type: Cum. Number of Failures

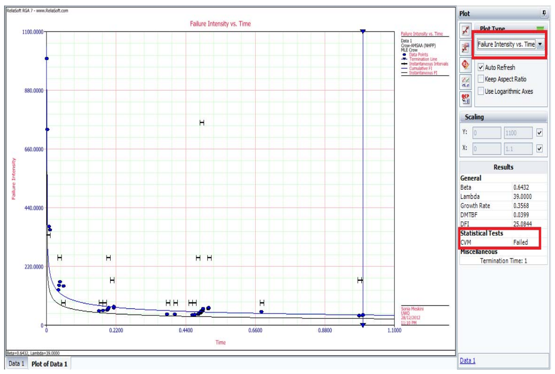
✓ Auto Refresh

Scaling Y: 0 40 ✓
X: 0 1.1 ✓

Results

General	
Beta	0.6432
Lambda	39.0000
Growth Rate	0.3568
DMTBF	0.0399
DFI	25.0844
Statistical Tests	
CVM	Failed
Miscellaneous	
Termination Time: 1	

Fig. 4. Cumulative number of failures per time (Skype)



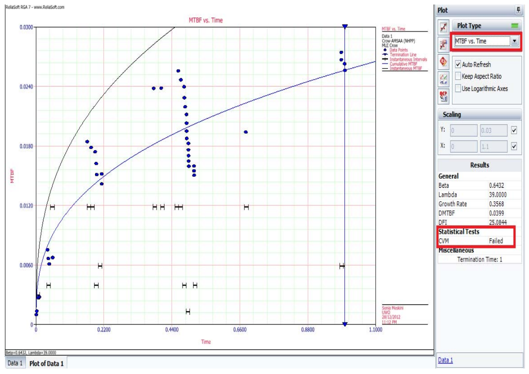
ReliaSoft RGA 7 - www.ReliaSoft.com
Failure Intensity vs. Time
Failure Intensity
Time
Data 1
Plot of Data 1

Plot Type: Failure Intensity vs. Time
☑ Auto Refresh
☐ Keep Aspect Ratio
☐ Use Logarithmic Axes

Scaling
Y: 0 1100 ☑
X: 0 1.1 ☑

Results	
General	
Beta	0.6432
Lambda	39.0000
Growth Rate	0.3568
DMTBF	0.0399
DFI	25.0844
Statistical Tests	
CVM	Failed
Miscellaneous	
Termination Time: 1	

Fig. 5. Failure intensity per time (Skype)

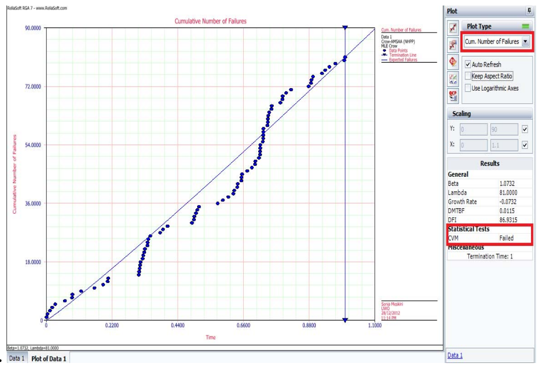


Plot Type: MTBF vs. Time
☑ Auto Refresh
☐ Keep Aspect Ratio
☐ Use Logarithmic Axes

Scaling
Y: 0 0.03 ☑
X: 0 1.1 ☑

Results	
General	
Beta	0.6432
Lambda	39.0000
Growth Rate	0.3568
DMTBF	0.0399
DFI	25.0844
Statistical Tests	
CVM	Failed
Miscellaneous	
Termination Time: 1	

Fig. 6. MTBF per time (Skype)



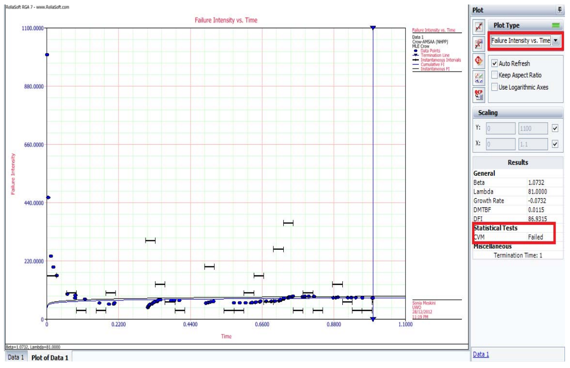
ReliaSoft RGA 7 - www.ReliaSoft.com
Cumulative Number of Failures
Cumulative Number of Failures
Time
Data 1
Plot of Data 1

