ISSN: 0067-2904

Monitoring and Enhancement of Mobile System Performance

Khalid S. Noori*, Assmaa A. Fahad
Computer Science Department, College of Science, University of Baghdad, Baghdad, Iraq

Received: 3/12/2019 Accepted: 27/3/2020

Abstract

Android operating system, since its first start, is growing very fast and takes a large space in smart devices market. It is built and developed on Linux and designed basically for touch screen devices such as, mobiles, tablets, etc. Mobile devices are markedly complicated and feature-rich; therefore they are prone to reliability of software and performance problems. Because of the small resources, smart devices, such as CPU, RAM, suffer from problems. One of these problems is Software Aging (SA). SA is recognized in long running OSs as a shortage in resources, performance retreating, and finally failure. SA is looked at from two sides, namely the poor response time of application which represents the end user side and the shortage in metrics related to device resources, such as RAM and storage. In this paper, a set of eight experiments is conducted to distinguish SA in Android mobiles. These experiments are conducted to find the correlation between Launch Time (LT) with RAM and storage metrics covered in this paper. Statistical methods, such as Mann Kendall test, Sen's slope, Spearman rank correlation, and Design of Experiment (DOE) are used to prove the correlation statistically. These experiments assist to detect SA, which will be helpful in the rejuvenation strategy of applications.

Keywords: Software Aging, Launch Time, Mann Kendall, Sen's slope, Spearman rank correlation, DOE.

تحسين و مراقبة اداء نظام الموبايل

خالد صباح نوري* , أسماء عبدالله فهد
قسم علوم الحاسوب, كلية العلوم، جامعة بغداد، بغداد, العراق

الخلاصة

نظام التشغيل اندرويد منذ بدايته, ينمو بسرعة عالية و يأخذ حيز كبير في سوق الاجهزة الذكية. الاندرويد بني و طور على نظام اللينكس و مصمم بشكل رئيسي للأجهزة ذات الشاشات القابلة للمس مثل, الموبايلات, الاجهزة اللوحية, .. الخ. أجهزة الموبايل معقدة بشكل ملحوظ و غنية بالميزات, لذلك فهي عرضة لمشاكل متعلقة بالاداء والاعتمادية على البرامج. بسبب امتلاكها لمصادر صغيرة مثل, وحدة المعالجة المركزية, ذاكرة الولوج العشوائي, الخ فهي تعاني من مشاكل. واحدة من هذه المشاكل هي شيخوخة البرامج. يتم التعرف على شيخوخة البرامج في انظمة التشغيل العاملة لفترات طويلة من خلال النقص في الموارد, تراجع الاداء, و اخيرا الفشل. شيخوخة البرامج ينظر لها من جانبين: وقت الاستجابة الضعيف للبرامج و الذي يمثل جانب المستخدم, و النقص في الموارد الخاصة بالجهاز مثل, ذاكرة الولوج العشوائي, و المساحة الخزنية و التي تمثل جانب النظام. هذا البحث, مجموعة من ثمان تجارب يتم إجراؤها لغرض تمييز شيخوخة البرامج

*Email: khalidsn82@yahoo.com

في اجهزة الموبايل العاملة بالاندرويد. هذه التجارب تجرى لغرض اثبات العلاقة بين وقت الانطلاق للتطبيق مع المقاييس الخاصة بذاكرة الولوج العشوائي و الخزن و التي سيتم التركيز عليها في هذا البحث. اختبار مان كندل, منحدر سين, ارتباط سبيرمان على أساس الرتبة, و تصميم التجارب أستخدمت لاثبات العلاقة أحصائياً. هذه التجارب تساعد في اكتشاف شيخوخة البرامج و التي تكون ذا فائدة في استراتيجية اعادة الشباب للتطبيقات.

1. Introduction

Mobile Operating Systems (MOSs) are systems designed specifically for mobile phones, tablets, and wearables. These are basically light weight OSs which requires low resources of power, storage space, CPU, RAM, etc [1]. They encompass all the essential components in personal computers, including WiFi, camera, video player, etc. , in addition to features useful for mobile use, such as the Subscriber Identity Module (SIM) tray for telephony and data connection [2].

Mobile OSs are built using various OS kernels, such as Linux, Unix, Windows, etc. The most famous and trending mobile OSs in the market are Android and iOS [1].

Android OS, just like other OSs, has many problems that include CPU utilization, power consumption, security threats, and RAM shortage. But fast response time is one of the important features that the user searches for. Poor response time, which is the representation of SA is one of these problems that is related to RAM shortage.

SA representations include resource exhaustion, performance degradation, and failure rising, from which the customers are suffering [3, 4, 5]. SA is a progressive performance degradation phenomenon or sudden failure of the system related to aging bugs such as fragmentation, accumulation of errors, and resources' exhaustion. It happens in long running systems that run applications for a long period of time. SA can be seen from two points of view; first, the user's view point which is represented by the Launch Time (LT) of the application that is directly perceived by the user; second, the system's view point, which is represented by the system's metrics. Different metrics are related to different resources. Some of memory metrics are those of "totally free" and "totally used", while some of the storage metrics are those of "reads completed" and "writes completed". These metrics are not perceived directly by the user and are used to check the influence of them to the LT [4]. Many software rejuvenation strategies are used to handle SA, such as system rebooting and application restart [6].

The rest of this paper is structured as follows: Section 2 discusses the related work. Section 3 discusses SA in resources and their metrics. Section 4 discusses testing utilities used to detect SA. Section 5 discusses statistical methods for SA detection process. Section 6 discusses the designed module of the paper. Section 7 discusses the results of the experiments. Section 8 includes the conclusions.

2. Related work

Application responsiveness is one of the main concerns of Android users which is caused by SA. Many researches were conducted on SA for different Oss, such as Unix and Linux, but only few were reported on Android system. This paper intends to provide an organized view of the recent achievements of SA in Android system research to cope with the rapid changes mentioned above. This paper focuses on investigating the recent evolutions of SA in Android system, concentrating on RAM and storage, as well as understanding the current trends and the directions of future Android SA research. The previous related works are described below.

An earlier work [7] introduced an empirical study to investigate the existence of SA in Android OS under various circumstances. Collecting system memory information was performed through an application developed for this purpose. Two Stress test experiments with fixed workload and exponential workload were conducted on Android OS to discover the SA phenomena. Collecting system information was performed every five minutes for twenty four hours for each experiment. In both experiments, GC_CONCCURENT (garbage collection that is activated by Android OS when the memory allocated is too large) was activated to collect garbage for releasing memory space when memory leaking happens. As a result of experiments, the SA existed in Android OS under the aforementioned stress tests. A previous article [8] considered the problem of SA in Android mobile OS. An experimental methodology was suggested that uses statistical methods (Mann Kendall test, one way Analysis Of variance (ANOVA), and Spearman's rank correlation coefficient) to recognize factors of the experimental plan. It involved an application set which represents application type to

stimulate the system device, which represents the physical mobile with its H/W and S/W configurations, workload and kill frequency. The experiment was performed every five and sixty seconds, with configuration of workload events such as touches, switches, using Monkey tool and storage space usage, in two modes, namely full and normal. Resource utilization metrics (memory and storage) with system operations (garbage collection and tasks) were tested in relation with SA. In addition, an empirical analysis was performed on recent Android devices and pointed out the processes (System Server, Surface Flinger, and System UI) and components of Android OS (such as Activity Manager) affected by SA. Metrics useful as indicators of SA to schedule software rejuvenation actions were also tested. As a result of those experiments, it was recommended that Android software rejuvenation should select a measurement-based approach to be familiar with the workload conditions. An earlier report [9] determined the main objective of the research to discover SA in and to study the influence of warm rejuvenation strategy (application restart) in Android OS. It was concentrated on available memory as aging-related phenomena and utilized five experiments; experiments one and two were conducted to confirm whether aging can happen in Android; experiments three and four verified whether warm rejuvenation can work on Android aging recovery; experiment five was performed to find out whether Android SA is reversible. Also, a mathematical modelling (Markov chains) was implemented to predict the anticipated time for Android OS to go into the aging condition. As a result of those experiments, the presence of aging in Android was noticed. In addition, the warm rejuvenation (application restart) had a small influence on aging recovery or alleviation process. The authors assumed that the only way to deal with this issue when occurs is to reboot OS or use other unknown technology strategies that may be developed in the future.