Plot Type: Cum. Number of Failures
☑ Auto Refresh
☐ Keep Aspect Ratio
☐ Use Logarithmic Axes

Scaling
Y: 0 90 ☑
X: 0 1.1 ☑

Results	
General	
Beta	1.0732
Lambda	81.0000
Growth Rate	-0.0732
DMTBF	0.0115
DFI	86.9315
Statistical Tests	
CVM	Failed
Miscellaneous	
Termination Time: 1	

Fig. 7. Cumulative number of failures per time (Vtok)



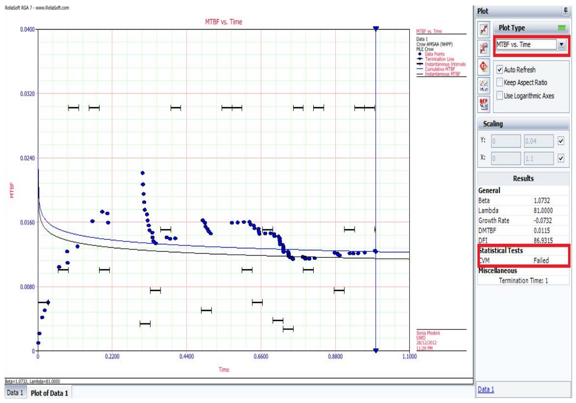
ReliaSoft RGA 7 - www.ReliaSoft.com
Failure Intensity vs. Time
Failure Intensity
Time
Data 1
Plot of Data 1

Plot Type: Failure Intensity vs. Time
☑ Auto Refresh
☐ Keep Aspect Ratio
☐ Use Logarithmic Axes

Scaling
Y: 0 1100 ☑
X: 0 1.1 ☑

Results	
General	
Beta	1.0732
Lambda	81.0000
Growth Rate	-0.0732
DMTBF	0.0115
DFI	86.9315
Statistical Tests	
CVM	Failed
Miscellaneous	
Termination Time: 1	

Fig. 8. Failure intensity per time (Vtok)



Plot Type: MTBF vs. Time
☑ Auto Refresh
☐ Keep Aspect Ratio
☐ Use Logarithmic Axes

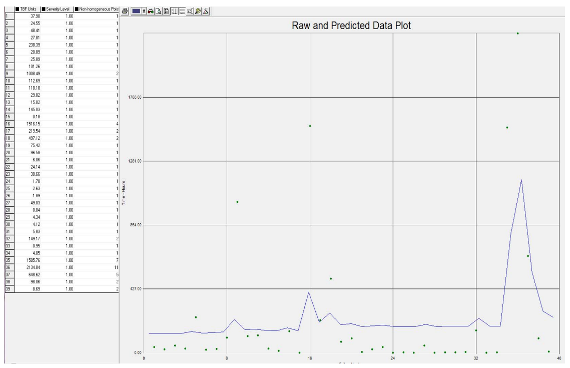
Scaling
Y: 0 0.04 ☑
X: 0 1.1 ☑

Results	
General	
Beta	1.0732
Lambda	81.0000
Growth Rate	-0.0732
DMTBF	0.0115
DFI	86.9315
Statistical Tests	
CVM	Failed
Miscellaneous	
Termination Time: 1	

Fig. 9. MTBF per time (Vtok)

Fig. 7, 8 and 9, represent respectively the cumulative number of failures per time, the failure intensity per time and the MTBF per time of Vtok. Again the NHPP model failed to fit the data. As mentioned before, we used Skype for a year and collected the failure data that contains some missing values, and Vtok was used for two months and collected the failure data with more than one failure per day. However the NHPP still fails to fit these two different types of data. One reason is that the failure data is a dynamic process for mobile applications which means that the occurring number of failures is unpredictable, sometimes decreasing and sometimes increasing, (for example in Fig. 7 from t = 0.20 until t = 0.309 the application didn't experience a failure and from t = 0.309 until t = 0.348 an important number of failures occurred).

In order to confirm our results we used a second tool, SMERFs, and we applied the NHPP model on the same data points. The result was the same which is the failure of the model each time (see Fig. 10).