3. SA Analysis and Metrics

This section will discuss the SA problem in main mobile resources such as RAM and storage.

A. SA in RAM

RAM is a valuable resource, precisely on smart phones. It is a critical resource for attaining good performance, but many studies showed that it is often affected by SA. Android provides a complicated mechanism to manage memory, both at the kernel level and the Android frame work level Some of these managements include application life cycle, application process caching, and reclaiming memory through OOMK and LMK [8]. When an application exits, activity manager service does not kill the process but instead places it in the Least Recently Used (LRU) list, in case the user returns to use it again to minimize application response time. The majority of Android devices have a restricted memory, and running applications in (out of memory) or (low memory) states will be high. Therefore, when most of the applications are launched, Android needs to free memory that influences the application LTs, which is the period between touching the application icon on screen (requesting an activity) until the first view of contents of screen for interaction. The LT of application plays a pivotal role in user experience of the application and, thus, memory metrics are included to analyse memory usage in the experiments and their effects on application LT. List of some memory metrics are shown in Table- 1[8].

Table 1- List of some memory metrics [8]

Metrics used	Measurement unit	Description
Total Free	KB	The size of the unused physical memory
Free Cached	KB	The size of the physical memory that is used as cache memory
Free Cached Proportional Set Size (PSS)	KB	The size of the physical memory that is recently used by PSS of cached processes (not in use)
Total Used	KB	The size of the used physical memory
Used PSS	KB	The size of the used physical memory by non-cached processes
Used Buffers	KB	The size of the used physical memory for file buffers
Used Shared Memory	KB	The size of the used physical memory by shared memory
Used Slab	KB	The size of the used physical memory by kernel to cache data
Lost RAM	KB	The size of the not free or used physical memory

B. SA in Storage Space

Internal storage is used to save files that are related to specific applications, where any other application cannot have access to them. The Android system provides an individual directory (private) for each application to organize any files the application requires [10]. The availability or not of storage space in experiments (in terms of free space) may or may not influence the app LTs. Hence, storage metrics are included to analyse the storage space usage. Some of storage metrics are shown in Table- 2 [6].

Table 2- Some of storage metrics [6]

Metrics used	Measurement unit	Description
Reads Completed	No. of reads	The total number of reads that are completed successfully
Reads Merged	No. of reads	The total number of reads which are adjacent to each other may be merged for efficiency
Sectors Read	No. of sectors	The total number of sectors that are read successfully
Reading Time	Milli second	The total number of time that is spent by all reads
Writes Completed	No. of writes	The total number of writes that are completed successfully
Writes Merged	No. of writes	The total number of writes which are adjacent to each other may be merged for efficiency
Sectors Written	No. of sectors	The total number of sectors that are written successfully
Writing Time	Milli second	The total number of milliseconds that is spent by all writes
Read Completion Time	Milli second/read	The average amount of time doing I/O read operation
Write Completion Time	Milli second/write	The average amount of time doing I/O write operation
Weighted I/O Time	Milli second	The number of I/O that are in progress times the number of milliseconds spent doing I/O

4. SA Testing Utilities

SA utilities that are used in the experiments to collect data include Monkey Tool [11], Dumpsys tool [12], Logcat tool [13], ADB tool [14], and /proc File System [15].

5. SA Statistical Methods

Several statistical methods are used to analyse the data collected over time. Some of statistical methods used in this paper include Mann Kendall (MK) test [16], Sen's slope estimator [16], Spearman's rank correlation coefficient [17], and Design Of Experiment (DOE) [18, 19].

6. The Designed Module

The designed module of the paper consists of many parts, including experiment setup, experiment design, experiment factors and levels, and experiment plan.

A. The Experiment Setup.

The following represents a complete view of the experiment platform which consists of several parts:

- The testbed: The conducted experiments were implemented on Samsung Note3 mobile phone equipped with 3GB of RAM and 32GB of internal storage. The mobile was installed with an Android 5.0 Lollipop operating system. A PC computer with 8GB RAM and 500GB hard disk was used to send user events and inject them into the mobile to gather the required information. The OS installed on the PC was 64-bit Ubuntu 18.04.1 LTS.
- User events generator: to simulate user events, Monkey tool was used to generate such events like, touch, motion, trackball, navigation, majornavigation, syskeys, appswitch, anyevent, flip, and pinchzoom. In each experiment, 10,000 events were injected into the mobile with 500 milli second (ms) throttle (delay) between each group of events. These events stress the mobile OS over a period of time in order to accelerate the appearance of SA.