	TBF Units	Severity Level	Non-homogeneous Pois
1	37.90	1.00	1
2	24.55	1.00	1
3	48.41	1.00	1
4	27.81	1.00	1
5	238.39	1.00	1
6	20.89	1.00	1
7	25.89	1.00	1
8	101.26	1.00	1
9	1008.49	1.00	2
10	112.69	1.00	1
11	118.18	1.00	1
12	29.82	1.00	1
13	15.02	1.00	1
14	145.03	1.00	1
15	0.18	1.00	1
16	1516.15	1.00	4
17	219.54	1.00	2
18	497.12	1.00	2
19	75.42	1.00	1
20	96.58	1.00	1
21	6.06	1.00	1
22	24.14	1.00	1
23	38.66	1.00	1
24	1.78	1.00	1
25	2.63	1.00	1
26	1.89	1.00	1
27	49.03	1.00	1
28	0.04	1.00	1
29	4.34	1.00	1
30	4.12	1.00	1
31	5.83	1.00	1
32	149.17	1.00	2
33	0.95	1.00	1
34	4.05	1.00	1
35	1505.76	1.00	7
36	2134.84	1.00	11
37	648.62	1.00	5
38	98.06	1.00	2
39	8.69	1.00	2

Fig. 10. NHPP model applied to Skype Time Between Failure (TBF) data

Fig. 11, 12 and 13 show the results of the same data from Skype and Vtok application respectively when applying the Musa-Basic and Musa-Okumoto models. Each time the models fail to fit the data. The models failed completely to fit the Vtok failure (Fig. 13). Fig. 14, 15 and 16 represent the results of the application of the NHPP model to the Windows phone failure data. Once again the RGA tool indicates the failure of the model

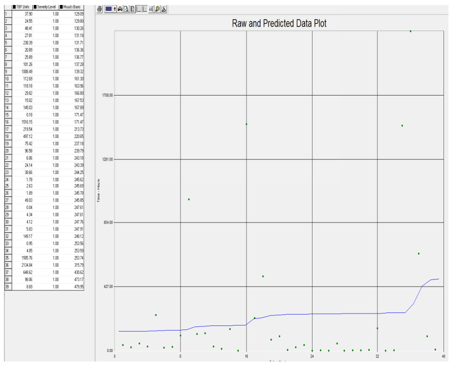


Fig. 11. Skype failure data and failure of the Musa-basic model

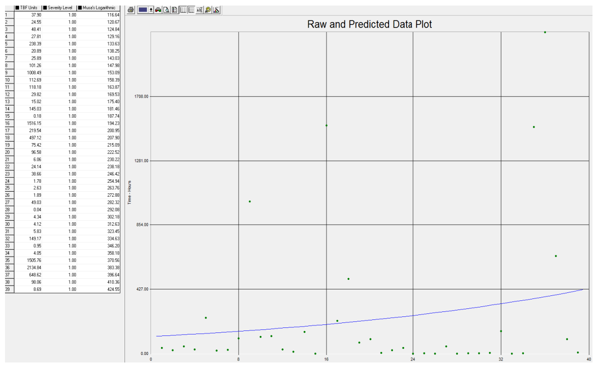


Fig. 12. Skype failure data and failure of the Musa-Okumoto model

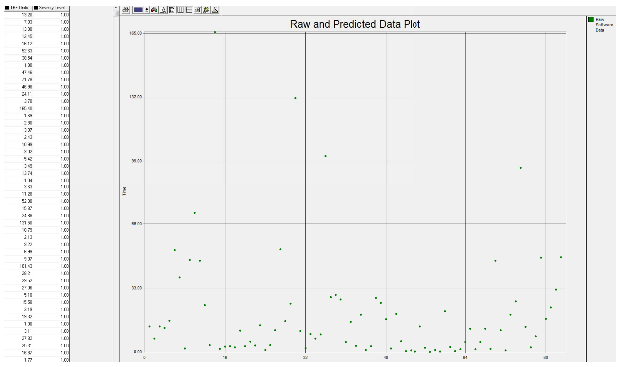
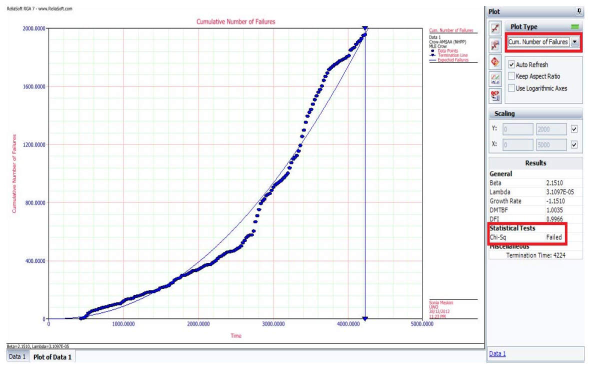