- Test applications: Two sets of applications were used in the experiments, namely the third party applications which were downloaded from play store. and system applications which were already installed by the manufacturer and cannot be uninstalled in the mobile phone.
- Data collection: In this part, eight experiments were conducted with different combinations of factors at each level. Each experiment was conducted t for one hour. During each experiment, five applications (third party applications or system applications) were launched and killed periodically every thirty seconds. A bash script was developed in order to launch and kill applications, generate events, collect data, and save the collected data to files in order to analyse them later using statistical methods mentioned in section 4. The collected data are:

- The applications LTs that are collected every thirty seconds and represent user perceived response metrics.
- RAM and Storage information that is collected every thirty seconds and represent system perceived response metrics.

In this paper, dumpsys, logcat and /proc were used to gather the aforementioned metrics. Statistical methods were used for analysing the collected data to discover the existence of SA in the mobile.

B. The Experiment Design

The experiment design (Figure-1) is passing through many steps. Collecting LTs of applications with RAM and storage metrics information was performed every thirty seconds. The median value was selected from the collected LTs of applications every thirty seconds at each time period. The MK test and the Sen's slope estimator were applied to the median LTs of applications with RAM and Storage metrics information every thirty seconds to check for trends and finding the slopes, respectively.

The Spearman's rank correlation coefficient was applied between the slopes of RAM metrics and the slopes of median LTs of applications collected every thirty seconds to find the metrics that affect the LTs of applications. The same method was applied to the slopes of storage metrics and the slopes of median LTs of applications collected every thirty seconds to find the metrics that affect the LTs of applications.