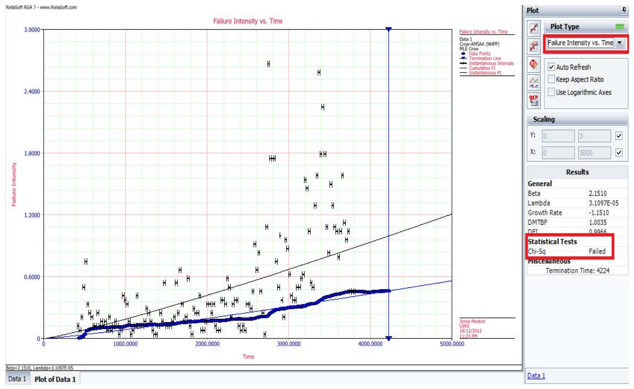
Fig. 13. Vtok failure data and failure of the three selected models



Results
General
Beta 2.1510
Lambda 3.1097E-05
Growth Rate -1.1510
DMTBF 1.0035
DFI 0.9966
Statistical Tests
Chi-Sq Failed
Miscellaneous
Termination Time: 4224

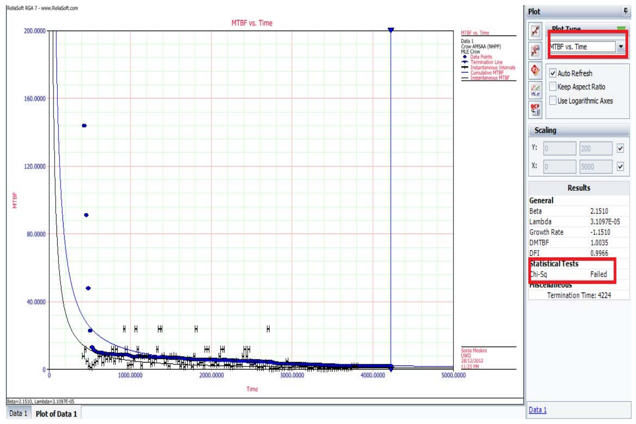
Fig. 14. Cumulative number of failure per time (Windows phone data)

One explanation of the failures of the models is the Goodness-of-fit tests that failed for each model application. Two tests were used; the Cramer-Von Mises (CVM) and the Chi-Square test. For both tests, the statistic (test value) has to be less than the critical value in order to have a successful fit otherwise it is a failure as it is the case in our experiments (this is confirmed in Fig. 7, by performing the CVM test where the model failed because the statistic (0.358) is greater than the critical value (0.173) same in Fig.14 when performing the Chi-Square test; the statistic (1282.005) is higher than the critical value (180.094) thus the model didn't fit the data) [14].



Results
General
Beta 2.1510
Lambda 3.1097E-05
Growth Rate -1.1510
DMTBF 1.0035
DFI 0.9966
Statistical Tests
Chi-Sq Failed
Miscellaneous
Termination Time: 4224

Fig. 15. Failure intensity per time (Windows phone data)



Results
General
Beta 2.1510
Lambda 3.1097E-05
Growth Rate -1.1510
DMTBF 1.0035
DFI 0.9966
Statistical Tests
Chi-Sq Failed
Miscellaneous
Termination Time: 4224

Fig. 16. MTBF per time (Windows phone data)

Thus, the most successful reliability models failed to fit all data and to predict the reliability in the mobile area for Smartphone applications. This failure can be traced back to the main differences between desktop and Smartphones. One of the mobile application failure characteristics is that they are application dependent in the sense that they are dynamic and non-homogenously spread in time. Moreover, they are unpredictable, sometimes decrease and sometimes increase. One possible explanation is that reliability depends on how the application is used (Fig. 3), where and when. Because the usage may differ from one user to another, from one country to another, from one condition and time to another, etc. which explains the uncertainty of usage of the application in the execution and release time, all these factors play an important role in the reliability of the applications.

Another reason is that the DLC of a mobile application is short and the programmer aims to develop the application as quickly to satisfy the time to market constraint which leads to skip phases from the DLC such as the design phase which is the most important phase in the DLC of the application. Thus, it would be difficult to identify the causes of errors, during the execution time, and find a convenient solution to fix them. Besides that, the failure or unreliability of the application may be caused by the technology used during the development process. Also, the skills of the developer and the tester play a huge role in the reliability of the application. Moreover, the device itself and its hardware characteristics such as the size of the screen, the performance, the keyboard, etc. can have a direct effect on the reliability of the application. For example to adjust the map size to a certain zoom level, a zoom in/out function is needed. However, to assure a perfect usage of this function the performance of the device has to be taken into consideration [9].