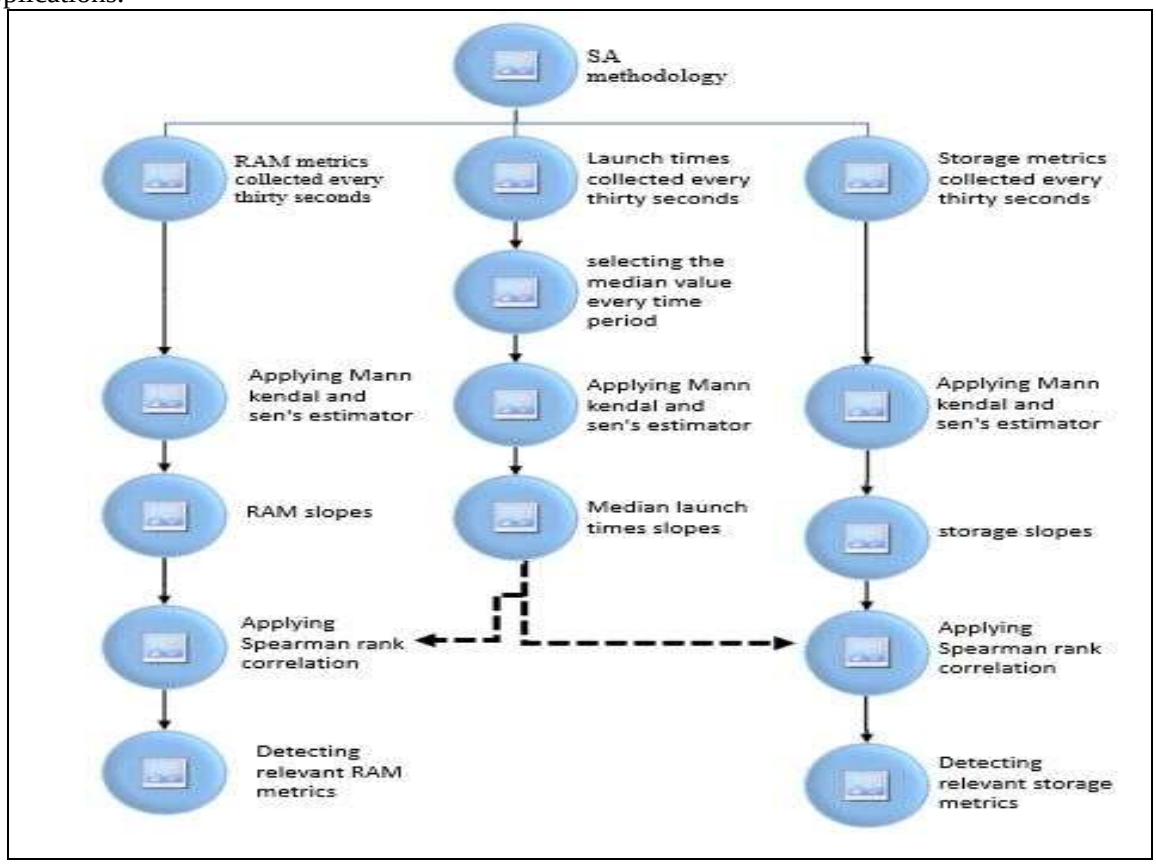

Figure 1- The experiment design.

C. The Experiment Factors and Levels

To stress the Android OS broadly and to bring out the hidden SA issues, the tests were run under many different configurations and stress applications, which represent the factors of the experiment plan. Three factors were considered in this paper and derived the experiment plan by varying the combinations of levels of these factors according to DOE (Table-3). A full factorial design of the experiments conducted on the mobile was adopted (Table-4).

Table 3 – The experiment factors and levels

Factor	Level	Description
Application	**third party applications**	- **io.faceapp** - **com.google.android.applications.translate** - **com.newpower.apkmanager** - **com.infraware.office.link** - **org.videolan.vlc**
	System applications	- **com.sec.android.applicationpopupcalculator** - **com.sec.android.applicationclockpackage** - **com.samsung.helphub** - **com.android.mms** - **com.sec.android.applicationmusic**
Events	**Events**	**Monkey tool sends touch, motion, trackball, navigation, majornavigation, systemkeys, switch, anyevent, flip, and pinchzoom**
	None	**Monkey tools is not sending events (not used)**
Storage	**Normal**	**Default storage space**
	Full	**Ninety percent storage space**

Table 4 - The experimental plan

Application	Events	Storage
third party applications	**Events**	**Normal**
System applications	**None**	**Normal**
System applications	**None**	**Full**
System applications	**Events**	**Normal**
third party applications	**Events**	**Full**
System applications	**Events**	**Full**
third party applications	**None**	**Normal**
third party applications	**None**	**Full**

7. Results and discussion

In this stage, the Sen's slope estimator for median LTs of applications, RAM metrics, and storage metrics was computed statistically. These Sen's slopes (values) were used to find the correlation between LT degradation with RAM and storage metrics, using Spearman's rank correlation coefficient. The Spearman's rank correlation coefficient was used to compute the correlation between median LT slopes of applications and memory metrics slopes across all the conducted eight experiments. The results of the correlation test are shown in Table-5. Correlation values between negative one to positive one, except zero value, imply that the correlation exists as the LT and memory metric both increase or decrease at the same time, or when one increases and the other decreases.

Table 5- Spearman's rank correlation coefficient between median LT slopes and memory metrics slopes

Memory metrics	Spearman correlation coefficient	P - value
Total Free	**-0.048**	**0.911**
Free Cached	**0.214**	**0.610**
Free Cached PSS	**0.286**	**0.493**
Total Used	**0.190**	**0.651**

Used PSS	**0.095**	**0.823**
Used Buffers	**0.048**	**0.911**
Used Shared Memory	**0.203**	**0.630**
Used Slab	**-0.657**	**0.156**
Lost RAM	**-0.143**	**0.736**

By taking a significance level of 95% ($\alpha = 0.05$), no one of memory metrics was found to be correlated to LT in a statistical significance way, according to the p-values of the metrics ($P>0.05$). The results of the correlation between LT and RAM metrics did not mean that there was no correlation at all in all metrics, for two important reasons:

- The duration of the experiment, which was one hour, could be increased to many hours or until the device will be in an unstable condition. This might show a correlation between LT and memory metrics.
- The type of workload could also be a cause of this correlation result.