Other reasons that may explain this dynamic aspect of the Smartphone applications are summarized in Table 1 that gives an idea about the different causes, external and internal, of the unreliability of the application. Due to the space limitation of the paper, a complete list of the causes and their description and examples can be found in [20].

TABLE I. CAUSES OF APPLICATIONS CRASH

Cause of Failure	Description
Code	Failures arise when not taking into consideration the limited resources of the device such as power and memory
Interfaces	WAP Gateway fails when converting WTP request to HTTP request
Hardware	Various models of devices: developers should take into consideration the specific platform and performance of each device
Non-executable files	Failure to open the help, demonstration or samples files of an application
Interaction	Thanks to the SOA, many application interfaces are located on a server. Thus, mobile applications have to connect to the server to accomplish data transfer and carry out tasks. Failure in the server may cause the crash of the application
Data input	The application has to be developed in a manner that the data input has to be optimized to ensure maximum efficiency for the user.
Third-party software failures	Smartphone application architecture uses third-party software applications (for example as Facebook and Adobe Photoshop Express to be able to modify and upload pictures). A crash/problem in the third-party application may cause the failure of the other application.
Wireless Network	The sudden loss of connection or failure in configuration may cause the failure of the application.
Mobile Database	Failure to connect to the database due to an error occurred in the database server.
OS version	Some Smartphone applications may not be compatible with upgraded OS version (for example Gas Prices Canada application is no longer available for iOS 6)
Software upgrades	Upgrading from a version to another may fix problems but cause others as it was the case with Skype 4.2.2601 and Skype 4.2.2604, the updated version crashes more often than the previous version when making calls

Based on different surveys and studies, reliability was identified as the most important quality attribute of the application software. Thus, the reliability of Smartphone applications needs to be assured since everyone is using their own Smartphones for daily life activities and tasks more than PCs. Our study confirms that a reliability growth model adapted to Smartphone applications is needed since the traditional reliability models turned out to be inefficient.

VII. CONCLUSIONS AND FUTURE WORK

Our work is a step toward the application and evaluation of traditional Software Reliability models in the mobile area. We selected three of the most used models that are known for their efficiency in the desktop area: the NHPP, Musa-Basic, Musa-Okumoto models. We used two iPhone applications, Skype and Vtok that were used and tested differently to evaluate the models under different conditions, and one Windows phone application that we didn't mention the name because of the company's confidential policies. It turned out that none of the selected SRGMs was able to account for the failure data satisfactorily.

Our study also highlighted the causes of the failure of the models and the need for a meticulous Software Reliability Growth Model for Smartphone applications; this is because the existing Software Reliability approaches are developed for traditional desktop software applications that are static and stable during their execution which it is not the case for Smartphone applications which have unknown operational profile, highly dynamic configuration and changing execution conditions. On a continuous background, the smartphone failures come in relatively short bursts from time to time which explain the abrupt changes in the observed cumulative failure number curves. This particular feature cannot be accommodated by the used SRGMs. Thus, in order to evaluate the reliability of Smartphone applications, new models, principles and tools are needed to incorporate the underlying uncertainties of such applications [21].

Our investigation of Smartphone application reliability through the use of well-known available growth models, suited primarily to desktop applications, is twofold: (i) highlight the versatile nature of mobile applications, their dynamic configuration, unknown operational profile and varying execution conditions in contrast to the static and stable desktop ones, (ii) stress the need for the design of new reliability models suited for mobile applications which take into account the inherent versatility of such applications.

As is well known, reliability is one of the most important features of an application and great efforts have been devoted to tailor and predict it through the study of recorded failure data. A non-reliable application means dissatisfied customers, loss of market share and significant costs to the supplier. For critical applications, such as banking or health monitoring, non-reliability can lead to great damage. Therefore, it is of great necessity to insure early detection and resolution of reliability issues in desktop applications as well as, now increasingly, in mobile applications.

Our future work will focus on analyzing more in depth these selected SRGMs and try to modify the closest one to the data and adapt in to Smartphone applications. Moreover, we will check if we need to have for each type of applications a specific model or one model is applicable to all the categories of Smartphone applications taking into consideration the severity of the failure.

Another future purpose is to evaluate the possibility of applying more than one model on the same failure data such as the Windows phone crash count by dividing the data into two or more categories and applying the convenient model to each category to predict the reliability of the application. Further investigations of Android failure data are also underway.

REFERENCES

[1] U.D. Perera, "Reliability of Mobile Phones", Annual Reliability and Maintainability Symposium, Washington, DC, USA, pp. 33-38, 1995.

[2] F. Ahmed and L.F. Capretz, "Managing the Business of Software Product Line: An Empirical Investigation of Key Business Factors", Information and Software Technology, Vol. 49, pp. 194-208, 2007.

[3] F. Ahmed, L.F. Capretz and J. Samarabandu, "Fuzzy Inference System for Software Product Family Process Evaluation", Information Sciences, Vol. 178, pp. 2780-2793, 2008.

[4] J. Pam, "Software Reliability", http://www.ece.cmu.edu/~koopman /des_s99/sw_reliability/, 1999.

[5] J. Xu, D.Ho and L.F Capretz, "An Empirical Validation of Object-Oriented Design Metrics for Fault Prediction," Journal of Computer Science, Vol. 4, No. 7, pp. 571-577, ANSInet, 2008.

[6] H.F. El Yamany, M.A.M. Capretz and L.F. Capretz, "A Multi-Agent Framework for Testing Distributed Systems", 30th IEEE International Computer Software and Applications Conference (COMPSAC), Chicago, MI, USA, Vol. II, pp. 151-156, 2006.

[7] Mharmer, "Modern Mobile Apps", http://blog.modernmobileapps.com/category/chips.aspx, 2011.

[8] L.F. Capretz and P.A. Lee, "Reusability and Life Cycle Issues within an Object-Oriented Methodology", Proceedings of the Eight International Conference on Technology of Object-Oriented Languages and Systems, Prentice-Hall, Santa Barbara, CA, USA, pp. 139-150, 1992.

[9] S. Jang and E. Lee, "Reliable Mobile Application Modeling Based on Open API", Proceedings of International Conference on Advanced Software Engineering and its Applications, Jeju Island, Korea, pp. 168-175, 2009.

[10] A. Wasserman, "Software Engineering Issues for Mobile Application Development," Worshop on the Future of Software Engineering Reasearh (FoSER), Santa Fe, NM, USA, ACM Press, 2010.

[11] M. R. Lyu, "Handbook of Software Reliability Engineering", Chapter 3, McGraw-Hill, 1996.

[12] Y. K. Malaiya and J. Denton, "What do the Software Reliability Growth Model Parameters Represent?", 8th IEEE International Symposium on Software Reliability Engineering, Albuquerque, NM, USA, pp. 124-135, 1997.

[13] R. Lai and M. Garg, "A Detailed Study of NHPP Software Reliability Models", Journal of Software, Vol. 7, No. 6, pp. 1296-1306, 2012.

[14] ReliaSoft, "Reliability Growth & Repairable System Data Analysis Reference", http://rga.reliasoft.com/, 2010.

[15] J.D. Musa, A.Iannino and K. Okumoto, "Software Reliability – Measurement, Prediction, Applications", McGraw-Hill, 1987.

[16] W. Xia, L.F. Capretz and D. Ho, "A Neuro-Fuzzy Model for Function Point Calibration", WSEAS Transactions on Information Science and Applications, Vol. 5, pp. 22-30, 2008.

[17] A.B. Nassif, D. Ho and L. F. Capretz. "Towards an Early Software Estimation Using Log-Linear Regression and a Multilayer Perceptron Model", Journal of Systems and Software, Vol. 86, pp. 144-160, 2013.

[18] A.B. Nassif, L. F. Capretz and D. Ho, "Software Estimation in the Early Stages of the Software Life Cycle", International Conference on Emerging Trends in Computer Science, Communication and Information Technology, Nanded, India, pp.5-13, 2010.

[19] A.B. Nassif, L. F. Capretz and D. Ho, "Estimating Software Effort Based on Use Case Point Model Using Sugeno Fuzzy Inference System", 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Boca Raton, FL, USA, pp. 393-398, 2011.

[20] A.K. Jha, "A Risk Catalog for Mobile Applications", Chapter 5, Florida Institute of Technology, Melbourne, FL, USA, 2007.

[21] S. Malek, R. Roshandel, D. Kilgore and I. Elhag, "Improving the Reliability of Mobile Software Systems through Continuous Analysis and Proactive Reconfiguration", 31st IEEE International Conference on Software Engineering, Vancouver, BC, Canada, Vol. ICSE-Companion, pp. 275-278, 2009.