In the same manner, Spearman's rank correlation coefficient was also used to compute the correlation between median LT slopes of applications and storage metrics slopes across all experiments. The results of the correlation are shown in Table-6. By taking a significance level of 95% ($\alpha = 0.05$), sectors read and reading time metrics were correlated to LT with a statistical significance (P-value<0.05). The results of memory metrics were also affected by the duration of the experiment and the type of the workload used during it.

Table 6- Spearman's rank correlation coefficient between median LT slopes and storage metrics slopes

Storage metrics	Spearman correlation coefficient	P - value
Reads Completed	**-0.690**	**0.058**
Reads Merged	**-0.690**	**0.058**
Sectors Read	**-0.786**	**0.021**
Reading Time	**-0.738**	**0.037**
Writes Completed	**-0.381**	**0.352**
Writes Merged	**-0.595**	**0.120**
Sectors Written	**-0.595**	**0.120**
Writing Time	**-0.595**	**0.120**
Read Completion Time	**None**	**none**
Write Completion Time	**-0.595**	**0.120**
Weighted I/O Time	**-0.595**	**0.120**

8. Conclusions

Eight experiments with different factors and levels were conducted in order to recognize the existence of SA in Android mobiles. LT, memory metrics, and storage metrics were used for the detection of SA. The results of the experiments showed that none of memory metrics has an influence on LT of the applications in experiments conducted every thirty seconds, according to p-values greater than 0.05. Also it was noted that sectors read and reading time metrics were correlated to LT of the applications in experiments conducted every thirty seconds.

References

1. Wukkadada, B., Nambiar, R. Nair, A. **2015**. Mobile Operating System: Analysis and Comparison of Android and iOS. *International Journal of Computing and Technology*, **2**(7).
2. Yoon, Y. **2012**. A Study on the Performance of Android Platform. *International Journal on Computer Science and Engineering* (IJCSE), **4**(04).
3. Huo, S., Zhao, D., Liu, X., Xiang, J., Zhong, Y. and Yu, H. **2018**. Using machine learning for SA detection in Android system. Tenth International Conference on Advanced Computational Intelligence (ICACI), Xiamen, pp. 741-746.
4. Inas A., Sumaya S. and Safa A. **2016**. "Using One-Class SVM with Spam Classification", *Iraqi Journal of Science*, **57**(1B)): 501-506.

5. Ayad R. Abbas, Asaad R. Kareem, **2018**. "Age Estimation Using Support Vector Machine", *Iraqi Journal of Science*, **59**(3C): 1746-1756.

6. Weng, C., Zhao, D., Lu, L., Xiang, J., Yang, C. and Li, D. **2017**. A Rejuvenation Strategy in Android. IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Toulouse. pp. 273-279.

7. Zhao, Y. et al. **2015**. An Experimental Study on SA in Android Operating System. 2nd International Symposium on Dependable Computing and Internet of Things (DCIT), Wuhan, pp. 148-150.

8. Cotroneo, D., Fucci, F., Iannillo, A..K, Natella, R. and Pietrantuono, R. **2016**. SA Analysis of the Android Mobile OS. IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, pp. 478-489.

9. Weng, C., Xiang, J., Xiong, S., Zhao, D. and Yang, C. **2016**. Analysis of SA in Android. IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Ottawa, ON, pp. 78-83.

10. Data and file storage overview | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/guide/topics/data/data-storage.

11. UI/Application Exerciser Monkey | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/test/monkey

12. Dumpy | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/command-line/dumpsys

13. Logcat command-line tool | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/command-line/logcat#alternativeBuffers

14. Android Debug Bridge (adb) | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/command-line/adb#IntentSpec

15. Proc (5) - Linux manual page. **2019**. Retrieved 31 October 2019, from http://man7.org/linux/man-pages/man5/proc.5.html.

16. Gilbert, Richard O. **1987**. *Statistical Methods for Environmental Pollution Monitoring*. Van Nostrand Reinhold Company Inc.

17. Corder, G.W. and Foreman, D.I. **2014**. *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons Inc., second edition.

18. Montgomery, D.C. **2013**. *Design and Analysis of Experiments.* John Wiley & Sons Inc., eighth edition.

19. Wagner, J.R., Mount, E.M., Giles, H.F. **2013**. *Extrusion*. Elsevier Inc.